

Eindverslag afstuderen

‘*Ontwikkelen van een servicelaag tussen Mendix en Windows Workflow Foundation’*

|  |  |  |
| --- | --- | --- |
| **Auteur** | : | Marcel Vuijk |
|  |  |  |
| **Bedrijf** | : | Centric B.V. |
| **Bedrijfsbegeleider** | : | Dhr. W. Engberts |
|  |  |  |
| **Begeleidend examinator** | : | Dhr. E.M. van Doorn |
| **Tweede examinator** | : | Dhr. M. Reijnhoudt |
|  |  |  |
| **Datum** | : |  |
| **Plaats** | : | Capelle aan den IJssel |
|  |  |  |

# Colofoon

|  |  |
| --- | --- |
| **Titel** | Eindverslag afstuderen |
| **Ondertitel** | Ontwikkelen van een servicelaag tussen Mendix en Windows Workflow Foundation |
| **Versie** | 0.3 |
| **Datum** | juni 2012 |
| **Samengesteld door** | Marcel Vuijk |
| **E-mail** | 08064334@student.hhs.nl |
| **Telefoon** | +31 (0) 6 423 69 144 |
|  |  |

# Referaat

Eindverslag van het project: ontwikkelen van een servicelaag tussen Mendix en Windows Workflow Foundation, uitgevoerd bij Centric B.V door Marcel Vuijk, in het kader van het afstuderen voor de opleiding Informatica aan de Haagse Hogeschool.

Descriptoren:

* Centric B.V.
* Mendix
* Mendix Business Modeler
* Windows Workflow Foundation
* .Net
* Java
* Sprintr
* SCRUM
* TMap

# Voorwoord

In dit verslag worden de resultaten en de verrichte werkzaamheden van mijn afstudeerproject gepresenteerd in het kader van mijn studie Informatica aan De Haagse Hogeschool, locatie Den Haag.

De opdrachtgever van de opdracht is Centric Software Engineering. Zij is een ICT-organisatie dat onderdeel uitmaakt van Centric B.V. Ik ben hiermee in aanraking gekomen via het bedrijf Mendix, waar ik in het derde jaar van mijn studie stage gelopen heb en nadien parttime in dienst ben geweest.

Vooraf was een duidelijke opdracht geformuleerd. Deze is: ‘Het ontwikkelen van een servicelaag tussen Mendix en Windows Workflow Foundation’.

Graag wil ik het zowel het management van als de collega's binnen Centric Software Engineering bedanken voor de prettige werkomgeving en de fijne samenwerking. Mijn begeleider, dhr. W. Engberts, wil ik bedanken voor zijn rol als Product Owner, de goede begeleiding en de vrijheid die ik kreeg bij het uitvoeren van de opdracht. Mevr. K. Visser wil ik bedanken voor haar bijdrage in het project als Scrum Master. Dhr. J. Luursema wil ik bedanken voor zijn technische ondersteuning. Tot slot wil ik mijn begeleidend examinator, dhr. E.M. van Doorn en mijn 2e examinator dhr. M. Reijnhoudt bedanken voor de begeleiding en de feedback op het afstudeerdossier.

Capelle aan den IJssel, juni 2012,

Marcel Vuijk

Inhoudsopgave

[Colofoon II](#_Toc326306774)

[Referaat III](#_Toc326306775)

[Voorwoord IV](#_Toc326306776)

[Samenvatting 1](#_Toc326306777)

[1 Inleiding 3](#_Toc326306778)

[2 Glossary 4](#_Toc326306779)

[4 Achtergrond 5](#_Toc326306780)

[4.1 Centric 5](#_Toc326306781)

[4.2 Mendix 6](#_Toc326306782)

[4.3 Windows Workflow Foundation 8](#_Toc326306783)

[5 De opdracht 12](#_Toc326306784)

[5.1 Aanleiding tot opdracht 12](#_Toc326306785)

[5.1.1 Achtergrond 12](#_Toc326306786)

[5.1.2 Probleemstelling 12](#_Toc326306787)

[5.2 Aanvangssituatie 13](#_Toc326306788)

[5.3 Gewenste situatie 13](#_Toc326306789)

[5.4 Deelproducten 16](#_Toc326306790)

[5.5 Eindproduct 18](#_Toc326306791)

[6 Aanpak 19](#_Toc326306792)

[6.1 Agile ontwikkeling 19](#_Toc326306793)

[6.2 Scrum 19](#_Toc326306794)

[6.2.1 Scrum team 20](#_Toc326306795)

[6.2.2 Sprint doorloop 21](#_Toc326306796)

[6.2.3 Sprintr 22](#_Toc326306797)

[7 Realisatie 25](#_Toc326306798)

[7.1 Initiatie fase 25](#_Toc326306799)

[7.1.1 Requirements bepalen 25](#_Toc326306800)

[7.1.2 Onderzoek 30](#_Toc326306801)

[7.1.3 Software architectuur opstellen 32](#_Toc326306802)

[7.1.4 Mastertestplan schrijven 38](#_Toc326306803)

[7.2 Sprint 1 40](#_Toc326306804)

[7.3 Sprint 2 44](#_Toc326306805)

[7.4 Sprint 3 47](#_Toc326306806)

[7.5 Sprint 4 48](#_Toc326306807)

[7.6 Sprint 5 54](#_Toc326306808)

[8 Evaluatie 56](#_Toc326306809)

[8.1 Proces 56](#_Toc326306810)

[8.2 Product 57](#_Toc326306811)

[8.3 Conclusie 58](#_Toc326306812)

[9 Wijze van aantonen beroepstaken 60](#_Toc326306813)

[10 Geraadpleegde literatuur 63](#_Toc326306814)

[11 Lijst met figuren 64](#_Toc326306815)

[12 Bijlages 65](#_Toc326306816)

[Bijlage A: Creëren van een webservice in de Mendix Business Modeler 66](#_Toc326306817)

[Bijlage B: Gebruik een Mendix webservice in een workflow 68](#_Toc326306818)

[13 Externe bijlages 71](#_Toc326306819)

# Samenvatting

Hieronder volgt een korte samenvatting van dit project.

Dit verslag beschrijft de uitvoering van de opdracht: ‘*Ontwikkelen van een servicelaag tussen Mendix en Windows Workflow Foundation voor Centric B.V’.*

Mendix is een jong bedrijf dat een framework ontwikkelt waarmee webapplicaties gemodelleerd kunnen worden. De modellen, die een declaratieve beschrijving zijn van de applicatie, zijn direct executeerbaar als werkende software. Bij het ontwikkelen van applicaties met het Mendix Framework komt normaliter vrijwel geen programmeerwerk aan te pas. De Business Modeler is de ontwikkelomgeving van het Mendix Framework. In de Business Modeler wordt, de structuur van de database, de user interface en de functionaliteiten bepaald.

In het bedrijfsleven komt het vaak voor de een proces een standaard reeks aan stappen is. Zo`n reeks aan standaard stappen is vast te leggen in één proces, wat een workflow wordt genoemd. Om zo`n reeks aan stappen digitaal vast te leggen kan Windows Workflow Foundation (WF) gebruikt worden. WF is een uitbreiding op het .Net framework is bedoeld voor het tekenen van een workflow, het toevoegen van acties/stappen en uitvoeren van de workflow. Een workflow kan uit allerlei soorten stappen bestaan, vaak echter is het een stap die wacht op input om verder te gaan. Bijvoorbeeld een workflow van een vergunningaanvraag bijvoorbeeld zou al kunnen beginnen met een stap die wacht tot de aanvraag ingediend wordt met bijvoorbeeld een naam als parameters.

Het aanroepen van zo`n stap kan eenvoudig gedaan worden vanuit.Net, de opdrachtgever wenst dit echter ook te kunnen vanuit een Mendix applicatie. Dat is waar de koppeling voor ontwikkeld moet worden. Het Mendix framework biedt standaard niet de mogelijkheid dit te doen, daarom zal dit doormidden van een Java servicelaag mogelijk gemaakt moeten worden. Behalve het aanroepen van een workflow-stap is het ook gewenst om in Mendix overzicht te hebben in welke stappen aangeroepen kunnen worden en wat de details (naam, parameters, etc.) van een bepaalde stap zijn. Hiervoor is een oplossing ontwikkeld waarmee de stappen in een database bijgehouden worden, die vanuit een Mendix applicatie via de servicelaag benaderbaar is. Bij de Java servicelaag zal ook een Mendix module[[1]](#footnote-1) worden opgeleverd die een aantal basis functionaliteiten en een user interface aanbiedt om met de servicelaag te communiceren.

Voor de ontwikkeling van dit project is gebruik gemaakt van de Scrum ontwikkelmethode. Scrum is een Agile ontwikkelmethode waarbij in korte periodes van twee tot vier weken gebouwd wordt aan nieuwe functionaliteiten. Eén zo`n korte periode wordt een sprint genoemd. Voorafgaan aan iedere sprint wordt een vergadering gehouden om te bepalen wat in de volgende sprint gedaan zal gaan worden. Deze indeling zal gemaakt worden op basis van de requirements die in de initiatie fase van het project zijn opgesteld. Na afloop van iedere sprint wordt teruggeblikt op de verrichtte werkzaamheden.

Voor de totstandkoming van het product zijn een aantal tussenproducten ontwikkeld. Zo is een project initiatie document opgesteld, een software architectuur ontwikkeld en zijn testplannen gemaakt. Samen met deze producten hebben de sprints geresulteerd in een koppeling van Mendix met Windows Workflow Foundation.

# Inleiding

Een belangrijk onderdeel van Centric Software Engineering is de ontwikkeling van maatwerkapplicaties. Ruim een jaar geleden is Centric partner geworden van Mendix, een bedrijf dat een modelgedreven ontwikkeltool met de naam Mendix Business Modeler aanbiedt. Sinds dien ontwikkelt Centric o.a. maatwerkapplicaties met behulp van Mendix. Voor dit project zal ook veel met de Mendix Business Modeler gewerkt worden. Zo zal met behulp van de Modeler een Mendix module ontwikkeld worden die de servicelaag en deels GUI (Graphical User Interface) bevat. Een Mendix module is een in de Mendix Business Modeler ontwikkelde module die als een los pakket wordt opgeleverd en geïmporteerd en gebruikt kan worden in de Mendix Business Modeler.

Het bedrijf Mendix is opgericht in 2005, met het doel een technologieplatform te ontwikkelen dat de bouw van softwareproducten volgens een modelgedreven aanpak mogelijk maakt. Door dit te benutten, kan de noodzaak van het handmatig schrijven van programmacode fors beperkt worden. Hierdoor is het mogelijk om in relatief weinig tijd volledig functionele applicaties te ontwikkelen. Het Mendix ontwikkel-platform biedt ondersteuning voor het complete traject van modelleren, uitvoeren tot gebruik van de eindapplicatie. Meer over het bedrijf Mendix is te lezen in hoofdstuk 2.2, waar nader wordt ingegaan op modelgedreven ontwikkeling met Mendix.

Een ander aspect dat aandacht vereist voor dit project, is Windows Workflow Foundation. Windows Workflow Foundation biedt een virtuele omgeving aan om workflows (bedrijfsprocessen) te tekenen, beheren en uit te voeren in .Net applicaties. In dit document zal Windows Workflow Foundation afgekort worden als WF aangezien WWF al voor andere doeleinden wordt gebruikt. Meer over WF in hoofdstuk 2.3.

Het verslag is opgedeeld in een aantal hoofdstukken. Hoofdstuk 2 beschrijft de achtergrond van het project. Centric, Mendix en Windows Workflow Foundation zullen hierin nader worden uitgelegd. Hoofdstuk 3 is een uitgebreide opdracht beschrijving. Het probleem, de aanvangssituatie, gewenste situatie en op te leveren producten worden hierin beschreven. Hoofdstuk 4 beschrijft de gehanteerde project aanpak. Algemene uitleg wordt gegeven over Agile en Scrum. Tevens wordt uitgelegd op wat voor manier Scrum is toegepast in dit project. Hoofdstuk 5 beschrijft de realisatie van het project. Op chronologische wijze wordt uitgelegd wat gedaan is, wat de belangrijkste beslissingen waren en tegen welke problemen zijn gelopen. Vervolgens wordt in hoofdstuk 6 het proces en opgeleverde eindproduct geëvalueerd. In hoofdstuk 7 wordt verdedigd op welke manier is voldaan aan het behalen van de vooraf opgestelde beroepstaken. Hoofdstuk 8 beschrijft de geraadpleegde literatuur en hoofdstuk 9 geeft een lijst met gebruikte figuren in het verslag. Tot slot zijn in hoofdstuk 10 een aantal bijlages te vinden en geeft hoofdstuk 11 een aantal verwijzingen naar externe bijlages.

# Glossary

Hieronder staan de meest gebruikte afkortingen en termen beschreven die gebruikt worden in dit rapport, welke tot mogelijke onduidelijkheid zouden kunnen leiden.

|  |  |
| --- | --- |
| **Term** | **Omschrijving** |
| Archtiectual Description | Rapport horend bij de ontworpen software architectuur |
| WF | Windows Worfklow Foundation. WF omdat WWF al voor andere doeleinden wordt gebruikt. |
| Werkvoorraad | Een lijst met workitems beschikbaar in een externe database. Deze lijst wordt opgebouwd vanuit de workflow en bestaat uit verwijzingen naar workflow-stappen. |
| Workitem | De werkvoorraad bestaat uit uit te voeren workflow-stappen. Deze worden bijgehouden in de Werkvoorraad database. Deze stappen worden in dit document workitem genoemd. Dit is in het engels omdat in de afbeeldingen ook altijd workitem gebruikt wordt, en hiermee verwarring voorkomen kan worden. |
| Mendix Business Modeler | Modelgedreven ontwikkelplatform |
| REST dataservice | De dataservice, ontwikkelt in .Net, die gebruikt wordt om de data uit de Werkvoorraad database te benaderen. |

# Achtergrond

*In dit hoofdstuk wordt beknopt achtergrondinformatie gegeven. Her eerste gedeelte is een beschrijving van het bedrijf Centric, waarbinnen de opdracht is uitgevoerd. Het tweede gedeelte analyseert de aanvangssituatie van de opdracht. Tot slot zal in worden gegaan op het Mendix ontwikkel-platform en Windows Workflow Foundation, twee systemen waar de opdracht om draait.*

## Centric

Centric Software Engineering, de divisie waarin dit onderzoek is uitgevoerd, is onderdeel van Centric B.V. waar bijna tienduizend medewerkers werkzaam zijn. Zij realiseerden in 2008 een omzet van 950 miljoen euro middels het aanbieden van gespecialiseerde totaaloplossingen op het gebied van consultancy, IT solutions, software engineering, e-business, systems integration, managed ICT services en training. Hiermee bedient Centric diverse branches (waaronder centrale en decentrale overheid, financiële dienstverlening, retail, woningcorporaties, groothandel en zorg.)

Centric is dan ook één van de grootste ICT-leveranciers binnen Europa en de grootste niet-beurs genoteerde ICT onderneming in Nederland. Centric haar activiteiten georganiseerd in drie hoofdtakken, respectievelijk: automatiseringsdiensten (Centric), financiële diensten (Finace) en advies- en ingenieursdiensten (Oranjewoud).

In het kader van dit project zal nader ingegaan worden op de tak automatiseringsdiensten. Deze zijn weer onderverdeeld in drie organisatorische eenheden, namelijk:

* *Managed ICT Services*

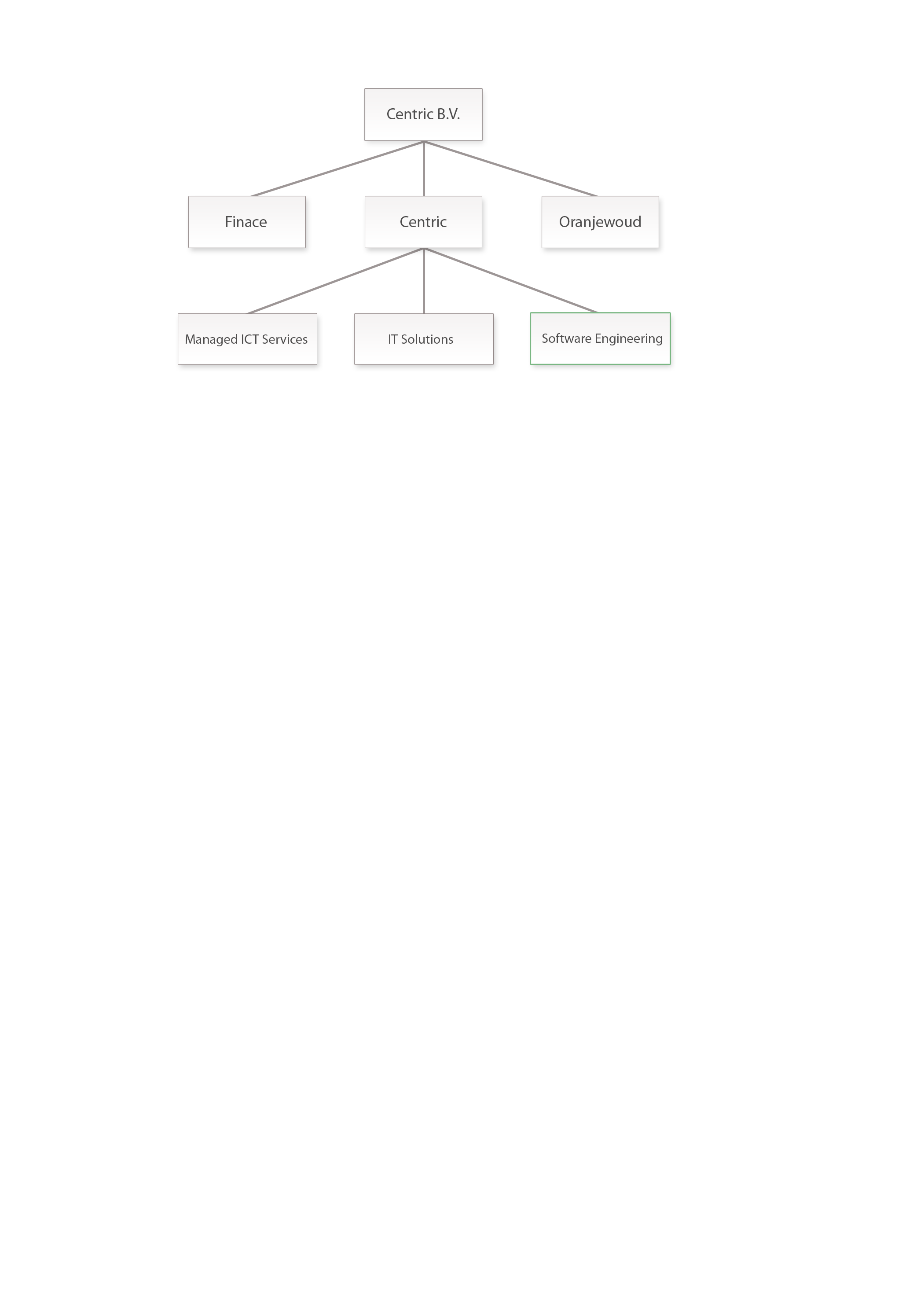
Totaaloplossingen op het gebied van kantoorautomatisering (detachering, outsourcing etc.).

* *IT Solutions*

Standaardoplossingen voor branchespecifieke en/of bedrijfkritische processen, integratie van bestaande en/of nieuwe oplossingen en de levering, implementatie en het onderhoud van deze producten en diensten.

* *Software Engineering*

Diensten voor renovatie, applicatie-integratie, (maatwerk-)systeemontwikkeling, testen, constructie van webapplicaties, technische automatisering, detachering en, outsourcing. Hierin zal dit project plaatsvinden.

Hieronder een illustratie ter verduidelijking van de organisatorische infrastructuur.   
  


Figuur 2.1: Infrastructuur Centric B.V.

## Mendix

Mendix is een jong bedrijf dat software ontwikkelt waarmee applicaties eenvoudig en snel gemodelleerd kunnen worden, de tool die hiermee geleverd wordt is de Mendix Business Modeler. Ter realisatie van de opdracht wordt een Mendix module ontwikkeld die deels gemaakt is met de Mendix Business Modeler zelf, en deels doormiddel van Java code als uitbreiding op de standaard mogelijkheden die de Mendix Business Modeler biedt.

Mendix is opgericht in 2005, met als doel een technologieplatform te ontwikkelen dat de bouw van softwareproducten volgens een modelgedreven aanpak mogelijk maakt. Door gebruik te maken van modelgedreven ontwikkeling kan de noodzaak van het handmatig schrijven van programmacode drastisch beperkt worden. Daardoor is het mogelijk om in een relatief korte periode een volledig functionele applicatie te ontwikkelen.

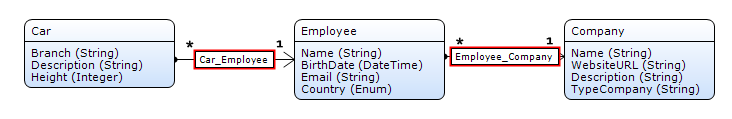
Mendix werkt met een eigen framework technologie, dat het “Mendix Framework” wordt genoemd. Om hier effectief mee te kunnen werken, heeft Mendix een ontwikkeltool (Mendix Business Modeler) ontwikkeld, die het mogelijk maakt om aan de hand van verschillende soorten modellen een applicatie te ontwikkelen.

Het Mendix framework bestaat uit vier hoofdcomponenten, te weten:

* de Mendix Business Modeler;
* de XML Application Server (XAS);
* de Connection Bus;
* en het Client System.

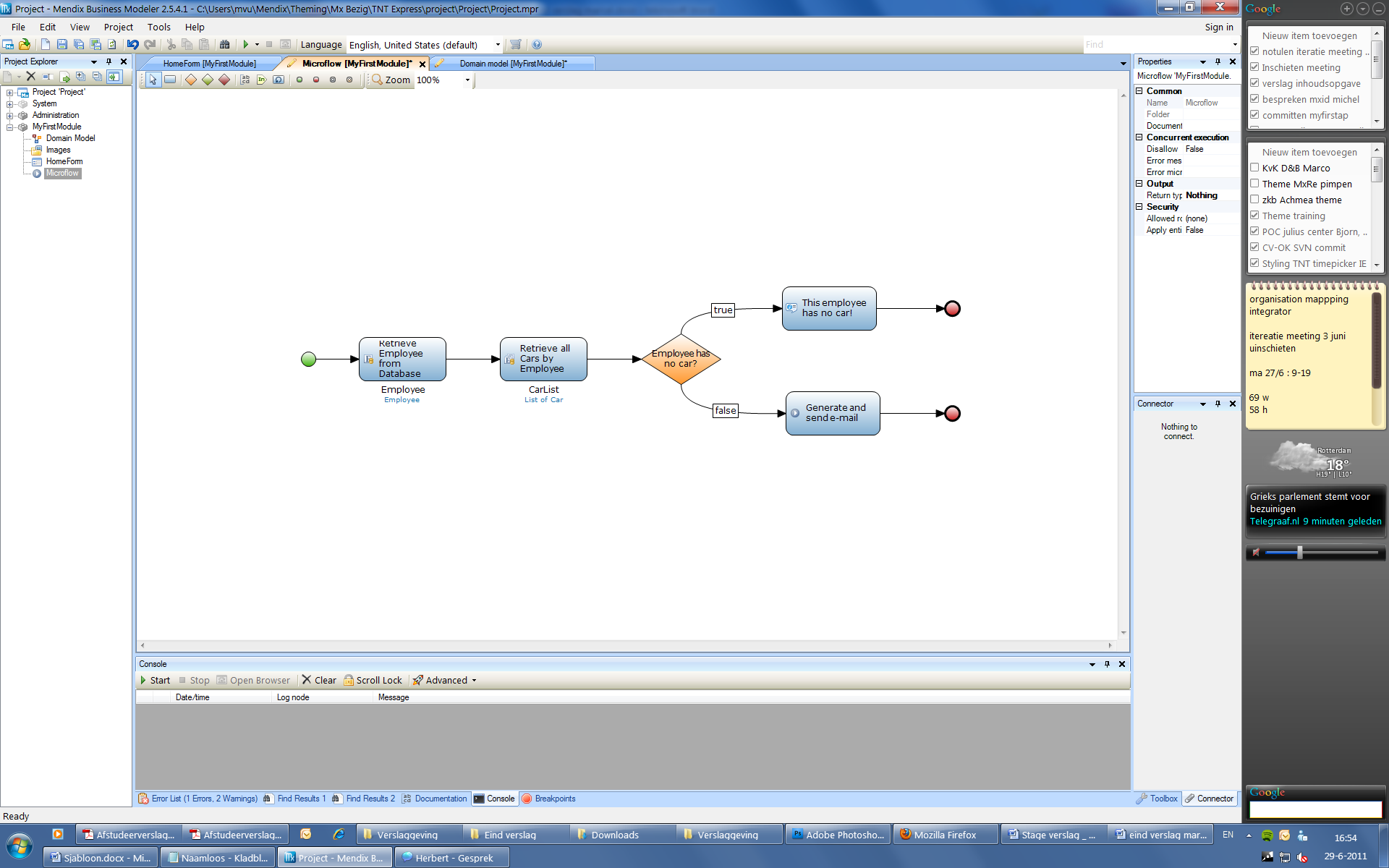
In het kader van dit project wordt de Mendix Business Modeler nader uitgelegd. De XAS, Connection Bus en het Client Systeem worden, gezien hun geringe relevantie in casu buiten beschouwing gelaten.  
 **De Mendix Business Modeler**De Business Modeler is de ontwikkelomgeving van het Mendix Framework, waarin aan de hand van modellen de gehele applicatie ontworpen wordt. Deze wordt hoofdzakelijk gebouwd aan de hand van de volgende modellen:

1. Het Domain Model: Het domeinmodel is een schematische weergave van de databasestructuur gebaseerd op UML (Unified Modeling Language)



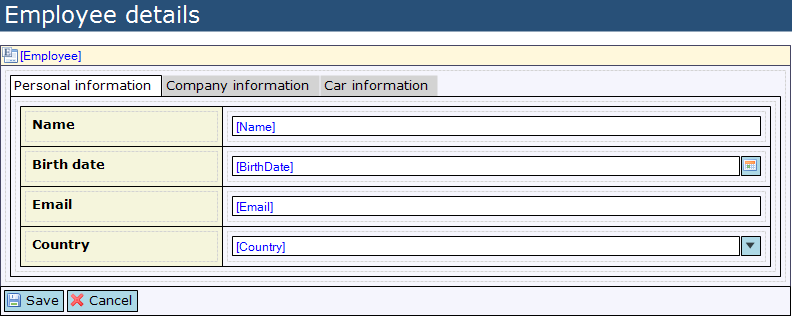
Figuur 2.2: Een Domein Model in de Mendix Business Modeler

Het Process Model: binnen de Mendix Business Modeler kunnen microflows (de term die Mendix zelf gebruikt voor dit soort modellen) aangemaakt worden waarin functionele stappenreeksen in een applicatie uitgetekend kunnen worden volgens vooraf gedefinieerde standaard acties. Een Mendix microflow is daarmee tot in zekere mate te vergelijken met een PSD (Program Structure Diagram)[[2]](#footnote-2), een PSD is echter bedoeld ter voorbereiding op het programmeren, waar het Process Model in feite het programmeren zelf is. Het Process Model (in de vorm van microflows) bepaalt het functionele gedeelte van een complete in de Mendix ontwikkelde applicatie.

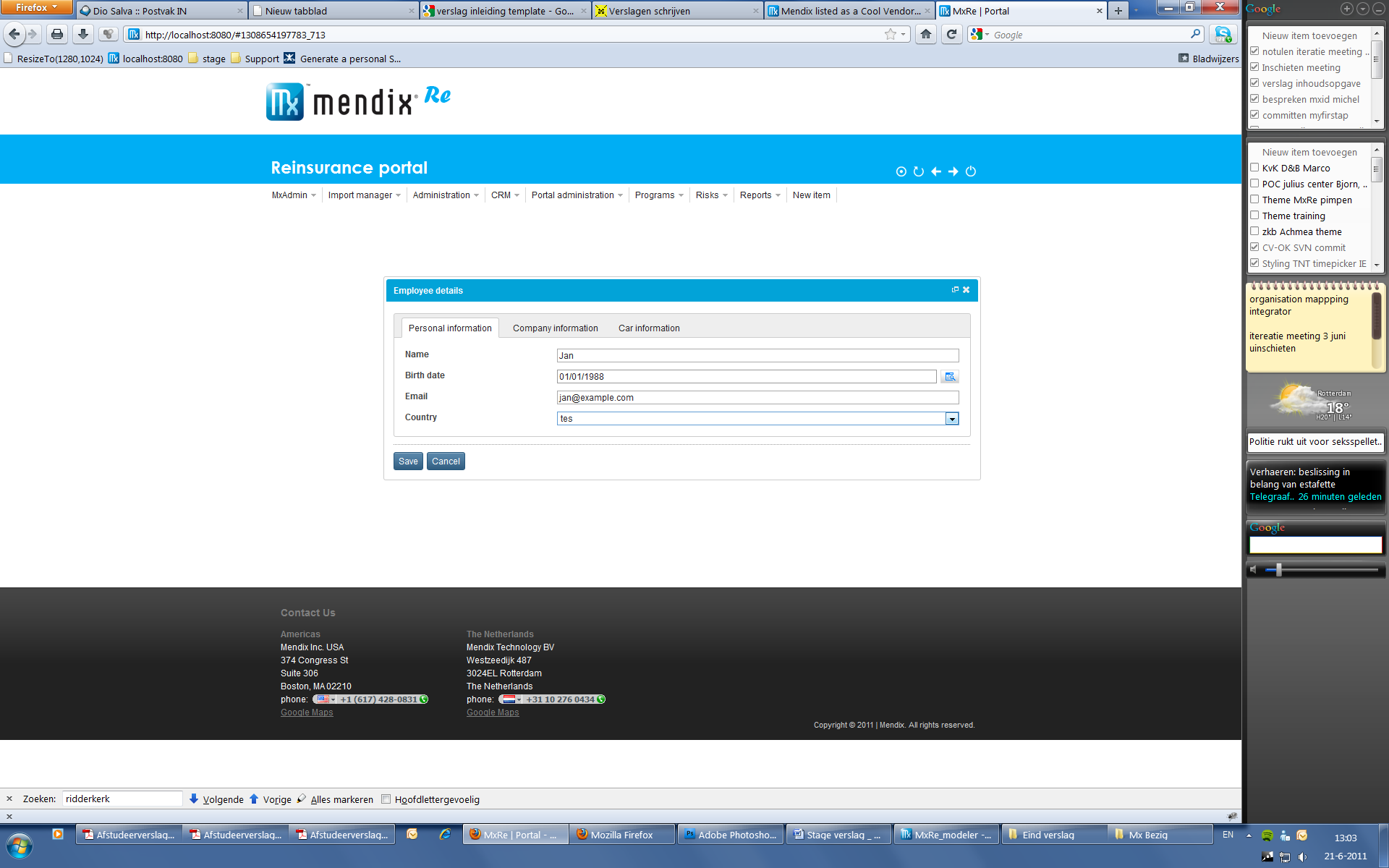


Figuur 2.3: Een microflow in de Mendix Business Modeler

1. Het Interface Model: dit is het gedeelte dat de eindgebruiker uiteindelijk te zien krijgt. De volgende afbeeldingen geven weer hoe het er in de Mendix Business Modeler uitziet en hoe het er in de webbrowser uitziet nadat extra stijl-informatie is toegevoegd.



Figuur 2.4: Een formulier in de Mendix Business Modeler



Figuur 2.5: Een formulier in de webbrowser

## Windows Workflow Foundation

Sinds het .NET 3.0 framework is een technologie op de markt gekomen die de naam Windows Workflow Foundation (WF) draagt.

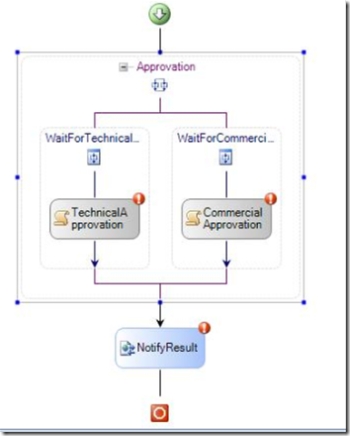
Tussen .NET 3.0 en 4.0 heeft een aantal significante veranderingen voor WF plaatsgevonden. Voor dit project is echter .NET 4.0 gebruikt, onderstaand uitleg baseert zich dan ook op WF in .NET 4.0.

Software binnen een bedrijf dient meestal ter ondersteuning van stappen in bedrijfsprocessen. Hierbij is te denken aan: een bedrijfsproces starten, documenten goedkeuren, uitzonderingen oplossen, etc. Een gehele reeks van dergelijke vastgelegde stappen in één proces wordt een workflow genoemd. Stappen in een workflow kunnen dan ook van alles zijn: een stukje programmacode dat wordt uitgevoerd, het versturen van een email, het wachten op invoer, enz.

Van belang is dat een workflow niet verward wordt met een Mendix microflow. Beide modellen beelden een proces uit, echter het type proces is wel degelijk verschillend. Een worfklow, getekend in Windows Workflow Foundation, is een bedrijfsproces zoals hierboven genoemd. Een microflow, getekend in Mendix, is een actie in een programma. Dit kan gezien worden als stukjes code wat van alles kan zijn, denk aan: if- of while-statements, een object ophalen of wijzigen, een bewerking op een lijst uitvoeren, enz.

WF biedt complete ondersteuning voor implementeren van langlopende processen als workflows in .NET applicaties*.* Het tekenen van een workflow, het toevoegen van acties/stappen en uitvoeren van de workflow kan in WF gedaan worden.

Onderstaande afbeelding laat een zeer eenvoudige workflow zien zoals het er in WF uit ziet.

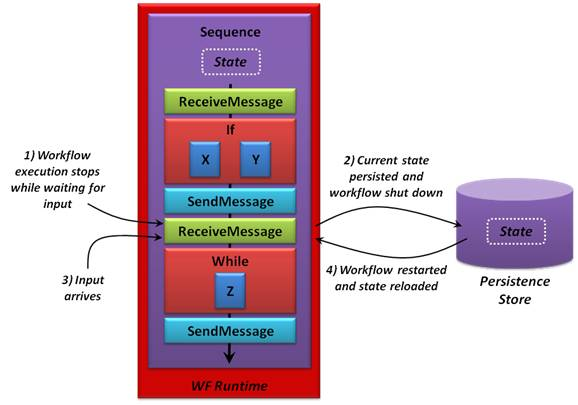


Figuur 2.6: Workflow in Windows Workflow Foundation[[3]](#footnote-3)

Bovenstaande afbeelding laat het proces van goedkeuring voor een document zien. Er moet een technische en commerciële goedkeuring gegeven worden aan een document. Doormiddel van de stappen in het plaatje kan afgevangen worden dat er pas een resultaatbericht wordt verstuurd op het moment dat beide goedkeuringen zijn verkregen. Nu is dit een zeer eenvoudig bedrijfsproces met slechts 1 stap, namelijk de goedkeuring van het document, echter het geeft goed de gedachte achter WF weer.

In bovenstaande afbeelding is een bedrijfsproces gemodelleerd. Na het modeleren kan de workflow ook daadwerkelijk uitgevoerd worden. Zodra de workflow (in bovenstaand geval is dat direct) aankomt bij de stap ‘Approvation’ zal de worfklow wachten tot technische en commerciële goedkeuring gegeven wordt. Zodra die goedkeuringen zijn ontvangen zal de workflow verder gaan naar de stap ‘NotifyResult’. Hoe deze goedkeuring wordt verkregen hangt af van hoe de ontwikkelaar van de workflow het geïmplementeerd heeft. Dit zou van alles kunnen zijn, bijvoorbeeld: via e-mail, handmatig op de knop drukken, via een ander programma, etc.

Het grote voordeel van werken met WF is dat een grote hoeveelheid activiteiten automatisch ‘onder de motorkap’ wordt afgehandeld door WF. Onderstaand figuur laat zien hoe de basis voor een workflow in WF werkt. Dit plaatje wordt tevens gebruikt in de uitleg die Microsoft zelf geeft aan WF.



Figuur 2.7: Windows Worfklow Foundation[[4]](#footnote-4)

Bovenstaande afbeelding is te concretiseren met de eerder uitgelegde figuur 2.6. Daarin was een proces te zien waarbij gewacht werd op technische en commerciele goedkeuring. Bovenstaande afbeelding (figuur 2.7) wacht ook op bepaalde momenten in de workflow, namelijk bij de ReceiveMessage stap. Figuur 2.6 liet alleen een reeks activiteiten in één specifieke workflow zien, bovenstaand figuur (figuur 2.7) laat zien wat door de Workflow engine (WF) gedaan wordt wanneer een workflow aankomt bij een ‘ReceiveMessage’ stap. Een ‘ReceiveMessage’ stap kan afhankelijk van de meegegeven parameters kan het ook voor van alles dienen. Wat echter altijd hetzelfde blijft, zijn de 4 stappen die door de workflow engine afgehandeld worden, zoals die in bovenstaande afbeelding te zien is.

Zoals bij alle processen stopt het proces op het moment dat invoer nodig is om verder te gaan. In bovenstaand figuur manifesteert dit zich als de workflow stopt op de ‘ReceiveMessage’ activiteit. De WF-runtime herkent dat. Vervolgens slaat zij de staat en de indicatie van waar de workflow verder behoort te gaan op in de ‘Persistence Store’ database (stap 1 en 2). Zodra nieuwe input voor de workflow binnenkomt herkent WF dat als zodanig. Vervolgens wordt de workflow opnieuw ingeladen, zodat hij verder gaat waar hij gestopt was (3 en 4). Worfklow-technisch gaat dit iets dieper, echter daar zal voor nu gezien het kader van dit project niet op ingegaan worden

# De opdracht

*Dit hoofdstuk beschrijft de aanleiding tot de opdracht, vervolgens de aanvangssituatie, de gewenste situatie, de opgeleverde deelproducten en het op te leveren eindresultaat (het doel van de opdracht). De opdracht luidt ´Ontwikkelen van een servicelaag tussen Mendix en Windows Workflow Foundation voor Centric B.V.´*

## Aanleiding tot opdracht

### Achtergrond

Sinds een aantal jaren is Centric partner van Mendix. Met regelmaat worden projecten in het Mendix framework ontwikkeld. Veel applicaties, bijvoorbeeld de meeste gemeentelijke applicaties dienen ter ondersteuning van langlopende bedrijfsprocessen. Om de bedrijfsprocessen te sturen en tot een definitief proces te assembleren worden workflows gemaakt. Hiervoor maakt Centric gebruik van Windows Workflow Foundation (WF). WF en het Mendix framework waren twee producten die tot op heden volledig los van elkaar gebruikt werden en geen interactie hadden.

### Probleemstelling

Zoals hoofdstuk 3.1.1 al beschrijft worden workflows ontworpen om bedrijfsprocessen vast te leggen en te sturen. In Mendix ontwikkelde applicaties worden gemaakt om stappen uit deze processen te ondersteunen of uit te voeren. Stappen in een workflow zouden kunnen vragen om een bepaalde actie, een workflow die gestopt is bij een bepaalde stap gaat verder op het moment dat er input komt. Het uitvoeren van zo`n stap en het geven van input aan de workflow zou iets kunnen zijn wat via een Mendix applicatie gedaan moet worden.

Het probleem is dat het op dit moment in een Mendix applicatie niet bekend is welke stappen in een workflow wachten op actie. Tevens kan niet op een standaard geautomatiseerde wijze data naar een workflow gestuurd worden om zo de gestopte workflow verder te kunnen laten gaan.

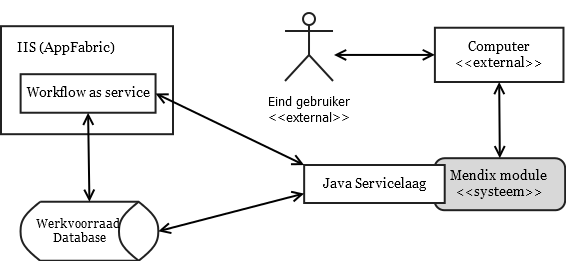
Op dit moment zal voor iedere keer dat vanuit een Mendix applicatie iets met WF gedaan moet worden, zoals het meegeven van data aan een stap om zo de workflow te kunnen laten continueren, handmatig een stuk code geschreven moeten worden die dit implementeert. Dit kost veel tijd en mankracht. Tevens is dit heel foutgevoelig; er is geen standaard, goed geteste oplossing om te gebruiken. Ook is er behoefte aan een aantal standaard functionaliteiten zoals wordt beschreven in hoofdstuk 3.3, welke nu nog ontbreken. De Mendix Business Modeler biedt namelijk geen standaard functionaliteit om met WF om te gaan. Ook heeft het Mendix framework geen standaard workflow engine om bedrijfsprocessen in vast te leggen, waardoor een koppeling met WF zeer gewenst is.

## Aanvangssituatie

In de aanvangssituatie was nog geen enkele standaard oplossing voor een koppeling tussen een in Mendix ontwikkelde applicatie en Windows Workflow Foundation. Enerzijds wordt WF gebruikt voor klanten om workflows te tekenen waarmee bedrijfsprocessen worden vastgelegd en door het uitvoeren van de workflow ook worden gestuurd. Anderzijds zijn er applicaties om bepaalde stappen uit die workflow te kunnen uitvoeren. Echter de koppeling tussen deze twee ontbreekt. Gezien het groeiende gebruik van het Mendix framework voor software ontwikkeling, verwacht men bij Centric dat ook voor Mendix applicaties de vraag naar een standaard workflow ondersteuning zal gaan stijgen.

## Gewenste situatie

In de gewenste situatie kan in een Mendix applicatie een lijst van workflow-stappen opgehaald worden die in een bepaalde workflow wachten op actie. Niet alle stappen in een workflow die wachten op actie zijn stappen die via een Mendix applicatie uitgevoerd moeten worden. Om de stappen bedoeld voor een Mendix applicatie te onderscheiden van de overige stappen wordt een aparte database, genaamd de Werkvoorraad database, bijgehouden die alle ‘Mendix-stappen’ bevat. Een Mendix-stap is een workflow-stap die actie vanuit een Mendix applicatie vereist alvorens de workflow kan continueren. Ter verduidelijking van waar de verschillende systemen zich ten opzichte van elkaar bevinden is hieronder een vereenvoudigd contextdiagram te zien van het systeem.



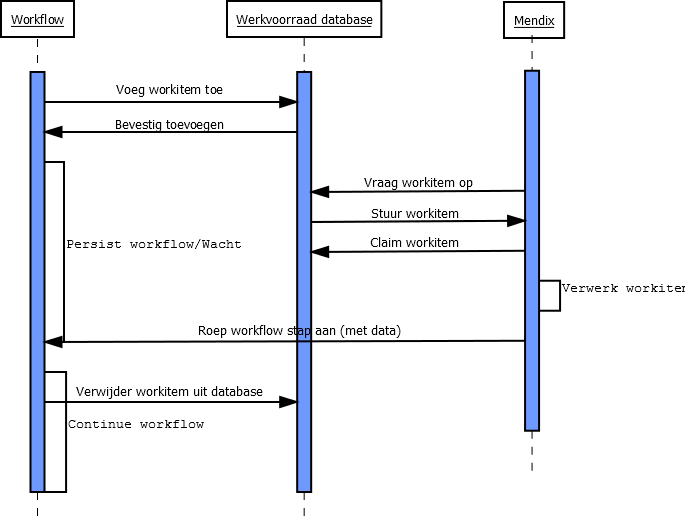
Figuur 3.1: Context diagram servicelaag

In bovenstaand figuur is te zien dat de Java Servicelaag de Mendix module overlapt. Dit is zo getekend omdat de servicelaag onlosmakelijk verbonden is met de Mendix Business Modeler module en specifiek bedoeld is om te communiceren met een Mendix applicatie.

Een Mendix-stap kan van alles zijn, en kan verschillende soorten actie vereisen. Zo kan er bijvoorbeeld een stap zijn die vraagt om mondelinge goedkeuring van een gebruiker waarbij iemand vervolgens in een Mendix applicatie op een knop kan klikken waardoor de workflow verder gaat. Maar het kan ook een meer geavanceerde stap zijn waarin een document goedgekeurd moet worden doormiddel van digitale ondertekening. De stap zal dan een verwijzing naar het document bevatten, en in het antwoord aan de workflow zal een verwijzing naar het goedgekeurde document met de handtekening moeten zitten.

Deze lijst met Mendix-stappen moet opgebouwd worden vanuit de workflow. Een workflow kan op een bepaald moment bij een stap aankomen die voor een Mendix applicatie bedoeld is. In dat geval zal de workflow die stap dus naar de Werkvoorraad database weg moeten schrijven zodat de Mendix applicatie hoogte krijgt van de wachtende workflow. Om een stap weg te schrijven naar de Werkvoorraad database moet een aangepaste workflow activiteit ontwikkeld worden die net als alle andere standaard activiteiten in een workflow geplaatst kan worden. Deze activiteit moet een unieke verwijzing naar de workflow instantie, een verwijzing naar de workflow-stap en eventuele zelf ingevulde overige gegevens naar de Werkvoorraad database kunnen schrijven.

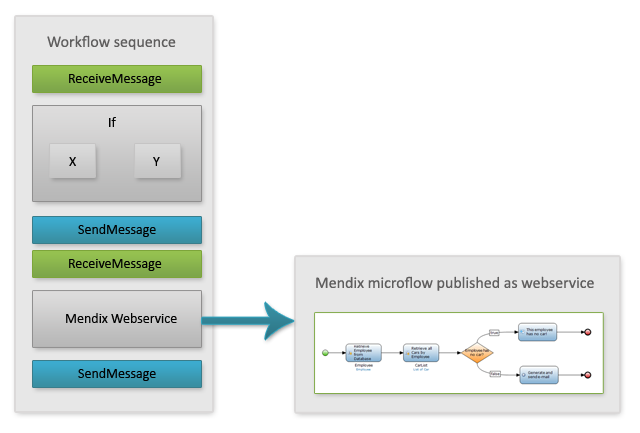
Na het voltooien van een bewerking moet de Mendix applicatie dat terugkoppelen naar een specifieke workflow instantie, zodat de workflow verder kan gaan en de stap uit de Werkvoorraad database gehaald wordt. Op de volgende pagina is een sequence diagram van het hierboven geschetste proces te zien zoals het er in een run-time omgeving uitziet.



Figuur 3.2: Sequence diagram

Bovenstaande diagrammen zijn slechts een benadering van de werkelijkheid, en geven op een vrij hoog niveau de werking van het systeem aan. In realiteit ligt het iets complexer, met meer systemen en meer tussenstappen, echter dat is in het kader van dit hoofdstuk eruit gelaten. Voor een gedetailleerder beeld van de werking van het systeem wordt verwezen naar de ontworpen software architectuur en naar hoofdstuk 5, Werkzaamheden.

Zoals eerder beschreven kan een workflow-stap van alles zijn, zo kan het ook een stuk code zijn die een aantal bewerkingen uitvoert. Zo een stuk code is een programma proces, wat vergelijkbaar is met een microflow in de Mendix Business Modeler. Het schrijven van zulke stukken code kost veel tijd, daarom is er ook de wens om in een workflow een Mendix microflow te kunnen gebruiken om zo niet zelf de code te hoeven schrijven. Dit kan gedaan worden door een Mendix microflow te maken, die als een webservice[[5]](#footnote-5) te publiceren, en deze Mendix-webservice aan te roepen vanuit een workflow. Dit is gewenst omdat in het Mendix platform in verhouding tot andere ontwikkelplatformen zeer snel ontwikkeld kan worden, en zo veel tijd en geld bespaard kan worden. Ter verduidelijking hieronder een afbeelding van de gewenste situatie.



Figuur 3.3: Mendix microflow als onderdeel van workflow

## Deelproducten

In het afstudeerplan zijn een aantal deelproducten vastgesteld die zullen worden opgeleverd. In de initiatie fase van dit project is gebleken dat behalve die producten nog een aantal producten opgeleverd zullen worden. In ieder geval de volgende producten zullen tussentijds worden opgeleverd:

* **Project Initiatie Document (PID)**

Het PID is in de eerste week van het project opgesteld als handvat voor de aanpak van het project. Alles dat hierin vastgelegd is, dient ter sturing en ondersteuning van het gehele project. Het PID geeft onder meer informatie over de achtergrond van het project, de gehanteerde aanpak en de samenstelling van het projectteam.

* **Beschrijving van de requirements**

De eisen en wensen van de opdrachtgever zijn vastgelegd in de vorm van stories, een lijst met uit te voeren taken die de gewenste functionaliteiten dekken. Deze zijn gezamenlijk met de Product Owner[[6]](#footnote-6) opgesteld in de eerste week van het project. De opgestelde stories fungeerden als leidraad voor de uit te voeren taken binnen het project.

* **Workflow activiteiten en Relationeel Implementatiemodel (RIM) voor de bijhorende database**

Om enkele stappen (conform hoofdstuk 3.3) vanuit een workflow op te slaan in een database zijn aangepaste workflow activiteiten ontwikkeld die als een normale activiteit opgenomen kunnen worden in een workflow. Concreet is één activiteit ontwikkeld voor het wegschrijven van een stap naar een database en één activiteit voor het weghalen van een stap uit de database.

Voor de bijhorende database is een RIM opgesteld die gebruikt kan worden om de database te implementeren. De database bestaat maar uit één tabel dus er is niet daadwerkelijk sprake van een relationele database, echter het is gewenst één standaard implementatiemodel te hebben voor de database.

* **Architectuurontwerp van de servicelaag als koppeling tussen Mendix en WF**

De servicelaag heeft veel interactie met andere systemen. Via verschillende protocollen gaat verschillende soorten data heen en weer tussen meerdere systemen. Om vast te stellen welke informatie precies waarheen gaat, is een software architectuur opgesteld. Deze bekijkt vanuit diverse optieken naar het geheel.

* **Mastertestplan en detailtestplannen**

Om het geheel aan testprocessen vast te leggen in een overkoepelend document is een zogeheten mastertestplan opgesteld. Met het mastertestplan wordt de algemene testaanpak, de testactiviteiten en de op te leveren test producten beschreven.

Er wordt op verschillende manieren getest, aan de hand van verschillende testsoorten. Voor een aantal van die testsoorten is een detailtestplan opgesteld waarin staat beschreven wat en hoe precies getest gaat worden.

* **Testrapporten**

De resultaten van alle uitgevoerde tests worden gedocumenteerd in testrapporten. Uiteindelijk zal voor elke testsoort een testrapport opgeleverd worden met de resultaten van de tests. Resultaten van eventuele her-tests zullen hier ook in beschreven worden.

* **Project documentatie**  
  Ter oplevering van het eindresultaat zijn twee soorten documentatie gemaakt, namelijk: gebruikers documentatie en technische documentatie. De gebruikers documentatie beschrijft hoe een gebruiker een workflow in combinatie met de Mendix module kan gebruiken en hoe de bij de module geleverde user interface werkt. De technische documentatie gaat in op de functionaliteiten van de servicelaag en is een uitgebreide beschrijving van het geheel. In de technische documentatie kan de ontwikkelaar alle benodigde informatie vinden om de opgeleverde module te kunnen gebruiken.

Meer over de realisatie van bovenstaande deelproducten is te lezen in hoofdstuk 5, waarin de uitgevoerde werkzaamheden nader worden uitgelegd en onderbouwd.

## Eindproduct

Het opgeleverde eindproduct is het resultaat van de geformuleerde opdracht en bovenstaand beschreven deelproducten.

Dat resulteert in een Mendix module, bestaande uit Mendix modellen en een servicelaag die de koppeling tussen een in Mendix ontwikkelde applicatie en WF mogelijk maakt. Om de basisfunctionaliteiten te kunnen demonstreren en gebruiken wordt ook een stuk GUI bij de module opgeleverd (de GUI is tevens ontwikkeld in de Mendix Business Modeler).

Er is de mogelijkheid om een werkvoorraad bij te houden van activiteiten die in een Mendix applicatie moeten worden uitgevoerd. Hiervoor is een aangepaste workflow activiteit ontwikkeld die net als standaard activiteiten in een workflow gezet kan worden. Deze activiteit biedt de mogelijkheid om een stap weg te schrijven naar de database die de werkvoorraad bijhoudt. Nadat een stap uitgevoerd is in een Mendix applicatie zal deze vanuit de workflow uit de database gehaald worden, ook via een zelf ontwikkelde activiteit. Ook is er een activiteit die de mogelijkheid biedt om een in Mendix ontwikkelde webservice aan te roepen.

De database voor de werkvoorraad, de workflow activiteiten en de dataservice om die database aan te spreken worden bij de Mendix module geleverd. Dit geheel biedt Mendix ontwikkelaars binnen Centric een handvat om op door te ontwikkelen voor een concrete realisatie van een koppeling tussen een Mendix applicatie en WF. Een Mendix ontwikkelaar binnen Centric is een ontwikkelaar die gespecialiseerd is in het ontwikkelen van applicaties met behulp van de Mendix Business Modeler

# Aanpak

*In dit hoofdstuk wordt de gehanteerde project aanpak beschreven. De gekozen ontwikkelmethode en de tools die daarbij gebruikt zijn zullen worden uitgelegd.*

## Agile ontwikkeling

Binnen Mendix ligt sterk de nadruk op Agile ontwikkeling van applicaties[[7]](#footnote-7). Vanwege het kader van dit project, is voor de realisatie van het project gekozen om ook een Agile-ontwikkelaanpak in te zetten.

Agile staat voor een flexibel ontwikkel proces. Letterlijk betekent Agile behendig of lenig. In 2001 zijn door 17 prominenten op het terrein van Agile ontwikkeling een aantal Agile principes bedacht die nader uitgewerkt zijn in het zgn. Agile manifest. Enkele principes uit het manifest van toepassing op dit project zijn:

* Klanttevredenheid door snelle levering van bruikbare software op continuele basis.
* Regelmatig aanbod van nieuwe werkende software (eerder per week dan per maand).
* Nauwe samenwerking op dagelijkse basis tussen ontwikkelaars en hun belanghebbenden.
* Direct persoonlijk contact
* Voortdurende aanpassing aan veranderende omstandigheden.

Deze principes lenen zich uitstekend voor de opgestelde opdracht. Het gaat namelijk om een servicelaag met veel verschillende functionaliteiten, zodat voor elke sprint de mogelijkheid is om nieuwe werkende software op te leveren. De vele vernieuwende elementen in de opdracht maakt dat het een grotendeels exploratieve ontwikkeling betreft. Door het werken in korte fases, het persoonlijke contact en het dynamisch vermogen van Agile past uitstekend in het projectkader.

## Scrum

Een zeer bekende Agile ontwikkelmethode is Scrum[[8]](#footnote-8). Scrum is een iteratieve en incrementele ontwikkelmethode die bedoeld voor ontwikkelprocessen met wijzigende eisen waarmee in korte periodes van 1-4 weken een werkend stuk software wordt opgeleverd. Door middel van intensief teamcontact en dagelijks contact met de Scrum master (zie 4.2.1, Scrumteam) kan zeer flexibel gewerkt worden. Scrum is een praktische uitwerking van de principes van Agile ontwikkeling.

Een ontwikkeltraject binnen Scrum wordt doorlopen door middel van sprints. Een sprint is een korte periode van 1 tot maximaal 4 weken. Een sprint bestaat uit een lijst met uit te realiseren stories, die vooraf opgesteld zijn. Een sprint wordt in overleg met het gehele Scrum team ingedeeld. De vooraf opgestelde stories zijn een lijst van requirements die onder de zogeheten Product Backlog vallen.

### Scrum team

Het Scrum team bestaat uit:

* Een Product Owner
* Het ontwikkelteam
* Een Scrum Master.

Een Scrum team is zelfstandig, flexibel en multifunctioneel. Het team bepaalt namelijk zelf hoe een probleem het beste opgelost kan worden. Hiervoor beschikt het team over alle benodigde competenties zonder afhankelijk te zijn van anderen buiten het team.

Onderstaande tabel geeft de samenstelling van het Scrum team in dit project weer.

|  |  |
| --- | --- |
| **Naam** | **Scrum rol** |
| Marcel Vuijk | Development Teammember |
| Dhr. W. Engberts | Product Owner |
| Mevr. K. Visser | Scrum Master |

Verder is ter begeleiding en ondersteuning van het project dhr. J. Luursema aangewezen als technisch expert.

**Development Team**Het Development Team bestaat uit meerdere Development Teammembers. In dit geval vanwege de individuele aard van de opdracht bestaat dat uit slechts 1 persoon. Een Development Teammember realiseert de implementatie van de stories voor een sprint. Hiermee is hij verantwoordelijk voor de incrementele ontwikkeling waarin aan het einde van iedere sprint een nieuw, functionerend softwaredeel opgeleverd wordt.

**Product Owner**  
De Product Owner is verantwoordelijk voor de inhoud aan stories in de Product Backlog. In dat kader zorgt de Product Owner voor optimalisatie van het werk dat de het Development Team uitvoert. Ook zorgt de Product Owner ervoor dat de backlog voor ieder teamlid duidelijk genoeg is om ermee aan de slag te kunnen gaan.

**Scrum Master**De Scrum Master is verantwoordelijk voor de controle op het correcte gebruik van Scrum. De Scrum Master staat het team bij op meerdere gebieden. Daarnaast is hij verantwoordelijk voor de planning van de Scrum bijeenkomsten.

### Sprint doorloop

In Scrum volgt elke sprint het volgende standaard doorlooppatroon:

1. Sprint Planning: voorafgaand aan iedere sprint, zal vastgesteld worden welke stories voor de betreffende sprint geïmplementeerd zullen worden. Tevens worden hierin de benodigdheden, risico’s en kansen geanalyseerd.
2. Doorlopen van de sprint: gedurende een sprint wordt gewerkt naar het opleveren van een nieuwe werkende versie van het op te leveren product. Hierin worden in ieder geval de stories verwerkt en getest die in de Sprint Planning waren opgesteld.
3. Daily Scrum: dagelijks zal met het Scrum team besproken worden wat gisteren is gedaan, wat vandaag gedaan gaat worden en of er problemen zijn geweest. De Daily Scrum duurt ongeveer 15 minuten. De Scrum Master ziet erop toe dat dit dagelijks gebeurt. Dit gebeurt normaal gesproken enkel met het Development Team, echter omdat dit project slechts één Development Teammember heeft wordt de Daily Scrum hierin gehouden met de Scrum Master. De Daily Scrum zal wegens het werken vanuit verschillende vestigingen van de Scrum Master meestal telefonisch gehouden worden.
4. Sprint Review: aan het einde van iedere sprint zal een Sprint Review meeting gehouden worden. Hierin wordt het resultaat dat de sprint heeft opgeleverd gedemonstreerd en worden indien nodig aanpassingen gedaan op de backlog.
5. Sprint Retrospective: in de Sprint Retrospective zal naar het team zelf gekeken worden. De Sprint Retrospective wordt mogelijk gecombineerd met de sprint review meeting. Hierin wordt gereflecteerd op de laatste Sprint en verbeterpunten voor een volgende sprint worden opgesteld.

Onderstaande afbeelding verduidelijkt de doorloop van een sprint.



Figuur 4.1: Sprint doorloop[[9]](#footnote-9)

### Sprintr

Om de ontwikkeling zo effectief en gestructureerd mogelijk te laten verlopen is gebruik gemaakt van een tool om de backlog en de sprints te beheren. Mendix heeft hier een eigen tool voor ontwikkeld, genaamd sprintr[[10]](#footnote-10). Sprintr maakt iteratieve en incrementele ontwikkeling mogelijk. Zo helpt sprintr ervoor zorgen dat zaken op het juiste tijdstip worden uitgevoerd met de hoogst mogelijk toegevoegde waarde. Daarom, en gezien het kader van dit project is gekozen om gebruik te maken van sprintr. In het volgende gedeelte wordt beschreven wat sprintr precies is en wat de mogelijkheden van sprintr zijn.

**Sprints, stories en backlog**Het hele project in sprintr wordt zoals het scrum betaamd onderverdeeld in sprints en een backlog. Dit wordt gedaan op basis van de opgestelde requirements en in overleg met het Scrum Team zal dit opgedeeld worden in sprints. Met behulp van sprintr kan uitgebreid beheer toegepast worden op de backlog, de sprints en de stories. Sprintr voert controle uit op dat sprints altijd na elkaar volgen en altijd een periode van 1 tot 4 weken zijn. Een sprint wordt opgebouwd uit stories (uit de backlog) en bevindt zich altijd in de staat todo, running of completed.

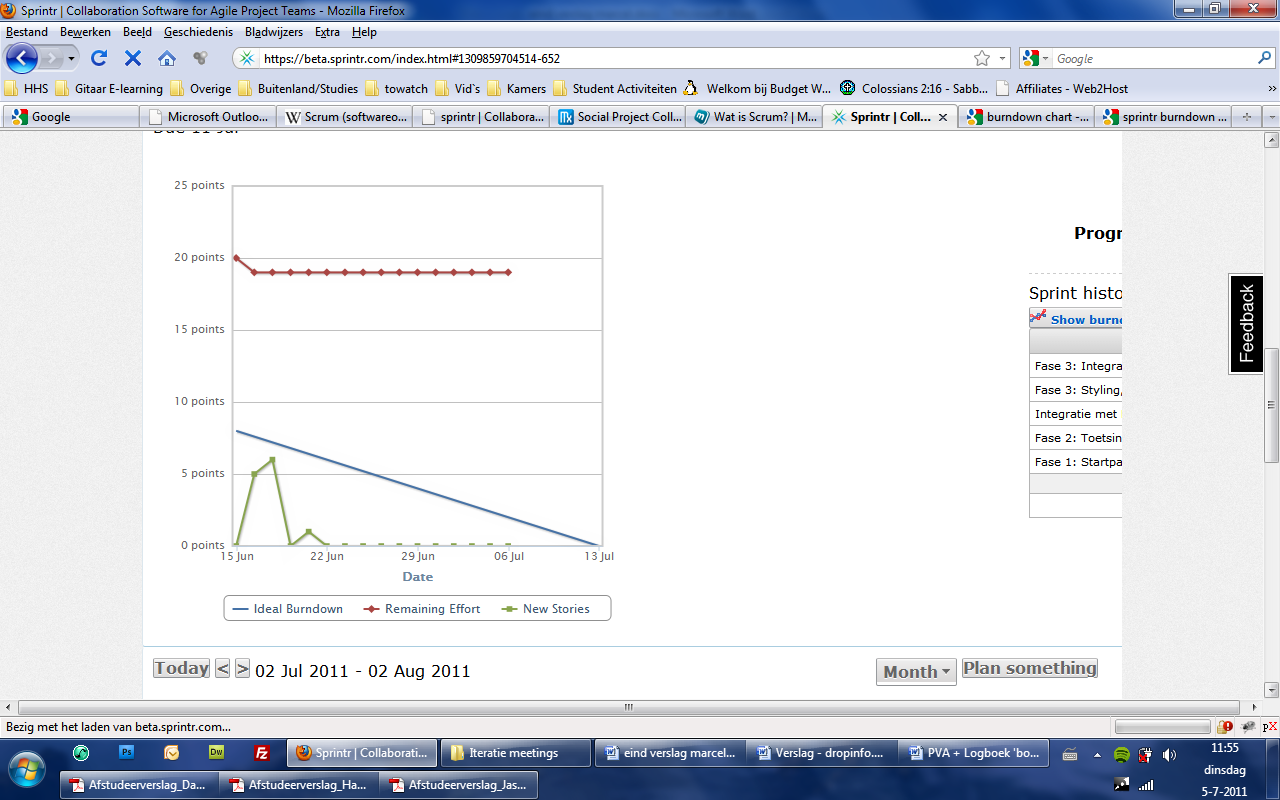
Door uitvoering van de stories wordt dus een sprint voltooid. Van elke story is te zien of, en zo ja, welk lid van het projectteam eraan werkt. De mate van voltooiing van een sprint wordt bepaald aan de hand van punten. Aan elke story wordt een puntental toegekend. De punten geven aan hoeveel werk een story in verhouding tot andere stories is.

**Project Wall**De sprintr Project Wall geeft de mogelijkheid om berichten achter te laten voor de members van het team en is daarmee vergelijkbaar met een digitaal forum. Ook is op de Project Wall te zien welke stories en/of sprints zijn aangemaakt.

**Planning**De planning is essentieel voor een ordelijk verloop van een project. Hoewel met Scrum geen sprake is van lange termijnplanningen, is een tactische planning van een sprint cruciaal. Sprintr biedt vier verschillende overzichten aan om de status van een project te monitoren. Hieronder een korte beschrijving van alle overzichten.

Het Sprint Status overzicht   
In het Sprint Status overzicht is in één oogopslag te zien hoeveel en welke stories voor een bepaalde sprint in de fase todo, running of completed staan.

Burndown Chart  
Het tweede overzicht bevat een burndown chart van de huidige sprint. De burndown chart toont het resterend werk per sprint. Het is een overzichtelijke weergave tussen de werklast en de tijdsdruk. De productiviteit kan op die manier altijd gemonitord worden en indien nodig kan worden bijgestuurd. Onderstaand figuur is een voorbeeld van een burndown chart in sprintr zoals hij er uit zou kunnen zien in een project.



Figuur 4.2: Een voorbeeld Burndown Chart in sprintr

Timeline  
In het timeline overzicht zijn de sprints te zien als in een agendaweergave. Er is te zien wanneer een sprint start en eindigt en in hoeverre een sprint is voltooid.

Release plan  
Het release plan laat in principe hetzelfde zien als de timeline. Wel is de weergave verschillend; het release plan laat alle sprints onder elkaar zien met erachter de release datum. Hierin is dus heel eenvoudig te zien wanneer een bepaalde sprint voltooid hoort te zijn en wat de planning is voor de sprints.

# Realisatie

*In dit hoofdstuk worden de werkzaamheden gepresenteerd die tijdens het project zijn uitgevoerd ter realisatie van het product. Allereerst wordt beschreven wat ondernomen is in de initiatiefase, en vervolgens zullen per sprint de werkzaamheden beschreven worden. Behalve de uitgevoerde werkzaamheden beschrijft dit hoofdstuk tevens de gemaakte keuzes en tegengekomen problemen.*

## Initiatie fase

Om een goed beeld bij het project te krijgen en om kennis te krijgen van de verschillende aspecten van het project is in overleg met de opdrachtgever besloten om te beginnen met een initiatie fase waarin de focus ligt op onderzoek en ontwerp.

De initiatiefase duurde 3 weken. In de eerste week is het Scrum projectteam samengesteld, is het project initiatie document opgesteld en zijn de requirements bepaald. De samenstelling van het team staat beschreven in hoofdstuk 4.2.1, Scrum team. Aan het eind van de eerste week is een sprint planning meeting gehouden om de eerste sprint in te delen, maar gezien de verschillende onbekende aspecten in de opdracht is besloten om de eerste 2 weken nog bij de initiatie fase te betrekken en te gebruiken voor onderzoek naar alles waar de opdracht betrekking op heeft.

Het project initiatie document is opgesteld om de opdracht (achtergrond, doelstelling en op te leveren resultaten) en de aanpak van het project vast te leggen. Voor de inhoud van dit document verwijs ik naar de externe bijlage ‘Project Initiation Document’ die bij het dossier is bijgevoegd.

### Requirements bepalen

Voor echt aan het project begonnen kon worden, moest een lijst met requirements bepaald worden. In eerste instantie heb ik deze lijst zelf opgesteld, om zo een beter beeld te krijgen van de opdracht en te kijken of het voor mezelf duidelijk genoeg was. Dit heb ik gedaan op basis van de kennis die ik al had van de opdracht en op basis van onderzoek dat ik in de eerste dagen had uitgevoerd. Aan de hand van de opgestelde requirements zullen de functionaliteiten ontwikkeld worden en zal naderhand geëvalueerd worden of het beoogde doel is behaald . De opgestelde requirements zijn nagekeken en waar nodig aangepast door de Product Owner, waarna ze als stories in sprintr aan de project-backlog zijn toegevoegd. De priorisering van de requirements bepaald tevens de indeling van de sprints. De precieze inhoud van een sprint wordt bepaald tijdens de zogenoemde sprint planning bijeenkomst.

Hieronder volgen de requirements zoals ze zijn opgesteld voor het project. De requirements zijn op de gebruikelijke Scrum methode opgesteld in de vorm van stories. In het afstudeerplan was aangegeven dat dit volgens de MoSCoW methode gedaan zou worden, echter in het kader van de gehanteerde project aanpak is toch gekozen om dit op de gebruikelijke Scrum manier te doen. Normaal gesproken worden de stories in de vorm van user-stories beschreven, waarin men uitgaat van een user als actor, echter omdat het project een servicelaag betreft waar in veel gevallen de actie niet van een user uitgaat, wordt hier niet altijd vanuit de gebruiker beschreven maar ook vanuit de workflow en Mendix applicatie.

Onderstaande actoren zijn gebruikt voor het beschrijven van de stories.

|  |  |
| --- | --- |
| **Actor** | **Omschrijving** |
| Eindgebruiker | De eindgebruiker van het opgeleverde product. Dit is de gebruiker op het user interface niveau. |
| Workflow | Een geïnstantieerde workflow in Windows Workflow Foundation. |
| Mendix ontwikkelaar | Een in Mendix gespecialiseerde ontwikkelaar die gebruik maakt van de opgeleverde Mendix module. |
| Workflow ontwerper | De persoon die verantwoordelijk is voor het ontwerpen en implementeren van een workflow in WF. |
| Mendix applicatie | De applicatie zelf, de applicatie is de runtime omgeving van de Mendix module met de service laag. |

Hieronder staan per actor de stories (requirements) beschreven. De priorisering is aangegeven met 1, 2, of 3 plusjes.

#### Als een Mendix applicatie

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| +++ | Een workflow instantiëren. | Er na een bepaalde handeling een nieuwe workflow instantie begint te lopen. |
| ++ | Excepties afvangen. | Als iets misgaat in de workflow of servicelaag de exceptie afgevangen wordt en verwerk tot een voor de gebruiker bruikbare exceptie. |

#### Als een eindgebruiker

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| +++ | Een lijst met workitems kunnen ophalen uit de Werkvoorraad database. | Ik weet welke stappen wachten op actie. |
| ++ | Een lijst bij het ophalen kunnen filteren. | Ik voor het ophalen van de lijst bepaalde criteria kan opgeven om de lijst zo in te korten naar wat mij interesseert en het ophalen te versnellen. |
| + | Een lijst na het ophalen kunnen filteren. | Ik na het ophalen van de lijst kan filteren op criteria waar ik geïnteresseerd in ben. |
| +++ | Op basis van een referentie uit de opgevraagde lijst een bepaalde stap kunnen ophalen. | Ik meer informatie over de betreffende stap kan krijgen en weet wat het precies inhoudt. |
| +++ | Op basis van een referentie uit de opgevraagde lijst een bepaalde stap kunnen claimen. | Ik de enige ben die aan die stap kan werken, en hij openstaat voor mij totdat ik of een administrator de claim loslaat. |
| +++ | Een geclaimde stap kunnen afmelden als zijnde voltooid. | De stap na het voltooien verdwijnt uit de werkvoorraad en de workflow verder kan lopen. |
| ++ | Een geclaimde stap kunnen annuleren. | De stap in de oude staat terugkomt in de werkvoorraad. |
| +++ | Wil gebruikersdocumentatie hebben. | Ik kan lezen hoe ik de module kan gebruiken als gebruiker zijnde. |
| + | Wil vanuit een geclaimde stap naar het bijhorende Mendix scherm kunnen gaan om de stap uit te voeren. | Ik niet zelf de juiste formulieren hoeft op te zoeken om de stap te completeren. |

#### Als een workflow

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| +++ | Informatie naar de Werkvoorraad database kunnen wegschrijven. | Er een bepaalde stap weggezet kan worden als een Mendix stap. |
| ++ | Wil een Mendix webservice actie kunnen aanroepen. | Bepaalde handelingen in de workflow via een Mendix webservice actie uitgevoerd kunnen worden. |

#### Als een Mendix ontwikkelaar

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| +++ | Documentatie hebben. | Ik kan lezen hoe ik de Mendix module horend bij de servicelaag kan gebruiken in combinatie met WF. |

#### Als workflow ontwikkelaar

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| +++ | Een ´Mendix activity´ kunnen toevoegen aan mijn workflow. | Er een bepaalde stap uit de workflow naar de Werkvoorraad database kan worden wegeschreven. |
| ++ | Documentatie hebben. | Ik kan lezen hoe ik een Mendix webservice kan aanroepen vanuit de workflow. |

Bovenstaande requirements zijn dus in de beginfase van het project opgesteld. Deze zijn voor het grootste deel van het project onveranderd gebleven. Echter in de voorlaatste sprint heeft een aantal veranderingen plaatsgevonden met betrekking tot de requirements en de uiteindelijk op te leveren stories. Tijdens het testen van een aantal processen is tegen een concurrency probleem aangelopen waardoor de kern van de servicelaag herschreven moest worden. Dit heeft veel tijd in beslag genomen waardoor helaas niet alle stories binnen de gewenste tijd afgerond konden worden. Het probleem en de oplossing hiervan staat nader beschreven in hoofdstuk 5.5, waarin de 4de sprint van het project beschreven staat.

Van de volgende twee stories is besloten ze buiten dit project te laten vallen.

#### Als een Mendix applicatie

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| ++ | Excepties afvangen. | Als iets misgaat in de workflow of servicelaag de exceptie afgevangen wordt en verwerk tot een voor de gebruiker bruikbare exceptie. |

#### Als een eindgebruiker

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| + | Wil vanuit een geclaimde stap naar het bijhorende Mendix scherm kunnen gaan om de stap uit te voeren. | Ik niet zelf de juiste formulieren hoeft op te zoeken om de stap te completeren. |

Alle stories met betrekking tot documentatie zullen wel opgeleverd worden, zodat alsnog een solide eindproduct opgeleverd kan worden die gebruikt kan worden als basis voor de koppeling van de Mendix Business Modeler met Windows Workflow Foundation.

Tevens is het, naarmate er een duidelijker eindresultaat kwam, steeds duidelijker geworden welke functionaliteiten de Mendix module (welke de servicelaag omvat) moet bieden. In overleg met de Product Owner zijn een aantal wijzigingen in doelstellingen en requirements gemaakt, die buiten de scope van dit project vallen, maar in een later stadium alsnog gewenst zijn. Deze requirements zijn:

#### Als een eindgebruiker

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| ++ | Bestaande (al in gebruik zijnde) workflows eenvoudig aan kunnen passen met een externe modelleer tool. | Ik niet het hele Visual Studio pakket hoeft te installeren om workflows te kunnen aanpassen. |

#### Als een administrator eindgebruiker

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| ++ | Gereed gemelde (afgeronde) workitems kunnen raadplegen in een separaat archief. | Ik daar analyses op uit kan voeren en de geschiedenis kan doorkijken. |

#### Als een Mendix applicatie

|  |  |  |
| --- | --- | --- |
| **Prio** | **Wil ik….** | **Zodat….** |
| ++ | Meerder workflows vanuit één applicatie kunnen benaderen. | Ik een activiteit te allen tijde kan aanroepen, in welke (Workflow Foundation) workflow dat dan ook moet zijn. |

De bovenstaande stories zijn pas in de laatste weken van het project naar voren gekomen en konden helaas niet meer in de planning worden opgenomen. Deze zullen echter open blijven staan en bij overdracht zal hier op geduid worden.

### Onderzoek

De opdracht omvat een groot aantal, voor mij, nieuwe elementen. Veel technieken en tools die gebruikt moesten worden had ik nog nooit of nauwelijks van gehoord. Om die reden zijn er twee weken gebruikt om in te lezen op en te experimenteren met alle tools en technieken die gebruikt moesten worden. De tools en technieken waarin ik me heb verdiept gedurende deze weken zijn:

* Windows Workflow Foundation (WF)
* Internet Information Services (IIS) en AppFabric
* Mendix Business Modeler
* Webservices
* JSON (Javascript Object Notation)
* XML (Extensible Markup Language)

**Windows Workflow Foundation (WF)**WF was voor mij volledig nieuw. WF is een uitbreiding op het .Net framework en kan met Visual Studio gebruikt worden. Alle begrippen omtrent WF zoals workflows, workflow activity, .Net, etc. waren ook vrij nieuw voor mij. Door veel te experimenteren met WF, tutorials te volgen en vooral veel te lezen ben ik erg veel te weten gekomen over WF.

Om een beter gevoel te krijgen met WF heb ik zelf ook een workflow ontworpen die tevens gebruikt zal worden als testcase voor het project. De ontworpen workflow is het proces van het aanvragen van een visvergunning. Het is een zeer eenvoudig proces met slechts enkele stappen, maar het heeft enorm geholpen om de techniek achter en de werking van WF te begrijpen.

**Internet Information Services (IIS) en AppFabric**WF is niet alleen bedoeld voor het ontwerpen van een workflow, het is tevens bedoeld om workflows te beheren en uit te voeren. Om zo`n workflow echter uit te kunnen voeren is IIS nodig. IIS biedt een aantal server diensten aan waarin de workflow gehost kan worden. Zodra een workflow op een server gehost wordt (dit kan vanuit Visual Studio gedaan worden), kan hij door middel van bijvoorbeeld een C# programma aangeroepen worden. Wanneer een workflow aangeroepen wordt, zal een nieuwe instantie van die workflow gaan lopen totdat het op een ‘ReceiveMessage’ stap aankomt. Deze stap kan van alles zijn, maar het gaat erom dat op dat punt de workflow zichzelf ‘uitlaad’ en in een standaard ‘persistence database’ wordt opgeslagen. Zodra de workflow weer wordt aangeroepen op de volgende stap (met de benodigde data), zal de workflow opnieuw ingeladen worden en verder lopen. Het wachten, uit en inladen en opslaan in de database is wat standaard gedaan wordt door WF.

AppFabric is een extra laag op IIS, speciaal voor het beheren van workflows en webservices. Een workflow gehost in IIS, is eigenlijk ook gewoon een webservice. Alle ‘ReceiveMessage’ stappen in een workflow worden als webservice methodes aangeboden en zijn net als normale webservices aan te spreken. Met behulp van AppFabric kan inzicht verkregen worden in welke workflows actief zijn, welke niet actief zijn en dus in de persistence database staan en welke workflows gestopt zijn. Tevens is te zien op welke stap een workflow is gestopt, en eventuele fouten kunnen via AppFabric opgespoord worden.

Het installeren van WF, IIS en AppFabric heeft voor de nodige problemen gezorgd. De computer bleek niet helemaal goed geïnstalleerd te zijn, waardoor bepaalde Windows onderdelen ontbraken die vereist waren voor het hosten van een workflow in IIS. Teven bleek SQL Server fout geconfigureerd te zijn, waardoor de persistence database niet benaderd kon worden. Aan het uitzoeken van de problemen, her-installeren, etc. is helaas veel tijd verloren gegaan.

**Mendix Business Modeler**De Mendix Business Modeler is de modelgedreven ontwikkeltool die Mendix aanbiedt om software mee te ontwikkelen. Door het uitvoeren van vooronderzoek was al een hoop duidelijk geworden over Mendix en de Mendix Business Modeler, zoals te lezen in hoofdstuk 2.2, Mendix. Echter om daadwerkelijk wat ervaring te krijgen met het ontwikkelen in Mendix, is in de eerste week tijd besteed aan het ontwikkelen van een aantal eenvoudige applicaties in de Mendix Business Modeler. Ook heb ik geëxperimenteerd met het aanmaken en importeren van webservices vanuit Mendix.

**Webservices**Aangezien een workflow volledig als een webservice aangesproken moet worden, en ik zelf nog nauwelijks ervaring had met webservices, leek het me zeer nuttig om eerst wat algemene kennis over webservices op te doen. Hiervoor heb ik een aantal tutorials gevolgd van W3 Schools over webservices, SOAP berichten[[11]](#footnote-11) en XML documenten. Inzicht in hoe een webservice werkt, aangesproken wordt, etc is van cruciaal belang gebleken gedurende de verdere ontwikkeling van het project.

### Software architectuur opstellen

De software architectuur is opgesteld ter ondersteuning van het ontwikkelproces en om stakeholders inzicht te geven in het verwachtte eindresultaat. Doormiddel van het benaderen van het project vanuit verschillende Viewpoints[[12]](#footnote-12). De gehele software architectuur is vastgelegd in een Architectual Description. Voor de inhoud van dit document verwijs ik naar externe bijlage Architectual Description.

Het project is vanuit aan aantal viewpoints benaderd die voor dit project het meest van nut waren. De viewpoints die zijn beschreven zijn:

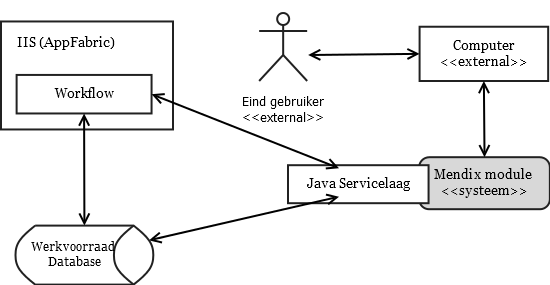
* Context View
* Functional View
* Information View

Het Architectual Description is gebaseerd op het boek ´Software Systems Architecture´, geschreven door Nick Rozanski en Eoin Woods. Dit boek beschrijft nog een aantal andere viewpoints, echter die waren voor dit project niet van toepassing of van extra waarde.

Hieronder zal per uitgewerkte view uitgelegd worden wat beschreven staat en wat de belangrijkste gemaakte architectuur keuzes hierin waren.

#### Context view

De context view van het systeem beschrijft waar het systeem staat in het geheel met zijn afhankelijkheden, relaties en interacties tussen het systeem en de omgeving. Om dit te visualiseren is een Context Diagram opgesteld. De Context diagram is in eerste instantie opgesteld zoals hieronder te zien.



Figuur 5.1: Context diagram Mendix module

De belangrijkste keuze die hierin is gemaakt is om de Werkvoorraad database als een aparte database bij te houden, en niet op te nemen in de Mendix module. Hiervoor is gekozen omdat in dit geval de Werkvoorraad niet noodzakelijk enkel vanuit Mendix benaderd hoeft te worden, maar er ook vanuit andere ontwikkelomgevingen eventueel aangesproken kan worden op de Werkvoorraad.

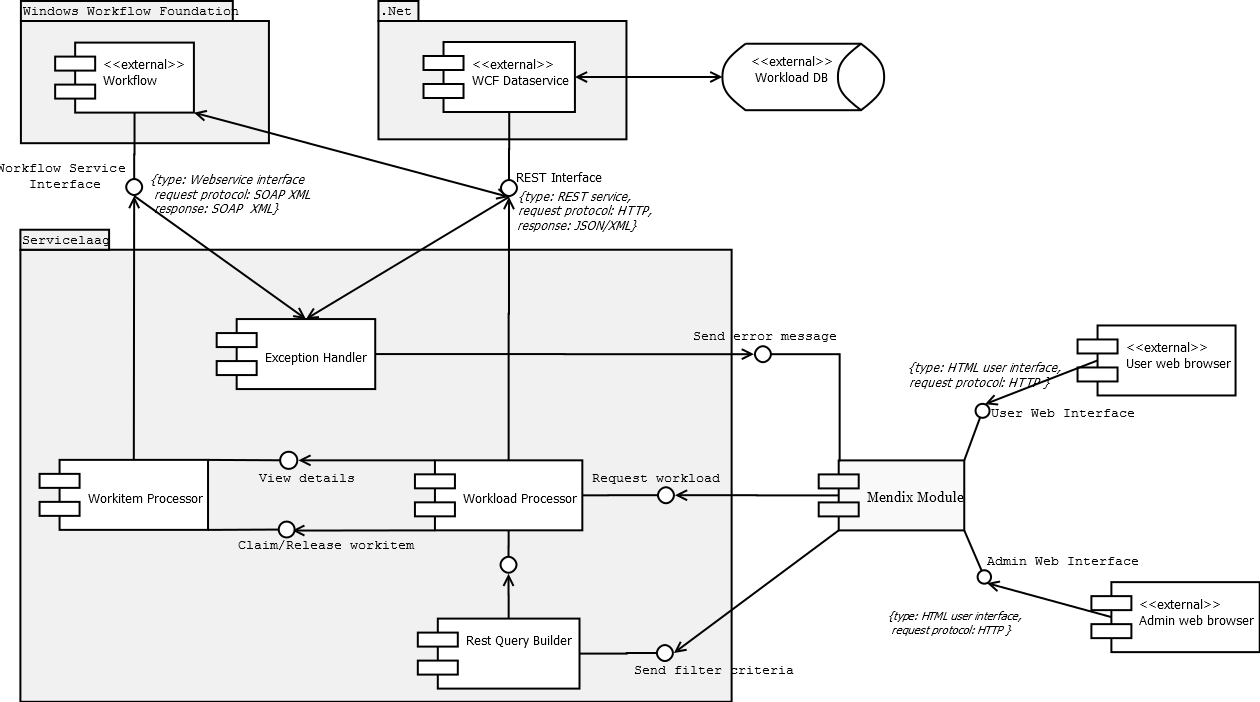
Later in het project is nog één extra aanpassing in de contextview aangebracht, namelijk dat de database niet direct aangesproken wordt maar dat er een REST dataservice tussen is geplaatst, die gebruikt kan worden om de database te benaderen. Er is gekozen om een REST dataservice te gebruiken omdat het direct benaderen van de database niet de meest veilige en effectieve manier is. Een REST dataservice zoals ontwikkeld voor dit project is een service die de data uit de Werkvoorraad database aanbiedt. De data is als XML representatie aangeboden en via een de URL van de dataservice benaderbaar. Doormiddel van eenvoudige HTTP-requests kan deze data verkregen worden van de dataservice.

Door het gebruiken van een REST dataservice kan de Werkvoorraad database vanuit allerlei systemen op eenzelfde wijze worden aangesproken. De communicatie met de database blijft hiermee ook altijd op één plek.

#### Functional view

Voor de functionele representatie van het systeem is gebruik gemaakt van het functional viewpoint. De functional view geeft de functionele elementen van het systeem, de verantwoordelijkheden daarvan en de afhankelijkheden ertussen weer.

Op de volgende pagina is het functionele model te vinden zoals deze is ontwikkeld voor het systeem. Het functionele model is door een aantal iteraties gegaan alvorens het in onderstaande afbeelding heeft geresulteerd.



Bovenstaand functioneel model laat zien welke functionele elementen zich waar in het geheel bevinden. Voor een uitgebreide uitleg van alle functionele elementen wordt verwezen naar de Architectual Description.

Zoals eerder vermeld heeft het ontwerpen van bovenstaande architectuur een aantal iteraties doorlopen. Feedback van de opdrachtgever en bepaalde inzichten die tijdens het ontwikkelen naar voren zijn gekomen hebben uiteindelijk geresulteerd in bovenstaand model.

De belangrijkste keuze gemaakt voor de functional view is de keuze om de REST dataservice vanuit de servicelaag aan te spreken op een dusdanige wijze dat een JSON response wordt gegeven. De REST dataservice, die in .Net ontwikkeld is, biedt de data in een tweetal vormen aan, namelijk middels XML en JSON. Er is gekozen om hier JSON voor te gebruiken om dat JSON (JavaScript Object Notation) in Java zeer eenvoudig en snel te interpreteren en verwerken is. XML vraagt een veel geavanceerdere uitlees methode in Java wat ook aanzienlijk meer van performance vraagt.

#### Information view

Tot slot is ook nog de information view gemaakt. De information view geeft aan wat voor informatie in het systeem is en welke informatie gedeeld wordt tussen de verschillende entiteiten in het systeem. Dit wordt gevisualiseerd met behulp van zogeheten Data-flow Diagrams die de informatiestromen weergeven.

Teven is een zogeheten Data Information Ownership Table gemaakt, die weergeeft welke entiteit welke rechten heeft op de data die in omloop gaat in de applicatie. Het gaat hier om de rechten van de verschillende systemen op de Workitem entitieit. De Workitem entiteit is in de de enige tabel in de Werkvoorraad database. Hierin worden alle workfow stappen bijgehouden die wachten op actie vanuit een ander systeem (in het geval van dit project vanuit een Mendix applicatie).

In het geheel zijn 2 entiteiten van Workitem te erkennen. Enerzijds is er de Werkvoorraad database met de tabel Workitem. Deze database wordt vanuit de workflow opgebouwd, de worklfow is dus sturend voor de data in deze database. Anderzijds is er in de Mendix Business Modeler ook een entiteit Workitem. Bij het ophalen van de workitems uit de Werkvoorraad database moeten namelijk alle uit de externe database opgehaalde workitems geconverteerd worden naar Mendix specifieke objecten om ze vervolgens te kunnen tonen in een stukje user interface.

Welke data waar in de applicatie rond gaat, waar de data wordt geconverteerd naar Mendix objecten, etc. staat beschreven in het Architectual Description.

Voor de inormation view is van een aantal functionele processen, waarbij de transitie van data/informatie een belangrijke rol speelt, een data-flow diagram gemaakt. Deze processen zijn:

* Ophalen werkvoorraad
* Workflow activity/stap aanroepen
* Parameters bij workitem operatie ophalen
* Workitem claimen

Vanwege het formaat van de diagrammen zijn ze niet opgenomen in dit rapport, maar wordt hiervoor verwezen naar de externe bijlage Architectual Description. Van alle processen wordt beschreven welke data in welke volgorde in het systeem circuleert.

Omdat voor de processen ‘Ophalen werkvoorraad’ en ‘Parameters bij workitem operatie ophalen’ een aantal belangrijke architectonische keuzes zijn gemaakt, zullen deze processen hieronder nader uitgelegd worden.

**Ophalen werkvoorraad**Het functionele proces: ophalen werkvoorraad, beschrijft het ophalen van de de lijst met workflow-stappen (workitems) die in de Werkvoorraad database wordt bijgehouden. Dit proces heeft interactie met verschillende systemen, aangezien de data uit een externe database opgehaald moet worden, wat zoals in hoofdstuk 5.1.3.1 beschreven staan, via een REST dataservice gebeurt.

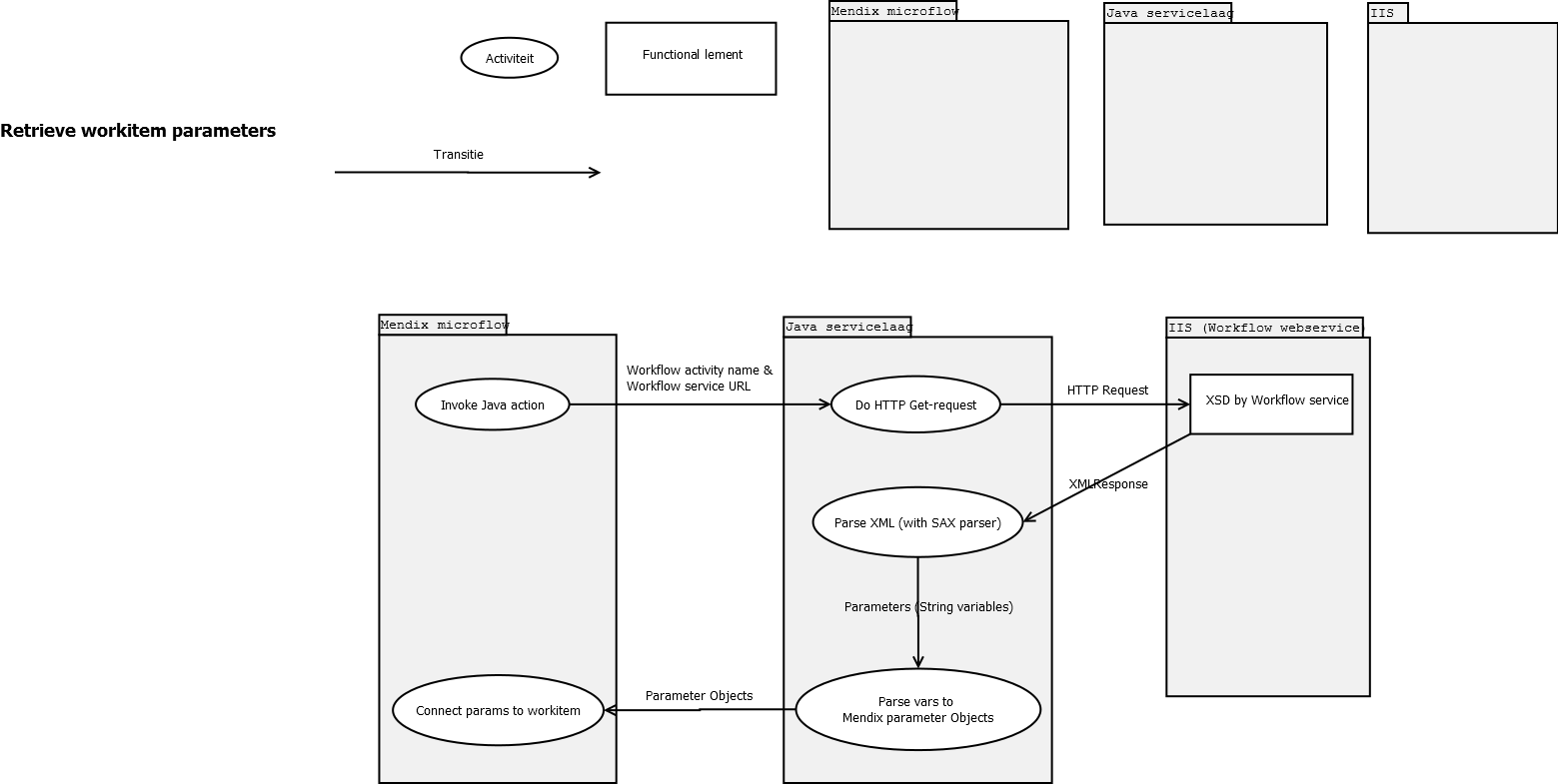
Iets wat van belang was voor de data-flow diagram van ‘Ophalen werkvoorraad’ was hoe ervoor gezorgd kon worden dat de gebruiker kon filteren bij het ophalen. Het is namelijk niet gewenst altijd alle werkvoorraad op te halen, dit kan de performance erg beïnvloeden. Er moet bij het ophalen gefilterd kunnen worden op de werkvoorraad attributen en er moet een maximaal aantal resultaten opgegeven kunnen worden. Om dit te realiseren is er gekozen om in het Mendix Domain Model (zie hoofdstuk 2.2) een WorkloadFilter entiteit aan te maken. Deze bevat precies dezelfde attributen als de Workitem entiteit, plus een extra attribuut om het maximaal op te halen resultaten op te geven. Met behulp van een stukje in de Mendix Business Modeler ontwikkelde user interface kan deze filter ingevuld worden en vervolgens zal in de servicelaag op basis van de opgegeven criteria een HTTP-request gegenereerd worden waarmee de data opgehaald kan worden. Een request is simpelweg een aangepaste URL, om bijvoorbeeld enkel de workitems van een bepaalde instantie op te halen is onderstaande URL benodigd voor de request.

<http://server:port/WorkloadDataService.svc/Workitem?$filter=InstanceID> eq "#instanceId"

Op bovenstaande manier kan op allerlei variabelen gefilterd worden in de werkvoorraad op basis van wat de gebruiker opgegeven heeft.

Iets wat pas later is opgenomen in de architectuur van ‘Ophalen werkvoorraad’ is het stuk Mendix microflow wat de opgehaalde lijst van de Werkvoorraad synchroniseert met de lijst aan reeds bestaande Mendix Workitem objecten. Tijdens sprint 4 is bij het testen aan het licht gekomen dat het niet werkt om een lijst met workitems op te halen, deze om te zetten naar Mendix objecten en ze vervolgens direct te tonen aan de gebruiker. Dit zorgde voor concurrency problemen, namelijk dat meerdere gebruikers tegelijk een workitem konden claimen. Om dit probleem tegen te gaan, is gekozen om in de Mendix microflow een synchronisatie op de lijst uit te voeren, zodat nieuwe objecten aan de Mendix database worden toegevoegd, maar reeds bestaande Workitem objecten niet nogmaals worden omgezet naar Mendix objecten. Op deze manier is er dus altijd slechts één lijst met workitems die altijd synchroon is met de externe Werkvoorraad database. Meer hierover is te lezen in hoofdstuk 5.5, waar Sprint 4 beschreven wordt. Om een beter beeld te krijgen bij het hierboven geschetste proces wordt met nadruk verwezen naar het Architectual Description waar dit met behulp van afbeeldingen nader uitgewerkt wordt.

**Parameters bij workitem operatie ophalen**Voor het proces ‘Parameters bij workitem ophalen’ is onderstaande data-flow architectuur ontworpen.



Figuur 5.2: Data-flow 'Paremeters bij workitem ophalen'

Van dit proces is pas vrij laat in het project (Sprint 4) gebleken dat het nodig was.   
Om te weten welke parameters een workflow activity nodig heeft in zijn aanroep moeten deze parameters bekend zijn aan de Mendix kant. Dit moet op een manier opgelost worden dat te allen tijde de juiste parameters in de aanroep worden gebruikt voor de aanroep van een specifieke workflow-stap. Aangezien de benodigde parameters voor workflow-stappen ook kunnen wijzigen is het niet mogelijk om deze parameters bijvoorbeeld vanuit de workflow op te nemen in de werkvoorraad. Daarmee zou namelijk een stap weggeschreven kunnen worden naar de werkvoorraad, vervolgens de parameters gewijzigd kunnen worden, en dan zouden de parameters al niet meer up-to-date zijn.

Om die reden is zoals in bovenstaande afbeelding te zien is gekozen de parameters op te halen uit de XSD (XML Schema Definition) die beschikbaar is bij de workflow. Elke workflow (die als webservice gehost wordt in IIS) heeft namelijk een geassocieerd XSD bestand waarin alle activiteiten en parameters van de workflow staan. De XSD zal opgehaald worden en Java met behulp van een SAX parser[[13]](#footnote-13) verwerkt worden. Meer over bovenstaande en de keuze voor het gebruik van SAX parser is te lezen in hoofdstuk 5.5, waarin Sprint 4 wordt beschreven.

Het proces ophalen van parameters kan op verschillende momenten aangeroepen. Zo zullen de parameters opgehaald moeten worden voordat een workflow activity aangeroepen wordt, maar kan het bijvoorbeeld ook aangeroepen worden om meer details van een workflow activity op te halen.

### Mastertestplan schrijven

Het mastertestplan is bedoeld als leidraad voor het testen, om de kwaliteit van het project te waarborgen en om betrokken op de hoogte te stellen van de gehanteerde testaanpak.

Voor de inhoud van het mastertestplan wordt verwezen naar de externe bijlage, het Mastertestplan.

Het mastertestplan beschrijft welke testtechnieken gehanteerd zullen worden, wat de algemene testaanpak is en heeft een testrisico analyse om de kritieke punten in het project te onderkennen. Deze risicoanalyse is gebaseerd op uitgevoerd onderzoek en gesprekken met de opdrachtgever en de betrokken expert, dhr. J.J Luursema. Gebleken is dat op een aantal punten een groot risico is dat problemen ontstaan. Op die punten zal dus extra gelet moeten worden bij het ontwikkelen. Tevens is met behulp van MoSCoW aangegeven wat de prioriteiten zijn van de te testen punten. Hiermee wordt aangegeven wat getest moet worden (Must), wat in principe getest zou moeten worden (Should) wat getest zou kunnen worden (Could) en wat niet getest zal worden (Will not).

Het mastertestplan is gecontroleerd door mevr. K. Visser die in Centric werkzaam is in de algemene functie van tester.

## Sprint 1

De eerste sprint heeft twee doeleinden gehad. Allereerst het installeren van alle benodigde software en het opzetten van de software-omgeving. En ten tweede het beginnen aan het daadwerkelijk bouwen van het systeem. Om te bepalen wat gedaan werd in de sprint, is een sprint planning meeting gehouden, waarin met het Scrum team (zoals beschreven in hoofdstuk 4.2) is bepaald wat in de sprint gedaan zou worden. Om een goede planning te kunnen maken is gebruik gemaakt van planningspoker, een techniek waarbij door middel van puntentoekenning aan stories, geschat kan worden hoe veel werk een story is. Deze aanpak is tevens voor alle onderstaande overige sprints gehanteerd.

Na de eerste sprint planning meeting is bepaald om de volgende taken en stories uit te voeren:

|  |  |
| --- | --- |
| **Naam** | **Omschrijving** |
| Taak | Software omgeving installeren, inclusief het ontwerpen van een voorbeeld workflow in Windows Workflow Foundation en het hosten daarvan in IIS. |
| Story 1 | Ik als Mendix Applicatie wil een workflow kunnen instantiëren. |
| Story 2a | Ik als Workflow wil workflow-stappen naar een Werkvoorraad database kunnen wegschrijven. |

**Software omgeving installeren:**Het installeren van de software omgeving ging niet zonder slag of stoot. Allereerst waren het voor mij een hoop nieuwe producten om mee te werken, maar daarbij zijn er ook meerdere malen problemen met de PC geweest waardoor systeembeheer erbij gehaald moest worden. Dit alles heeft veel tijd in beslag genomen.

Het installeren van en verdiepen in WF heeft voor deze taak het meeste tijd gekost. Het hele gebied van workflows was voor mij nog onbekend. Als testcase voor het project was bedacht zelf een workflow te ontwerpen die de aanvraag van een visvergunning simuleert. Het tekenen van de workflow en het hosten ervan in IIS heeft de nodige tijd aan uitzoekwerk gekost. Om hier meer over te leren, heb ik tutorials gevolgd van Microsoft waarin stap voor stap wordt uitgelegd hoe WF werkt, hoe workflows getekend kunnen worden, etc.

Later is ook nog AppFabric geïnstalleerd. Dit is een tool die als uitbreiding werkt op IIS. Met AppFabric kunnen webservices en workflows eenvoudig bekeken worden. Allerlei beheersmatige activiteiten kunnen hiermee op workflows uitgevoerd worden. Zo is te zien welke workflows actief zijn, welke workflows ‘idle’ zijn en welke worfklows afgebroken zijn vanwege een bepaalde fout. Als een workflow idle is, betekent het dat hij niet meer loopt, maar ook niet afgebroken is. De workflow is dus gepauzeerd en wacht dus op actie om verder te gaan. Van afgebroken workflows kan ook gezien worden waar en waarom het fout is gegaan. AppFabric is dus een zeer handige tool die veel gebruikt is voor dit project.

**Instantiëren van een workflow**Het instantiëren van een workflow kan gedaan worden door middel van het aanroepen van de eerste stap in een workflow. De workflow moet dus zo gemodelleerd worden dat de eerst stap/activity in de workflow een ReceiveMessage[[14]](#footnote-14) activiteit is met bijvoorbeeld de naam StartWorkflow. Hier kunnen eventueel nog parameters aan opgegeven worden om de workflow mee te starten. Om dit te kunnen testen is in de eerder gemaakte visvergunning workflow een stap geplaatst met de naam StartAanvraag. Op het moment dat deze stap wordt aangeroepen zal automatisch een nieuwe instantie van de workflow gaan lopen in IIS.

Uit de productrisico analyse (zie Mastertestplan) is gebleken dat dit punt erg risico gevoelig was. Hier is dan ook extra tijd besteed aan het uitdenken van een oplossing met een zo`n laag mogelijk risico op problemen.

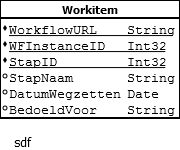
Om te weten hoe de workflow aangeroepen kan worden, is gebruik gemaakt van de eerder opgedane kennis met betrekking tot webservices en SOAP. Aangezien een workflow gehost in IIS op exact dezelfde wijze benaderbaar is als een normale SOAP webservice, was het niet heel moeilijk om dit uit te zoeken.

Het aanroepen van een workflow-stap kan op twee manieren gebeuren. De eerste mogelijkheid is om het gewoon via een SOAP HTTP request te doen. De andere manier is om gebruik te maken van bestaande mogelijkheden in Java om een proxy client te genereren, waardoor de workflow als zijnde Java methodes kan worden aangeroepen. Om het aanroepen van de workflow te testen, is eerst een stukje C# code geschreven in Visual Studio waarin de workflow als een service-referentie wordt geïmporteerd. Dit is gelijk aan het genereren van een proxy client in Java, echter in Visual Studio werkt het een stuk eenvoudiger. Vandaar dat daarmee is begonnen.

De belangrijkste vraag voor het aanroepen van de workflow activity was welke methode gebruikt zou gaan worden. Het generen van een proxy client zal namelijk veel minder foutgevoelig, en wellicht makkelijker te implementeren zijn. Echter, hiermee kan onmogelijk iets generieks opgezet worden. Zodra een workflow-stap andere parameters of een andere operatie naam krijgt, zal de hele proxy opnieuw gegenereerd moeten worden. Hierdoor wordt de aanpasbaarheid en schaalbaarheid drastisch ingeperkt. Door gebruik te maken van een SOAP request kan dit een stuk generieker opgezet worden. Een SOAP bericht zou bijvoorbeeld in de code gegenereerd kunnen worden, afhankelijk van variabelen. Om die redenen is uiteindelijk gekozen om SOAP te gebruiken voor het aanspreken van de workflows.

Het gebruik van SOAP brengt wel met zich mee dat de foutgevoeligheid heel hoog ligt. Door ergens net een klein foutje in het SOAP bericht te hebben kan de hele request mislukken zonder dat het heel duidelijk is waar het aan ligt. Het standaard soap bericht kan opgesteld worden en getest worden, dusdanig dat het geen fouten meer bevat. Echter, zodra er bijvoorbeeld variabele parameters opgenomen moeten worden in het bericht, is het van belang dat dit op een correcte wijze gebeurd.

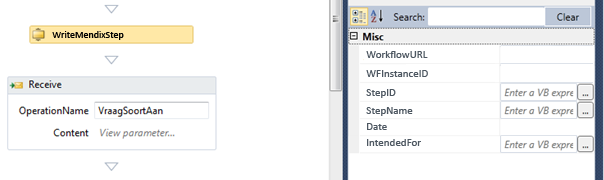
Een workflow wordt op basis van een aantal variabelen aangeroepen. De URL van de workflow is nodig en de operatie naam van de aan te roepen stap. Indien het om het instantiëren gaat hoeft enkel de stap aangeroepen te worden en zal hierdoor een nieuwe instantie beginnen te lopen. Echter wanneer het om een stap gaat verder in een workflow zal teven de instantie ID meegeven moeten worden in de aanroep, zodat IIS weet om welke precieze workflow instantie het gaat. Als oplossing om achter deze variabelen te komen vanuit een Mendix applicaties is besloten een externe database te ontwikkelen waarin een lijst met stappen kan worden bijgehouden. Voor de implementatie van de database is onderstaand klasse-representatie gebruikt.



Figuur 5.3: Tabel Werkvoorraad

Door in een Mendix applicatie de data uit bovenstaande tabel op te halen, is altijd up-to-date informatie over welke workflow, instantie en stap het gaat bij het aanroepen. De StapID uit de tabel bepaald namelijk welke stap aangeroepen moet worden. Mocht in de workflow iets wijzigen dan zal dit bij het wegschrijven automatisch mee gewijzigd worden en zal dit voor het SOAP bericht geen problemen opleveren, aangezien deze gegenereerd wordt op basis van de variabelen uit bovenstaande tabel. Hoe deze informatie weg wordt geschreven naar de database vanuit de workflow staat hieronder beschreven.

**Workflow stappen wegschrijven naar de database vanuit de workflow**Een worfklow bestaat uit stappen. Een stap kan van alles zijn. Windows Worfklow Foundation biedt een groot aantal voor gedefinieerde stappen aan waarmee een workflow ontworpen kan worden. Het wegschrijven van gegevens naar de database kan op twee manieren gedaan worden. De eerste mogelijkheid is om een standaard ‘code-acitivty’ aan de workflow toe te voegen, en daarin de koppeling met de database te leggen. Dit heeft echter het nadeel dat het niet herbruikbaar is, en mocht er een aanpassing nodig zijn, in wat weggeschreven wordt naar de database, zal dit overal aangepast moeten worden. Daarom is in overleg met de workflow expert besloten om zelf aangepaste workflow activities te ontwikkelen die dit realiseren. In WF kunnen zelf activities gemaakt worden die net als andere voorgedefinieerde activities in een workflow gesleept kunnen worden. Zodoende zijn er twee activities ontwikkeld die in een workflow gebruikt kunnen worden. De eerste heet WriteMendixStep en de andere ReleaseMendixStep. Met de WriteMendixStep kunnen een aantal variabelen opgegeven worden en kan data naar de database weggeschreven worden. Deze workflow activities zijn geschreven in de C# programmeertaal en worden met de Mendix module opgeleverd. Het grote voordeel van het gebruik van activities is dat het slechts op één plek gewijzigd hoeft te worden als dat nodig is, namelijk in de activity zelf. Zie onderstaande afbeelding voor hoe dit er in de workflow uit ziet.



Figuur 5.4: Custom activity workflow

Zoals hierboven te zien is kunnen de variabelen stapID, stapNaam en bedoeldVoor (IntentedFor) ingevuld worden in de workflow. De andere variabelen worden automatisch ingevuld in de activity. Met behulp van deze activity kan dus een stap weggeschreven worden naar de database. Door bij StepName in bovenstaande afbeelding de naam ‘VraagSoortAan’ in te vullen zal de volgende stap in de workflow (VraagSoortAan) worden weggescheven naar de database.

## Sprint 2

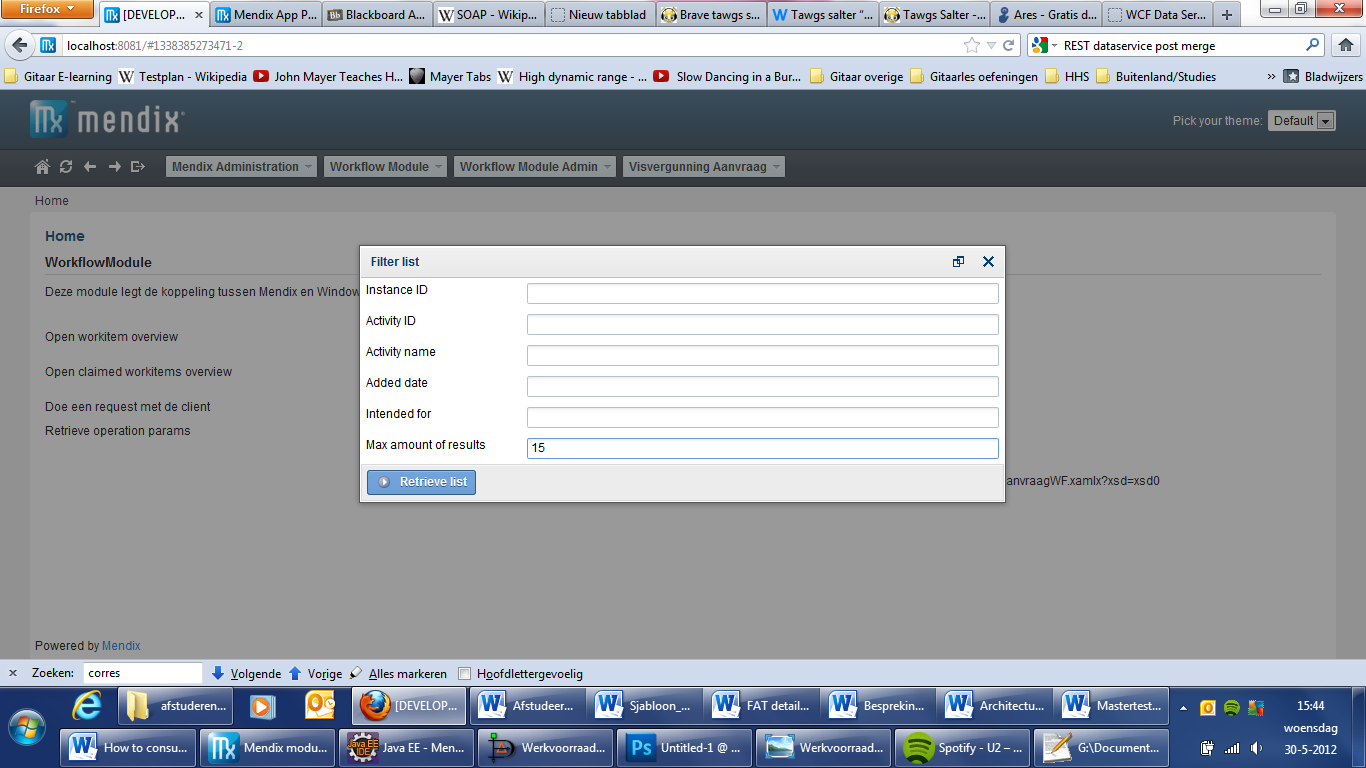
Voor de tweede sprint is in de sprint planning meeting besloten de volgende punten in de betreffende sprint af te handelen.

|  |  |
| --- | --- |
| **Naam** | **Omschrijving** |
| Story 2b | Ik als Mendix applicatie wil op basis van filter criterium een lijst met werkvoorraad ophalen. |
| Story 3 | Ik als Mendix applicatie wil een stap uit de lijst {2b} ophalen en claimen. |
| Story 7 | Ik als Mendix applicatie wil een geclaimde stap annuleren zodat hij weer als ongeclaimd terug komt in de werkvoorraad. |
| Story 20 | Ik als Gebruiker wil een geclaimde stap annuleren vanuit de in Mendix ontwikkelde user interface. |

De reden dat de stories geen opeenvolgende nummers zijn, is dat in de Sprint Planning meeting is besloten dat het voor dat moment beter was om story 7 eerder te implementeren.

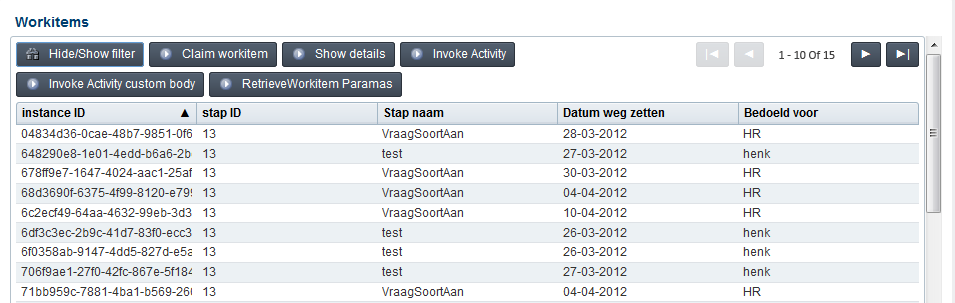
**Lijst met werkvoorraad ophalen op basis van filter criterium**Het ophalen van een lijst met werkvoorraad is een belangrijk onderdeel van de applicatie. Zoals in hoofdstuk 5.2 beschreven staat, is een database waarin een lijst met workitems wordt bijgehouden. Deze lijst wordt vanuit de workflow opgebouwd. Deze database moet benaderbaar zijn vanuit het Mendix systeem. Over deze storie is al uitleg gegeven in hoofdstuk 5.1.3.3 waarin de architectuur hiervan uitgelegd wordt. Voor meer hierover wordt dan ook verwezen naar dat hoofdstuk. In hoofdstuk 5.5 zal nog wat uitgelegd worden over een belangrijk probleem wat later in het project is opgetreden met betrekking tot het claimen van een stap uit de lijst met workitems.

Om het ophalen en filteren van de lijst met werkvoorraad te kunnen testen is in de Mendix Business Modeler een stukje user interface ontwikkeld die met de servicelaag, die de werkvoorraad ophaalt, communiceert. Ter realisatie hiervan is een Mendix formulier gemaakt die de mogelijkheid biedt filtercriteria op te geven. Na het opgeven van de filtercriteria zal een microflow (gemaakt in de Mendix Business Modeler) aangeroepen worden die de Java actie voor het ophalen van de workitems in de servicelaag aanroept. Na het terugkrijgen van deze workitems zullen de workitems aan een Mendix formulier meegegeven worden die deze workitems kan weergeven. In de browser ziet dit er uiteindelijk uit zoals in onderstaande screenshots. De eerste screenshot is het filter scherm wat de gebruiker krijgt alvorens het ophalen van de data, de tweede is een overzicht van de opgehaalde workitems.



Figuur 5.5: Filter popup scherm voor het ophalen van werkvoorraad

Zoals in bovenstaand afbeelding te zien is kan op alle attributen van de workitem gefilterd worden en kan tevens een maximaal aantal op te halen resultaten opgegeven worden.



Figuur 5.6: Lijst met workitems in een Mendix applicatie

Bovenstaande afbeelding laat zien hoe een lijst met Workitem objecten, opgehaald uit de Werkvoorraad database, er in een Mendix applicatie uit ziet.

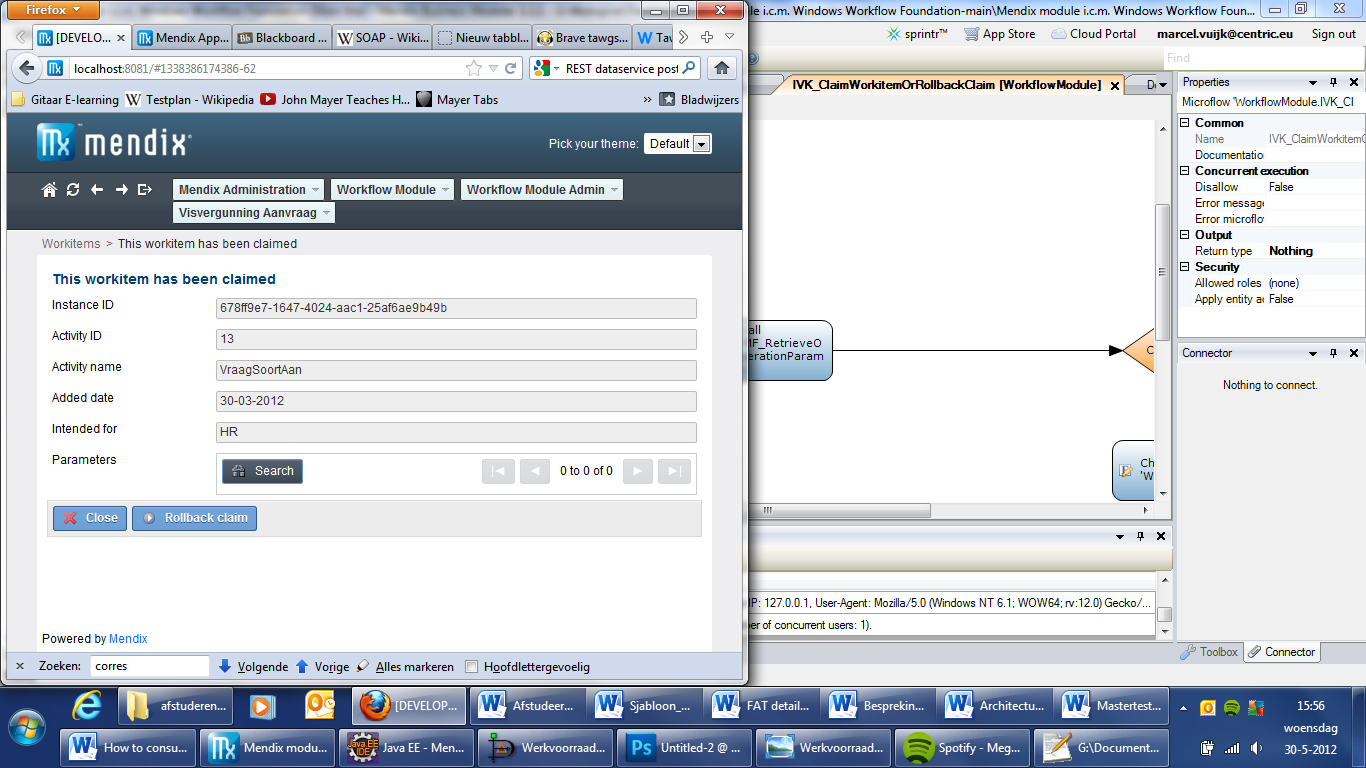
**Workflow-stap ophalen en claimen of een reeds geclaimde stap annuleren**   
Het claimen van een stap is een vrij eenvoudig proces. Door de stap op te halen uit de Werkvoorraad database en zoals in hoofdstuk 5.1.3.3 te converteren naar een Mendix object kan eenvoudig een variabele aangepast worden die de status van de stap op ‘geclaimd’ zet. Hiermee kan in de Mendix Business Modeler vervolgens ook eenvoudig afgevangen worden dat enkel de niet geclaimde stappen te zien zijn, waardoor dus een geclaimde stap na het claimen niet zichtbaar is voor anderen.

Het lastige aan dit proces is dat de status van zowel de stap in de Mendix lijst als de stap in de externe wekvoorraad database gewijzigd zal moeten worden. Mocht de Werkvoorraad database bijvoorbeeld nog vanuit een ander systeem benaderd worden, dan moet het niet mogelijk zijn dat de stap daarin nog als ongeclaimd in staat.

In hoofdstuk 5.1.3.3 is al uitgelegd hoe data opgehaald kan worden via de REST dataservice. Echter voor het geval van het claimen van een stap zal de data niet opgehaald moeten worden, maar de data zal ge-update moeten worden. In de externe bijlage Architectual Description is duidelijk te zien hoe dit precies werkt. Met behulp van data-flow diagrams wordt daarin duidelijk gevisualiseerd via welke entiteiten de data uiteindelijk gewijzigd wordt.

Het wijzigen van de data zal ook via de REST dataservice gedaan worden. De REST dataservice is op een aantal manieren aan te spreken. Door middel van HTTP requests kan data opgehaald, verwijderd, toegevoegd of gewijzigd worden. Hierin corresponderen de POST, MERGE en DELETE requests respectievelijk met een INSERT, UPDATE en DELETE statement op de database. Middels HTTP headers kan aangegeven worden welk response type gewenst is (JSON of XML).

Ook voor dit gedeelte is een stuk Mendix user interface ontwikkeld om de geclaimde stap te kunnen bekijken. Hieronder een screenshot van hoe dat er in de webbrowser uitziet.



Figuur 5.7: Formulier voor geclaimd workitem in een Mendix applicatie

Zoals in bovenstaande afbeelding te zien is, biedt dit formulier ook de mogelijkheid om de claim te annuleren (Rollback claim). Daarmee wordt precies hetzelfde gedaan als wanneer een workitem geclaimd wordt, echter hier zal de status van geclaimd weer terug gezet worden op ‘open’.

Alle hierboven beschreven functionaliteiten zijn tevens getest zoals beschreven in het mastertestplan, zie externe bijlage: Mastertestplan. Er zijn functionele acceptatie tests uitgevoerd om de functionaliteiten te testen op de correcte werking. De testen staan beschreven in het Functionele Acceptatie Testrapport.

## Sprint 3

De derde sprint is vooral bedoeld om de reeds opgeleverde applicatie delen te optimaliseren. Er is dus in de derde sprint geen nieuwe functionaliteit opgeleverd, wat eigenlijk ook tegen de SCRUM regels ingaat. Maar gezien het kleine projectteam, met alleen ikzelf als development teammember, is hier en daar iets van de standaard SCRUM aanpak afgeweken.

Het eerste gedeelte van sprint 3 is benut om alle reeds geschreven code en ontworpen Mendix modellen te refactoren. Dat houdt eigenlijk een grote schoonmaak van de code in. De code was hiermee ingekort en overzichtelijk gemaakt. Ook zijn unit-tests toegevoegd aan de Java code. Dit is gedaan met behulp van JUnit[[15]](#footnote-15), een framework speciaal voor het testen in Java. Met unittesten worden afzonderlijke delen van het systeem getest. Dit wordt getest op een manier waarbij het stuk systeem zoveel mogelijk los staat van andere systemen. Met unittesten wordt de kwaliteit van de code gewaarborgd en kan snel en eenvoudig getest worden met verschillende soorten input.

Tijdens het opschonen van de code en unittesten zijn geen problemen aan het licht gekomen. Wel zijn een aantal stukjes code aangepast om bijvoorbeeld excepties af te vangen of de foutgevoeligheid te verlagen.

Verder is in de derde sprint tijd genomen om de reeds gemaakte rapportage uit te breiden en te verbeteren. Zo is de information view van de architectuur verder uitgewerkt en is het mastertestplan verbeterd naar aanleiding van feedback van de accordeerder.

Sprint 3 was een vrij korte, maar absoluut belangrijke en nuttige sprint.

## Sprint 4

Hieronder de stories die tijdens de sprintplanning meeting opgesteld waren om af te handelen in sprint 4.

|  |  |
| --- | --- |
| **Naam** | **Omschrijving** |
| Story 6 | Ik als applicatie wil een workflow activiteit met parameters kunnen aanroepen op basis van een referentie uit de werkvoorraad lijst. |
| Story 8 | Ik als Applicatie wil excepties vanuit WF kunnen afvangen en als bruikbare excepties tonen in Mendix. |
| Story 12 | Ik als Workflow wil een Mendix webservice kunnen aanroepen. |

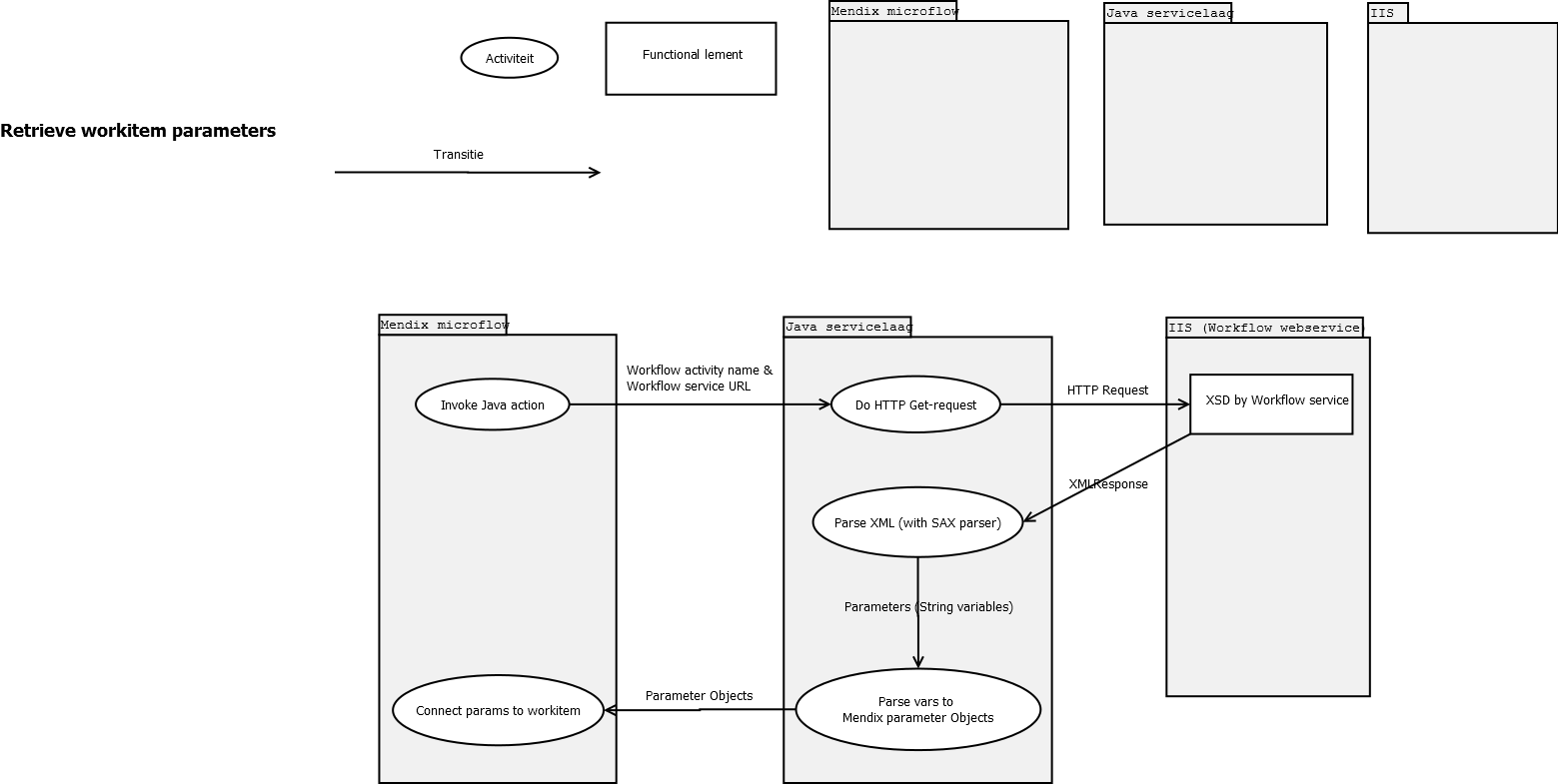
**Workflow activiteit aanroepen vanuit Mendix applicatie**Het aanroepen van een workflow activiteit vanuit een Mendix applicatie staat al beschreven in hoofdstuk 5.2, waarin sprint 1 wordt beschreven. Hierin wordt uitgelegd hoe een workflow geïnstantieerd kan worden door de eerste activity in een workflow aan te roepen. Het aanroepen van een andere, willekeurige activity kan op precies dezelfde manier gedaan worden. Het probleem wat echter ontstaat bij het aanroepen van een activity is dat een bepaalde workflow-stap in vrijwel alle gevallen gegevens nodig heeft. Als een worfklow gestopt is bij een bepaalde stap zal dit in de meeste gevallen zijn omdat de workflow informatie nodig heeft om verder te gaan. Deze informatie kan door middel van parameters aan een stap meegegeven worden.

Het grote probleem waar bij het ontwikkelen van deze functionaliteit tegen aan werd gelopen, was het probleem dat in een Mendix applicatie geen kennis is van welke parameters een bepaalde workflow-stap heeft.

Na enig onderzoek is gebleken dat om dit te realiseren er twee mogelijkheden zijn om deze parameters te achterhalen. De eerste mogelijkheid is om ze vanuit de workflow op één of andere manier door te geven aan de Mendix applicatie. De andere mogelijkheid is om ze vanuit de Mendix applicatie te achterhalen.

Beide mogelijkheden zijn onderzocht en uiteindelijk is enkel de laatste mogelijkheid een serieuze optie gebleken. Het is namelijk onmogelijk om vanuit de workflow automatisch op de hoogte te zijn van de parameters bij een bepaalde stap. De enige optie is om de parameters hard-coded in de ‘WriteMendixStap’[[16]](#footnote-16) op te nemen. Deze optie is echter zeer foutgevoelig, door een simpele typefout zou het hele proces van aanroepen van een workflow activiteit al niet meer werken namelijk. De andere mogelijkheid, waar na enig onderzoek achter is gekomen, is om de parameters te achterhalen via een aan de workflow geassocieerde XML Schema Definition (XSD). Elke workflow webservice heeft namelijk een geassocieerd XSD bestand waarin alle activiteiten en parameters van de workflow staan. Door middel van het uitlezen van dat bestand zouden de parameters achterhaald moeten kunnen worden.

Dit proces is uitgedacht in de Information view van de ontworpen software architectuur. Onderstaande afbeelding is daarbij gemaakt.



Figuur 5.8: Information view diagram voor ophalen parameters bij workitem

Te zien is hoe vanuit de Java servicelaag door middel van een HTTP request de XSD wordt opgehaald. Vervolgens zal deze in Java verwerkt worden. Uit het hele XSD bestand zijn namelijk enkel de parameters nodig voor de betreffende workflow-stap. Te zien is dat het uitlezen van deze parameters wordt gedaan met behulp van een SAX (Simple API for XML) parser. Er zijn twee manieren om XML bestanden mee uit te lezen. De eerste manier is door het gebruik van een DOM (Document Object Model) parser, de tweede manier is door gebruik te maken van de SAX parser. De keuze voor SAX parser is vooral performance gerelateerd. DOM parser is eenvoudiger te begrijpen en implementeren dan SAX parser, echter DOM parser zal veel meer van de performance vereisen indien het grotere XSD bestanden betreffen. De onderbouwing hiervan is te lezen in het Architectual Description waarin de keuze voor SAX parser nader wordt uitgelegd vanuit de performance perspective. Hieronder zal kort beschreven worden hoe de implementatie van de SAX parser er uitziet zoals ontwikkeld voor dit project.

**Sax parser implementatie**Een SAX parser werkt met event-handlers. Door een XML document met de SAX parser te verwerken wordt het hele document doorlezen en voor ieder deel dat de parser herkent als een standaard element wordt een methode aangeroepen. SAX kent de volgende event-handlers:

* startDocument: Deze methode zal aangeroepen worden bij de start van ieder document.
* endDocument: Deze methode zal aangeroepen worden bij het einde van ieder document.
* startElement: Deze methode zal aangeroepen worden bij de start van elk element. Een element is een willekeurige XML tag.
* endElement: Deze methode zal aangeroepen worden bij het einde van elk element.
* characters: Deze methode zal worden aangeroepen worden wanneer extra karakters, zoals spaties of enters, worden tegengekomen in het document.

Een XSD schema van een workflow ziet er ongeveer zo uit:

|  |
| --- |
| <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">  <xsd:element name="exampleActivityName">  <xsd:complexType>  <xsd:element name="exampleParameterName">  <xsd:element name="exampleParameterName2">  </xsd:complexType>  </xsd:element>  </xsd:schema> |

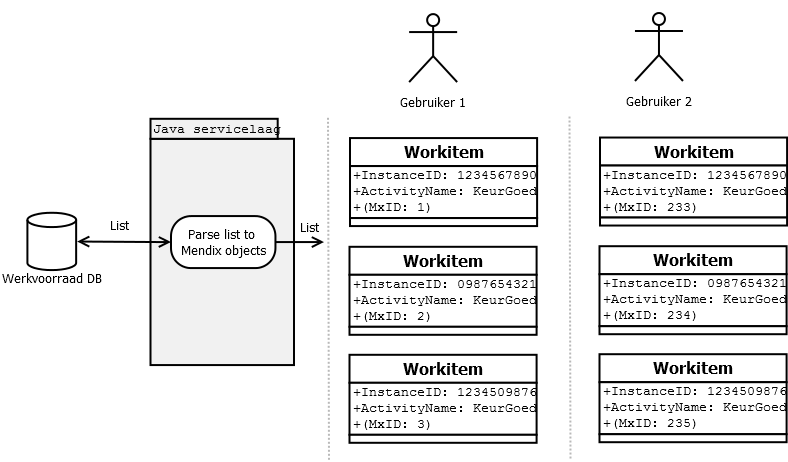
Het doorlopen van dit document met behulp van de SAX parser ziet er dan ongeveer zo uit:

|  |  |
| --- | --- |
| **Tag** | **Methode aanroep** |
| <xsd:schema | startDocument() |
| <xsd:element | startElement() |
| <xsd:complexType | startElement() |
| <xsd:element | startElement() |
| </xsd:complextype | endElement() |
| </xsd:element | endElement() |
| </xsd:schema | endDocument() |

In bovenstaand voorbeeld heeft het XSD schema slechts één activity. In de meeste gevallen zullen dit er uiteraard veel meer zijn. Om nu achter de juiste parameters te komen voor één specifieke activity zijn in Java de startElement en endElement methodes opgenomen. Hierin wordt afgevangen wanneer bij een bepaalde tag is aangekomen met de activity naam die gezocht wordt. Vervolgens zullen van alle element tags in het activity element de waarde van het attribuut ‘name’ opgehaald worden, dit wordt doorlopen totdat de endElement van het activity element aangeroepen wordt. De gevonden parameter waardes zullen naar Mendix objecten omgezet worden en in een Mendix microflow aan de betreffende workitem gekoppeld worden.

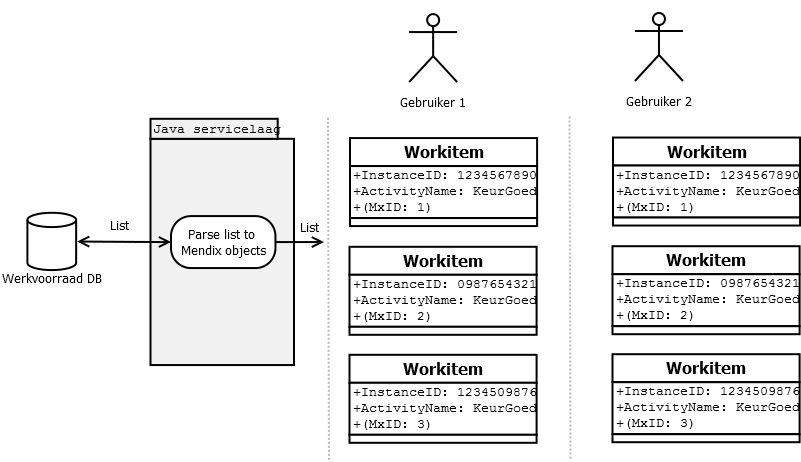
Met bovenstaande oplossing kan van alle worfklow-stappen (die in de Werkvoorraad database staan) te allen tijde achterhaald worden welke parameters benodigd zijn voor de aanroep. Nadat de hierboven beschreven oplossing geïmplementeerd was, kon getest worden.

Tijdens het testen is gebleken dat het aanroepen van een acitvity op basis van de opgehaalde parameters prima werkte. Echter, er was wel een andere belangrijke fout geconstateerd. Indien er namelijk meerdere gebruikers een lijst met werkvoorraad ophalen, kunnen al die gebruikers een bepaalde stap aanroepen/claimen. Dit kan omdat de opgehaalde lijst met workitems uit de Werkvoorraad database telkens voor het ophalen wordt geconverteerd naar een lijst van Mendix objecten. Telkens bij het ophalen wordt dan voor ieder Workitem object een tijdelijk Mendix object aangemaakt. Op deze manier heeft dus iedere gebruiker die de lijst ophaalt een eigen lijst van de werkvoorraad. Hieronder een afbeelding om het duidelijker te maken:



Figuur 5.9: Concurrency probleem bij ophalen werkvoorraad

In bovenstaande afbeelding is te zien dat de verschillende gebruikers dezelfde lijst van workitems zien. Echter wat verschilt, is de MxID die de workitems hebben; ze zien dezelfde workitems maar het zijn andere objecten. Als dus de status van een bepaald workitem bij Gebruiker 1 wordt gewijzigd naar ‘geclaimd’, betekent dit niet dat dit ook voor de workitem van Gebruiker 2 gebeurt.

Voor het aanroepen van een worfklow-stap levert dit niet direct een functioneel probleem op, namelijk zodra een workflow-stap wordt aangeroepen zal de workflow verder lopen, en de volgende die deze workflow aanroept zal een error krijgen omdat de workflow-stap reeds is aangeroepen en dus niet meer wacht op actie. Waar het echter wel een probleem oplevert, is bij het claimen van een stap. Het claimen van een stap betekent enkel het wijzigen van een variabele in de Workitem tabel. Het kan dus gebeuren dat die variabele meerdere keren naar ‘geclaimd’ wordt gewijzigd. Hierdoor kunnen meerder gebruikers denken dat ze de stap geclaimd hebben, en kunnen meerdere gebruikers aan de slag gaan om die stap af te handelen. De oplossing hiervoor is om te zorgen dat de verschillende gebruikers niet alleen dezelfde lijst met workitems zien, maar ook daadwerkelijk dezelfde Mendix objecten zien. Dit ziet er uit zoals hieronder afgebeeld.

Figuur 5.10: Concurrency probleem bij ophalen van werkvoorraad opgelost

Dit lijkt eenvoudig, echter de implementatie hiervan ligt vrij ingewikkeld. In plaats van dat er elke keer een complete lijst opgehaald kan worden uit de werkvoorraad moet de opgehaalde lijst (uit de externe database) gesynchroniseerd worden met de reeds in het Mendix datamodel opgenomen workitems. Hoe het proces van ophalen en synchroniseren precies werkt staat nader beschreven in het Architectual Description (Information View). Voor de uitgevoerde tests en de uitslag van die testen wordt verwezen naar de externe bijlage, het FAT Detailtestplan. Daarin worden alle Functionele Acceptatie Testen beschreven.

Het oplossen van het hierboven beschreven probleem heeft veel tijd gekost en daarmee voor een vrij grote vertraging in het project gezorgd. De overige stories die in eerste instantie gepland waren voor deze sprint, is dan ook niet meer aan toegekomen. Deze stories zijn in overleg met de Product Owner terug naar de backlog verplaatst. Dit waren de volgende twee stories:

|  |  |
| --- | --- |
| **Naam** | **Omschrijving** |
| Story 8 | Ik als Applicatie wil excepties vanuit WF kunnen afvangen en als bruikbare excepties tonen in Mendix |
| Story 12 | Ik als Workflow wil een Mendix webservice kunnen aanroepen. |

Story 12 is later verplaatst naar Sprint 5 en story 8 is op de backlog blijven staan, en zal op een later moment geïmplementeerd moeten worden.

In de Sprint Review meeting (Zie hoofdstuk 4.2.2) van sprint4 is teruggekeken op de tot nu toe behaalde resultaten. Gebleken is dat de basis gelegd was, echter hoe meer een eindproduct tot stand kwam, hoe meer de opdrachtgever een beeld kreeg van wat daadwerkelijk gewenst was. De requirements zijn hierbij zoals beschreven in hoofdstuk 5.1.1 (Requirements bepalen) nog enigszins aangepast.

## Sprint 5

Sprint 5 was de laatste sprint van het project. Hierin is in het bijzonder aandacht besteed aan documentatie en aan de overdracht van de opdracht.

|  |  |
| --- | --- |
| **Naam** | **Omschrijving** |
| Story 14 | Ik als Mendix Ontwikkelaaar wil kunnen lezen hoe ik de Mendix module met de servicelaag kan gebruiken. |
| Story 16 | Ik als Gebruiker wil kunnen lezen hoe ik de module moet gebruiken. |
| Story 15 | Ik als Workflow Ontwerper wil kunnen lezen hoe ik vanuit een workflow een Mendix webservice kan aanroepen. |

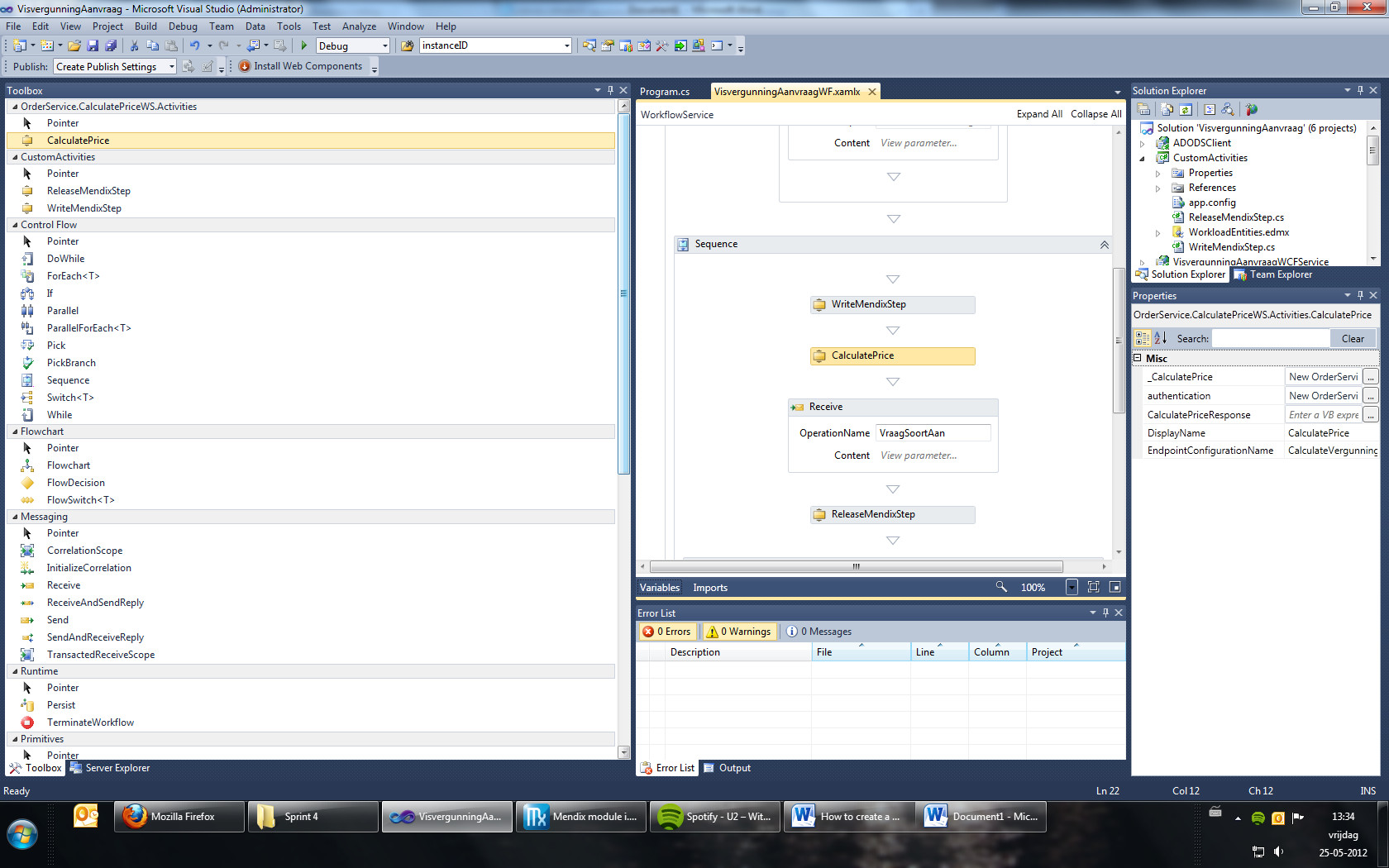
De eerste twee story’s waren vrij eenvoudig, en vooral een kwestie van schrijfwerk. Er is documentatie voor zowel de Ontwikkelaar als de Gebruiker opgeleverd. Met behulp van die documentatie heeft de toekomstige gebruiker van de Mendix module een beeld hoe de module gebruikt kan worden en wat de functionaliteiten ervan zijn. Voor de Mendix ontwikkelaar is beschreven hoe de module in elkaar zit, op welke punten gebruik wordt gemaakt van de servicelaag, etc. Op basis van deze documentatie zal de Mendix ontwikkelaar kunnen door ontwikkelen aan de module. In eerste instantie is de documentatie vooral bedoeld voor de overdracht van de opdracht. Na deze vijfde sprint zal het project namelijk worden overgedragen en zal er intern verder aan ontwikkeld worden.

Voor story 15 moest beschreven worden hoe vanuit een workflow een Mendix webservice aangeroepen kan worden. Dit is opzich een eenvoudig iets, aangezien WF mogelijkheden biedt om webservices te importeren en vanuit de workflow zeer eenvoudig aan te roepen. Ook het ontwikkelen van een webservice in Mendix is een zeer eenvoudig proces waar de Mendix Business Modeler zeer goede mogelijkheden in biedt. Het enige probleem was dat, om dit te kunnen documenteren, het ook getest moest worden. Daarom is een voorbeeld webservice ontwikkeld in Mendix die een eenvoudige berekening uitvoert op een input variabele. De berekende waarde wordt vervolgens teruggegeven. In de workflow is deze webservice geïmporteerd. Importeren in de workflow betekend dat de webservice-aanroep als een normale workflow-stap wordt opgenomen in de workflow, en dat een aantal velden kunnen worden opgegeven met benodigde waardes om de webservice aan te roepen. Hier is tegen een klein probleem aangelopen, namelijk dat de operatie geen veld had die de door de webservice gevraagde input vertegenwoordigde. Het had echter wel een veld waarin een webservice object kon worden meegegeven. Na enig onderzoek bleek het zo te zijn dat de input meegegeven kon worden door middel van een Visual Basic object expressie. Dat wordt gedaan op onderstaande manier:

New PackageNaam.WebserviceReference.CalculatePrice() With {.WebserviceParameter = #}

Er wordt dus een nieuw object aangemaakt van de webservice operatie en daarin worden de benodigde parameters opgegeven.

Een webservice geimporteerd in een worklfow ziet er zo uit:



Figuur 5.11: Mendix webservice geïmporteerd in een workflow

CalculatePrice is dus de geïmporteerde Mendix webservice operatie. Rechts is te zien welke variabelen opgegeven kunnen worden. In het veld \_CalculatePrice kan een object meegegeven worden met als parameters de input die de webservice operatie nodig heeft.

Voor een uitgebreide beschrijving van het aanroepen van een Mendix webservice vanuit de workflow wordt verwezen naar de interne bijlages A en B, waarin het creëren van een Mendix webservice en het aanroepen daarvan in de vorm van een gebruikers-tutorial wordt uitgelegd.

# Evaluatie

*In dit hoofdstuk zal het uitgevoerde project worden geëvalueerd. Zowel het proces (ontwikkelaanpak) als product zal worden geëvalueerd.*

## Proces

Omdat de opdracht een groot aantal, voor mij, onbekende elementen bevatte is aan het begin veel tijd geweest aan onderzoek. Zoals in hoofdstuk 4 beschreven staat is voor dit project gewerkt aan de hand van de Scrum ontwikkelmethode. Volgens de Scrum ontwikkelmethode wordt eigenlijk enkel gewerkt aan de hand van sprints, echter voor dit project is besloten om voorafgaand aan het ontwikkelen met sprints een initiatie fase te starten waarin onderzoek wordt gedaan en het project wordt voorbereidt.

Tijdens de initiatie fase is het Project Initiatie Document opgesteld, de software architectuur ontworpen en zijn testplannen geschreven. Tevens is de hele software omgeving geïnstalleerd en is al begonnen met experimenteren met de verschillende producten.

Het hebben van een initiatie fase heb ik zelf als zeer nuttig en prettig ervaren. Na de informatie fase was de opdracht een stuk duidelijker en was alles geïnstalleerd en voorbereidt om aan het ontwikkelen te beginnen.

Omdat de opdracht een heel aantal onbekende elementen bevatte vond ik het in het begin was lastig te begrijpen. Ik begreep het doel van de opdracht, maar kon het nog niet echt plaatsen. Vanuit Centric is echter voldoende tijd gegeven om me hierop in te lezen en na de initiatie fase was dit geen probleem meer.

Na de initiatie fase is gestart met het doorlopen van sprints volgens de Scrum ontwikkelaanpak. Het was voor mij voor het eerst dat ik op redelijke serieuze wijze gebruik maakte van een Agile ontwikkelaanpak. Het Agile, wat staat voor flexibel, ontwikkelen heb ik als zeer prettig ervaren. In Scrum wordt gewerkt met een bepaald project team die bestaat uit een Scrum master, een product owner en een development team. Omdat dit project een individuele opdracht was kon het development team slechts uit één teammember, namelijk ikzelf, bestaan. Bij Scrum horen ook de ‘daily Scrum meetings. Dit zijn standup meetings van maximaal 15 minuten. Tijdens zo`n meeting wordt besproken wat de vorige dag gedaan is en wat die dag gedaan gaat worden. Een daily Scrum meeting wordt normaal gesproken gehouden met het development team, echter omdat in het geval van dit project slechts één teammember was, is besloten om de daily scrum meetings te houden met de Scrum master. Aangezien de Scrum master meestal buiten de deur werkte is dit voor de meeste gevallen telefonisch gebeurd. Het houden van Daily scrum meetings heeft ervoor gezorgd dat problemen in het project tijdig besproken konden worden. De Scrum master heeft hierin voor uitstekende begeleiding gezorgd.

Het werken volgens de algemene scrum aanpak, met de scrum doorloop zoals beschreven in hoofdstuk 4.2.2, is ook als zeer prettig ervaren. In de initiatie fase is in overleg met de opdrachtgever (product owner) bepaald wat gedaan moest worden, en dit is in de vorm van Scrum stories uitgewerkt. Voor de start van iedere sprint werd een Sprint planning meeting gehouden waarin werd besproken wat in de komende sprint aan Scrum stories afgehandeld zou worden. Het inplannen van een sprint werd gedaan met behulp van planningspoker. Op deze manier bleef het proces zeer flexibel, en werd slechts op zeer korte termijn gepland. In de ‘sprint review meeting’, werd teruggeblikt op de afgeronde sprint. Gekeken werd of de planning realistisch was, en de aandachtspunten voor een volgende sprint worden besproken. Het flexibele werken, op korte termijn inplannen, de regelmatige terugblik momenten en het directe contact en feedback met het team zijn zeer goed bevallen. Aangezien het een zeer exploratief ontwikkelproces was vermoed was de keuze voor Scrum voor dit project absoluut een goede keuze.

In de voorlaatste sprint is het project iets uitgelopen vanwege een onverwacht probleem met betrekking tot concurrency. Echter dankzij de agile aanpak was dit geen groot probleem. In de overige sprints is vaak tegen het probleem aangelopen de het implementeren van een story een grote hoeveelheid uitzoek werk nodig had. Dit was soms lastig aangezien voor veel stories meer tijd nodig was voor het uitzoeken van hoe iets opgelost moest worden, en wat de beste oplossing was, dan dat er tijd benodigd was om de storie te ontwikkelen. Echter door dat dit goed in de planningen was meegenomen heeft dit niet voor problemen gezorgd. Tevens is door gebruik te maken van de juiste literatuur, en de beschikbaarheid van de technisch expert, de tijd besteed aan onderzoek zo klein mogelijk gehouden.

## Product

Het doel van de opdracht was: ‘Ontwikkelen van een servicelaag tussen Mendix en Windows Workflow Foundation voor Centric B.V’*.* De servicelaag moet aansluiten op een Mendix module die basis functionaliteiten biedt voor een koppeling met Windows Workflow Foundation.

Om deze doelstelling te behalen zijn een aantal tussenproducten opgeleverd. De belangrijkste tussenproducten waren het Project Initiatie Document (PID), de Architectual Description en de testplannen. Verder werd voor het einde van iedere sprint een nieuw stuk werkend systeemdeel opgeleverd en gedemonstreerd aan het project team.

Het PID en het Architectual Description zijn een belangrijke basis voor de ontwikkeling geweest. In het PID zijn de requirements opgesteld op basis waarvan ontwikkeld zal worden. Het Architectual Description beschrijft de belangrijkste architectonische keuzes en geeft op een visuele manier weer welke functionele componenten het systeem heeft en welke systeem delen wat voor interactie hebben. Beide documenten zijn door de opdrachtgever goedgekeurd, om zo te verzekeren dat mijn beeld bij het project in overeenstemming was met die van de opdrachtgever. De testplannen hebben ervoor gezorgd dat de kwaliteit van de opgeleverde functionaliteiten gewaarborgd was. In het Mastertestplan is een product-risico analyse gemaakt waarin de risico gevoelige punten in het systeem worden beschreven. Teven staat daarin beschreven welke testen worden uitgevoerd, en welke rapportage daarbij hoort. Het uitvoeren van de functionele acceptatie testen en unittesten heeft er toe geleid dat fouten ontdekt werden en de code optimaal was.

De opgeleverde tussenproducten zijn van groot belang geweest voor het project. Zonder deze producten had het opgeleverde eindresultaat van een stuk mindere kwaliteit geweest. Tevens hebben de tussenproducten ervoor gezorgd dat de verschillende stakeholders (opdrachtgever, Scrum team) op de hoogte waren van wat ontwikkeld ging worden. Tijdens de ontwikkeling is in het bijzonder de ontworpen architectuur enorm nuttig geweest. Deze is tussentijds wel nog een aantal keer aangepast, maar het vooraf vaststellen van de architectuur helpt enorm bij het ontwikkelen.

Het uiteindelijk opgeleverde eindproduct bevat helaas niet alle functionaliteiten die waren opgesteld om te implementeren. Dit vanwege sprint 4, waar door een onverwachts probleem vertraging is opgelopen. De storie die hierdoor niet meer binnen het project gerealiseerd heeft kunnen worden is : ‘*Ik als Applicatie wil excepties vanuit WF kunnen afvangen en als bruikbare excepties tonen in Mendix’*. Buiten deze story om is alles echter wel geïmplementeerd.

In de laatste fase is gebleken dat de opdrachtgever nog een aantal wensen had met betrekking tot de servicelaag (zie hoofdstuk 5.1.1, waarin de requirements beschreven staan), hierdoor is besloten om de opdracht intern over te dragen zodat de laatste functionaliteiten van de servicelaag ontwikkeld kunnen worden. Helaas kan ik dit niet zelf doen aangezien ik al van een andere baan ben voorzien na dit afstudeerproject, echter de overdracht zal ik nog wel door mij worden afgehandeld.

## Conclusie

De conclusie van het project is dat zowel ik en de opdrachtgever zeer tevreden zijn met het eindresultaat. Een goede basis voor de koppeling van een Mendix applicatie met Windows Workflow Foundation is gelegd. De opdracht was vrij complex, met veel systemen en veel nieuwe elementen, echter dankzij de goede begeleiding vanuit Centric is het project tot een succesvol einde gekomen.

Tevens is van het eind resultaat een presentatie gegeven voor het Mendix Competence Center[[17]](#footnote-17), daar werd zeer positief op gereageerd en vanuit verschillende hoeken bleek er interesse te zijn voor de ontwikkelde Mendix module.

# Wijze van aantonen beroepstaken

Voor het afstuderen zijn een aantal beroepstaken opgesteld die behaald moesten worden gedurende het afstuderen. Hieronder de beroepstaken zoals ze zijn opgenomen in het afstudeerplan:

|  |  |  |
| --- | --- | --- |
| **Beroepstaak** | **Omschrijving** | **Bijhorend product** |
| **1.1** *Selecteren methoden, technieken en tools* (**Niveau 3**) | Er zal tijdens het afstuderen gekozen worden welke methoden en technieken gebruikt gaan worden, zoals welke UML modellen en ontwikkelmethode gebruikt zullen worden. Er is al gekozen om Scrum te gebruiken. Centric wil namelijk dat gebruik wordt gemaakt van een agile methode. | **5** |
| **1.2** *Voorbereiden en opstarten softwareontwikkeltraject* (**Niveau 3**) | De servicelaag moet vanaf niets ontwikkeld worden, daarom zal een softwareontwikkeltraject voorbereid en opgestart worden. | **1, 2** |
| ***3.1*** *Ontwerpen van softwarearchitectuur* (**Niveau 4**) | Op basis van de requirements zal een architectuur ontwerp gemaakt worden voor de servicelaag als koppeling tussen een Mendix applicatie en WF. | **3, 4, 5** |

De getallen gebruikt bij ‘bijhorend product’ corresponderen met de volgende opgeleverde producten:

1. PID (Project Initiatie Document)
2. Sprintplanningen (priorisering van stories)
3. Beschrijving van requirements
4. UML Diagrammen
5. Architectuur ontwerp van de Servicelaag als koppeling tussen een Mendix applicatie en WF

Hieronder zal per beroepstaak aangegeven worden op welke wijze eraan gewerkt is om deze te behalen.

**Beroepstaak 1.1: Selecteren methoden, technieken en tools (Niveau 3)**Zoals in hoofdstuk 5 te lezen is zijn er verschillende methoden, technieken en tools gebruikt. De keuzes voor deze methodes zijn deels door de opdrachtgever, deels door mij en deels gezamenlijk in overleg met het Scrum team gemaakt. De belangrijkste keuzes worden hieronder nogmaals gepresenteerd:

* Scrum als ontwikkelaanpak
* REST dataservice voor het benaderen van de data
* Gebruik van Eclipse voor Java ontwikkeling
* Gebruik van SOAP request in plaats van proxy client laten genereren
* Gebruik van XSD voor ophalen van parameters bij workflow-stap
* Gebruik van JSON response voor het ophalen van de data uit de Werkvoorraad database

De verdediging van deze keuzes staat beschreven in hoofdstuk 5, Realisatie.

**Beroepstaak 1.2: Voorbereiden en opstarten softwareontwikkeltraject (Niveau 3)**Er was nog geen enkele basis beschikbaar voor de ontwikkeling van het project. Doormiddel van het bepalen van de project aanpak, het opleveren van het Project Initiatie Document en het opstellen en opleveren van de requirements is aangetoond dat er succesvol een softwareontwikkeltraject is opgestart.

**Beroepstaak 3.1: Ontwerpen van softwarearchitectuur (Niveau 4)**Het project heeft interactie met veel verschillende systemen. Een Java servicelaag die onderdeel is van een Mendix module en weer communiceert met een REST dataservice en workflows. Een workflow die ook weer communiceert met de dataservice, enz. Het is daarom van uiterst belang dat vooraf goed nagedacht wordt over hoe het systeem precies in elkaar zit, welke systemen bij het project betrokken zijn, welke onderdelen van het systeem op wat voor manier met elkaar communiceren, enz.

Om dit vast te leggen is een Architectual Description opgesteld, zoals ook beschreven in hoofdstuk 5.1 (Initiatie fase). Vanuit verschillende viewpoints wordt naar het geheel gekeken en zo is een architectuur opgesteld en goedgekeurd door de opdrachtgever, aan de hand waarvan het project ontwikkeld is. Voor de inhoud van het Architectual Description wordt verwezen naar de externe bijlage in het afstudeerdossier ‘Architectual Description’.

Zoals in hoofdstuk 5 te lezen is, is er ook ruim aandacht besteed aan het testen van de opgeleverde producten. Zo is een mastertestplan opgeleverd, een detailtestplan voor de functionele acceptatie tests en zijn er unit-tests in de code opgenomen. Zie externe bijlages voor de rapportage hiervan. Hiermee is dus ook aandacht besteed aan de beroepstaken 3.4 en 3.5, die gaan over het voorbereiden, plannen en uitvoeren van het testproces.

# Geraadpleegde literatuur

Onderstaande literatuur is geraadpleegd tijdens de ontwikkeling van het project.

|  |  |  |  |
| --- | --- | --- | --- |
| **Literatuur naam** | **Versie** | **Datum** | **Auteur(s)** |
| TMap® Next voor resultaatgericht testen | 1e druk | 2006 | T. Koomen, L. van der Aalst, B. Broekman en M. Vroon |
| Reader ‘Testing in Agile Software Development Environments with TMap NEXT®’ | 1.0 | 2010 | Sogeti Nederland B.V. |
| TestGoal. Leerboek resultaatgedreven software testen | 1e druk | 2008 | Derk-Jan de Grood |
| Software Systems Architectuur ‘Working with Stakeholders Using Viewpoints and Perspectives’ | 4e druk | 2008 | Nick Rozanski en Eoin Woods |

# Lijst met figuren

[Figuur 2.1: Infrastructuur Centric B.V. 6](#_Toc326306751)

[Figuur 2.2: Een Domein Model in de Mendix Business Modeler 7](#_Toc326306752)

[Figuur 2.3: Een microflow in de Mendix Business Modeler 7](#_Toc326306753)

[Figuur 2.4: Een formulier in de Mendix Business Modeler 8](#_Toc326306754)

[Figuur 2.5: Een formulier in de webbrowser 8](#_Toc326306755)

[Figuur 2.6: Workflow in Windows Workflow Foundation 9](#_Toc326306756)

[Figuur 2.7: Windows Worfklow Foundation 10](#_Toc326306757)

[Figuur 3.1: Context diagram servicelaag 13](#_Toc326306758)

[Figuur 3.2: Sequence diagram 15](#_Toc326306759)

[Figuur 3.3: Mendix microflow als onderdeel van workflow 16](#_Toc326306760)

[Figuur 4.1: Sprint doorloop 22](#_Toc326306761)

[Figuur 4.2: Een voorbeeld Burndown Chart in sprintr 24](#_Toc326306762)

[Figuur 5.1: Context diagram Mendix module 33](#_Toc326306763)

[Figuur 5.2: Data-flow 'Paremeters bij workitem ophalen' 37](#_Toc326306764)

[Figuur 5.3: Tabel werkvoorraad 42](#_Toc326306765)

[Figuur 5.4: Custom activity workflow 43](#_Toc326306766)

[Figuur 5.5: Filter popup scherm voor het ophalen van werkvoorraad 45](#_Toc326306767)

[Figuur 5.6: Lijst met workitems in een Mendix applicatie 45](#_Toc326306768)

[Figuur 5.7: Formulier voor geclaimd workitem in een Mendix applicatie 46](#_Toc326306769)

[Figuur 5.8: Information view diagram voor ophalen parameters bij workitem 49](#_Toc326306770)

[Figuur 5.9: Concurrency probleem bij ophalen werkvoorraad 51](#_Toc326306771)

[Figuur 5.10: Concurrency probleem bij ophalen van werkvoorraad opgelost 52](#_Toc326306772)

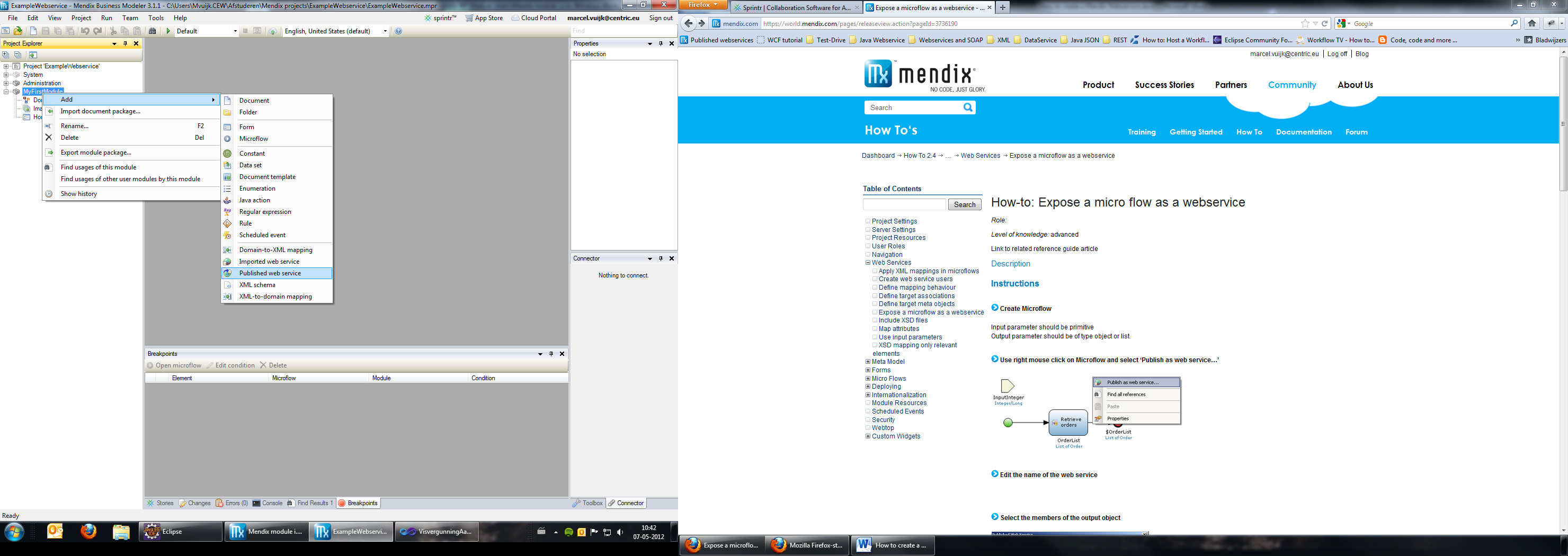
[Figuur 5.11: Mendix webservice geïmporteerd in een workflow 55](#_Toc326306773)

# Bijlages

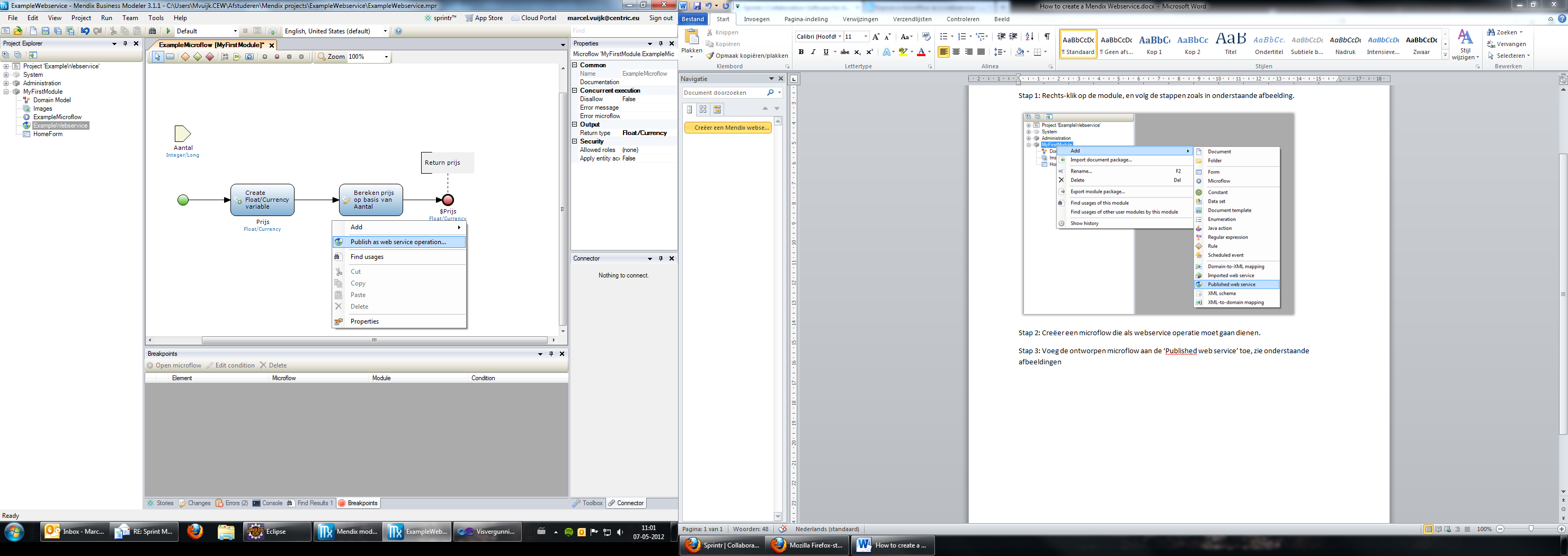
Hier volgen de internet bijlages horend bij dit rapport.

## Bijlage A: Creëren van een webservice in de Mendix Business Modeler

Onderstaande tutorial is gebaseerd op Mendix 3.x.

Stap 1: Rechts-klik op de module, en volg de stappen zoals in onderstaande afbeelding.  


Stap 2: Creëer een microflow die als webservice operatie moet gaan dienen.

Stap 3: Voeg de ontworpen microflow als webservice operatie aan de ‘Published web service’ toe. Open de microflow, rechts-klik en selecteer: ‘Publish as web service operation…’. Selecteer vervolgens de aangemaakte ‘Published web service’  


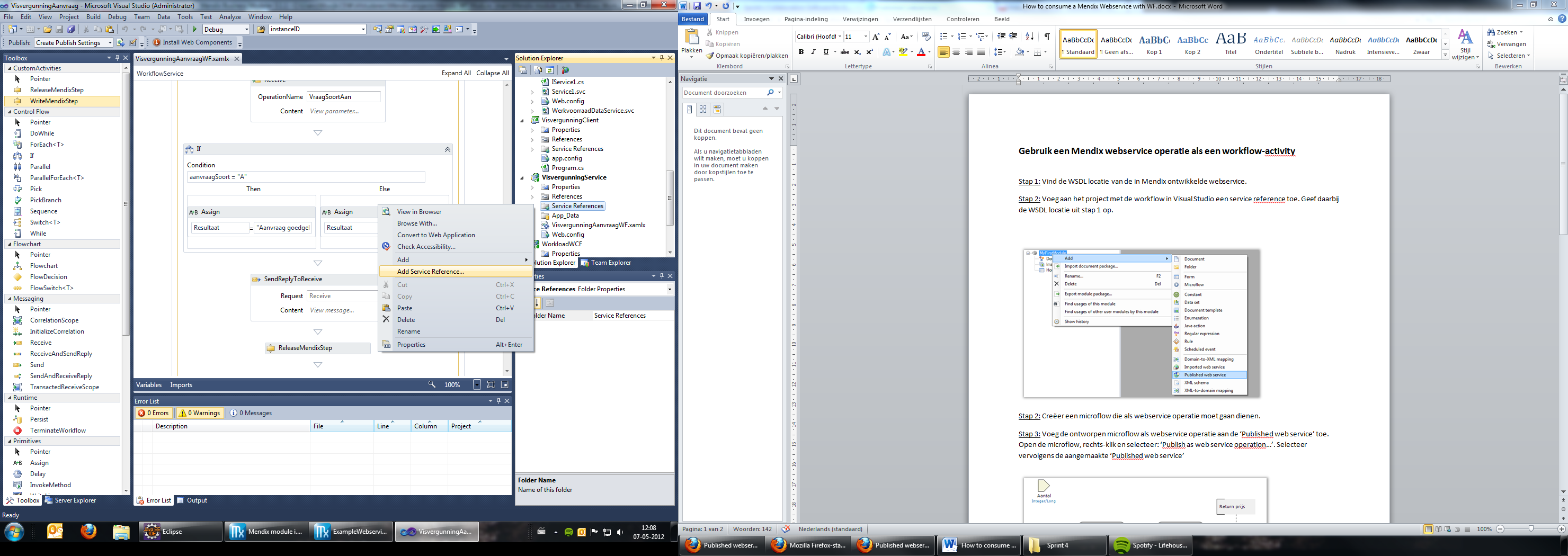
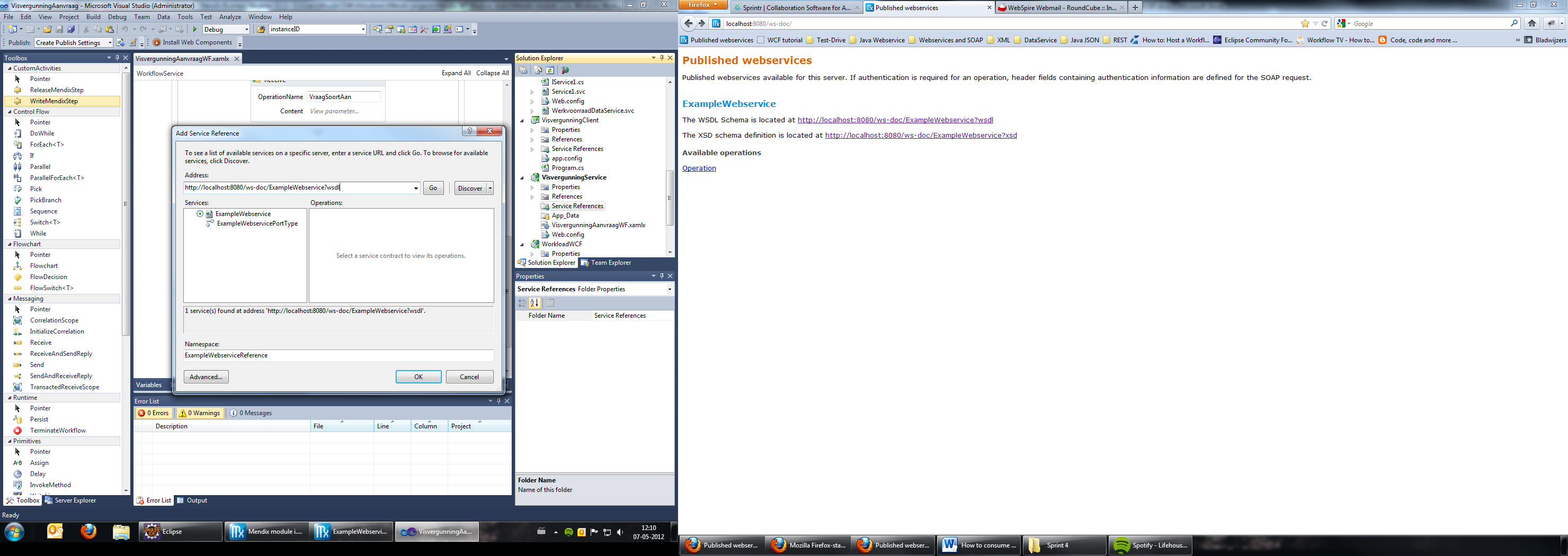
Stap 4: Geef de naam van de webservice operatie, namespace en eventueel documentatie op, klik ok en sla het project op. De web service is nu aangemaakt en heeft één operatie, namelijk de geselecteerde microflow. Informatie over en de webservice is te vinden via: <http://url:8080/ws-doc/>.

Om de webservice te kunnen aanroepen vanuit andere applicaties moet in het Mendix systeem een webservice-user aangemaakt worden die voor de authenticatie header gebruik kan worden.

## Bijlage B: Gebruik een Mendix webservice in een workflow

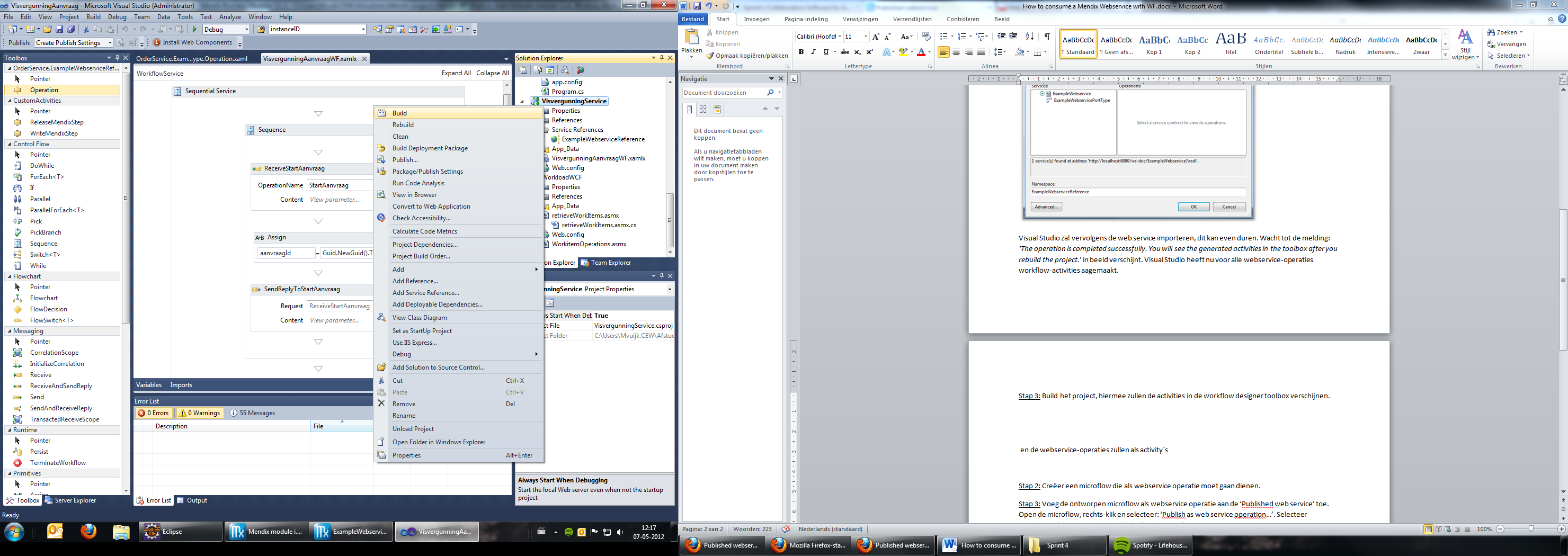
Stap 1: Vind de WSDL locatie van de in Mendix ontwikkelde web service.

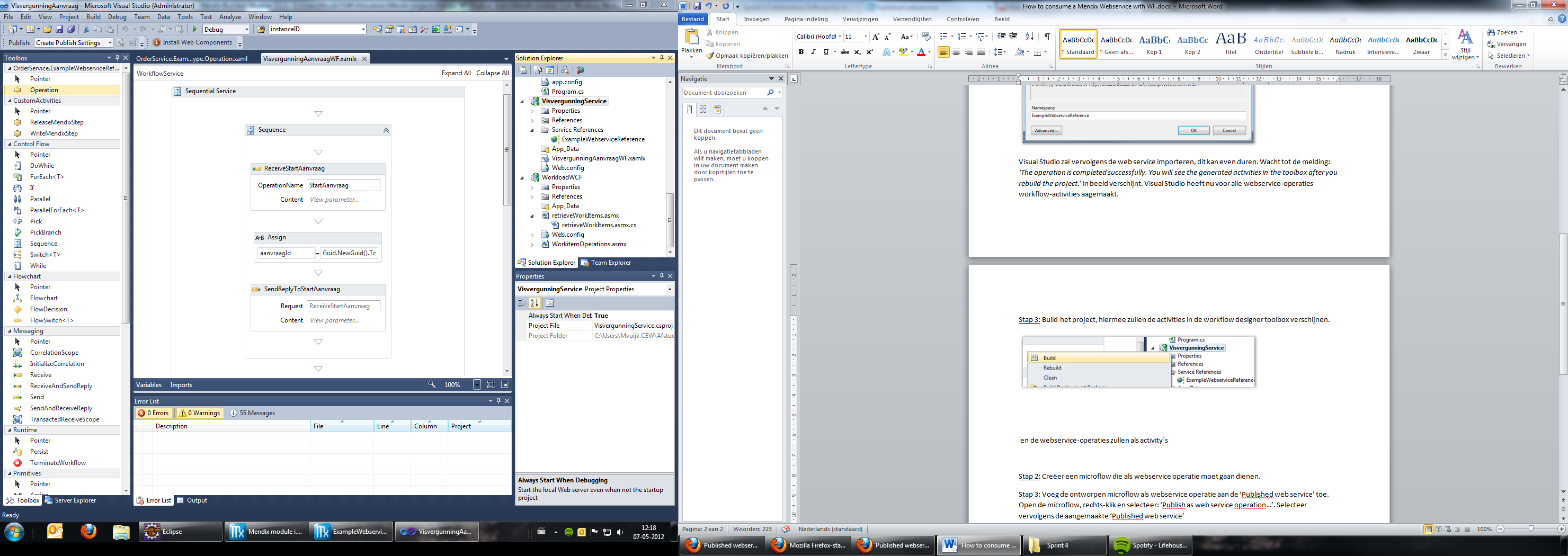
Stap 2: Voeg aan het project met de workflow in Visual Studio een Service Reference toe d.m.v ‘Add Service Reference..’. Geef vervolgens de WSDL locatie uit stap 1 op en klik op ‘Go’ en vervolgens ‘OK’.

Visual Studio zal vervolgens de web service importeren, dit kan even duren. Wacht tot de melding: *‘The operation is completed successfully. You will see the generated activities in the toolbox after you rebuild the project.*’ in beeld verschijnt. Visual Studio heeft nu voor alle webservice-operaties workflow-activities aagemaakt.

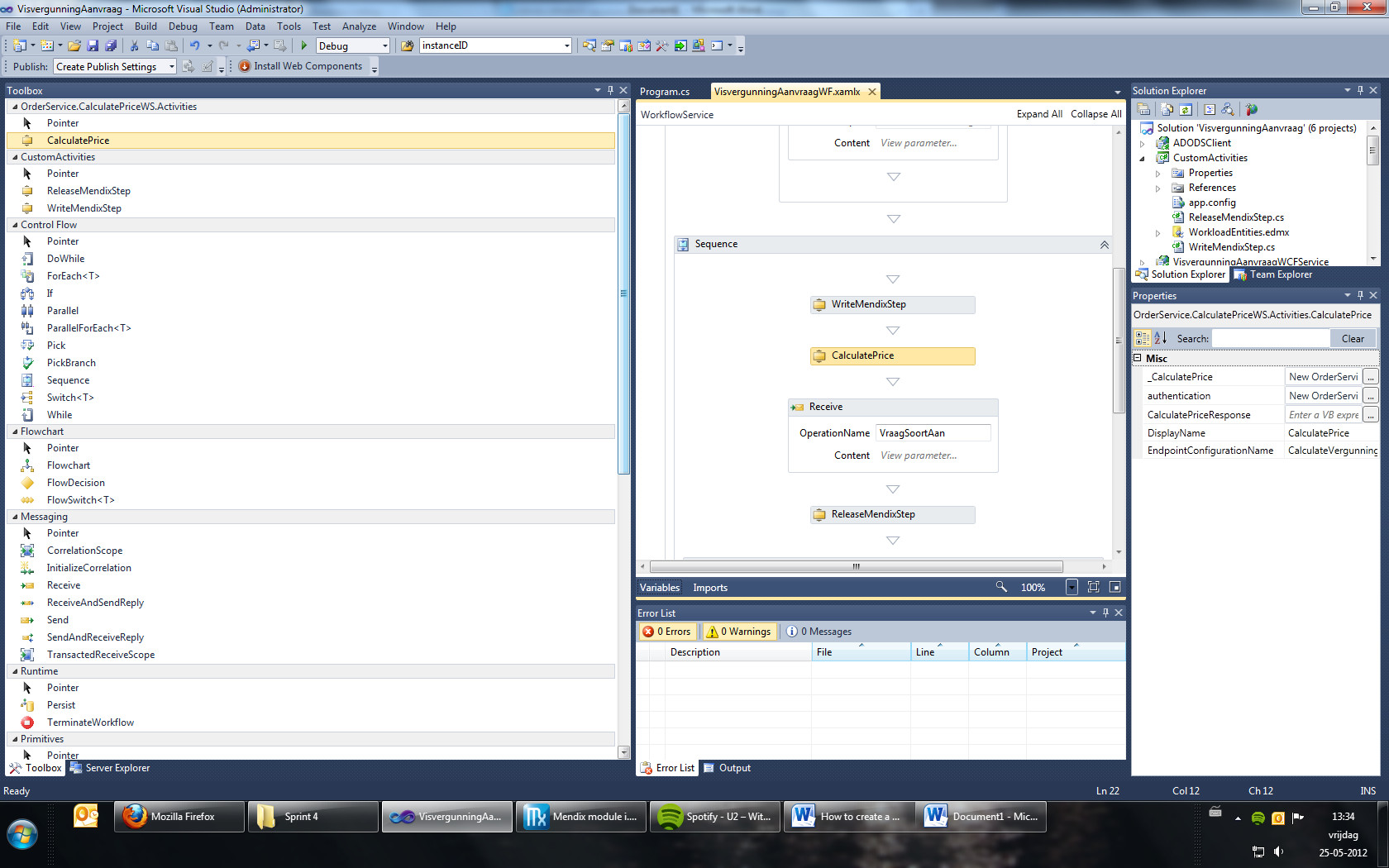
Stap 3: Build het project, hiermee zullen de activities in de workflow designer toolbox verschijnen.





Zoals hierboven te zien is, is de webservice-operatie als een activty in de toolbox verschenen.

Stap 4: Sleep de webservice-operatie activity in een workflow sequence, en open de properties van de activity. Hier kunnen middels Visual Basic expressies de parameters ingevuld worden.



De parameters en authenticatie headers van de webservice-operatie kunnen niet direct opgegeven worden in de properties, maar moeten met behulp van Visual Basic in een object aangeroepen worden. In bovenstaand geval wordt dus een nieuwe ‘CalculatePrice’ gemaakt met in de aanroep de parameters die de workflow verwacht. Voor de authenticatie moet een nieuwe ‘authentication’ aangemaakt worden met de username en password als gegevens. Hieronder een voorbeeld expressie in Visual Basic:  
 New PackageNaam.WebserviceReference.CalculatePrice() With {.WebserviceParameter = #}

Als alle properties correct zijn ingevuld wordt de webservice aangeroepen zodra de workflow bij de betreffende activiteit aankomt.

Zoals in bovenstaande afbeelding ook te zien is heeft de operatie ook een ‘CalculatePriceResponse’ property, door het resultaat van die functie aan een variabele te koppelen kunnen zo ook resultaten uit de webservice aanroep verwerkt worden.

# Externe bijlages

De volgende externe bijlages zijn aan het dossier toegevoegd:

**Project Initiatie Document**

Dit is het project initiatie document dat opgesteld is om de aanpak en requirements van de opdracht vast te leggen.

**Architectual Description**

Dit document beschrijft aan de hand van verschillende views de software architectuur voor het systeem.

**Mastertestplan**

Het mastertestplan beschrijft de algemene testaanpak, testtechnieken en testtools die gebruikt gaan worden.

**FAT detailtestplan**Het FAT detailtestplan beschrijft de uitgevoerde functionele acceptatie tests en de resultaten van die tests.

1. Een Mendix module is een stuk applicatie ontwikkelt in de Mendix Business Modeler dat als een losse module geïmporteerd kan worden in projecten. [↑](#footnote-ref-1)
2. Zie ook: <http://nl.wikipedia.org/wiki/Programmastructuur-diagram> [↑](#footnote-ref-2)
3. Zie ook: <http://geeks.netindonesia.net/blogs/jimmy/WindowsLiveWriter/K2blackpearlLessonLearntPartDeux_CEEC/image_2.png> [↑](#footnote-ref-3)
4. Bron: <http://i.msdn.microsoft.com/dynimg/IC295131.jpg> [↑](#footnote-ref-4)
5. Zie ook: <http://nl.wikipedia.org/wiki/Webservice> [↑](#footnote-ref-5)
6. Zie ook: hoofdstuk 4.2, de verschillende rollen binnen het projectteam. [↑](#footnote-ref-6)
7. Zie ook: <http://nl.wikipedia.org/wiki/Agile-software-ontwikkeling> [↑](#footnote-ref-7)
8. Zie ook: <http://nl.wikipedia.org/wiki/Scrum_%28softwareontwikkelmethode%29> [↑](#footnote-ref-8)
9. Bron: <http://nl.wikipedia.org/wiki/Scrum_%28softwareontwikkelmethode%29> [↑](#footnote-ref-9)
10. Zie ook: <http://www.sprintr.com/> [↑](#footnote-ref-10)
11. Zie ook: <http://nl.wikipedia.org/wiki/SOAP> [↑](#footnote-ref-11)
12. Een viewpoint is een bepaald oogpunt van waaruit de software architectuur benaderd wordt. [↑](#footnote-ref-12)
13. Zie ook: <http://nl.wikipedia.org/wiki/Simple_API_for_XML> [↑](#footnote-ref-13)
14. Zie ook hoofdstuk 2.3 waarin Windows Workflow Foundation en workflows nader worden uitgelegd. [↑](#footnote-ref-14)
15. Zie ook: http://en.wikipedia.org/wiki/JUnit [↑](#footnote-ref-15)
16. De WriteMendixStap is de eigen ontwikkelde workflow activity die de mogelijkheid biedt data naar de Werkvoorraad database weg te schrijven. Zie ook hoofdstuk 5.2. [↑](#footnote-ref-16)
17. Het Mendix Competence Center is een bijeenkomst binnen Centric waarin de belangrijkste ontwikkelingen van Mendix worden gepresenteerd. [↑](#footnote-ref-17)