https://www.sogeti.nl/sites/all/themes/sogeti/logo-screen.png

06-Jun-14

Joël Harkes

Sogeti BV

Afstudeerdossier

Ontwikkeling .Net package naar behoefte van Sogeti Developers

Gegevens

|  |  |
| --- | --- |
| Afstudeerblok | 2014-1.1 - 2014-1.2 |
| Startdatum | 10 februari 2014 |
| Inleverdatum | 6 juni 2014 |
|  |  |
| Studentnummer | 10009477 |
| Achternaam | dhr. Harkes |
| Voorletters | J I |
| Roepnaam | Joël |
| Adres | Abraham van Beyernstraat 103 |
| Postcode | 2525 TC |
| Woonplaats | Den Haag |
| Privé emailadres | [jlharkes@gmail.com](mailto:jlharkes@gmail.com) |
|  |  |
| Opleiding | Informatica |
| Locatie | Den Haag |
| Variant | Voltijd |
|  |  |
| Naam studieloopbaanbegeleider | A. Wieman |
| Naam begeleidend examinator | A.M.J.J. Lousberg |
| Naam tweede examinator | G.M. Tuk |
|  |  |
| Naam bedrijf | Sogeti |
| Afdeling bedrijf | Stage Unit |
| Bezoekadres bedrijf | Plotterweg 31-33 |
| Postcode bezoekadres | 3821 BB |
| Postbusnummer | 76 |
| Postcode postbusnummer | 4130EB Vianen |
| Plaats | Amersfoort |
| Internetsite bedrijf | [https//www.sogeti.nl/](https://www.sogeti.nl/) |
|  |  |
| Achternaam opdrachtgever | dhr. Verhaaf |
|  |  |
| Achternaam bedrijfsmentor | mw. Vernooij |

Voorwoord

Het bedrijf Sogeti heb ik leren kennen via de Bedrijvenbeurs HBO ICT Job Event in Rotterdam. Na mijn stage in een klein bedrijf, werd ik benieuwd welke mogelijkheden een groot bedrijf zoals Sogeti mij te bieden had en wat ik hen kon bieden. Ik zag mijn afstudeerperiode als een mooie mogelijkheid om te ontdekken of een groot ICT bedrijf iets voor mij zou kunnen zijn om later te gaan werken.

Als eerste wil ik Sogeti bedanken voor de opdracht die ik bij hen kon uitvoeren. Doordat het bedrijf mij flexibele werktijden bood kon ik naast mijn bestuurstaak als Ab Actis (secretaris) van de Navigators Studentenvereniging Den Haag ook beginnen aan mijn afstudeertraject. Het is voor mij een erg drukke periode geweest, maar ik kijk terug op een periode waar ik veel geleerd heb. Ik ben dankbaar dat ik in mijn expertise kon verbreden door mij te verdiepen in ontwikkeltalen technieken en tools van Microsoft.

Verder wil ik Bert, de andere stagiair bij Sogeti op de Microsoft afdeling, met wie ik kon sparren en terecht kon met vragen, bedanken.

Ook wil ik mijn medebestuursleden van de Navigators Studentenvereniging Den Haag bedanken. Ze hadden begrip voor dat ik in een drukkere periode kwam en namen taken over die ik niet meer kon uitvoeren. Bedankt voor de vriendschap en de motiverende woorden, die hielpen om het traject goed af te ronden.

Door de afstudeerperiode ben ik enthousiast geworden over de mogelijkheden van de Microsoft producten. Ze zijn goed doorontwikkeld en bieden veel functionaliteit. Wat voor mij de Microsoft producten uniek maakt, is de gebruiksvriendelijkheid. Om de producten te gebruiken hoeft weinig geconfigureerd te worden, terwijl er wel heel veen configuratieopties zijn. Daarnaast is voor alles wel een tool die de ontwikkelaar kan gebruiken, als hij niet de commandline wil gebruiken.

Ik heb veel geleerd en goede keuzes gemaakt en ben daarom erg tevreden over mijn afstudeertraject bij Sogeti.

Inhoudsopgave

[1 Inleiding 1](#_Toc389805645)

[2 Bedrijfscontext 2](#_Toc389805646)

[3 Plan van aanpak 4](#_Toc389805647)

[4 Onderzoek 8](#_Toc389805648)

[5 Package omschrijving 13](#_Toc389805649)

[6 Expertise verwerving .Net 16](#_Toc389805650)

[7 Requirementsfase 21](#_Toc389805651)

[8 Ontwikkeling prototype 23](#_Toc389805652)

[9 Ontwerpfase 35](#_Toc389805653)

[10 Ontwikkelfase 40](#_Toc389805654)

[11 Testfase 46](#_Toc389805655)

[12 Distributiefase 50](#_Toc389805656)

[13 Procesevaluatie 51](#_Toc389805657)

[14 Evaluatie beroepstaken 55](#_Toc389805658)

[15 Verklarende woordenlijst 58](#_Toc389805659)

[Bijlage A Initiële Planning Ganttr 59](#_Toc389805660)

[Bijlage B Onderzoeksplan 63](#_Toc389805661)

[Bijlage C Onderzoeksrapport 66](#_Toc389805662)

[Bijlage D Klassendiagram versie1 77](#_Toc389805663)

[Bijlage E Klassendiagram versie2 80](#_Toc389805664)

[Bijlage F Sequentiediagram FilterProvider 83](#_Toc389805665)

[Bijlage G Testrapport derde iteratie 84](#_Toc389805666)

[Bijlage H Goedgekeurde afstudeeropdracht 90](#_Toc389805667)

[Bijlage I Formulier bespreking concept afstudeerdossier 93](#_Toc389805668)

[Bijlage J Formulier tussentijds assessment 95](#_Toc389805669)

# Inleiding

Het eindverslag is geschreven in het kader van mijn opleiding, Informatica aan de Haagse Hogeschool. De afstudeeropdracht is uitgevoerd bij het bedrijf Sogeti. Ik voerde deze opdracht uit binnen de afdeling ANT Microsoft.

Dit verslag beschrijft het proces van het project dat ik deed in mijn afstudeerperiode. Het verslag is geschreven voor de examinatoren om inzicht te geven in het proces en de gemaakte keuzes, om zo een beeld te geven van de diepgang van de opdracht en de behaalde competenties.

Het verslag bevat een omschrijving van de bedrijfscontext waarin gewerkt is en de opzet van het plan van aanpak. Daarna bevat het een verslag van de uitgevoerde werkzaamheden en de gemaakte keuzes. Dit is opgedeeld in de fases waarin het project is uitgevoerd.

Verder bevat het verslag onder het hoofdstuk ‘Expertise verwerving .Net’ een uitleg over het gebruikte framework. Wanneer u expertise heeft in het .Net framework heeft, zou u het hoofdstuk kunnen overslaan. In het hoofdstuk wordt onder andere uitgelegd wat annotaties zijn. Deze kennis zal helpen bij het begrijpen van de rest van het verslag. Ik raad u dan ook aan dit hoofdstuk te lezen.

Bij het hoofdstuk ‘Ontwerpfase’ en ‘Ontwikkelfase’ is het ontwerp gemaakt dat te vinden is als in de bijlage als: Bijlage E Klassendiagram versie2. In dit hoofdstuk heb ik geprobeerd zo duidelijk mogelijk uit te leggen hoe het ontwerp is opgezet. In het hoofdstuk zelf zijn onderdelen van het diagram opgenomen ter verduidelijking. Voor het volledige overzicht kunt u deze bijlage raadplegen.

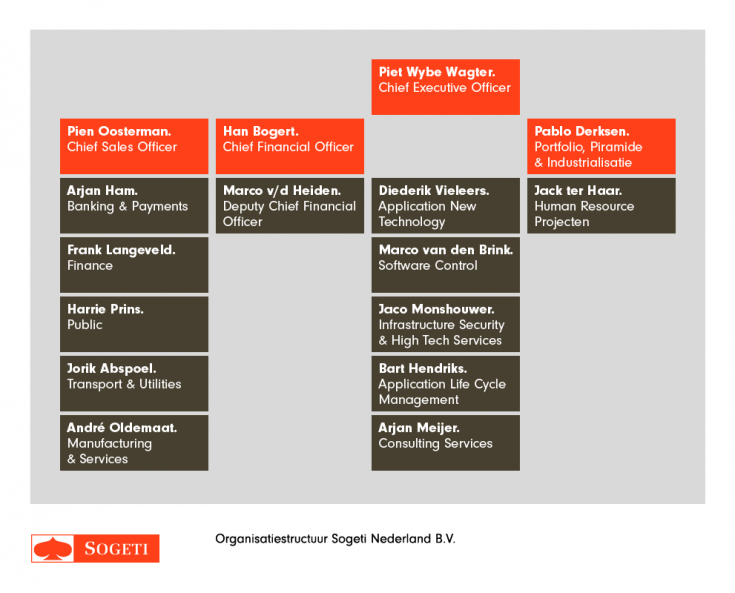
# Bedrijfscontext

Sogeti is opgericht in oktober 1967 door Serge Kampf te Grenoble (Frankrijk). In 1973 neemt Sogeti een meerderheidsbelang in het Europese bedrijf CAP. In 1974 wordt het Amerikaanse bedrijf Gemini Computer Systems overgenomen.

In 1975 worden de drie bedrijven samengevoegd tot CAP Gemini Sogeti. 1996 is het jaar waarin de naam versimpeld wordt naar Capgemini. In 2002 wordt de merknaam Sogeti opnieuw in de markt gezet door Capgemini. Sogeti is binnen de Benelux sinds 2002 een zusterbedrijf van Capgemini.

In Nederland is op 1 augustus 2002 Sogeti Nederland B.V. ontstaan door fusie met de bedrijven IQUIP Informatica B.V. (IQUIP), Gimbrère en Dohmen Software B.V. (G&D) en Twinac Software B.V. (Twinsoft). Twee jaar later kwam op concernniveau de fusie tussen Sogeti en Transiciel tot stand. Dit betekende voor Sogeti Nederland dat de UCC Groep en Flow onderdeel gingen uitmaken van de organisatie. In 2010 nam Sogeti SAP-dienstverlener Uphantis over.

Binnen Sogeti Nederland B.V. is de Test Management Approach (TMap) softwaretestaanpak, de DYA architectuurmethodiek, de infrastructuur projectmethodiek InFraMe en de implementatieaanpak Regatta ontwikkeld en gepubliceerd. Het bedrijf heeft zijn Nederlandse hoofdvestiging in Vianen en heeft kantoren in Diemen, Groningen, Amersfoort en Maastricht.

Sogeti Nederland BV heeft ongeveer 3000 medewerkers in dienst. Kenmerkend voor Sogeti is haar organisatiemodel. De lijnorganisatie heeft een op expertises ingedeelde divisiestructuur. De verkoop en deliveryorganisatie zijn op basis van sectoren ingericht. 

Figuur 1 Ogranisatiestructuur van Sogeti Nederland B.V.

## Afdeling

Het afstudeertraject is gedaan op de afdeling ANT (Application New Technology) in de divisie Microsoft. De Afdeling ANT is opgedeeld in 4 divisies: Mobile, BI & Analytics, Microsoft en JAVA. Dit zijn vier losse afdelingen die individueel van elkaar functioneren.

In de divisie Microsoft worden opdrachten voor klanten gerealiseerd door gebruik van Microsoft oplossingen. Enkele gebruikte Microsoft oplossingen zijn asp.NET, Sharepoint en Azure. De afdeling bevindt zich in de vestiging in Amersfoort. De afdeling is ingericht met 30 flexplekken waar de werknemers werken die momenteel beschikbaar zijn. Dit betekent dat er per week verschillende collega’s op de afdeling kunnen zitten en dat er een grote pool aan diverse Microsoft expertise aanwezig is op de afdeling.

Naast dat op deze locatie de werknemers werken die op dat moment beschikbaar zijn, werken er mensen aan het doorontwikkelen van Microsoft producten.

Binnen Sogeti was ik ingeschreven in de afdeling: stage/Student unit. Hieronder vallen alle studenten die op dat moment stagelopen of afstuderen bij Sogeti.

## Werkwijze ANT

Sogeti is een ICT Detacheringsbedrijf. Dit betekent dat zij haar werknemers uitleent aan andere bedrijven. Wanneer een werknemer van Sogeti is ingezet op een opdracht dan werkt hij op de locatie van de derde partij. Wanneer de opdracht afgerond of gestopt is, wordt de werknemer weer ‘beschikbaar’. Wanneer een werknemer van Sogeti beschikbaar is, wordt hij ingezet op interne opdrachten en werkt hij op de locatie van Sogeti. Wanneer een opdracht beschikbaar wordt voor deze werknemer zal hij weer aan de slag gaan bij de derde partij. Externe opdracht gaan altijd voor interne opdrachten.

Daarnaast werken er op de afdeling mensen die ingezet zijn op het doorontwikkelen van de Microsoft producten. Dit is waarschijnlijk voor het samenwerkingscontract dat Sogeti heeft met Microsoft. Deze groep ontwikkelaars word ook wel de Centre of Excellence genoemd (CoE).

# Plan van aanpak

## Probleemstelling

Binnen het bedrijf Sogeti werken veel ontwikkelaars die gebruik maken van Microsoft technologieën. Het .Net framework is één van deze technologieën die veel gebruikt wordt voor ontwikkeling van web applicaties en services. Een van de grote krachten van het framework is de mogelijkheid van het downloaden en installeren van packages.

Een package kan niet alleen een code-bibliotheek of een interface naar andere applicaties (API) bevatten (zoals bij de meeste frameworks), maar bevat ook de mogelijkheid om code toe te voegen en te wijzigen van het project waarin de package geïnstalleerd wordt. Dit betekent dat de mogelijkheid wordt geboden niet alleen code(bibliotheken) toe te voegen maar ook om deze automatisch te integreren in het project.

De utopie is wanneer een ontwikkelaar een opdracht zou krijgen, hij de benodigde packages uitzoekt en installeert waarna de applicatie draait. Dit is helaas nog lang geen werkelijkheid. Bij het ontwikkelen van applicaties komt er nog steeds veel handwerk kijken. Er zijn veel mogelijke oorzaken te benoemen waarom we nog zo ver weg zijn van deze utopie. Het zou bijvoorbeeld zo kunnen zijn dat er simpelweg nog te weinig packages ontwikkeld zijn om het mogelijk te maken of dat de ontwikkelaars de juiste packages niet kunnen vinden.

## Doelstelling

De doelstelling van mijn project is om bij te dragen naar het streven van de realisatie van deze utopie. Concreet verwoord zal er een package ontwikkeld gaan worden waar behoefte aan is onder de ontwikkelaars.

Een uitgangspunt is dat één van de problemen van het niet behalen van de utopie is dat niet alle packages aanwezig zijn. Hiervoor is gekozen omdat uit ervaring blijkt dat er altijd wel iets is waarvoor nog geen package ontwikkeld is. Bij volwassen frameworks zijn de meest generieke onderdelen wel gedekt maar voor meer specifieke eisen moet (het) al snel handmatig ontwikkeld worden.

De andere reden waarom gekozen is te kijken naar welke packages nog niet ontwikkeld zijn, is het behalen van de beroepstaken. Ik had mijn volledige afstudeertraject kunnen onderzoeken waarom de bestaande packages niet optimaal gebruikt worden maar dit zou niet binnen beroepstaken vallen die vereist zijn vanuit de opleiding. Een van de gekozen beroepstaken voor dit project is het ontwikkelen van een applicatie/package. Ik met deze doelstelling waarborg ik, dat ik ook aan deze beroepstaak zal voldoen.

## Opdracht omschrijving

Om een package te ontwikkelen naar de behoefte van de ontwikkelaars zal allereerst onderzocht moeten worden welke behoefte er is. Uit dit onderzoek zal één idee geselecteerd worden waarvan binnen de afstudeerperiode een package ontwikkeld zal worden. Er zal bij deze selectie dus niet alleen gekeken moeten worden naar wat uit het onderzoek naar boven kwam, wat de grootste behoefte bevat maar ook of het haalbaar is dit idee uit te voeren in het gestelde afstudeertermijn.

Naast het ontwikkelen van de package zal het ook belangrijk zijn deze package beschikbaar te stellen, zodat deze gedownload kan worden en om daarnaast een ontwikkelomgeving op te stellen waarin deze package doorontwikkeld kan worden. Stel Microsoft updatet het .Net framework waardoor de package niet meer werkt, dan moet deze wel aangepast kunnen worden.

Binnen Sogeti wordt niet altijd de nieuwste versie van het .Net framework gebruikt in haar projecten. Verder wordt er in Sogeti gebruik gemaakt van een referentie architectuur genaamd SARA (Sogeti Application Reference Architecture). Met beide dient, wanneer nodig blijkt, rekening gehouden te worden bij het ontwikkelen van de package.

## Projectmanagement

Om het project in goede banen te leiden, is er gekozen het project te managen volgens de PRINCE2 (PRojects IN Controlled Environments, version 2) methodiek. Het is een methodiek ontwikkelt door de overheid van het Verenigd Koninkrijk. Het is een flexibele methodiek die veel handvatten biedt om het project in goede banen te lijnen.

Ik heb ervoor gekozen om voor dit project PRINCE2 te gebruiken omdat ik al ervaring had in het gebruik hiervan wat er voor zou zorgen dat er geen tijd nodig zou zijn een andere projectmanagement techniek te leren. Daarnaast is PRINCE2 erg flexibel. Dit betekent vrijheid om te kiezen voor dit project welke methoden uit de methodiek wel en niet gebruikt worden.

Voor dit project heb ik gekozen het traject op te delen in fases volgens de waterval methode om zo een vaste structuur/lijn in het project aan te brengen. Dit biedt voor mij goede handvatten om mee te werken, omdat ik vanuit mijzelf niet altijd gestructureerd werk. PRINCE2 biedt hierin handvatten om per fase structuur te houden.

### Gebruik PRINCE2

Als er in dit project gebruik gemaakt zou worden van alle handvatten en processen die PRINCE2 voorschrijft, zou er bijna geen tijd over zijn om inhoudelijk aan het project te kunnen werken. Daarom is het noodzakelijk een aantal processen en technieken te kiezen die voor dit project genoeg toegevoegde waarde leveren. Ik heb gekozen voor dit project de volgende documenten en processen te gebruiken:

* PID (Project Initation Document). Dit is het plan van aanpak en biedt overzicht en inzicht in het project.
* BC (Businesscase). De Businesscase bevat een project rechtvaardiging maar wat vooral van belang is zijn de benodigde middelen, tijdsduur, kosten en de nu al geconstateerde risico’s. Dit geeft een goede start van het project omdat duidelijk is vastgelegd binnen welke kaders het project zal (moeten) functioneren.
* Riskregister. Het riskregister worden alle geconstateerde risico’s genoteerd met mogelijke voorzorgsmaatregelen. Omdat het een project met een krappe planning (korte fases) is, is het van belang dat er goed ingespeeld wordt op mogelijke risico’s. Daarom is er voor gekozen in dit project actief gebruik te maken van risicomanagement.
* Binnen elke fase:
  + Highlight Report. Aan het eind van de fase wordt een Highlight Report geschreven. Dit bevat de uitgevoerde werkzaamheden en behaalde mijlpaalproducten. Daarnaast wordt er beschreven welke maatregelen er genomen moeten en moesten worden om onverwachtse tegenslagen te corrigeren. Het Highlight Report wordt onder andere gebruikt om de begeleider/opdrachtgever te informeren en daarnaast mogelijkheid te bieden om aanpassingen te maken in de planning. Omdat het van te voren nog niet vastgelegd is wat voor package ontwikkeld wordt, is de kans groot dat de planning wijzigt gedurende het project.

### Buiten de scope

Naast deze technieken biedt PRINCE2 nog veel meer technieken waarbij bij een project rekening gehouden hoort te worden. Zo kent PRINCE2 een grote hoeveelheid rollen. Deze dienen verdeeld te worden. Maar aangezien er maar slechts twee personen bij dit project betrokken zijn, de opdrachtgever/begeleider en ik, zou dit voor veel onduidelijkheid zorgen. Er moet namelijk telkens vanuit een andere rol gehandeld worden. Voor een project van deze omvang zou dit zorgen voor overbodige complexiteit. Bijvoorbeeld: het aanstellen van een Teammanager is redelijk overbodig wanneer het team bestaat uit één persoon.

Verder maakt PRINCE2 gebruik van managen bij exceptie. Zo zouden er per fase toleranties afgesproken moeten worden met betrekking tot tijd, geld, kwaliteit, scope, risico en baten. Wanneer hier een wijziging komt, moet hiervoor een RfC (Request for Change) ingediend worden, die dan afgehandeld moet worden. Omdat dit een stageperiode is, ga ik er niet vanuit dat alles in één keer vlekkeloos verloopt. In tegendeel, er zullen waarschijnlijk vaak aanpassingen gedaan moeten worden met betrekking tot de planning, scope etc. Dit zou betekenen dat er iedere keer een RfC aangemaakt zou moeten worden. Al zou dit wel zorgen voor een beter overzicht van het verloop van het proces, dit zou wel erg veel tijd kosten. Daarom heb ik gekozen dit niet mee te nemen in dit project en het daarentegen mee te nemen in het Highlight Report.

## Softwareontwikkelingsmethode

In de ontwikkelingsfase wordt gebruik gemaakt van sprints. Dit zijn wekelijkse fases. Dit houdt in dat er per week wordt gekeken wat die week gedaan kan worden en zal er gestreefd worden aan het eind van deze week een werkende versie van de package op te leveren. Om te bepalen of het ‘werkt’ worden de nieuwe functionaliteiten en de onderdelen die aangepast zijn, aan het eind van iedere sprint getest.

Op deze manier wordt er iteratief ontwikkeld. Het voordeel hiervan is dat er gedwongen wordt iedere keer een werkende versie af te leveren en te ontwikkelen in kleine stappen. Ervaring uit mijn stageperiode vorig jaar was dat ik te snel over ging naar het ontwikkelen van nieuwe onderdelen van de applicatie alvorens eerst een werkende versie van het vorige onderdeel op te leveren, wat ervoor zorgde dat op gegeven moment geen overzicht meer was wat wel werkte en wat nog niet. Dit hoop ik op deze manier te voorkomen.

De ideologie achter de SCRUM sprint biedt hier een goed methode. Omdat SCRUM zelf bedoeld is voor ontwikkelteams van ongeveer 10 man, is het volledig gebruik van SCRUM geen optie. Bijvoorbeeld een daily meeting met het projectteam is overbodig wanneer het projectteam bestaat uit één man.

In een sprint zal net als bij Scrum aan het begin bepaald worden welke userstories of requirements er binnen deze sprint uitgevoerd zullen worden. Deze zullen gedurende sprint doorgevoerd worden in de applicatie. Aan het eind van de sprint moet er een werkende versie zijn. Requirements die niet doorgevoerd konden worden, kunnen meegenomen worden naar de volgende sprint.

## Planning

Voor dit project heb ik de projectmanagementmethode PRINCE2 gekozen. Hierin wordt het project beheert per fase. Om gestructureerd te kunnen blijven werken, heb ik gekozen om het project op te delen in verschillende fases:

|  |  |  |
| --- | --- | --- |
| Tijdsduur | Fase | Omschrijving |
| 1 week | Initiatiefase | Opstart project, opstellen plan van aanpak. |
| 3 weken | Onderzoeksfase | Onderzoek doen naar behoefte van de ontwikkelaar. |
| 1 week | Requirementsfase | Wensen en eisen ophalen en verwerken. |
| 2 weken | Ontwerpfase | Uitwerken van wensen en eisen tot werkbaar ontwerp. |
| 5 weken | Ontwikkelfase | Iteratief ontwikkelen van de package. |
| 3 weken | Testfase | Testen van het gebruik van de package. |
| 1 week | Distributiefase | Beschikbaar stellen van de package en de broncode. |

Alvorens ik een package zal kunnen gaan ontwikkelen zal ik eerst moeten onderzoeken welke behoefte er speelt. Om ervoor te zorgen dat het onderzoek niet de volledige tijd in beslag neemt heb ik er voor gekozen dit te limiteren tot drie weken. De opvolgende fases zijn de fases die vele software ontwikkelmethodes kennen. De fases zijn overgenomen vanuit SDL (Microsoft Security Development Lifecycle).

Het ontwikkelproces zal volgens het waterval principe doorlopen worden. Dit betekent dat de fases eenmalig en na elkaar uitgevoerd worden. Eerst zullen de requirements opgehaald worden en de ontwerpen gemaakt worden, voordat er ontwikkeld wordt. Er is gekozen om een testfase in te plannen zodat er getest kan worden hoe gemakkelijk de package te implementeren is en in de verschillende (ontwikkel)omgevingen. De versies van .Net waarin de package goed werkt kunnen meegenomen worden in de distributie fase om ervoor te zorgen dat de package niet te downloaden is voor versies waarin de package niet werkt. In de distributiefase zal de package vrijgegeven worden zodat de ontwikkelaars van Sogeti deze kunnen downloaden en gebruiken. Daarnaast zal de broncode vrijgegeven worden binnen het bedrijf zodat het doorontwikkeld kan worden.

Voor het maken van de planning is gebruikt gemaakt van een Gannt-chart. Deze is te vinden in de bijlage, Bijlage A Initiële Planning Ganttr.

# Onderzoek

Om een package te kunnen ontwikkelen die aansluit bij de behoefte van de .Net ontwikkelaars binnen Sogeti zou eerst onderzocht moeten worden, welke behoefte er speelt op de afdeling. Om te kunnen bepalen wat deze ‘behoefte’ inhoudt en of deze behoefte relevant is, was er gekozen om de behoefte te meten aan de hoeveelheid tijd die het zou besparen, wanneer deze behoefte vervult wordt. Zo wordt de behoefte meetbaar. De hoofdvraag is als volgt geformuleerd:

*Op welk onderdeel kan een package ontwikkeld worden waarmee de .NET ontwikkelaars van Sogeti het meeste tijd gaan besparen in bouwen van .Net web applicaties?*

Het doel van dit onderzoek zou zijn, dat er een keuze gemaakt kon worden wat voor package ik zou gaan ontwikkelen. Deze keuze zal gemaakt worden op basis van: de haalbaarheid om de package te ontwikkelen in de afstudeer periode, of er genoeg diepgang mogelijk is op de beroepstaken en of de package relevant genoeg is voor de ontwikkelaars.

Om de hoofvraag te kunnen beantwoorden was er gekozen voor de volgende volgorde in werkwijze:

1. Onderzoeken welke en wat voor packages nu binnen de afdeling gebruikt worden.
2. Onderzoeken welke packages er gemist worden en welke ideeën ontwikkelaars hebben voor het ontwikkelen van een package.
3. Onderzoeken hoe relevant deze ideeën zijn. (Welke alternatieve zijn er? Hoeveel tijd kost het om te ontwikkelen? Hoe vaak zal de package gebruikt worden?)

Door allereerst te verdiepen in welke packages gebruikt worden op de afdeling kon achterhaald worden op welke gebieden er al packages bestaan die gebruikt worden. Dit zou een kader leveren waarin gewerkt kon worden. Dit zou ook voorkomen dat een idee voor een package verder uitgewerkt zou worden, terwijl andere ontwikkelaars al zeiden dat zij daarvoor een bestaande package gebruiken.

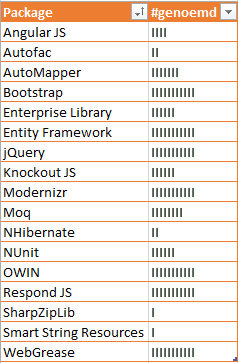
Er was gekozen de ontwikkelaars om ideeën te vragen voor packages omdat zij degene zijn die het uiteindelijk zullen moeten gebruiken. De ontwikkelde package zal aan moeten sluiten bij hun behoefte. Omdat ik zelf net het bedrijf binnenkwam, had ik geen ervaring wat er speelde binnen het bedrijf en kon ik dus niet zelf bepalen welke behoefte er speelde.

Door de ideeën verder uit te werken zou ik beter inzicht krijgen in de relevantie van het uitwerken van dat idee tot een package. In eerste instantie kan het een mooi idee lijken, maar achteraf kan dit toch tegenvallen doordat het bijvoorbeeld: uiteindelijk niet gebruikt wordt of niet haalbaar is om uit te werken in de afstudeerperiode. Ook kan het bijvoorbeeld zo zijn dat er al packages bestaan die in minder of meerdere mate hetzelfde doen. Om te voorkomen dat één van deze ideeën uitgewerkt zou worden werd in het onderzoek ook aandacht besteed aan het uitwerken van de ideeën. Hierin is gekeken naar mogelijke alternatieve packages, de tijd die het ontwikkelen zou kosten en de tijdsbesparing die het op zou leveren door onder andere te kijken in welke context de package nuttig is (en hoe vaak die context voorkomt). Dit leverde de volgende deelvragen op:

1. Welke onderdelen/packages worden er gebruikt door de .Net ontwikkelaars van Sogeti?
2. Welke ideeën hebben de .Net ontwikkelaars voor het ontwikkelen van de package? (Welke onderdelen missen de .NET ontwikkelaars nu in het ontwikkelen van hun projecten?)
3. Welke alternatieve packages zijn er beschikbaar voor de aangedragen ideeën?
4. Hoeveel tijd gaat het ontwikkelen van een package voor dit onderdeel uiteindelijk besparen?
   * Hoe vaak gaat de package gebruikt worden?
   * Hoeveel tijd gaat het per keer besparen?
5. Hoeveel tijd kost het ontwikkelen van de ideeën tot een package?

In de bijlage is het uitgevoerde onderzoek te vinden als: Bijlage B Onderzoeksplan en Bijlage C Onderzoeksrapport. In het onderzoeksplan is beschreven hoe het onderzoek uitgevoerd zal worden en met welke limieten en kaders rekening gehouden moet worden bij het onderzoek en welke keuzes hierin zijn gemaakt.

## Werkzaamheden

Tijdens het uitvoeren van het onderzoek werd tegen meer problemen aangelopen dan verwacht. Als eerste had ik de interviews gehouden volgens de structuur die opgesteld is in het plan van aanpak. Dit bood een goede structuur en zorgde ervoor dat de vragen gesteld zijn die ik in ieder geval gesteld wilde hebben. Uit de interviews kwam minder respons dan ik had verwacht. In Figuur 2 zijn de resultaten te zien van het turven van de gebruikte packages in het laatste project. Uiteindelijk konden er elf personen geïnterviewd worden binnen de afdeling. Deze elf personen dragen drie verschillende ideeën aan voor het ontwikkelen van een package. Dit was minder dan ik had verwacht maar dit leken op het eerste opzicht wel goede ideeën en meer ideeën verzamelen zou meer tijd kosten. Daarom besloot ik om door te gaan.

Figuur 2 Resultaat gebruik van NuGet packages

### Uitwerken packages

De lijst met ‘packages die gebruikt werden’ heb ik verder onderzocht door te kijken naar wat elke package bevatte aan functionaliteit. Dit deed ik omdat het voor een aantal packages niet direct duidelijk was wat ze inhielden. Om te achterhalen wat de packages deden heb ik via het packagemanagementsysteem, [NuGet.org](https://www.nuget.org/), de packages opgezocht. Via deze website kon de link naar de website van de package gevonden worden of een website met documentatie over de package. Zo werd achterhaald wat de packages zou toevoegen.

Daarnaast had ik de packages gecategoriseerd om een overzicht te krijgen wat voor soort packages gebruikt te worden. Opvallend hieruit was dat de meeste packages onder de categorie ‘programmeerhulp’ vielen. Dit waren de packages die niet meer functionaliteit toevoegde dan functionaliteit om efficiënter, eleganter of meer gestructureerd te kunnen programmeren. Dit is een package die voor de eind gebruiker van de web applicatie niet direct extra functionaliteit toevoegt maar zorgt voor extra functionaliteiten voor de ontwikkelaar om de web applicatie te ontwikkelen.

### Aangedragen Ideeën

Het meest opvallende van de interviews was voor mij, dat de ontwikkelaars maar weinig ideeën aandroegen voor het ontwikkelen van packages. Ik had verwacht dat elke ontwikkelaar toch wel een idee voor ontwikkelen van een package zou aandragen. Uit eigen ervaring weet ik dat ik bij elk project dat ik uitgevoerd heb een package kon bedenken die voor dat en meerdere projecten goed van toepassing zou zijn. Dit bleek hier dus niet het geval.

Uit de interviews werden drie ideeën aangedragen:

* Ontwikkelen van een module of service waarmee Word, PDF en HTML templates beheerd kunnen worden. Deze module zou dan bijvoorbeeld gebruikt moeten worden om HTML e-mails te versturen of Pdf’s te genereren.
* Een Uitbreiding schrijven op de Autorisatie in het .Net framework. De autorisatie in .Net is erg simplistisch, kan zorgen voor veel werk en mist overzicht.
* Een Cache-manager voor Windows Communication Foundation (WCF). WCF wordt gebruikt voor service georiënteerde applicaties. Dit betekent dat hiervoor performance erg belangrijk is. Er wordt gedacht dat er performancewinst behaald kan worden door processen of onderdelen van processen te cachen.

Voor deze ideeën had ik een korte omschrijving beschreven in het rapport om een duidelijk overzicht te geven welke ideeën zijn aangedragen en wat de bedoelingen waren van deze ideeën. Op basis hiervan heb ik in NuGet gezocht naar mogelijk vergelijkbare packages. Zou dit zo zijn, dan zou het overbodig zijn om hiervoor een package te ontwikkelen. Als het wiel al bestaat, hoeft deze niet opnieuw uitgevonden te worden. Voor de cachemanager van de WCF applicatie werd geen vergelijkbare packages gevonden. Voor beide andere packages werden wel packages gevonden, die mogelijk gebruikt konden worden als onderdeel bij het ontwikkelen van een nieuwe package. Voor autorisatie bleek wel een alternatief te zijn maar deze verving de volledige autorisatie en authenticatie van .Net.

### Bepalen relevantie van ontwikkeling

Om verder te bepalen hoe relevant het zou zijn om de package te ontwikkelen, had ik besloten te kijken naar:

* Hoe vaak de package gebruikt zal gaan worden.
* Hoeveel tijd het gebruik van deze package bespaart.
* Hoeveel tijd het ontwikkelen van de package zal kosten.

Het bleek lastiger dan gedacht om deze vragen te beantwoorden. Om een beter overzicht over de package te krijgen, leek het me verstandig om allereerst waarde van de kwaliteitseisen voor de packages vast te stellen. De kwaliteitseisen zijn de niet functionele eisen van elk systeem onder de ISO9126 norm[[1]](#footnote-1). Ik had hiervoor gekozen omdat dit me meer inzicht zou geven welke kwaliteitseisen van belang zouden zijn bij het ontwikkelen van de package. Hoe hoger het belang van de kwaliteitseisen, hoe belangrijker dat de package goed ontwikkeld wordt. En waarschijnlijk ook hoe belangrijker de package, omdat van deze package meer verwacht wordt. Achteraf gaven deze resultaten wel meer inzicht in welke kwaliteitseisen belangrijk zouden zijn in de packages, maar gaf het minder inzicht in de drie eerder opgestelde vragen.

### Hoeveelheid gebruik van de package

Om te kunnen bepalen hoe vaak de package gebruikt zou kunnen gaan worden, had ik gekeken naar de context waarin de package bruikbaar zou zijn. Dit had ik gedaan omdat ik zo kon achterhalen hoe vaak de packages gebruikt zouden kunnen worden. Door daarna te kijken naar de projecten die uitgevoerd worden en in hoeveel procent van deze projecten de package toegepast zou kunnen worden, kon dan bepaald worden hoe vaak de package gebruikt zou worden. De context bepalen van de package bleek geen probleem. Uiteraard zullen er waarschijnlijk altijd uitzonderingen zijn op de context, maar dat kon in dit stadium niet ingecalculeerd worden. Het lastige bleek het percentage achterhalen waarin deze context van toepassing was. Ik had verwacht gegevens op te kunnen vragen van projecten die uitgevoerd waren afgelopen jaar, maar dit bleek er niet te zijn of in ieder geval niet beschikbaar voor iemand van mijn functie binnen het bedrijf. De drie andere opties die ik had waren: gebruik van de interne CV applicatie om percentage te berekenen, zelf een percentage bepalen op basis van ervaring, percentage bepalen op basis van expertise van ontwikkelaars van de afdeling.

Als eerste leek mij de interne CV applicatie de beste oplossing. Hierin werden de projecten ook genoteerd met ‘tags’ die aan zouden moeten geven wat voor project uitgevoerd was. Het voordeel van het gebruik van de CV applicatie zou zijn dat de resultaten meetbaar en herhaalbaar zouden zijn. (althans: de Cv’s worden constant aangepast dus herhaalbaar op een later tijdstip zou wel andere resultaten opleveren). Op advies van de ontwikkelaars op de afdeling heb ik geen gebruik gemaakt van deze applicatie. Het bleek dat de tags bepaalde waar de ontwikkelaar ervaring in had. Op basis van deze ervaring werd de ontwikkelaar dan ingezet op nieuwe opdrachten. Hierdoor werden de tags vaak niet naar waarheid ingevuld: tags worden weggelaten waar de ontwikkelaar niet op ingezet wil worden en tags worden toegevoegd waar hij wel op ingezet wil worden.

Omdat mijn kennis binnen het bedrijf en de afdeling nog zeer beperkt was, leek het mij niet verstandig om zelf te bepalen hoe vaak de package gebruikt zou worden. Daarom had ik gekozen hierbij de expertise van de ontwikkelaars op de afdeling te gebruiken. Ik had een aantal van deze ontwikkelaars gevraagd wat zij dachten hoe vaak het gebruikt zou worden. Door hiervan het gemiddelde te nemen heb ik bepaald hoe vaak de applicatie gebruikt zou kunnen worden. Doordat het resultaat niet bepaald is op basis van cijfers had ik besloten geen percentage op te nemen in het onderzoeksrapport maar een indicatie op de schaal: altijd, vaak, gemiddeld, soms, zeldzaam.

### Tijdsbesparing

Voor de tijdsbesparing die de package op zou leveren kon alleen een schatting gedaan worden. Dit kwam als eerste doordat het op dit stadium niet vast stond welke functionaliteiten de packages zouden gaan bevatten en hoe deze geïmplementeerd zouden moeten. Daarnaast zou niet vastgesteld kunnen worden, hoe vaak deze functionaliteiten gebruikt zouden gaan worden. Daarom had ik besloten niet te richten om harde cijfers te geven op deze antwoorden, maar te kijken naar de voordelen en nadelen die het gebruik van de package zou moeten gaan leveren. Zo zou bijvoorbeeld het gebruik van de autorisatie package de tijd van het configureren van de autorisatierechten kunnen verlagen. Door na te denken en te beschrijven welke waarde het gebruik van de package zou kunnen hebben, kon al een beter beeld gevormd worden op de relevantie van de package.

### Ontwikkelingstijd package

Als laatste is bepaald hoeveel tijd het ontwikkelen van de packages zou gaan kosten. Bij dit hoofdstuk had ik op basis van mijn eigen expertise en de omschrijving van het idee (dat ontwikkeld moet worden tot package) een schatting gedaan naar de verwachte ontwikkeltijd van de package. Ik had ervoor gekozen dit te doen op basis van eigen ervaring omdat ik, door alle interviews en gesprekken zelf het beste overzicht had in hoeveel tijd het zou gaan kosten. Daarnaast wist ik zelf het beste welke ervaring en expertise ik zelf al had en dus of er veel ingewerkt zou moeten worden om de package te ontwikkelen. Alleen voor het ontwikkelen van een package voor de WCF applicatie vond ik het lastig om een schatting te geven omdat voor dit idee het nog totaal niet duidelijk was wat de praktische invulling zou (kunnen) gaan worden.

## Conclusie

Na het uitwerken van de deelvragen had ik geconcludeerd dat de meest geschikte package om te ontwikkelen, het idee was om de autorisatie van .Net Framework uit te breiden. De redenen dat ik heb gekozen voor deze packages zijn:

* Het idee is door meerdere ontwikkelaars aangedragen.
* Er is al een basis aanwezig in .Net. Dit betekent dat er een uitbreiding ontwikkeld kan worden als uitbreiding op de bestaande code. Zo wordt het mogelijke risico voorkomen dat het ontwikkelen van de eerste versie erg veel tijd kost.
* Het uitwerken van de package geeft veel diepgang (en persoonlijke uitdaging) doordat er veel in het framework .Net ‘gekeken’ zal moeten worden omdat de uitbreidingen goed aan zullen moeten sluiten op het framework.
* Er is voldoende ruimte om bezig te gaan met de beroepstaken (vanuit de opleiding).
* Autorisatie is een onderdeel dat bijna elke applicatie moet bevatten. Dit betekent dat het een redelijk algemeen applicatie onderdeel is.
  + De kans dat de package gebruikt gaat worden na mijn afstudeerperiode is relatief hoog.
  + De ervaring die ik op doe is erg nuttig.
* Het trok mijn persoonlijke interesse omdat autorisatie iets is wat bijna elke applicatie bevat en daarom een goede expertise is om in verdiept te hebben.

## Discussie

Wat ik persoonlijk jammer vond, was dat uit het onderzoek weinig meetbare resultaten kwamen. Cijfers en statistieken zijn resultaten die mij (en de meeste technische personen) vaak erg boeien. Het onderzoek had mij wel voldoende inzicht gegeven in welke package ik zou ontwikkelen. Daarmee is het onderzoek succesvol afgerond.

Doordat er een kwalitatief onderzoek is uitgevoerd op een relatief kleine steekproef, waren de resultaten niet betrouwbaar genoeg om dit te generaliseren naar de volledige populatie. Naar mijn verwachting zouden er bij het interviewen van de volledige populatie nog meer ideeën voor het ontwikkelen van packages naar boven komen. Voor mijn afstudeerperiode was het niet nodig de volledige wensen van de ontwikkelaars op dit gebied vast te leggen en boden de resultaten van dit onderzoek mij genoeg inzicht om een keuze te maken voor het ontwikkelen van een relevante package. Al zou het bedrijf gaan willen investeren in het ontwikkelen van interne package om te efficiëntie van de ontwikkelaars te verhogen, zou ik adviseren een grotere steekproef te interviewen. (Het door mij uitgevoerde onderzoek zou daarvoor onvoldoende input leveren).

Tijdens de interviews had ik bijgehouden hoe vaak de verschillende packages worden gebruikt. Dit had voor mijn onderzoek niet meer waarde dan te ontdekken wat de belangrijkste packages waren. Wel werd mij tijdens de interviews duidelijk dat ontwikkelaars vaak verschillende packages gebruikte. Dit zou bijvoorbeeld kunnen komen doordat er geen ideale kennisoverdracht is binnen de afdeling of gewoon door persoonlijke voorkeur. Hier zou verder onderzoek naar gedaan kunnen worden. Dit onderzoek zou kunnen leiden tot een onderzoek naar verbeteren van de kennisoverdracht op de afdeling en meer uniformiteit.

## Afronding

De onderzoeksfase is afgerond met het schrijven van het Highlight Report. De hoogtepunten uit deze fase waren de mijlpaalproducten: het onderzoeksplan en onderzoeksrapport. Verder werd in het Highlightsreport kort beschreven dat de onderzoeksfase een week was uitgelopen. Dit kwam onder andere doordat bleek, dat ik voor sommige packages niet voldoende kennis van .net had om te achterhalen wat de gebruikte packages inhielden. Daarnaast moesten voor sommige deelvragen alternatieve oplossingen verzonnen worden, omdat de geplande werkwijze niet mogelijk bleek te zijn.

Verder kwam ik tijdens het onderzoek tot de conclusie dat ik meer kennis en expertise van het framework nodig zou hebben om de package te kunnen ontwikkelen. Dit kwam doordat ik nog weinig inzicht had in welke functionaliteiten het framework bood en hoe autorisatie in het framework toegepast werd. Zonder deze kennis zou het onmogelijk zijn een uitbreiding hierop te ontwikkelen, laat staan ontwerpen. Daarom had ik besloten eerst een week in te plannen om mijzelf in te werken in het framework en de nodige kennis op te doen. Dit is ook meegenomen in het Highlight Report en een nieuwe planning is opgesteld voor de resterende afstudeerperiode.

Om een beter beeld te kunnen krijgen bij wat voor package er ontwikkeld zal gaan worden, is het volgende hoofdstuk gewijd aan de omschrijving van de ontwikkelde package.

# Package omschrijving

Autorisatie is het proces waardoor bepaald wordt of de persoon toegang heeft om een bepaalde actie uit te voeren. Door middel van autorisatie kan er bijvoorbeeld vastgesteld worden dat een administrator meer rechten heeft (en dus meer kan) dan een gewone gebruiker.

Autorisatie in .Net wordt gedaan door middel van rollen. In de database worden accounts en rollen opgeslagen. De rollen kunnen aan accounts gekoppeld worden. Een account kan meerdere rollen hebben en een rol kan aan meerdere accounts gekoppeld worden. Zo worden de rollen aan de accounts gekoppeld. Aan een actie (bijvoorbeeld: persoonsgegevens bekijken) kan bijvoorbeeld de rol Administrator gekoppeld worden. Dit zorgt ervoor dat alleen de gebruikers die ingelogd zijn met een account die gekoppeld zijn aan de rol ‘Administrator’ deze persoonsgegevens kunnen bekijken.

De rollen worden gekoppeld aan deze acties door middel van (Authorize) annotaties. Hoe dit werkt, wordt uitgelegd in hoofdstuk 6.2.2 en 6.2.3.

## Probleemstelling

Bij kleine applicaties die eenmalig opgezet worden zijn de huidige functionaliteiten van autorisatie meestal voldoende. Wanneer er grotere web-portals ontwikkeld moeten worden, mist het vaak aan functionaliteit en wordt het configureren van de autorisatie erg veel werk.

Het eerste probleem treedt op wanneer er rechten van een rol gewijzigd moeten worden. Als bijvoorbeeld blijkt dat het toch niet verstandig is dat alle ‘managers’ alle financiële gegevens van het bedrijf kunnen bekijken, moet dit aangepast worden. Dit moet dan gebeuren door de annotaties in de code aan te passen, waarna de code opnieuw gecompileerd en gedistribueerd moet worden. Het nadeel bij grote projecten is, dat er tussen de ontwikkel- en productie-omgeving vaak een testomgeving zit of is er met het bedrijf afgesproken dat er eens in de zoveel tijd een nieuwe versie uitgebracht worden. Het wachten op de nieuwe versie of het testen kan veel tijd en werk kosten. Dat allemaal voor het wijzigen van één of een paar recht voor een rol is erg omslachtig.

Een andere constatering die gedaan is, is dat het definiëren van de autorisatieconfiguratie (het vastleggen welke rechten, welk rollen hebben) exponentieel meer werk wordt bij grotere applicaties. Dit komt omdat, aan de ene kant het aantal acties die in de applicatie uitgevoerd worden groeit en aan de andere kant het aantal rollen in de applicatie meer wordt. Elk recht tot het uitvoeren van een actie die een rol heeft, moet los gedefinieerd worden. Dit leidt tot exponentiële groei van de configuratiemogelijkheden. Al zegt dit niet per definitie dat het aantal configuraties dat gedaan moet worden direct een kwadratisch verband heeft met de applicatiegrootte, vaak blijkt toch dat het in grotere applicaties veel meer werk kost om de autorisatie te configureren.

*Voorbeeld: Heeft een applicatie 4 acties en 4 rollen. Dan zijn er 4x4=16 autorisatie configuratiemogelijkheden. Zijn er 8 acties en 8 rollen dan zijn er 8x8=64 mogelijkheden. Een verdubbeling van de applicatiegrootte leidt tot een verviervoudiging van de autorisatie configuratiemogelijkheden.*

Er zijn nog veel meer kleinere nadelen aan de huidige implementatie van autorisatie. Eén die ik ook wil noemen, is de mogelijkheid om rechten terug te kunnen trekken. Een groep acties wordt bij elkaar gebundeld in controllers (zie hoofdstuk 6.2.1). Rollen kunnen in plaats van aan een actie gekoppeld te worden, ook aan de een controller gekoppeld worden. Dit zorgt ervoor dat alle acties in deze controller gekoppeld worden aan deze rol. Er is geen mogelijkheid om dan bij een aantal acties deze rechten terug te trekken. Is het dus zo, dat als een rol alle, behalve één of een paar, acties uit mag voeren, dan moet voor elke actie die de rol wél mag uitvoeren een configuratie gemaakt worden. Het zou veel configuratie werk kunnen schelen wanneer acties buitengesloten konden worden.

## Doelstelling

De doelstelling van deze package is om voor deze tekortkomingen, en de tekortkomingen die nog geconstateerd worden, een oplossing of verbetering te ontwikkelen. Tijdens de Requirementsfase zal er geprobeerd worden nog meer tekortkomingen te achterhalen en alle tekortkomingen om te zetten in mogelijke oplossingen. De nieuwe functionaliteiten, of wanneer nodig een volledig nieuwe systeemdelen, zullen dan gebundeld worden tot een NuGet package die ontwikkelaars dan kunnen downloaden en installeren in hun ontwikkelprojecten.

## Opdracht omschrijving

Het doel is om een package te ontwikkelen die de autorisatie uitbreidt. Tijdens het onderzoek kwam ik er achter dat, om dit te bewerkstelligen, ik meer expertise nodig moest hebben van het framework. Daarom had ik ervoor gekozen mij allereerst te gaan verdiepen in het framework op de gebieden die relevant zijn voor deze package.

Na het verdiepen in de huidige structuur zullen de wensen en eisen, die aangedragen werden tijdens de interviews, verwerkt worden tot requirements. Op basis van deze requirements zullen er diagrammen ontworpen worden van hoe de package eruit zal moeten komen te zien. Dit zal gedaan worden zodat er van te voren goed nagedacht wordt over de structuur en ingewikkelde processen (sequenties) van de applicatie.

Hierna zal de package iteratief ontwikkeld worden aan de hand van deze diagrammen die opgesteld zijn. Er is gekozen om de package te ontwikkelen voor de nieuwste versie van het .Net framework. Hiervoor is gekozen zodat de package ontwikkeld wordt voor de huidige structuur van het framework. Blijkt eventueel later dat de package niet werkt in oudere versies, dan zou dit altijd later nog opgelost kunnen worden (backward compatible gemaakt kunnen worden). Het voordeel hiervan, ten opzichte van de package ontwikkelen voor een oudere versie, is dat de package gebaseerd wordt op de meest moderne structuur. De kans is dan kleiner dat de package in de nieuwere versies van het framework niet meer werkt of dat er ‘hacks’ ingebouwd moeten worden, om het te laten werken. Omdat in de nieuwere versie van het framework veel meer functionaliteiten bevat, wil je stimuleren dat ontwikkelaars de meest recente versie gebruiken van het framework. Door de package voor de recente versie te ontwikkelen en waar nodig ‘hacks’ in te bouwen voor oudere versies, wordt dit gestimuleerd.

Tijdens het ontwikkelen van de package zal aan het eind van elke iteratie getest worden of de functionaliteiten goed werken. Na het ontwikkelen van de package zal er een testperiode in gegaan worden. In deze test periode zal getest worden voor welke versies van het .Net framework de package voldoende werkt en zal daarnaast getest worden of de package gemakkelijk geïmplementeerd kan worden in één van de huidige projecten van Sogeti.

Komen er uit deze tests grote problemen naar voren en is er tijd over dan zou er voor gekozen kunnen worden om deze problemen op te lossen. Zo niet, dan zal doorgegaan worden naar de distributiefase. In deze fase zal de package omgezet worden naar een NuGet package. NuGet is het package distributiesysteem van Microsoft. Door middel van NuGet kunnen packages gemakkelijk gedownload en (automatisch) geïnstalleerd worden in .Net projecten. Zo kan, door het omzetten van de package naar NuGet, de package gemakkelijk gedistribueerd worden. Deze package zal beschikbaar gesteld worden op een interne NuGet server bij Sogeti, zodat alle ontwikkelaars van Sogeti deze package kunnen gebruiken. Ook zal in deze fase de broncode beschikbaar gesteld worden en zal er een handleiding voor het gebruik van de package en een overdrachtsdocument, die de ontwikkelaar zal helpen met het doorontwikkelen van de package, geschreven worden.

# http://www.arvixe.com/images/landing_pages/net_framework_service_pack_1_hosting.pngExpertise verwerving .Net

Door een minor Game Design die ik gedaan heb op de Haagse Hogeschool, had ik al ervaring opgedaan met de programmeertaal C#. Daarom heb ik gekozen mij direct te verdiepen in het .Net framework. Ik heb gekozen dit te doen door video-tutorials. Omdat ik visueel ben ingesteld, begrijp ik dingen sneller door ze te zien dan door het lezen van een boek of handleiding.

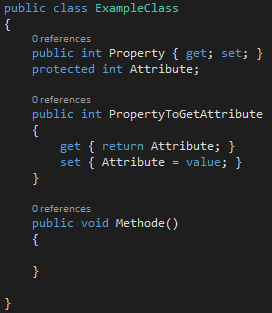
Deze video-tutorials heb ik bekeken vanaf de website: [channel9.msdn.com](file:///C:\Users\joel\AppData\Roaming\Microsoft\Word\channel9.msdn.com). Dit is een website met video’s over applicaties ontwikkelen met Microsoftproducten. Ik had niet zozeer deze website gekozen omdat dit nu eenmaal een goede website was maar omdat ik uit kwam op deze website bij het zoeken naar geschikte video tutorials. Om de tutorial beter te begrijpen en de kennis die geleerd wordt beter te kunnen onthouden, had ik gekozen ook dit direct toe te passen. Terwijl ik de tutorial bekeek heb ik mee gedaan door een MVC .Net project op te zetten en dezelfde stappen uit te voeren als werd laten zien in de video.

Ik had al een klein beetje ervaring met .Net door twee workshops, die gegeven waren in het major blok architectuur van mijn opleiding. Maar omdat deze kennis mij niet voldoende inzicht gaf in het framework en omdat de workshops uitgevoerd waren op een oudere versie van het .Net framework, had ik gekozen een tutorial te zoeken die tegelijkertijd de basis van het framework uitlegt en daarnaast een toelichting geeft van de nieuwe features in de laatste versie van het framework. Ik vond deze tutorial: <http://channel9.msdn.com/Series/Building-Web-Apps-with-ASP-NET-Jump-Start/Building-Web-Apps-with-ASPNET-Jump-Start-01-Whats-New-in-ASPNET-45>.

Ik had allereerst gekozen om mij breed te verdiepen in het .Net framework om zo een goed beeld en overzicht te krijgen van hoe .Net werkt en wat het framework bevat. Daarnaast had ik mij verdiept in hoe de autorisatie werkt in het .Net framework. Dit had ik gedaan om twee redenen:

* Door te begrijpen hoe de huidige autorisatie werkt kon ik beter begrijpen wat de zwakke kanten zijn en waarom het niet werkt.
* Door uit te zoeken hoe de autorisatie is opgebouwd, kon ik bepalen of het mogelijk was deze zwakke kanten ‘aan te passen’ door de structuur uit te breiden of dat de volledige autorisatie vervangen zou moeten worden.

## De programmeertaal

.Net is ontwikkeld op de Microsoft programmeertalen, waaronder C#. Het verschil van deze programmeertaal vergeleken bij andere talen (zoals JAVA en PHP) is dat een klasse niet alleen functies en attributen kan bevatten, maar ook ‘properties’. In JAVA is het gebruikelijk de attributen van een klasse te maskeren door gebruik te maken van Get en Set methodes. Hiervoor heeft C# properties.

Figuur 3 Voorbeeld van een klasse in C#

Properties worden gedefinieerd zoals te zien in figuur 4. Zoals te zien in het figuur kunnen properties gekoppeld worden aan attributen door de get en set in te vullen. Wordt dit niet gedaan dan maakt c# automatisch een attribuut aan voor de property bij het compileren. Dus achter de ‘Property’ zit een ook attribuut.

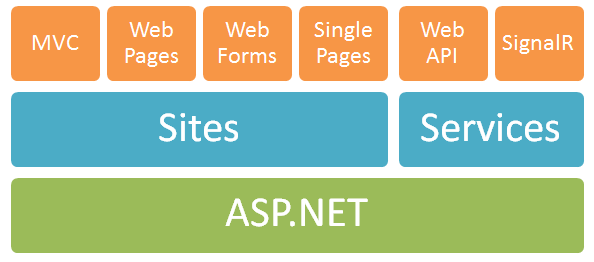
Een van de grootste nadelen van C# (en .Net) is dat het (op dit moment) alleen op Windows apparaten draait.

## http://upload.wikimedia.org/wikipedia/commons/thumb/d/d3/DotNet.svg/900px-DotNet.svg.pngHet framework

Figuur 4 Geschiedenis van ontwikkeling .Net framework

Het .Net framework bestaat sinds 2000 mei en is nu door ontwikkeld tot versie 4.5.1. in Figuur 4 is te zien welke nieuwe functionaliteiten bij welke versie ontwikkelt zijn.

Het .Net framework is een software framework dat draait alleen in Windows omgevingen. Het bevat een groot aantal codebibliotheken en programmeertaal interoperabiliteit ook wel bekend als CLR (Common Language Runtime). In deze bibliotheken zitten een aantal sjablonen voor het ontwikkelen van web applicaties en web services. De bekendste sjablonen zijn te zien in Figuur 5.

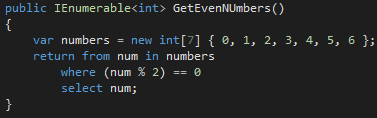


Figuur 5 Sjablonen in het .net framework

Waar in eerdere versies van het framework deze sjablonen los van elkaar gebruikt moesten worden, heeft Microsoft in de nieuwere versies van .Net, de sjablonen steeds dichter bij elkaar gebracht en mogelijkheden geboden een aantal sjablonen door elkaar te gebruiken.

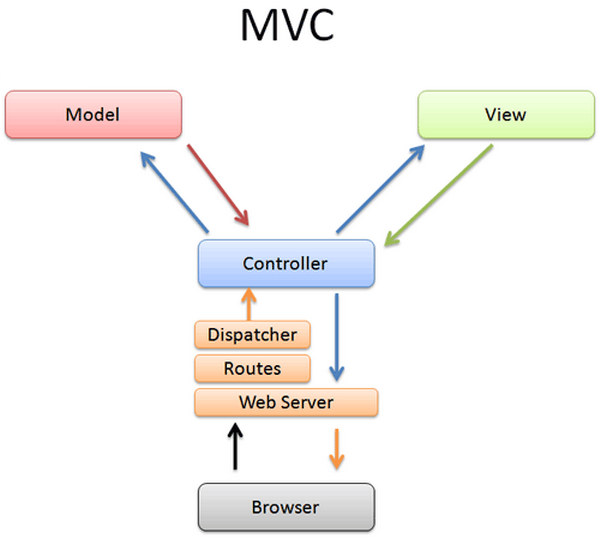
Het sjabloon waar ik mij op zal richten is het MVC sjabloon. Dit sjabloon wordt veel gebruikt voor het ontwikkelen van web-portals. Het biedt een simpele model-view-controller architectuur waarin de web-applicatie ontwikkeld kan worden. Ik heb gekozen voor het ontwikkelen en testen gebruik te maken van het MVC sjabloon omdat dit een simpel sjabloon is. Het is een handig sjabloon voor testen omdat het verder weinig tot geen configuratie vereist om te laten draaien en snel code geschreven kan worden om visueel te testen. Een andere mogelijkheid zou zijn om een web-service sjabloon te gebruiken. Ik had hier niet voor gekozen omdat het dan lastig wordt visueel te kunnen zien wat er gebeurd. Een web service geeft alleen maar opmaaktaal terug (zoals JSON of XML). De meeste andere sjablonen zijn uitgebreid door gebruik van externe bibliotheken (bijvoorbeeld Angular JS). Om deze sjablonen te gebruiken zou ik eerst kennis moeten verwerven van de externe bibliotheken. Ik kon op internet nies vinden waaruit ik af kon leiden hoeveel deze sjablonen werden gebruikt daarom had ik gekozen voor het MVC sjabloon omdat deze vanuit het bedrijf het meest gebruikt werd en geen overbodige complexiteit toevoegde.

Het voordeel dat .Net sjablonen dichter bij elkaar brengt is dat waarschijnlijk de autorisatie in verschillende sjablonen hetzelfde werkt. Dit betekent, dat wanneer de package ontwikkeld wordt voor een MVC sjabloon, er een grote kans is dat deze ook werkt in de andere sjablonen. Om hiervan zeker te zijn zal dit meegenomen moeten worden in de testfase.

Verder kent het framework talloze functionaliteiten. Eén van de grootste onderscheidingen die het framework maakt vergeleken bij andere frameworks, is de functionaliteit LINQ. LINQ betekent: ‘Language Integrated Query’. LINQ biedt de mogelijkheid om SQL query’s direct in de code te schrijven. Het kan gebruikt worden om data op te vragen of te veranderen. Data kan komen vanuit een database, direct vanuit de code (zoals in Figuur 6) of uit andere bronnen.

Figuur 6 Voorbeeld van LINQ om alle even nummers uit een reeks nummers te halen

### Model-View-Controller

Model view controller (MVC) is een bekende architectuur binnen de softwareontwikkeling die veel gebruikt wordt voor het ontwikkelen van applicaties. Op gebied van web-applicaties zijn er tegenwoordig al meerdere (en nieuwere) verschillende architecturen in omloop zoals PAC (presentation-abstraction-control) en MOVE (model-operation-view-event). Toch heeft Microsoft gekozen voor deze architectuur te ondersteunen.

Figuur 7 MVC structuur in .Net

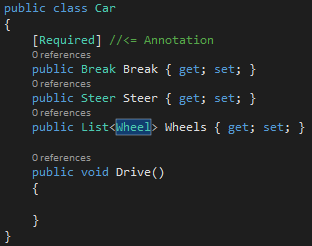
De MVC architectuur deelt de applicatie op in drie delen (zoals te zien in Figuur 7), met elk een eigen verantwoordelijkheid:

* Model: bevat de data van de applicatie (die vaak wordt opgeslagen in een database).
* View: bevat statische code van hoe de webpagina er uit komt te zien en gebruikt een model voor dynamische code.
* Controller: In komende actie afhandelen (bijvoorbeeld data van een model aanpassen). Het model maken, doorgeven aan de view en de view renderen terugsturen naar de gebruiker/browser.

Het nadeel van MVC is dat er vaak een overlap zit tussen de onderdelen en het niet precies duidelijk is wat er in de model, controller of view moet gebeuren. Een voorbeeld hiervan is de discussie of de database opslag moet gebeuren door de controller of het model of de kwestie of er in de view nog acties uitgevoerd mogen worden om gegevens uit de database te halen. Een opvallende conventie die Microsoft aanhoudt bij deze architectuur is dat één view altijd één model gebruikt om zijn view te genereren. Als de view meer data nodig heeft dient hiervoor een zogenaamde ‘ViewModel’ gemaakt te worden dat door de controller opgevuld en meegegeven wordt aan de view.

### Annotaties

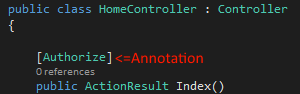
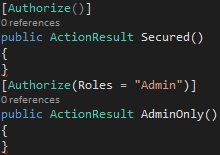
Een belangrijke feature in het .Net framework die ook gebruikt wordt voor de autorisatie is de mogelijkheid van het toekennen van Annotaties (Annotations) aan klasse, attributen en methode. Een annotatie is een soort van aantekening of metadata bij een klasse, attribuut of methode die extra functionaliteit definieert.

Stel: je hebt een website waar allemaal auto’s op hebt staan. Je hebt een klasse Auto waarvan alle objecten opgeslagen worden in de database. Je wilt verplichten dat een auto alleen maar opgeslagen wordt in de database wanneer de informatie volledig is. Je wilt vastleggen dat de informatie alleen volledig is, wanneer de Rem aanwezig is. Dit zou door de Controller gecontroleerd moeten worden. Maar bij veel verplichtingen zou dit snel oplopen tot onoverzichtelijke code. De andere optie is om een Annotatie toe te voegen bij de klasse Auto. .Net kent de ‘Required’ Annotatie. Deze zorgt ervoor, dat er eerst gecontroleerd wordt of de Rem aanwezig is, voordat een object van de klasse Auto opgeslagen wordt in de database. In Figuur 8 is te zien hoe dit in C# geprogrammeerd wordt.

Figuur 8 Gebruik van annotations in de klasse Car

Het gebruik van deze annotaties kent vele mogelijkheden. In .Net zelf zijn al vele mogelijkheden ingebouwd. Daarnaast is het zelf mogelijk eigen annotaties te schrijven.

### Autorisatie

In .Net gebeurt ook de autorisatie via annotaties. Door een Annotatie ‘Authorize’ toe te kennen aan een methode in een Controller, kan een actie afgeschermd worden tegen niet ingelogde gebruikers.

Figuur 9 Gebruik van annotaties voor autorisatie

Stel: je hebt een pagina die ‘Secured’ heet binnen je website die alleen ingelogde gebruikers mogen lezen, dan kun je door boven de Secured methode de annotatie ‘Authorize’ toe te voegen (zie Figuur 12), er voor zorgen dat deze pagina afgeschermd wordt. Wanneer een niet ingelogde gebruiker probeert naar deze pagina te gaan wordt hij doorgestuurd naar het inlog scherm.

Verder is er de mogelijkheid om aan deze Annotatie rollen toe te kennen. Door een rol mee te geven aan de autorisatie annotatie wordt er vastgelegd dat alleen een gebruiker met dezelfde rol deze actie uit mag voeren. Zoals te zien in Figuur 9: alleen gebruikers met de rol ‘Admin’ kunnen de actie ‘AdminOnly’ uitvoeren.

# Requirementsfase

In de onderzoeksfase is er een package gekozen en tijdens het inwerken had ik mij verdiept in het framework en hoe autorisatie is geïmplementeerd in het framework. Dit gaf mij al enig inzicht in wat er verwacht werd en op welke manieren dit geïmplementeerd zou kunnen worden, maar er waren verder nog geen concrete wensen en eisen opgesteld. Daarvoor was ik deze fase in gegaan om alle wensen en eisen boven water te halen, deze concreet formuleren, en prioriteren.

Om de wensen en eisen te achterhalen voor de package had ik gekozen om met de ontwikkelaars die dit idee aandroegen nog eens om de tafel te zitten om al hun wensen en eisen op tafel te krijgen. De wensen en eisen die hieruit naar boven kwamen, werden genoteerd in een spreadsheet. Deze lijst werd daarna geprioriteerd op basis van de MoSCoW prioritering. Dit leverde een bruikbare lijst met requirements.

Uit de interviews zelf waren al redelijk aantal wensen en eisen naar boven gekomen. Deze waren ook genoteerd en in deze lijst meegenomen. De requirements die uit de interviews naar boven waren gekomen waren vaak omschreven als voorbeeld van een probleem in de huidige structuur terwijl uit het gesprek al meer oplossingsgerichte wensen naar boven kwamen. Door voor iedere requirement te zoeken naar de achterliggende wens en deze zo te herschrijven konden vele requirements samen gevoegd worden. Zo werd bijvoorbeeld de requirement dat rollen rechten van andere rollen konden erven ook omschreven als dat het in het huidige systeem het probleem was dat er vaak voor acties veel meerdere rollen vastgelegd moesten worden of dat het rollen systeem een hiërarchie moest krijgen. Deze beide functionaliteiten zitten ook in de requirement dat rollen rechten van andere rollen kunnen erven. Daarom had ik ervoor gekozen deze duplicaten te verwijderen. Dit leverde een klein aantal concrete wensen en eisen op voor de autorisatie package. In Figuur 10 zijn deze wensen en eisen beknopt beschreven.

|  |  |  |
| --- | --- | --- |
| **Code** | **Omschrijving** | **Prioriteit** |
| F01 | Autorisatie rechten moeten dynamisch toegekend kunnen worden aan rollen | Must |
| F02 | Er moeten bedrijfsregels toegekend kunnen worden | Should |
| F03 | Aan rollen moeten rechten teruggetrokken kunnen worden. | Would |
| F04 | Rollen moeten rechten van andere rollen kunnen erven. | Should |
| F05 | De autorisatie moet uitgebreid kunnen worden. | Should |
| F06 | Een administrator moet permissies aan rollen toe kunnen kennen | Would |
| F07 | Een administrator moet rollen toe kunnen kennen aan gebruikers | Would |
| F08 | De autorisatie(configuratie) moet zichtbaar zijn in één overzicht of bestand. | Should |

Figuur 10 Wensen en eisen van de autorisatie package

## Conclusie

Na het opstellen van de requirements kwam ik er achter, dat ik voor een heel aantal van de wensen en eisen nog niet concreet kon bepalen hoe dit in het framework ingepast zou moeten worden. Ik had me wel al verdiept in hoe het framework werkte maar, wist daarin niet veel meer dan wat zich aan de oppervlakte van het framework bevond. Zo wist ik bijvoorbeeld dat er een UserStore gebruikt werd om gegevens van gebruikersaccounts op te halen en in te loggen, maar wist ik niet hoe het inloggen dan precies werkte of hoe de rollen van de gebruiker gecontroleerd werden met de rollen van de aangevraagde actie. Dit betekende dat ik nog geen overzicht had op welke onderdelen van het framework ik een uitbreiding zou moeten schrijven. Om toch door te kunnen gaan met het project had ik besloten om eerst een prototype te ontwikkelen, voordat ik door zou gaan naar de ontwerpfase.

Het schrijven van het prototype zou gericht zijn op het implementeren van een aantal van de belangrijkste requirements. Bij het ontwikkelen van het prototype zou ik mij niet richten op het bouwen in de juiste structuur. Wel werd er gekeken op welke manier de functionaliteit in het framework ingebouwd zou moeten worden. Zo kreeg ik expertise in hoe de functionaliteiten in het framework zouden moeten passen en kreeg ik ondergrondse kennis van het autorisatie deel van het framework. Er werd tijdens het ontwikkelen van het prototype niet gelet op het bouwen in de juiste structuur omdat hier tijdens de ontwerpfase aandacht aan besteed zou worden. Zo kon ik mij allereerst focussen op hoe en waar de functionaliteiten in het framework geplaatst zouden moeten worden. Daarna zou ik mij (in de ontwerpfase) kunnen richten op ‘wat hiervoor de juiste structuur zou zijn’. Hierna zou het prototype in de ontwikkelfase omgebouwd kunnen worden naar de nieuwe structuur. Naar verwachting zou het ombouwen niet veel tijd kosten. Daarom had ik gekozen om voor het ontwikkelen van het prototype drie weken in te plannen. Ik verwachte zoveel tijd nodig te hebben omdat er niet alleen ontwikkeld moest worden, maar ook verder ingewerkt moest worden in het framework.

# Ontwikkeling prototype

## Opzet

Het ontwikkelen van het prototype is gedaan aan de hand van de belangrijkste wensen en eisen uit de lijst met requirements. Ik had ervoor gekozen om dit iteratief uit te voeren en voor elke sprint één of een aantal requirements te nemen. De requirements die ik deze fase door wilde voeren waren:

* F01: Autorisatie rechten moeten dynamisch toegekend kunnen worden aan rollen.
* F02: Er moeten bedrijfsregels toegekend kunnen worden.
* F04: Rollen moeten rechten van andere rollen kunnen erven.
* F06: Een administrator moet permissies aan rollen toe kunnen kennen.
* F08: De autorisatie(configuratie) moet zichtbaar zijn in één overzicht of bestand.

Ik had voor deze requirements gekozen omdat: deze relatief hoge prioriteit hadden, relatief gezien het minste werk kosten om te implementeren en daarnaast voor uitgezocht moest worden hoe dit in het framework zou moeten passen. De ontwikkeling van het prototype had ik opgedeeld in vier sprints. Ik had voor vier sprints gekozen omdat ik bij het bedenken van de requirements vier verschillende functionaliteiten/onderdelen ontdekte, die zouden leiden tot het implementeren van deze requirements. De sprints had ik als volgt gepland:

|  |  |  |
| --- | --- | --- |
| Sprint | Functionaliteit | Requirements |
| 1 | Uitbreiding op de rollenstructuur zodat de rollen hiërarchisch aan elkaar gekoppeld kunnen worden | F04 |
| 2 | De configuratie van de autorisatie vast kunnen leggen in een configuratiebestand | F08, F06, (F01) |
| 3 | Een abstracte klasse die overgeërfd kan worden om een bedrijfsregels annotaties te maken. | F02 |
| 4 | De configuratie van de autorisatie vast kunnen leggen in de database. | F01, F06 |

### Testen

Om te testen of de functionaliteit die ingebouwd is werkt, was ervoor gekozen een MVC web-applicatie op te zetten in de nieuwste versie van het .Net framework. In dit project zouden dan de functionaliteiten ingebouwd worden en direct getest kunnen worden. Het testen zou gebeuren door een aantal controller acties aan te maken die de toegevoegde functionaliteiten testten. Bijvoorbeeld: een actie die altijd uitgevoerd mag worden en een actie die alleen uitgevoerd mag worden wanneer de gebruiker ingelogd is. Zo kon getest worden of dit daadwerkelijk ook het geval was bij de configuratie van autorisatie in het bestand of de database.

Na het ontwikkelen van het prototype kon ervoor gekozen te worden de onderdelen eruit te halen die nodig waren voor de package, of andersom: alles weghalen wat niet in de package zou moeten.

Ik had gekozen om geen gebruik te maken van unit tests omdat het veelal over kleine functionaliteiten gingen die veel sneller handmatig getest konden worden. Op deze manier zou ik ook direct visuele feedback krijgen. Voor het schrijven unit tests zou ik veel tijd kwijt zijn om mij in te werken. Daarnaast zouden er ook veel ‘mock’-objecten gemaakt moeten worden, wat het schrijven van unit tests veel werk zou maken. Al deze mock’-objecten bevonden zich al in het MVC sjabloon en zo kon op deze manier snel getest worden.

### .Net Framework inkijken

**Losse koppeling**

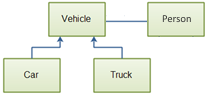
Losse koppeling houdt in dat een in de klasse alleen de abstracte functionaliteit gedefinieerd wordt en pas tijdens het draaien van de applicatie specifieke klasse toegekend wordt. Bijvoorbeeld Een klasse Persoon heeft een attribuut Voertuig. Tijdens het draaien van de applicatie kan aan een persoon een Auto of een Truck toegekend worden. 

Figure 11 Voorbeeld van losse koppeling

Het .Net framework heeft een uitgebreide codebibliotheek en bevat veel klassen die gebruikt en overgeërfd kunnen worden. Zo kan er snel ontwikkeld worden in een goede structuur. Om te zorgen dat het prototype goed aansluit op het framework zal veelvuldig gebruik gemaakt moeten worden van deze klassen. Daarnaast is het verspilde tijd om een onderdeel te schrijven wat al bestaat in de codebibliotheek. Het probleem waar ik al snel tegenaan liep was dat het framework niet volledig openbaar was. In het framework kon alleen de declaraties van de functies en attributen bekeken worden. Er kon niet gekeken worden hoe de functies geïmplementeerd werden. Op zich zou dit geen probleem zijn, wanneer slechts gebruik gemaakt moest worden van de functies. Er was namelijk genoeg documentatie aanwezig wat elke functie deed. Helaas voor het ontwikkelen van de functionaliteiten in het prototype zouden een aantal van deze klassen en functies aangepast of uitgebreid moeten worden.

Wanneer een functie uitgebreid moest worden, werd het wel zeer relevant om te kunnen zien hoe deze functies geïmplementeerd werd. Op basis hiervan kon bepaald worden of de functie slechts uitgebreid of volledig vervangen moest worden. Wat de declaratie van de code soms ook erg onduidelijk maakte was het veelvuldig gebruik van interfaces. Het voordeel van deze interfaces was dat alles erg losgekoppeld was en verwisseld kon worden. Het nadeel was dat het soms niet duidelijk was wat voor soort implementatie van deze interfaces verwacht werd. Soms werden namelijk erg abstracte interfaces gebruikt voor specifieke implementaties (bijvoorbeeld een IPrincipal interface voor het ophalen van de ingelogde gebruiker).

Tijdens mijn afstudeerperiode kwam er wel veel in het nieuws dat Microsoft haar frameworks en producten die ontwikkelaars veel gebruiken opensource aan het uitbrengen was. Sommige onderdelen waren al open source en telkens meer onderdelen werden uitgebracht met opensource licentie. Helaas kon ik destijds van het .Net framework geen opensource versie of broncode vinden.

Toen ik geen variant van de broncode kon vinden ben ik op zoek gegaan naar een andere oplossing. Ik ontdekte dat er een gratis applicatie was die codebibliotheken van C# kon decompileren. Dit maakte het mogelijk de code van deze bibliotheken in te kijken. Deze applicatie heette dotPeek. Door de .Net bibliotheken te in dotPeek te importeren en de juiste klasse op te zoeken kon ik toch de code ingekeken worden. Het nadeel van deze applicatie was dat het soms omslachtige code teruggaf en heel soms de code niet goed decompileerde, waardoor het soms er lastig werd te achterhalen, hoe de functie geïmplementeerd was. Maar in de meeste gevallen gaf dit direct genoeg inzicht om te kunnen bepalen of de functie uitgebreid of vervangen moest worden.

## Rollenstructuur

Als eerste had ik ervoor gekozen de functionaliteit van requirement F04 te implementeren in het prototype. Ik koos deze requirement omdat dit van één van de requirements was, waarvan ik al grotendeels wist hoe deze geïmplementeerd zou moeten worden en het framework.

In MVC .Net worden de rollen opgeslagen in de database. Daarnaast wordt in de database opgeslagen welke gebruikersaccounts welke rollen hebben. Om de rollen een hiërarchische structuur te geven, zodat ze rechten van elkaar kunnen erven, zijn er twee dingen nodig:

1. Er moet opgeslagen kunnen worden welke rollen boven andere rollen staan (Parents) en/of andersom (Childs).
2. bij het autoriseren moet er gecontroleerd worden op deze rollen hiërarchie.

Figuur 12 Uitbreiding op de Rol klasse

*Stel: Er is een rol ‘Admin’. Deze heeft een rol onder zich (een kind), de rol: ‘Manager’. Als een gebruiker de rol Admin toegekend heeft gekregen moet hij ook de acties uit kunnen voeren van de Manager. Dus wanneer een actie uitgevoerd die alleen de rol Manager uitvoeren mag hij deze ook uitvoeren.*

Om de rollen-hiërarchie op te kunnen slaan is de klasse ‘IdentityRole’ uitgebreid met kinderen en ouders (te zien als Childs en Parents in IdentityRoleExt in Figuur 16). Zo kunnen de rollen, die erven van en de rollen die geërfd worden, opgevraagd worden. Er is gekozen voor een meer op meer relatie, omdat er zo mogelijkheid geboden wordt dat een rol rechten van meerdere rollen erft en daarnaast niet gelimiteerd wordt dat een rol maar één keer geërfd kan worden.

Een gevaar waar op gelet zou moeten worden, is dat er een oneindige cyclus kan ontstaan wanneer de rollen-hiërarchie verkeerd geconfigureerd wordt. Om dit te voorkomen zijn er twee opties: of er moet een beveiliging ingebouwd worden die controleert bij het opslaan van de rollen of geen cyclus ontstaat, of er wordt voor gekozen dat deze verantwoordelijkheid ligt bij de ontwikkelaar die de package implementeert. In deze prototype is deze beveiliging niet opgenomen en ligt de verantwoordelijkheid voor nu dus bij degene die de package implementeert.

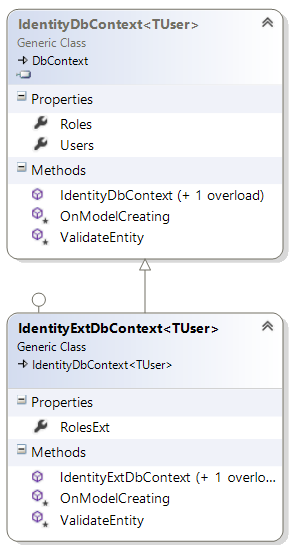
Om in .Net modellen op te slaan in de database wordt gebruik gemaakt van een ‘DbContext’ klasse. In deze klasse wordt vastgelegd welke modellen in de database opgeslagen moeten worden en welke relaties er tussen deze modellen bestaan. Voor het prototype moest de DbContext uitgebreid worden, zodat de nieuwe rol modellen opgevraagd konden worden en relatie tussen de rollen toegevoegd werd in de database.

Figure 13 Uitbreiding op de databasecontext

Nu was het mogelijk de ‘kinderen’ te definiëren en op te slaan in de database, maar deze moest ook uitgelezen worden wanneer een aanvraag geautoriseerd werd. In .Net. Dit bleek te gaan via de Authorize annotatie. Door deze annotatie wordt de klasse Authorize aangemaakt, die aangeroepen wordt voordat de betreffende controller-actie aangeroepen wordt en controleert of deze functie aangeroepen mag worden. Op basis van deze kennis had ik besloten de Authorize klasse uit te breiden en de functie, die na gaat of de ingelogde user de actie uit mag voeren, aan te passen. De uitbreiding die ik schreef vroeg aan de database de rollen op die boven de rollen stonden, die aan de actie gekoppeld waren. Ook voor deze rollen werd gecontroleerd of de gebruiker deze had.

Functioneel gezien werkte dit goed er bleek ook een andere oplossing te zijn. De andere mogelijkheid was om van de gebruiker alle rollen en kinderrollen op te halen en te kijken of hiertussen een overeenkomst is met de toegekende rollen aan de controller-actie. Voor beide zou het betekenen dat voor elke aanvraag uit de database, alle rollen en kinder- of ouderrollen opgehaald zouden moeten worden. Dit zou in grote applicaties voor veel vertraging kunnen zorgen, zeker wanneer er een grote hiërarchie-boom van rollen bestaat.

Door met dotPeek naar de code van de inlogfunctie te kijken kwam ik er achter, dat wanneer een gebruiker inlogt, de rollen die hij heeft, gecached worden in een string. Aan deze string kan gevraagd worden of deze een bepaalde rol bevat. Dit gebeurt in de UserStore klasse. Ik heb toen besloten niet de Authorize klasse (die elke aanvraag opgeroepen wordt) aan te passen, maar de UserStore inlog functie uit te breiden, zodat deze ook alle kinderrollen cachet (wat alleen gebeurd tijdens het inloggen). Op deze manier kon ook altijd direct opgevraagd worden of een gebruiker een bepaalde rol rechten voor had en veranderde er verder niets bij het gebruik van de package. Alleen wanneer de ontwikkelaar alleen de direct gekoppelde rollen van een gebruiker wilde weten kon deze functie niet meer hiervoor gebruikt worden. Dit was in principe geen probleem, omdat dit via de User-klasse diende te gebeuren en deze functie niet hiervoor bedoelt was.

**Caching**

Het opslaan van gegevens op een sneller medium om sneller toegang tot deze data te hebben wordt caching genoemd. De term cache wordt meestal gebruikt voor zowel de data die gecachet worden als voor de opslagplaats waar deze data gecachet wordt.

## https://wiki.opscode.com/download/attachments/13172814/configuration-settings.png?version=1&modificationDate=1306647437000&api=v2Configuratie

Voor het tweede deel van het prototype werd gekeken naar een Implementatie van requirement F01 en F08. Een oplossing voor deze requirements zou zijn, dat het mogelijk werd om de autorisatieregels op te slaan in een configuratiebestand. Een andere mogelijkheid zou zijn, het op te slaan in de database of ander medium en het te laten zien in een overzicht door een controller-actie. Het nadeel hiervan was dat de configuratie dan wel ingezien kon worden, maar niet direct aangepast kon worden. Een ander nadeel was dat, wanneer er gekozen zou worden voor een minder populair medium, de kans een stuk groter werd dat ook de communicatie met dit medium zelf ontwikkeld zou moeten worden. Dit zou veel extra werk op leveren.

Uit ervaring wist ik dat het gebruik van een configuratiebestand veel gebruikt wordt in (PHP) frameworks zoals Zend en Yii. In deze frameworks zitten duizenden uren ontwikkeltijd en kan daarom vanuit gegaan worden dat ook hierover goed is nagedacht.

### Bestaande package

In de onderzoeksfase kwam naar boven dat voor deze functionaliteit al een NuGet package[[2]](#footnote-2) beschikbaar was. Omdat het mogelijk is om in een package referenties naar een andere package te maken (ook wel ‘dependencies’ genoemd) werd er gekeken of deze package voldeed aan de wensen en eisen en gebruikt kon worden als onderdeel van mijn package (mogelijk met enige aanpassingen). Wanneer deze package toegevoegd zou worden als dependecy zou deze package bij installatie van mijn package ook geïnstalleerd worden. Het voordeel hiervan is dat onderdelen los van elkaar geüpdatet kunnen worden. Deze package zou geüpdatet kunnen worden, zonder dat mijn package geüpdatet hoeft te worden.

Om te bepalen of deze package geschikt was om te gebruiken is de package geïnstalleerd in een nieuw .Net project en is gekeken naar de functionaliteiten en uitbreidbaarheid van de package. De package bleek niet geschikt omdat:

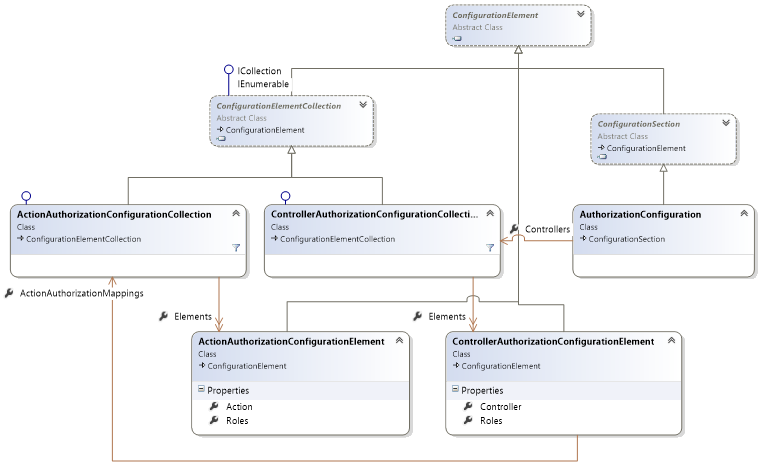
* Het omslachtige features bood, die niet aansloten bij de wensen en eisen van mijn package.
* De package meer dan een half jaar niet was geüpdatet. Dit betekende dat deze package niet meer geüpdatet is na de release van de nieuwste versie van .Net. Het is dus mogelijk dat deze niet meer goed aansluit bij de huidige versie van .Net. Daarnaast is er een kans dat wanneer een bug ontdekt wordt deze niet meer opgelost wordt.

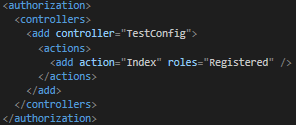
Het verwijderen van de omslachtige functionaliteit zou waarschijnlijk meer moeite kosten dan zelf een eigen implementatie te schrijven. Dit kwam doordat het .Net framework al veel handvatten levert om dingen op te slaan in een configuratie bestand. Daarom besloot ik deze package niet te gebruiken maar het zelf te ontwikkelen.

### Ontwikkeling

Het ontwikkelen van deze uitbreiding bleek tamelijk eenvoudig. Na enig onderzoek bleek dat .Net zelf al de opzet bevatte waarmee het configuratiebestand van de web-applicatie (Web.Config) uitgebreid kon worden. Ik had ervoor gekozen deze manier te gebruiken om deze functionaliteit te ontwikkelen omdat dit precies was wat ik zocht. Andere opties zouden waarschijnlijk meer omslachtig zijn om te implementeren omdat hiervoor de communicatie met het configuratiebestand zelf ontwikkeld zou moeten worden.

De .Net codebibliotheek bevat de klassen ConfigurationSection, ConfigurationElement en ConfigurationElementCollection. Door klasse te definiëren die overerven van deze klasse kon een nieuwe sectie in het configuratiebestand gedefinieerd worden. In Figuur 15 is te zien hoe de configuratie geïmplementeerd wordt door gebruik te maken van overerving van deze drie klassen (die grijs aangegeven zijn in het diagram).



Figuur 15 Ontwerp van configuratie onderdeel

Figuur 14 Implementatie van de configuratie onderdeel

Door dit te ontwikkelen was het mogelijk in het configuratie bestand een autorisatie sectie te definiëren zoals te zien in Figuur 14. Deze nieuwe sectie bevat een collectie van Controller elementen die elke weer een collectie van Action elementen bevat. (De oranje pijlen in Figuur 15 komen overeen met de verdieping in Figuur 14, Elements => <add>)

#### Filter provider

Naast het feit dat de autorisatieregels gedefinieerd moet worden in het configuratie bestand, moet deze ook uitgelezen worden, zodat deze gebruikt worden en de autorisatie ook daadwerkelijk uitgevoerd wordt. Tijdens het zoeken naar mogelijke oplossingen werd redelijk snel duidelijk dat de beste optie was om gebruik te maken van een Filterprovider. De Filterprovider biedt de mogelijkheid om alvorens elke gebruikersaanvraag uitgevoerd wordt, extra filters toe te kennen aan deze aanvraag. Zo kunnen dus op basis van de configuratie in het configuratiebestand en de soort aanvraag die gedaan wordt, extra Autorisatie annotaties toegevoegd worden.

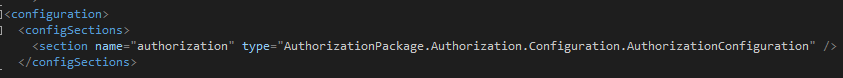
*Stel: Een gebruiker doet de aanvraag naar de pagina van zijn persoonsgegevens: mijnwebsite.nl/Gebruiker/Gegevens. Dit filterprovider kan in de configuratie bekijken of deze controller (Gebruiker) en actie (Gegevens) gedefinieerd is, en zo ja kan deze de Authorize annotatie toevoegen. Zo zal voordat deze actie uitgevoerd wordt, deze annotatie eerst aangeroepen worden en gecontroleerd worden of de gebruiker ingelogd is en of hij de juiste rol bezit.*

In .Net worden filters gebruikt om logica uit te voeren voor bepaalde controller acties (gelijk aan een gebruikers aanvraag) te kunnen vastleggen. In deze filters kan gedrag vastgelegd worden voor en/of na het uitvoeren van de controller actie.

De autorisatie annotatie is een voorbeeld van een filter. Deze filter controleert voordat de actie uitgevoerd wordt of deze uitgevoerd mag worden en zo nee stuurt deze door naar een andere actie.

**Filters**

Bij het ontwikkelen van dit systeemdeel waren er geen grote onverwachte problemen. Wel kostte het wat moeite om uit te zoeken hoe precies de configuratie-elementen gebruikt moesten worden om een eigen configuratie sectie te maken. Daarnaast bleek het configuratiebestand de gemaakte configuratie eerst niet te accepteren. Dit kwam doordat er in het configuratie bestand ook de sectie vastgelegd moest worden zoals te zien in Figuur 16.



Figuur 16 Configuratie van een configuratie sectie

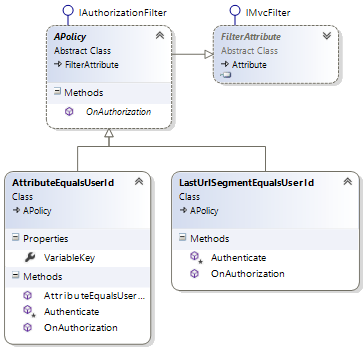
Om te testen of het werkte werd de configuratie van Figuur 14 en Figuur 16 toegevoegd aan het configuratie bestand en werd er een Controller gemaakt met de volgende acties:

* Een actie die door iedereen uitgevoerd mocht worden.
* Een actie die alleen uitgevoerd mocht worden wanneer de gebruiker ingelogd was.
* Een actie die alleen uitgevoerd mocht worden wanneer de ingelogde gebruiker de rol Admin bezat.
* Een actie die alleen uitgevoerd mocht worden wanneer de ingelogde gebruiker de rol Registered (of indirect de rol Admin) bezat.

Hierbij stond de rol Registered onder de rol Admin, door de functionaliteit ingebouwd uit de vorige sprint.

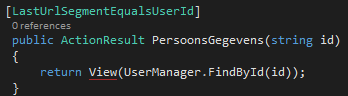
Door niet ingelogd, wel ingelogd zonder rollen en wel ingelogd met de Admin rol deze acties uit te voeren had ik vastgesteld of deze functionaliteit werkte. Toen alle functionaliteiten naar behoren bleken te werken, ben ik doorgegaan naar de volgende sprint.

## Bedrijfsregels

Naast rollen en rechten die een gebruiker heeft, zijn er vaak ook nog andere bedrijfsregels waarmee wordt bepaald of de gebruiker de actie mag uitvoeren. Een voorbeeld dat door de ontwikkelaars werd aangedragen was, dat een gebruiker bijvoorbeeld alleen zijn eigen persoonsgegevens mag aanpassen of dat sommige acties alleen uitgevoerd mogen worden vanuit het intranet.

Figuur 18 Ontwerp Bedrijfsregels (policies)

Het was niet mogelijk om tijdens het ontwikkelen van het prototype al alle mogelijke bedrijfsregels te bepalen. Elk project en elke applicatie heeft vaak zijn eigen bedrijfsregels. Vaak zijn de bedrijfsregels bedrijfsspecifiek en daarom niet van te voren te bepalen. Daarom was het van belangrijker dat er een structuur opgezet werd waarin de ontwikkelaar zijn eigen bedrijfsregels kan vastleggen. De structuur die hiervoor ontwikkeld is, is te zien in Figuur 20.

Ik koos ervoor om een abstracte policy klasse te maken die overgeërfd kon worden om zo specifieke bedrijfsregels te maken. Een andere optie zou zijn geweest om de controller uit te breiden en daarin bedrijfsregels toe te voegen die aangeroepen zouden kunnen worden. De nadelen hiervan waren:

Figuur 17 Gebruik bedrijfsregels

* Controller zou niet meer atomair zijn (één verantwoordelijkheid)
* De configuratie van de bedrijfsregels zou omslachtiger zijn.

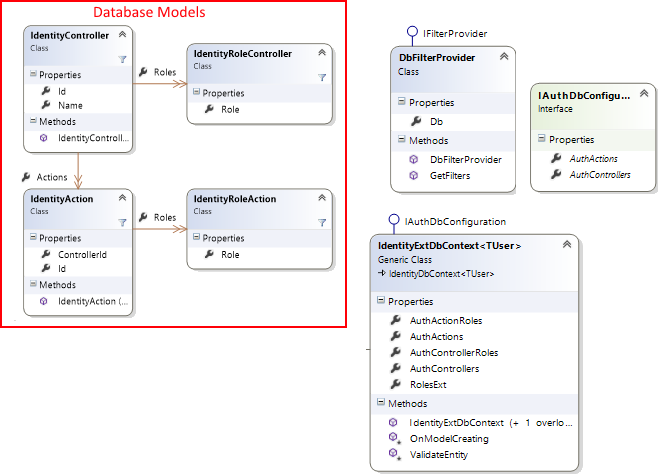
Daarom heb ik ervoor gekozen om de bedrijfsregels te implementeren op dezelfde manier als de Authorize annotatie. Het autorisatie proces kan gezien worden als een bedrijfsregel en bood daarom een uitstekend voorbeeld van hoe deze bedrijfsregels geïmplementeerd konden worden. Het bijkomende voordeel zou zijn, dat de bedrijfsregels ook gemakkelijker te implementeren zijn in andere uitbreidingen die ik geschreven had (denk aan het configuratiebestand, gemaakt in de vorige sprint).

Om te testen of de structuur werkte, zijn er twee concrete implementaties van een mogelijke bedrijfsregel gemaakt. Beide controleren of iets overeenkomt met het gebruikersidentificatienummer. De ene controleert het tegen een meegestuurde attribuut en de andere tegen de laatste sectie van de URL. Met deze bedrijfsregels kan bijvoorbeeld vastgelegd worden, dat de gebruiker alleen een bepaalde actie mag uitvoeren als het nummer overeen komt met zijn eigen identificatienummer. Denk bijvoorbeeld aan het bekijken van de accountgegevens, om te zorgen dat je alleen je eigen gegevens in kunt kijken.

Voor deze bedrijfsregels werden er controlleracties gemaakt, waarbij deze annotaties gedefinieerd werden zoals te zien in Figuur 17. Door in de browser de URL aan te passen kon gekeken worden of deze functionaliteit werkte. Eindigde de URL met het identificatienummer van de gebruiker, dan zou hij wel de actie mogen uitvoeren, anders niet. Dit bleek te werken en daarom ben ik doorgegaan naar de volgende sprint.

Het enige waar nog wel rekening mee gehouden zou moeten worden, is dat het nu nog niet mogelijk was deze bedrijfsregels toe te voegen in het configuratiebestand.

## Databaseconfiguratie

De laatste functionaliteit die ingebouwd was in het prototype is de mogelijkheid om de autorisatie configuratie op te slaan in de database. Dit zou de mogelijkheid bieden om de configuratie aan te passen terwijl de applicatie draait. Ook zou het een opzet zijn voor de functionaliteit dat een administrator rechten van rollen zou kunnen aanpassen, requirement F06.

Figuur 19 Ontwerp database-opslag van autorisatie

In Figuur 19 is het ontwerp te zien van het ontwikkelde onderdeel in deze sprint. De structuur lijkt erg op die van de sprint van het configuratiebestand. Dit komt omdat ze beiden een structuur bieden om de zelfde autorisatieconfiguratie op te slaan, alleen op een ander medium. Ook voor dit onderdeel is een filterprovider ontwikkeld om de configuratie om te kunnen zetten naar Authorize annotaties.

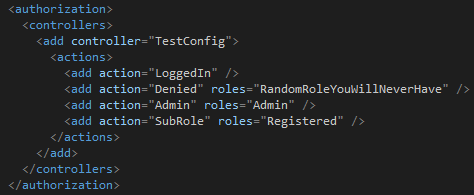
Een groot voordeel dat het implementeren van deze configuraties had, was dat ze naast elkaar konden draaien. Zo kon een deel van de configuratie opgeslagen worden in de database, een deel in het configuratie bestand en een deel in de code. Ook al is dit niet aangeraden het biedt wel meer functionaliteiten. Denk bijvoorbeeld aan wanneer de autorisatieconfiguratie in het configuratiebestand gedefinieerd wordt maar er moet een recht toegevoegd worden, dan kan dit zolang de applicatie niet geüpdatet wordt, toegevoegd worden in de database. Zo kan er snel een probleem opgelost worden voor de klant wanneer deze meer toegang nodig heeft en hoeft de klant niet te wachten op een nieuwe versie van de applicatie. Daarnaast biedt het de ontwikkelaar meer mogelijkheden om te kiezen, omdat elke manier van configuratie zijn eigen voor- en nadelen heeft.

### Testen

Deze sprint werd het zelfde getest als de tweede sprint. Er werd weer een controller opgezet met de volgende acties:

* Index, actie waar geen autorisatie configuratie voor is, waar iedereen bij kan.
* LoggedIn, Actie waar alleen ingelogde gebruikers bij kunnen.
* Denied, Denied is een rol die de gebruiker niet toegekend heeft gekregen en dus niet bij kan.
* Admin, Admin is de rol die de gebruiker toegekend heeft gekregen en dus bij kan.
* Registerd, is een rol waarvan de Admin rechten erft. Dus hier mag de gebruiker ook bij.

Door de web-applicatie te laten draaien en de acties uit te voeren voor en na ingelogd te zijn kon bepaald worden of de configuratie werkt. Wanneer de gebruiker geen toegang had werd hij doorgestuurd naar de inlog pagina. Zo kon bepaald worden of de configuratie ook daadwerkelijk werkte zoals zou moeten.



Figuur 20 Autorisatie configuratie in het configuratiebestand van de test applicatie

## Conclusie

Aan het begin van deze fase had ik aangegeven nog veel kennis te missen om een ontwerp te kunnen maken van de package. Achteraf gezien klopte dit en had ik de juiste beslissing gemaakt allereerst een prototype te ontwikkelen. Tijdens het ontwikkelen van het prototype heb ik nog veel dingen moeten uitzoeken. Dit heeft dan ook de meeste tijd gekost in deze fase. Dit kwam onder andere doordat de broncode niet direct in te kijken was, maar daarnaast ook door de uitgebreide structuur van .Net en doordat ik gewoon nog erg weinig kennis bezat van het framework. Het voordeel van deze uitgebreide structuur van het framework is, dat het ontwikkelen van nieuwe functionaliteiten relatief weinig code nodig was.

Het bouwen van het prototype en het tegelijkertijd uitzoeken hoe dit in het .Net framework het beste zou passen, gaf me veel kennis en inzicht in het .Net framework. Ik stond versteld hoeveel ik geleerd had in deze korte periode. In vergelijking tot de stageperiode eerder in de opleiding (waarin ik in het Zend framework een web portal had opgezet), heb ik met veel meer structuur gewerkt. In de eerdere stage had ik vaak onderdelen moeten vervangen om binnen de structuur van het framework te blijven en was het vaak onduidelijk waar welk onderdeel geplaatst moest worden. Dit had ik tot nu toe niet vaak gehad. Dit kwam waarschijnlijk mede doordat ik nu eerst beter uitzocht hoe de structuur opgebouwd was, alvorens een onderdeel van het prototype te ontwikkelen (en niet zomaar aan de slag te gaan).

Het ontwikkelen van het prototype gaf mij voldoende kennis en input om het ontwerp te maken voor de structuur van de package. Daarom had ik besloten door te gaan naar de ontwerpfase.

De fase van de ontwikkeling van het prototype is afgerond met het Highlightsreport waarin het opgeleverde prototype benoemd is. Verder waren er geen opvallende grote wijzigingen voor het project. Er waren geen wijzigingen nodig in de planning.

# Ontwerpfase

Na het ontwikkelen van het prototype had ik een week ingepland om een ontwerp te ontwikkelen. Deze fase was niet gericht op de inhoud en functionaliteiten van de package maar op de structuur, de manier waarop de package moeten worden gevormd. Zo wordt werd stil gestaan bij bepaalde kwaliteitseisen zoals uitbreidbaarheid en er werd nagedacht hoe deze kwaliteit gewaarborgd kon worden. Naar kwaliteit is gekeken aan de hand van SOLID[[3]](#footnote-3). SOLID is een afkorting voor vijf principes:

* Single responsibility principle - Atomaire verantwoordelijkheid: Elke klasse heeft één verantwoordelijkheid
* Open/closed principle - Open voor uitbreiding, gesloten voor aanpassing
* Liskov substitution principle - Objecten moeten vervangen kunnen worden door een instantie van een subklasse
* Interface segregation principle - Interface segregatie principe: Client specifieke interfaces zijn beter dan één algemene interface
* Dependency inversion principle - Inversie van afhankelijkheid: klasse zijn afhankelijk van abstracties in plaats van concrete klasse.

Het gebruik van de SOLID principes tijdens het ontwikkelen van de package werd aangedragen door mijn begeleider. Door uit te zoeken wat SOLID precies inhield kwam ik erachter dat dit de vijf basis principes zijn voor goed object-georiënteerde applicaties. Het bleek veel gebruikt te worden en bij goed gebruik kwalitatief goede code op te leveren met een losse koppeling en hoge mate van uitbreidbaarheid.

## Planning

In de initiële planning stond aangegeven een integratie model te maken. Het integratiemodel zou in kaart brengen in welke omgevingen de package zou moeten functioneren. Aangezien de package slechts in de standaard .Net omgeving zou moeten functioneren en rekening zou moeten houden met de SARA architectuur was dit diagram niet erg relevant. Zou ik een service ontwikkelen die moest aansluiten op veel verschillende systemen, dan zou dit model erg interessant zijn geweest. Daarentegen had ik gekozen mij vooral te focussen op het ontwerp van het klassendiagram en sequentiediagrammen van de FilterProviders, aan de hand van het SOLID principe.

Allereerst zou ik een eerste versie van het klassendiagram ontwikkelen, waarna deze herzien zou worden aan de hand van SOLID. Indien hier relevante opmerkingen naar boven zouden komen, dan zou aan de hand van deze opmerkingen een tweede versie van het klassendiagram ontwikkeld worden.

Voor het ontwikkelen van de diagrammen is gekozen de ontwerp functionaliteiten van Visual Studio te gebruiken. Deze sloot goed aan op de programmeeromgeving (namelijk Visual Studio is de ontwikkelomgeving) en biedt voldoende functionaliteiten om UML diagrammen te maken.

## Klassendiagram

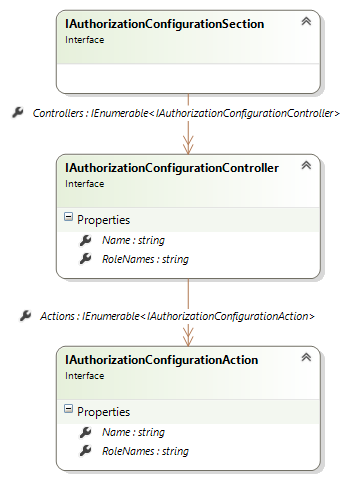
Bij het ontwerpen van het klassendiagram was allereerste gekozen het model op te delen in vier onderdelen. Zo bleef het model overzichtelijk en werkbaar. De onderdelen waren:

1. Uitbreidingen op .net
2. Autorisatieconfiguratie in configuratiebestand
3. Autorisatieconfiguratie in database
4. Structuur bedrijfsregels

Het klassendiagram is te vinden als bijlage genaamd: Bijlage D Klassendiagram versie1. Ik koos voor deze verdeling omdat de koppeling tussen de vier onderdelen laag was en ieder onderdeel een losse functionaliteit toevoegde. Bij het ontwikkelen van de structuur had ik mij zoveel mogelijk gehouden aan de structuur van het prototype, zodat er tijdens de ontwikkelfase niet overbodig veel veranderd zou moeten worden. De grootste veranderingen waren de naamgeving die niet overal consistent was en de opdeling van de package. De opdeling in het prototype was anders dan in het diagram. In het prototype had ik ervoor gekozen de filterproviders samen te groeperen. Omdat dit echter voor een hoge koppeling tussen filterproviders en de beide autorisatieconfiguratie onderdelen, besloot ik de filterproviders bij de autorisatieconfiguratie onderdelen in te voegen.

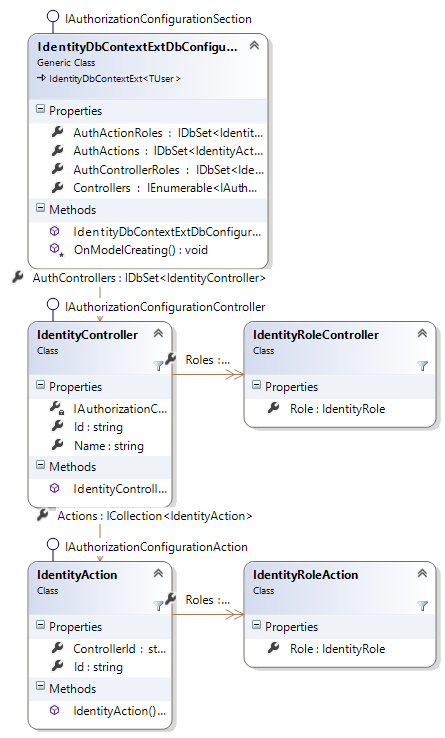
In plaats van alles in één databasecontext te stoppen, besloot ik dit op te delen over twee klassen. De hoofdklasse zou de functionaliteit toevoegen voor de rollen-hiërarchie structuur. De ander, een subklasse hiervan, zou ervoor moeten zorgen dat de autorisatieconfiguratie toegevoegd werd in de database. Zo had de ontwikkelaar de keus om wel de functionaliteit van de rollen-hiërarchie te gebruiken zonder dat de autorisatieconfiguratie onderdeel ook geconfigureerd werd in de database. Dit zou bijvoorbeeld handig zijn wanneer er voor gekozen werd om de database autorisatieconfiguratie helemaal niet te gebruiken. Dan zouden deze onderdelen ook niet toegevoegd worden aan de database.

### Review

Na het ontwikkelen van de eerste versie is het klassendiagram herzien. Dit hield in dat ik het diagram kritisch bekeken heb op alle SOLID aspecten. Daarnaast besprak ik dit met een aantal aanwezige ontwikkelaars, die feedback konen geven op basis van hun ervaring in het vakgebied.

Figuur 21 Structuur van de interfaces voor autorisatie configuratie

Wat duidelijk uit de review naar boven kwam, is dat er meer gedaan kon worden aan ‘Dependency inversion’. De autorisatieconfiguratie en filterproviders hadden een eigen verantwoordelijkheid en toch waren ze aan elkaar ‘vastgebonden’. Een structuur met een losse koppeling zou het mogelijk moeten maken om met één filterprovider alle soorten van configuratie aan te spreken. Hiervoor zou wel nodig zijn dat elke soort van configuratie de zelfde structuur zou aanhouden. Om deze structuur vast te leggen is gebruik gemaakt van interfaces. Er werden een drietal interfaces vastgelegd waarmee de structuur van de configuratie werd gedefinieerd (zie Figuur 21).

Deze structuur gaat als volgt:

* Er is een sectie die een lijst met objecten bevat die de controller-interface implementeren.

Figuur 22 Implementatie van de interface structuur op de autorisatieconfiguratie voor de database

* De controller-interface legt vast dat het een lijst met objecten bevat die de action-interface implementeren. Daarnaast heeft de controller een naam en rollen.
* De action-interface legt vast dat de action objecten een naam en rollen hebben.

Deze structuur (uit Figuur 21) zou dan geïmplementeerd kunnen worden in de structuur van het configuratiebestand en de database. Dit werd gedaan en leverde de structuur op zoals te zien in Figuur 22. (Het ‘rondje met een streep naar beneden’ laat zien welke interface geïmplementeerd wordt).

Door het gebruik van deze interfaces kon er een filterprovider ontworpen worden die alleen hier van gebruik hoefde te maken. Zo konden beide manieren van autorisatieconfiguratie gebruikt worden door dezelfde filterprovider. Door aan de constructor van deze filterprovider een object mee te geven die de Interface ‘IAuthorizationConfigurationSection’ implementeert, kon elke soort van configuratie meegegeven worden aan de filterprovider. Dit had een aantal voordelen:

* Geen duplicaten: Er werd nu maar één keer de logica gedefinieerd hoe de autorisatieconfiguratie omgezet moest worden naar Authorize annotaties.
* Hogere inwisselbaarheid: Een ontwikkelaar kon op deze manier gemakkelijk zelf een autorisatieconfiguratie toevoegen en deze gebruiken.

Het volledige klassendiagram (tweede versie) is opgenomen als Bijlage E Klassendiagram versie2. In deze tweede versie behield ik de onderdelen, maar besloot ik een extra onderdeel toe te voegen: de structuur van de autorisatieconfiguratie. Onder dit onderdeel vielen de interfaces en de filterprovider. Dit leverde de volgende onderdelen:

1. Uitbreidingen op .net.
2. Structuur autorisatieconfiguratie.
3. Autorisatieconfiguratie in configuratiebestand.
4. Autorisatieconfiguratie in database.
5. Structuur Bedrijfsregels.

Op dit punt besloot ik door te gaan en te beginnen aan de sequentiediagrammen. Het klassendiagram had naar mijn mening voldoende kwaliteit en ik verwachte dat de structuur niet veel meer verbeterd zou kunnen worden zonder veel complexiteit toe te voegen. Daarnaast zou het vanaf nu relatief gezien veel tijd kosten om de kwaliteit een beetje te verhogen.

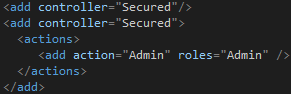
## Sequentie diagrammen

Doordat Microsoft een solide structuur levert waar de uitbreidingen gemakkelijk op in te voegen zijn, betekende dit dat er maar weinig uitgebreide code sequenties geschreven hoefde te worden. Dit werd bijvoorbeeld zichtbaar in de uitbreidingen op het configuratiebestand. Omdat Microsoft zelf al de sequentie heeft geschreven over hoe het configuratiebestand uitgelezen moet worden, hoefde ik slechts uitbreidingen op de structuur toe te voegen (Zie hoofdstuk 8.3). Hiervoor had ik wel de kennis nodig hoe Microsoft dit geïmplementeerd heeft, maar deze kennis is in het algemeen sneller opgedaan dan zelf hiervoor een implementatie te schrijven.

Echter op één punt van de package is er wel een uitgebreide sequentie. Dit is de sequentie waar bepaald wordt, hoe de autorisatieconfiguratie vertaald moet worden naar daadwerkelijke autorisatie. Dit gebeurde door de filterprovider.

Voor deze sequentie is een diagram ontworpen. Zo werd er goed nagedacht werd over hoe deze sequentie uitgevoerd zou moeten worden. Dit diagram is opgenomen in Bijlage F Sequentiediagram FilterProvider. Bij het ontwerpen van deze sequentie werd gelet op de logica van de autorisatie annotatie.

Het enige onderdeel waar een beslissing op gemaakt moest worden was, wat er moest gebeuren wanneer er in de configuratie een controller gedefinieerd was, zonder rollen én zonder acties. Omdat het ook mogelijk was in de code Authorize annotaties toe te voegen aan een controllerklasse (wat er voor zorgde dat alle acties van deze controller beschermd werden), had ik ervoor gekozen dezelfde logica hier te gebruiken.

De enige functionaliteit die de annotatie wel heeft en niet op deze manier geïmplementeerd kon worden was wanneer geconfigureerd moest worden dat:

Figuur 23 Voorbeeld van dubbele autorisatieconfiguratie voor een controller

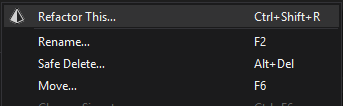
* de controller alleen gebruikt kon worden door ingelogde gebruikers,
* en deze controller een actie bevat die alleen door een bepaalde rol uitgevoerd kon worden.

Om dit op te lossen is gekozen het mogelijk te maken dat controllers meerdere malen gedefinieerd kunnen worden in de autorisatieconfiguratie. Zo werd dit wel mogelijk. Dan kon de configuratie genoteerd worden zoals in Figuur 23.

## Conclusie

Ook deze fase werd afgesloten door het schrijven van een Highlightsreport. Het rapport beschreef de ontwikkelde diagrammen en aanpassingen op de planning. De fase werd verder goed verlopen zonder dat er grote onverwachte problemen naar boven kwamen. Interessant was de feedback op het klassendiagram, waardoor de structuur verbeterd kon worden en er een hogere mate van inwisselbaarheid in het model kwam. Dit betekende wel dat er meer van de code ‘refactored’ zou moeten worden, maar dit zou niet voor problemen moeten zorgen. Het refactoren van het prototype naar de nieuwe structuur zou gebeuren in de ontwikkelfase, de volgende fase.

‘code refactoring’ is een gedisciplineerde techniek voor het herstructureren van bestaande programmeercode, het veranderen van de interne structuur zonder haar externe gedrag te veranderen.



**Refactoring**

Verder kwam uit één van de gesprekken van collega’s naar boven dat het omzetten van code naar een NuGet package ook zeer waarschijnlijk enige programmeerwerk zou kosten. Dit zou betekenen dat dit meer werk zou kosten dan verwacht. Daarom had ik tijdens deze fase tijd ingepland om dit uit te zoeken, om zo te bepalen hoe groot dit risico zou zijn. Dit bleek een bijna onvermijdelijk risico. Bij het constateren van het risico is het risico opgenomen in het risicoregister en zijn de mogelijke voorzorgsmaatregelen bedacht en vastgelegd. De voorzorgmaatregel die mij het meest geschikt leek, was het inplannen van tijd in de ontwikkelfase om de package te ontwikkelen, of op zijn minst de structuur op te zetten waarin de package geplaatst kan worden zodat deze omgezet kon worden naar een NuGet package tijdens de distributiefase.

# Ontwikkelfase

Van de ingeplande vijf weken voor de ontwikkelingsfase waren er uiteindelijk twee weken over gebleven. Dit kwam onder andere doordat er drie weken gebruikt waren om het prototype te ontwikkelen.

Ik had besloten om deze twee weken op te delen in twee losse weken, in twee ‘sprints’. De eerste week zou ik de package omzetten naar de nieuwe structuur en de tweede week de package omzetten naar een NuGet package. Ik had besloten om niet de package door te ontwikkelen en nieuwe features toe te voegen, omdat er dan waarschijnlijk te weinig tijd zou zijn om het project af te ronden en de package te kunnen distribueren. Het omzetten naar de nieuwe structuur zou zorgen voor een betere uitbreidbaarheid en het gemakkelijker maken voor de ontwikkelaar om zelf uitbreidingen te schrijven. Daarom koos ik ervoor dit wel nog te doen.

## Sprint 1 Toepassen nieuwe structuur

De ontwikkelomgeving, Visual Studio, biedt de mogelijkheid om te ‘refactoren’. Wanneer er een klasse of functienaam gewijzigd moet worden, kan er gebruik gemaakt worden van ‘refactoren’. Deze feature past niet alleen de naam aan, maar zoekt waar deze klasse en functie gebruikt wordt en past ook daar de naam aan. Deze functionaliteit werkt ook voor verplaatsen van bestanden. Hierdoor kon gemakkelijk de folderstructuur en de naamgeving aangepast worden. Maar alleen refactoren zou niet genoeg zijn voor het implementeren van de nieuwe structuur van het klassendiagram.

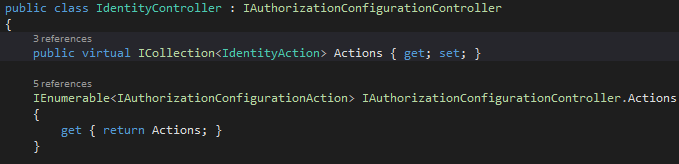
Om de nieuwe structuur te realiseren had ik de volgende stappen gedefinieerd:

1. Ontwikkelen van de nieuwe structuur, de interfaces
2. Implementeren van de nieuwe structuur in de configuraties
3. Samenvoegen van de filterproviders door het gebruik van de interfaces.

Als eerste ben ik begonnen met de huidige code te refactoren waar nodig. Op zich ging dit redelijk gemakkelijk, maar om een onbekende reden crashte de Visual Studio soms tijdens het uitvoeren van het refactoren. Wanneer dit gebeurde werden alle bestanden die op dat moment bij refactoring betrokken waren corrupt. Dit betekende dat deze bestanden niet meer geopend konden worden. Uiteraard maakte ik gebruik van versiebeheer om de applicatie bij te houden en op te slaan en kon dus de vorige versie van het bestand teruggehaald worden. Wel moest dan de meeste refactor-acties hierdoor meerdere malen uitgevoerd worden. Dit zorgde ervoor dat het een stuk meer tijd kostte dan verwacht.

Daarna begon ik met het realiseren van de nieuwe structuur. Het ontwikkelen van de nieuwe structuur lukte zonder enige problemen. Bij het implementeren van deze nieuwe structuur werd wel tegen een aantal problemen aangelopen. Dit kwam vooral terug in gebruik van de IAuthorizationConfigurationController interface bij de database configuratie.

Deze interface verplichte de klasse om een lijst (IEnumberable) te implementeren van de interface IAuthorizationConfigurationAction’s. Nu bevatte deze klasse een lijst van klassen die dit interface implementeert, maar dit bleek niet expliciet genoeg. Ik vond hiervoor een oplossing op de website stackoverflow waar uitgelegd werd, dat dit expliciet gedefinieerd moest en kon worden. In Figuur 24 toont hoe dit eruit zag in de code. Als eerste is de ‘Actions’ attribuut vastgelegd voor de IdentityController klasse. Daaronder is vastgelegd dat, wanner de klasse aangesproken wordt vanuit de interface, de Actions Attribuut een andere lijst met klasse teruggeeft. Zo werd, zoals te zien in Figuur 24 een abstractie van de klasse ‘Actions’ gedefinieerd.



Figuur 24 Code expliciet definiëren van interface eigenschappen

Een andere mogelijkheid zou zijn om de concrete ICollection<IdentityAction> een andere atribuutnaam te geven, bijvoorbeeld ‘RealActions’ en de attribuut naam Actions te definiëren als een IEnumerable<IAuthorizationCOnfigurationAction> die, wanneer opgevraagd, de RealActions teruggeeft.

De eerste optie leek mij eleganter omdat er zo niet overbodig een extra attribuutnaam gedefinieerd werd en het precies doet wat er verwacht wordt. Eigenlijk had ik niet verwacht tegen dit probleem aan te lopen en verwacht dat de compiler zou zien dat ICollection<IdentityAction> subklassen zijn van IEnumerable<IAuthorizationConfiguractionAction>.

Het ‘samenvoegen’ van de filterproviders gaf geen problemen. Omdat de sequentie al uitgedacht was in de ontwerpfase was deze snel geïmplementeerd.

Deze sprint verliep vlot. Er waren wel problemen tijdens het ontwikkelen maar omdat ik al vaker tegen soortgelijke problemen was aangelopen tijdens de ontwikkeling van het prototype, had ik rekening gehouden met de mogelijkheid dat deze problemen zich voor zouden doen. Daarom had ik niet teveel in één week ingepland.

Ik heb de sprint afgerond door te testen of alle functionaliteiten nog werkten. Hierbij heb ik gebruikt gemaakt van de tests die opgesteld waren tijdens de ontwikkeling van het prototype. Om deze tests uit te kunnen voeren hoefde alleen de registratie van de filterproviders aangepast te worden. In plaats van twee verschillende filterproviders, kon nu twee keer dezelfde filterprovider geregistreerd worden met een andere configuratiesectie. Dat er zo weinig aangepast hoefde te worden kwam doordat het externe gedrag van de package niet aangepast werd door het gebruik van de nieuwe structuur.

Alle functionaliteiten bleken nog steeds te werken. Dit was voor mij een teken om door te kunnen gaan naar de volgende sprint: Het omzetten van de package naar een distribueerbare NuGet Package.

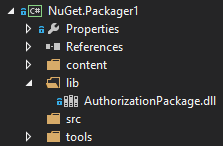
## Sprint 2 Ontwikkeling NuGet package

Voor de tweede sprint had ik gepland om de NuGet package op te zetten. Een NuGet package is meer dan een simpele codebibliotheek. De package bevat onder andere de mogelijkheid om code toe te voegen aan het project waar de package geïnstalleerd wordt. Daarnaast biedt het de mogelijkheid om afhankelijkheden toe te voegen van andere packages en de mogelijkheid voor het toevoegen van verschillende bibliotheken voor verschillende versies van de .Net framework.

Een NuGet package is eigenlijk niets meer dan een mappenstructuur die gezipt is. Op basis van deze structuur weet NuGet wat het moet doen bij het installeren van de package. De mappenstructuur bevat drie folders:

* Lib, deze map bevat de gecompileerde code bibliotheek die aan het project toegevoegd moet worden. Hierin komt de code die tot nu toe ontwikkeld is.
* Content, In deze map kunnen bestanden geplaatst worden die bij het installeren toegevoegd worden aan het project waar de package geïnstalleerd wordt.
* Tools, deze map bevat scripts die uitgevoerd worden bij het installeren en verwijderen van de package.

Verder bevat de package een xml bestand waar informatie over de package in geplaatst kan worden, zoals: De naam, de ontwikkelaars, de uitgever, versie en de licentie waarmee de package verstrekt wordt.

Visual Studio bleek een template te hebben die deze folderstructuur opzette in een nieuw project. Ik heb ervoor gekozen dit ook vanuit Visual Studio te doen zodat alle code in één overzicht (en één solution) blijft.

Figuur 25 NuGet Project in VS2013

Het eerste onderdeel van het ontwikkelen van de NuGet package zou zijn om de code bibliotheek in de Lib folder toe te voegen. Dit bleek niet moeilijk. Door het ontwikkelproject te compileren werd deze bibliotheek automatisch aangemaakt in de ‘bin’ folder van het project. Deze kon gekopieerd worden naar de lib folder van het NuGet project zoals te zien in Figuur 25. Visual studio bood ook mogelijkheden om een script uit te voeren elke keer wanneer het project gecompileerd wordt. Op deze manier zou automatisch de codebibliotheek gekopieerd kunnen worden naar de NuGet package. Ik had besloten om deze functionaliteit niet nu toe te voegen, omdat dit waarschijnlijk nog best wat tijd zou kosten om dit uit te zoeken en andere dingen die nog moesten gebeuren aan de package meer prioriteit hadden.

Nu kon de package gezipt worden, waarna deze geïnstalleerd kon worden in een .Net web-applicatie. Het nadeel was dat de ontwikkelaar nu zelf nog zijn eigen project aan de code bibliotheek moest koppelen. De kracht van een NuGet package zat hem juist in het feit dat de NuGet package dit zelf kan. Het idee zou zijn dat, wanneer de package geïnstalleerd wordt, het project waar de package geïnstalleerd is, direct gebruikt maakt of kan maken van de nieuwe functionaliteiten. Om dit te kunnen bewerkstelligen waren er nog twee onderdelen nodig:

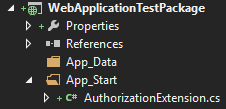
1. Code moest toegevoegd worden aan het project om de Filterproviders te registeren.
2. Code die aangepast moet worden zodat er gebruik gemaakt wordt van de extensies op het .Net framework die de package bevat. Zo moet bijvoorbeeld gebruik worden gemaakt van de nieuwe DatabaseContext en de aangepaste UserStore.

### Contents

NuGet biedt de mogelijkheid om code toe te voegen aan het project waar de package geïnstalleerd wordt door code toe te voegen in de content folder. Om ervoor te zorgen dat de package direct gebruikt kan worden wanneer deze geïnstalleerd wordt, moest er code toegevoegd worden. De onderdelen die toegevoegd moesten worden waren:

* Het Registreren van de filterproviders
* Een Autorisatie sectie in de web.config

Figuur 26 Structuur package voor automatische configuratie

Door bestanden met een .pp extensie toe te voegen aan de content folder werden de bestanden in het project toegevoegd wanneer de package geïnstalleerd werd. In Figuur 27 is een deel van het resultaat te zien wanneer de package vanuit Figuur 26 geïnstalleerd wordt (zie: ‘AuthorizationExtension.cs’). In dit bestand werden de filterproviders geregistreerd zodat deze direct werkte wanneer de package geïnstalleerd werd. Het nadeel van deze .pp bestanden is dat het een redelijk nieuwe feature is in NuGet. Dit was te zien aan het feit dat de ontwikkelomgeving nog geen ondersteuning bood voor dit soort bestanden (zie de blanke iconen zijn bij Figuur 26).

Figuur 27 Structuur project met geïnstalleerde package

De NuGet package bood uitgebreide ondersteuning om het web.config bestand aan te passen. Dit moest gedaan woorden door het web.config.transform te noemen en hierin alle xml code te plaatsen, die toegevoegd zou moeten worden aan de web.config van de applicatie. Met dit bestand werd er een sectie gedefinieerd voor de autorisatieconfiguratie in het configuratiebestand.

### Review

Ik had gehoopt dat ik alle functionaliteiten kon implementeren door gebruik van de .pp functionaliteit die de NuGet package bood. Helaas bleek dit niet het geval. Zoals beschreven bij ontwikkeling van het prototype was er een onderdeel ontwikkeld voor de rollen hiërarchie. Dit onderdeel paste de inlogprocedure aan. Dit was een uitbreiding op de UserStore klasse uit .Net. De accountcontroller, die aangemaakt wordt door het MCV sjabloon, bevat deze klasse. Het zelfde gold voor de database context en andere uitbreidingen vanuit de package op het framework. Op deze punten zouden de klassen overschreven moeten worden door de subklassen uit mijn package. Hiervoor zou deze functionaliteit niet werken. Deze (de .pp functionaliteit) bood alleen de mogelijkheid om bestanden te overschrijven. De bestanden waar deze klasse in gebruikt worden, zijn bestanden die vaak ook door de ontwikkelaar zelf aangepast worden. Stel: de package zou halverwege de ontwikkeling van de applicatie geïnstalleerd worden, dan zouden alle aanpassingen die de ontwikkelaar had gedaan, overschreven worden. Daarnaast zouden ook de subklassen gebruikt moeten worden waar de ontwikkelaar de superklasse gebruikt.

De NuGet package bood wel een andere mogelijkheid om dit te doen. De NuGet package bevat een folder ‘tools’ waarin scripts geplaatst kunnen worden die uitgevoerd worden bij het installeren en verwijderen van de package in het project. Deze scripts zijn gebaseerd op powershell. Dit is een uitbreiding is op de terminal van Windows (de commandline). Ik had besloten dat ik voor deze sprint geen tijd zou hebben om dit uit te zoeken. Ik zou mij hiervoor eerst moeten inwerken in Powershell voordat ik dit script kon ontwikkelen. Het risico zou te groot zijn dat dit niet zou lukken binnen de tijd voor deze sprint en ik aan het eind van deze sprint geen werkende versie kon opleveren. Daarom had ik gekozen deze sprint hiermee af te ronden, om zo een werkende package te behouden en nog in deze sprint te kunnen testen of de package werkte tot nu toe. Dit bleek zo te zijn.

Dit betekende dat deze sprint afgerond werd zonder deze laatste functionaliteit in te bouwen. Naar verwachting zou dit nog een korte week duren om dit te implementeren. Ik had wel enige ervaring met de terminal en had al een idee hoe het script eruit zou moeten komen te zien. Ook omdat de vorige sprints eerder waren afgelopen en er dus nog tijd over was, koos ik ervoor de planning aan te passen, zodat de package goed afgrond kon worden. Dit betekende dat aan de ontwikkelfase nog een sprint toegevoegd werd. Dit zou waarschijnlijk wel betekenen dat er enkele dagen van de testfase nodig zouden zijn om deze laatste sprint af te ronden. Ik heb hiervoor gekozen omdat tijdens het ontwikkelen ook telkens getest werd of de functionaliteiten werkte en ik hierdoor bij het testte geen grote problemen verwachtte. Daarnaast zou dit een essentiële functionaliteit zijn voor het gebruikersgemak van de ontwikkelaar. In het Highlightsreport van deze fase zou dit besluit ook meegenomen worden.

## Sprint 3 Script NuGet Package

Wanneer een nieuwe web-applicatie gebouwd wordt met .Net, wordt er code gegenereerd die gebruikt maakt van de databaseContext en de UserStore klasse. In de package die ik ontwikkeld heb, werden deze klasse uitgebreid. In een NuGet package wordt de mogelijkheid geboden om door middel van powershell, scripts te schrijven om aanpassingen door te voeren in de web-applicatie waar de package geïnstalleerd wordt.

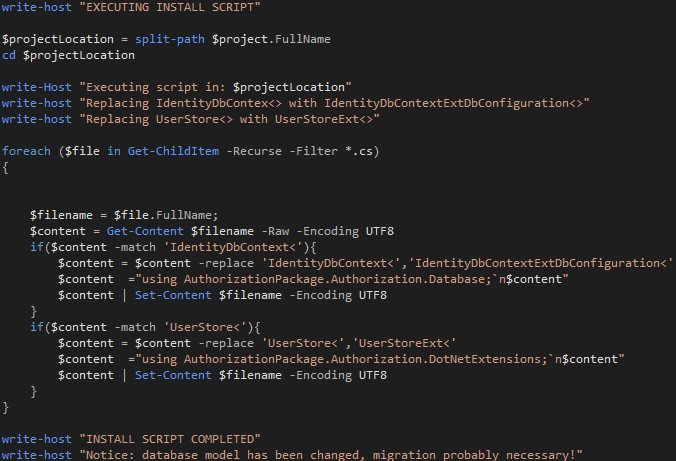
Omdat ik geen ervaring had met powershell scripts, maar wel wat met de terminal verwachte ik redelijk gemakkelijk het script te kunnen schrijven. Ik wist al hoe het script vorm zou krijgen, namelijk:

1. Ophalen van alle code bestanden in het project.
2. Vervangen van de klassen met de subklasse vanuit de package.
3. Het toevoegen van de referentie naar de package.
4. Het opslaan van de bestanden wanneer er iets aangepast was in het bestand.

Ik koos ervoor te zoeken naar vergelijkbare scripts en onderdelen die ik nodig had voor mijn scripts en deze zo aan ta passen als ik nodig had. Dit bleek niet zo gemakkelijk te gaan. Als eerste lukt het niet om het script werkend te krijgen en daarnaast gaf het systeem weinig feedback van wat er verkeerd ging.

Om het script dit toch aan de praat te krijgen, heb ik ontwikkelaars gevraagd op de afdeling, of één van hen hier ervaring mee had. De aanwezig ontwikkelaars bleken hierin niet in staat te helpen, mede doordat ze hierin zelf weinig expertise hadden. Ik besloot twee acties uit te voeren. Als eerst mijn probleem voor te leggen aan andere ontwikkelaars via de stackoverflow website en daarnaast door te gaan met het script te ontwikkelen. Door in kleine stappen iedere keer het script op te bouwen en mij verder te verdiepen in powershell en mogelijke oplossingen voor het script zou ik uiteindelijk tot het script komen wat ik nodig had. Op de gestelde[[4]](#footnote-4) vraag op stackoverflow kreeg ik een antwoord. Dit hielp mij voldoende om het script werkend te krijgen. Ook de kennis die ik intussen had opgedaan hielp erg bij het aanpassen van het script.

Om te testen of het script werkt, werd iedere keer een nieuwere versie van de package gecompileerd en geïnstalleerd in een test project. Door te kijken naar wat er veranderd was in het testproject na het installeren van de package kon bepaald worden of de script werkt.



Figuur 28 Powershell script einde van sprint 3

Het resultaat van dit installeerscript is te zien in Figuur 28. Daarnaast is er ook een script geschreven dat uitgevoerd wordt, wanneer de package verwijderd wordt. Dit script draait de veranderingen, die het installeer-script doet, terug.

## Conclusie

Toen het script bleek te werken zoals ik verwachtte kon deze sprint afgesloten worden en ook deze fase. Er stond een package waarbij alle onderdelen die gepland waren, werkten. Er werd een codebibliotheek toegevoegd, die na het installeren direct in gebruik genomen werd door de applicatie waar deze in geïnstalleerd was.

Ook deze fase werd weer beëindigd met het schrijven van het Highlightsreport. Hierin werd het opgeleverde product, de package, beschreven en de bijzondere gebeurtenissen van de fase. Daarnaast werd beschreven dat door een derde sprint toe te voegen er maar twee weken zouden zijn voor testen. Hiervoor was onder andere gekozen omdat er tijdens de ontwikkelfase ook aan het eind van iedere sprint functioneel getest werd.

# Testfase

Omdat de functionaliteit van de package al aan het eind van elke sprint getest werd, zag ik geen noodzaak om iedere functionaliteit opnieuw te testen. Ik had daarom besloten mij vooral te richten op het testen van de integratie van de package op verschillende omgevingen.

Andere mogelijkheden waren om een gebruikerstest uit te voeren of exploratietest. De gebruikerstest leek mij in eerste instantie minder relevant, omdat de gebruikers van de package ontwikkelaars zijn. Deze hebben in het algemeen meer ervaring op ICT gebied, wat betekent dat er minder rekening gehouden hoeft te worden met de missende expertise en ervaring van de gebruiker. Gebruikersgemak was nog steeds wel belangrijk, maar minder relevant dan het functioneel goed opereren van de package. De exploratietest (en het doen van een risicoanalyse) wordt vooral gebruikt om foutgevoelige functionaliteiten van het systeem te testen. Aangezien ik niet verwachte dat de fouten zich zouden op doen in specifieke functionaliteiten maar in de verschillende omgevingen waarin de package geplaatst zou worden, heb ik gekozen integratietests uit te voeren.

## Initiatie

Om structureel te werk te gaan had ik als eerst een testrapport op gesteld. Hierin heb ik vastgelegd wat het doel was van de tests, op welke wijze de tests uitgevoerd zouden worden en in welke omgevingen de tests uitgevoerd zouden worden. Op basis hiervan heb ik de tests uitgevoerd. De resultaten van de tests en de conclusie ook in dit rapport opgenomen na het uitvoeren van de tests. De conclusie beschrijft ook mogelijke vervolgstappen.

## Iteratie 1

Voor het uitvoeren van de tests had ik besloten om voor elke versie van het .Net framework, vanaf versie 2.0, te testen of de package daarin zou werken. Deze versie kwam uit november 2005 en de package zou dan meer dan genoeg ondersteuning bieden, als deze hierin ook zou werken. Omdat in mijn package gebruikt werd gemaakt van ‘generics’ en deze feature pas in 2.0 is toegevoegd is aan het framework zou de package sowieso niet werken in lagere versies.

Om ervoor te zorgen dat er niet te veel tijd besteed zou worden aan het testen van oudere versies van .Net, besloot ik om te beginnen bij de meest recente versie en van daaruit steeds verder terug te werken. Zouden de tests mislukken door problemen die ontstaan doordat de oudere versies van het framework benodigde ondersteuning mist, kan er vanuit gegaan worden dat deze ook gemist worden in oudere versies en is het dus niet relevant om ook voor deze versies nog te testen. Zo kon, wanneer er problemen ontdekt weren, de tijd gebruikt worden om deze problemen op te lossen, in plaats van door te gaan met testen.

Tijdens het uitvoeren van de tests kwamen er meer problemen naar voren dan ik had verwacht. Zelfs in de meest recente versie van .Net bleek de package niet te werken. Er kwamen allereerst een paar foutmeldingen naar boven die direct verholpen konden worden. Maar daarna kwamen er ook een foutmelding naar boven dat een methode die de package gebruikte niet meer gevonden kon worden in het framework.

Om te bepalen of dit alleen iets van de laatste .net versie was had ik besloten om ook nog te testen in de vorige versie van het .Net framework, versie 4.5.0. Ook bij deze tests kwamen er foutmeldingen naar boven. De foutmelding die bij deze test naar boven kwam, was dat de .Net versie van de package hoger was dan die van de ontwikkelomgeving. Omdat deze foutmelding zich zeer waarschijnlijk zou doorzetten in de lagere versies van het framework, had ik besloten deze versies niet meer te testen en door te gaan.

Ik had ervoor gekozen om nog wel te proberen de package te testen in een ontwikkelproject van Sogeti om te kijken of hierbij dezelfde foutmeldingen naar boven zouden komen. Hiervoor had ik een applicatie gekregen van mijn begeleider. Bij het opstarten van het project kwamen er allerlei foutmeldingen naar boven en het lukte mij niet de applicatie voldoende werkend te krijgen. Omdat dit waarschijnlijk nog veel tijd zou kosten dit werkend te krijgen en ik al voldoende foutmeldingen had, die opgelost moesten worden had ik besloten hiermee deze testfase te beëindigen en eerst deze problemen op te lossen.

Tijdens het testen kwam ik erachter dat niet alleen de .Net versie invloed had op de werking van de package in de tests. De ontwikkelde package maakte ook gebruik van een aantal packages waaronder de package MVC. Deze package bleek geüpdatet te zijn tussen de start van de ontwikkeling van de package en de testfase. De conclusie van dit rapport was dat de package niet voldoende werkte om vrijgegeven te worden. Om de problemen op te lossen moesten waarschijnlijk de externe packages in mijn codebibliotheek geüpdatet worden.

Om de package toch vrij te kunnen geven, had ik besloten na deze tests de problemen die naar boven kwamen op te lossen en dan opnieuw te testen.

### Probleem oplossing

Om de NuGet package te laten werken op de meest recente versie zou de codebibliotheek in de package geüpdatet moeten worden naar de meest recente versie. Die code bibliotheek werd opgebouwd in een ontwikkelproject. Hierin had ik twee mogelijkheden:

1. In dit ontwikkelproject alle packages updaten.
2. Een nieuw project opzetten en daarin de bestanden uit het oude project kopiëren.

Ik had gekozen voor de tweede optie om twee redenen. Op deze manier was er altijd nog een ‘werkende’ back-up, gebruikt zou kunnen worden als het fout zou gaan. Daarnaast bevatte het eerste project al een aantal sub-projecten waarin de applicatie getest was in het prototype- en ontwikkelfase. Hierdoor werd dit project steeds minder overzichtelijk. Het nadeel wel van de package opzetten in een nieuwe omgeving is dat de versie geschiedenis van he project opgedeeld werd. Omdat de package in het project zelf wel goed werkte maar niet in de testprojecten, had ik gekozen hiervoor een nieuwe project op te zetten. Op deze manier hoopte ik op de zelfde foutmelding te lopen als bij het testen, zodat deze problemen opgelost zouden kunnen worden.

Vanuit deze nieuwe project werd de NuGet package weer opgezet. Er bleken een paar problemen te zijn met de package, omdat de MVC package geüpdatet was, maar deze waren snel verholpen. Verder ontdekte ik dat bij de projectopties acties uitgevoerd konden worden nadat het project succesvol gecompileerd was. Hierbij had ik de code toegevoegd dat de codebibliotheek automatisch gekopieerd werd naar de NuGet package. Deze acties waren powershell acties. Omdat ik mij hier eerder in had verdiept kon ik dit snel opzetten.

Als laatste had ik de NuGet installatiescripts nog aangepast, omdat uit de tests bleek dat deze ook niet helemaal goed werkte. Hierna heb ik opnieuw de package samengesteld en was ik doorgegaan naar de volgende fase.

## Iteratie 2

Ook voor de tweede iteratie van het testen had ik een testrapport opgesteld. Hierbij was de werkwijze onaangepast gebleven, maar zijn de omgevingen waarin getest zou worden wel aangepast. uit de vorige test iteratie kwam naar voren dat de package niet zou werken in oudere versies. Omdat hiervoor niets is gedaan om dit op te lossen had ik besloten niet opnieuw te gaan testen of dit zou werken. Ik had besloten om eerst te testen of de verschillende versies van de externe NuGet packages ook nog verder invloed hadden op de werking van de package. Om te kunnen bepalen of deze (versies van externe package) voorruit en/of achteruit problemen op zouden leveren had ik ook een NuGet package gemaakt waarin de code bibliotheek gecompileerd werd op basis van een oudere versie van de MVC package (versie 5.0.0). Zo kon er getest worden of een oudere, dezelfde en een nieuwe versie van een externe package in het ontwikkelproject problemen op zou leveren.

Uit de tests kwam naar voren dat de NuGet package nu wel naar behoren werkte, maar niet werkte wanneer de codebibliotheek uit de package een nieuwere versie van de MVC package gebruikte dan de ontwikkelomgeving. Dit betekende dus dat, omdat alle packages in mijn NuGet package volledig geüpdatet waren, mijn package alleen zou werken in een ontwikkelomgeving die gebruik maakte van de zelfde packages en deze packages ook volledig geüpdatet had naar de meest recente versie. Voordat er getest zou worden, had ik verwacht dat mijn package overal in zou werken. Nu zou het betekenen dat mijn package alleen in de meest recente versie zou werken. Om ervoor te kunnen zorgen dat mijn package in zo veel mogelijk omgevingen zou werken zouden er twee dingen moeten gebeuren:

1. Het aantal afhankelijke packages reduceren naar een zo’n laag mogelijk aantal.
2. De versienummers van de afhankelijke packages zo ver mogelijk reduceren.

Met deze conclusie ben ik doorgegaan en had ik mijn NuGet package verder aangepast.

### Probleem oplossing

Als eerste was ik gaan kijken naar het minimaliseren van de afhankelijkheden van externe packages. Dit had ik gedaan door een nieuw, leeg project op te zetten en hierin alle code van de codebibliotheek vanuit mijn package in te plakken. Door te kijken naar de referenties die deze code maakt, is gekeken welke package er geïnstalleerd moesten worden. Dit bleken twee packages te zijn:

* Microsoft.AspNet.Identity.EntityFramework
* Microsoft.AspNet.Mvc

Na dit ontdekt te hebben, moesten de versienummers van deze packages zo veel mogelijk verlaagd worden om zoveel mogelijk dekking te kunnen krijgen. Hierbij zou het belangrijk zijn dat mijn package wel zou blijven werken. Dit heb ik gedaan door telkens één van beide packages te de-installeren en een lager versie nummer te installeren. Om te controlleren of mijn package nog zou werken heb ik de code gecompileerd. Al kwamen hierbij geen foutmeldingen naar boven ging ik er vanuit dat de package nog zou werken. Bij de volgende test iteratie zou dit dan bevestigd moeten worden. Dit bleek bij de package van EntityFramework al snel het geval te zijn. Bij een lagere versie dan 2.0.0 kon mijn package niet meer gecompileerd worden.

Daarna is gekeken tot welke versie het .Net framework gebruikt kon worden voordat het compileren foutmeldingen zou geven. Dit bleek versie 4.5.0 te zijn. Bij versie 4.0 kwamen gaf de package foutmelding over dat de package EntityFramework een lagere versienummer had dan versie 2.0.0.

Toen bepaald was tot op welke versies van de packages mijn package nog werkte heb ik deze afhankelijkheden toegevoegd aan mijn NuGet package. Nu zou NuGet rekening houden met deze afhankelijkheden en deze packages direct installeren of updaten, wanneer mijn package geïnstalleerd zou worden.

## Iteratie 3

In deze derde testiteratie wilde ik bepalen of de package werkte in de omgevingen die ik nu vastgelegd had. Dit waren de omgevingen met versie nummer MVC 5.0 en hoger, EF 2.0 en hoger en .Net 4.5.0 en hoger. Om te bepalen of mijn package werkt in deze versies en hoger had ik besloten om voor elke package sowieso de laagste en de hoogste versie te testen. Om zeker van de zijn dat mijn package werkt ook in de tussengelegen versies, had ik besloten steekproefsgewijs nog een aantal versies te testen. Op basis van deze eisen had ik de testomgevingen bepaald. De werkwijze van het uitvoeren van de tests was verder hetzelfde gebleven.

Bij het uitvoeren van de tests verliep alles succesvol op één omgeving na. Dit was de omgeving met de MVC package versie 4.0. Dit kwam doordat het niet mogelijk was de testomgeving werkend te krijgen. Eerst gaf het ontwikkelproject foutmelding dat de configuratie bestanden niet klopte omdat hierin de versienummers niet overeen kwamen met de geïnstalleerde packages. Toen dit opgelost was, bleken andere packages niet goed aan te sluiten op deze versie van de MVC package. Toen had ik deze versienummers verlaagd maar kwamen ook toen kwamen er foutmeldingen. Het lukte deze in de gestelde termijn op te lossen, daarom besloot ik door te gaan naar de volgende test. Dit betekende dat voor deze omgeving (MVC4) niet getest kon worden of de package hierin zou werken.

Omdat alle tests die wel uitgevoerd waren, succesvol voltooid waren besloot ik deze iteratie af te ronden en te concluderen dat de package van voldoende kwaliteit was om te distribueren.

## Conclusie

Aan het einde van de derde test iteratie had ik besloten dat mijn package van voldoende kwaliteit was om op te leveren. Ook speelde in deze beslissing mee dat de tijd voor de testfase bijna ten einde liep en het project nu afgerond moest gaan worden. Om nog voldoende tijd te hebben om het project voldoende af te ronden besloot ik, dat ik hiermee de fase moest afronden.

Ik had deze fase afgerond met het schrijven van het Highlightsreport met daarin kort hoe en waarom ik meerdere iteraties van testen had uitgevoerd.

# Distributiefase

## Planning

In de distributie fase heb ik goed gekeken naar de tijd die nog beschikbaar was en wat ik hiermee nog kon doen. Dit was de laatste fase van het project en tegelijkertijd ook de laatste periode van mijn afstudeertraject. Dit betekende dat ik zowel het project moest afronden en tegelijkertijd dit afstudeerverslag moest schrijven en het dossier afronden. Ook aan het verslag moest nog veel gebeuren daarom heb ik besloten om de distributiefase in te perken tot drie kleine handelingen:

* Het schrijven van een handleiding voor gebruik van de package
* De NuGet package en broncode beschikbaar maken binnen het bedrijf.
* Collega’s attenderen op de ontwikkelde package.

Deze laatste stap had ik ook meegenomen omdat ik ontdekt had, dat elke ontwikkelaar op zijn eigen manier werkt en zich niet snel breed oriënteert naar wat er binnen het bedrijf beschikbaar is. Daarnaast worden de projecten van stagiaires, net zoals mijn package vaak in een databank gezet waar nauwelijks meer naar gekeken wordt. Daarom is het belangrijk om ook het bestaan van mijn package onder de aandacht te brengen. Daarnaast zou dit een goed moment zijn om afscheid te nemen van mijn collega’s

## uitvoering

Ik had een handleiding geschreven voor het gebruik van de package omdat niet voor een ontwikkelaar direct duidelijk zou zijn welke nieuwe functionaliteiten mijn package aan zijn project zou toevoegen. Ik ontdekte dat NuGet de mogelijkheid bood om een readme.txt bestand toe te voegen aan de package. Deze zou dan automatisch geopend worden wanneer de package geïnstalleerd wordt. Omdat ontwikkelaars vaak niet eerst de handleiding opzoeken maar het direct uitproberen, had ik besloten op deze manier de handleiding onder de aandacht te brengen. Deze moest niet te lang zijn omdat deze dan niet gelezen zou worden maar wel de belangrijkste informatie bevatten. Daarom heb ik zo veel mogelijk beschreven wat de package bevat. De ontwikkelaar zou zelf de vertaalslag moeten doen naar hoe dit goed gebruikt zou moeten worden.

Ik had verwacht dat er een interne NuGet server aanwezig zou zijn op de afdeling waar iedere ontwikkelaar van Sogeti gebruik van zou maken. Dit bleek niet het geval. Omdat er niet voldoende tijd was om dit zelf op te zetten heb ik dan ook besloten dit niet te doen. Ik heb de package en de broncode opgeleverd aan de opdrachtgever die deze dan zou kunnen distribueren.

Ik zag geen mogelijkheid om de bereikbaarheid van de package nog in deze periode te kunnen verhogen. Daarom en omdat ik verwachte dat bereikbaarheid alleen niet voldoende was om ervoor te zorgen dat ontwikkelaars op de afdeling de package zouden gaan gebruiken had ik gekozen de bekendheid van de package te verhogen. Dit heb ik gedaan door in gesprekken te voeren met de aanwezig ontwikkelaars op de afdeling en bij hen de package onder de aandacht gebracht. Daarnaast was dit gesprek een mooie gelegenheid om ook afscheid te nemen omdat hiermee mijn project en afstudeertraject bij Sogeti ten einde zou lopen.

Met het schrijven van het Highlightsreport en contact met de opdrachtgever heb ik deze fase afgesloten.

# Procesevaluatie

## Keuze afstudeeropdracht

Het proces was begonnen met het zoeken van een geschikte opdracht en leuk bedrijf waar ik mijn afstudeertraject zou gaan uitvoeren. Omdat ik in mijn eerdere stageperiode bij een klein ICT bedrijf had gezeten, wilde ik deze keer bij een groot ICT bedrijf mijn opdracht doen. Op deze manier zou ik, voordat ik het echte ‘werkleven’ in zou gaan, ervaring hebben opgedaan in een klein en in een groot bedrijf. Dit zou mij een beter beeld geven of een groot of klein bedrijf meer iets voor mij is. Daarom had ik mij gefocust om een stageopdracht te krijgen bij een groot bedrijf.

Omdat ik, tijdens het proces van het zoeken naar een afstudeeropdracht, het erg druk had met mijn bestuurstaken van de studentenvereniging had ik ervoor gekozen niet veel sollicitaties af te gaan. Toen ik bij Sogeti ‘binnen’ was had ik daarmee direct genoegen genomen. Er was op dat moment geen concrete opdracht voor mij beschikbaar, wat ervoor zorgde dat ik een opdrachtomschrijving moest opstellen met daarin een onderzoek wat de vervolgstappen van mijn project zou bepalen. Dit betekende dat het voor mij vooraf niet duidelijk was wat ik nar het onderzoek precies zou gaan ontwikkelen, terwijl mij dit wel gevraagd werd bij het opstellen van de opdrachtomschrijving.

Achteraf gezien had ik misschien beter verder kunnen kijken en kunnen zoeken naar een concrete opdracht die binnen het afstudeertraject uit te voeren zou zijn. In plaats daarvan heb ik mij erg laten leiden door het idee dat het bedrijf belangrijker was dan de opdracht. Dit zorgde ervoor dat de eerste drie weken was het voor mij erg zoeken was op welke manier ik de rest van mijn afstudeertraject vorm zou geven. Daarnaast was de afdeling van het bedrijf in Amersfoort, wat anderhalf uur reistijd was.

Er waren ook grote voordelen aan dit bedrijf en de opdracht. Als eerste heb ik mijzelf kunnen bewijzen. Ik heb bijna volledig zelfstandig gewerkt en moest zelf advies vragen wanneer ik dat nodig had. Hierin werd mijn assertiviteit veel gestimuleerd. Wanneer ik hulp vroeg waren er vaak wel ontwikkelaars aanwezig die hulp konden bieden. Daarnaast bood deze afstudeerplek flexibele werktijden wat er voor zorgde dat ik ruimte had om daarnaast mijn bestuurstaken (bij de studentenvereniging) uit te kunnen voeren. Ik moest dus werken vanuit intrinsieke motivatie. Een andere grote voordelen aan het bedrijf was, dat ik inzicht kreeg in het reilen en zeilen van een groot ICT bedrijf. Daarnaast heb ik door deze opdracht erg veel nieuwe dingen geleerd en expertises verworven. Ik heb veel technieken en talen gebruikt waar ik nog weinig of geen ervaring in had. Dit betekent dat ik mijn kennis op ICT vakgebied erg verbreed heb. Daarnaast was er bij zo’n groot ICT bedrijf veel expertise aanwezig en konden er diepgaande gesprekken gevoerd worden over onderwerpen uit dit vakgebied. Dit maakt dat ik achteraf gezien erg blij ben, dat ik deze opdracht heb uitgevoerd.

## Onderzoeksfase

In de onderzoeksfase had ik onderzoek gedaan om er achter te komen wat een relevante package was te ontwikkelen om zo concreet te krijgen wat ik zou gaan ontwikkelen. Dit proces liep in het begin erg traag. Dit kwam mede omdat ik een onderzoeksplan had opgesteld wat niet realistisch bleek te zijn. Ik had hierin veel beter vooronderzoek kunnen doen naar de mogelijkheden (de resources die aanwezig waren). Verder had ik achteraf gezien, beter minder tijd in het onderzoek kunnen steken zodat er meer tijd zou zijn geweest de package te ontwikkelen en uit te breiden. Door onderzoek te doen verplichte ik mijzelf een onderzoeksplan en rapport op te stellen wat ook veel tijd kostte. Het voordeel hiervan was wel dat ik snel een goed beeld kreeg waar de ontwikkelaars zich mee bezig hielden en hoe en op welke manier ze werken (bijvoorbeeld welke packages ze gebruiken).

## Expertise verwerving

Ik had een week ingelast om mij in te werken in het framework. Dit was een goede keuze. Voor deze fase had ik nog maar weinig kennis van het framework. Door mij te verdiepen in het framework leerde ik hoe het gebruikt diende te worden en leerde ik een beetje hoe het framework opgebouwd was. Dit gaf mij al veel meer inzicht maar omdat ik nog weinig kennis had in hóé het framework opgebouwd was kon ik nog niet zien hoe ik mijn package hierin zou ontwikkelen. Daarom had ik daarna ook besloten eerst een prototype te ontwikkelen.

Achteraf gezien had ik een verstandige keuze gemaakt door expertise te verwerven door gebruik van beeld en audio (video) materiaal. Hierdoor leerde ik erg snel en kreeg had ik een totaalplaatje van het framework. Omdat ik visueel veel sneller verbanden leg dan tekstueel, heb ik hiermee denk ik veel tijd bespaart (vergeleken met leren vanuit een boek).

## Prototype

Normaal gesproken gebruik je een framework om (web)applicaties te ontwikkelen en wanneer het framework niet meer voldoet aan je eisen, dan pas je onderdelen aan. In dit geval had ik nog maar zeer weinig ervaring in het gebruik van het framework en had ik al besloten om een onderdeel van het framework uit te breiden. Het nadeel hiervan was dat ik nog niet alle ins en outs van het framework kende en daarom soms veel moest uitzoeken, voordat ik ontdekte en kon besluiten, hoe dingen opgelost moesten worden. Dit koste soms veel tijd om uit te zoeken. Het voordeel hiervan wel was dit ik heel snel het framework leerde kennen.

Het ontwikkelen van een prototype was een goede keuze. Ik had geen goed ontwerp kunnen maken omdat ik geen idee had wat ik zelf zou moeten ontwikkelen en wat al aanwezig was in het framework. Het prototype leverde en een goede werkende applicatie met de werkende functionaliteiten en de kennis in hoe dit het best geïmplementeerd moest worden. Het voordeel hiervan was dat het prototype direct gebruikt kon worden in de ontwikkelfase en het een goede basis vormde voor het ontwerp.

## Ontwerpfase

Door alsnog een ontwerpfase in te gaan, werd er nog eens goed gekeken naar de structuur van het ontwikkelde systeem (de package). Vooraf zag ik niet veel voordeel in het uitvoeren van deze fase, behalve dat er nog eens goed nagedacht werd over de structuur van het prototype. Omdat dit vanuit de opleiding verplicht werd en ik wel wilde laten zien dat ik volgens geleerde structuur kon werken, besloot ik deze fase toch uit te voeren. Zeker omdat er nu al een werkend prototype aanwezig was met, in eerste opzicht een goede structuur, leek mij deze fase overbodig.

Achteraf gezien was het toch goed dat deze fase uitgevoerd is. Door na te denken over de structuur konden zwakke punten ontdekt worden en positieve veranderingen aan e package aangebracht worden. Zo werd er een onderdeel aangepast wat zorgde voor meer inwisselbaarheid en een hogere kwaliteit van de package. Dit betekende ook meer uitbreidbaarheid. Achteraf gezien ben ik blij en tevreden dat ik deze fase had uitgevoerd. Niet zozeer om ‘de ontwerpen te maken’, maar omdat er hierdoor goed is nagedacht over de structuur van de package.

## Ontwikkelfase

Omdat er al een prototype ontwikkeld was met basis functionaliteit, was het niet noodzakelijk dat er in ontwikkelfase nieuwe functionaliteiten toegevoegd werden voor de eerste versie van de package. Daarom had ik mij direct gericht op het omzetten van de package naar het ontwerp en het ontwikkelen van de NuGet package.

Achteraf ben ik erg blij dat ik het ontwikkelen van een NuGet package al gedaan had in de ontwikkelfase. Het bleek nog best wat programmeerwerk te kosten. Al had ik dit pas gedaan tijdens de distributiefase was ik er tijdens de testfase niet achter gekomen dat het belangrijk was de afhankelijkheden (dependencies) van de NuGet package goed vast te leggen. Ik had dan tijdens de testfase überhaupt geen integratietests uit kunnen voeren en ik had niet de problemen ontdekt, die ik nu tijdens de testfase had ontdekt.

Ook had ik verstandig aan gedaan niet meer extra functionaliteiten aan de package toe te voegen. Anders had ik waarschijnlijk te weinig tijd gehad voor de testfase.

Het gebruik van sprints om iteratief te ontwikkelen heeft mij erg geholpen structuur aan te brengen in het ontwikkelproces en alles werkend op te leveren. Zo verplichte ik mijzelf eerst een werkende versie neer te zetten, voordat ik door ging naar de volgende sprint. Uit ervaring van mijn voorgaande stageperiode wist ik dat ik snel geneigd ben door te gaan met het ontwikkelen van nieuwe functionaliteiten alvorens de vorige afgerond en goed getest waren.

## Testfase

Bij de testfase had ik niet veel problemen verwacht. Ik had verwacht dat er uit de tests hoogstens een paar kleine foutjes uit zouden komen. Ik had deze verwachting omdat ik tussendoor ook iedere keer functioneel getest had en de problemen die hieruit naar boven kwamen, allemaal opgelost had. Al zouden er zoals naar mijn verwachting, weinig problemen naar boven komen, zou ik voldoende tijd hebben om nog de distributiefase uit te voeren en het project rustig af te ronden. Deze verwacht bleek helaas niet te kloppen.

De eerste keer kon er geen enkele test succesvol voltooid worden. Zelfs in de meest geüpdatete omgeving deed de package het niet. Dit kwam doordat afhankelijke packages geüpdatet waren. Ik wist niet dat deze packages zoveel invloed hadden op mijn package. Hier kwam ik pas tijdens het testen achter. Gelukkig had ik voldoende tijd ingepland om onverwachtse problemen op te kunnen lossen en opnieuw te testen.

Achteraf gezien had ik al eerder uit kunnen zoeken of al eerder kunnen testen of de package zou werken in de omgevingen die ik voor ogen had. De prototype-fase had hiervoor een juiste tijd geweest. Dit was namelijk vooral een fase om dingen uit te zoeken. Het was ook verstandig geweest dit in die fase uit te zoeken.

Verder heb ik de juiste keus gemaakt door integratietests uit te voeren en geen gebruik te maken van uitgebreide testtechnieken. Het gebruik van deze test technieken zou ervoor gezorgd hebben dat ik meer tijd kwijt was geweest met testen en de ontdekte problemen waarschijnlijk nier meer zou kunnen oplossen. De integratietest leverde de concrete resultaten waarmee ik nog onderdelen van de package kon verbeteren.

Bij de tweede test iteratie had ik een kort script geschreven wat hielp bij het opzetten van de test omgevingen. Als het project groter was geweest, had ik er waarschijnlijk voor gekozen het testen te automatiseren door gebruik van Unittests en de Moq package. Achteraf had ik de juiste keuze gemaakt dit niet te doen. Dit had veel tijd gekost dit uit te zoeken, waarschijnlijk de volledige test periode. Ik had dan geen tijd meer gehad om de problemen in de package op te lossen.

## Distributiefase

Bij het vastleggen van de afstudeeropdracht had ik wel rekening gehouden dat ik eventueel zelf de omgeving op moest zetten waardoor de package gedistribueerd kon worden. Ik had hier tijdens het project geen rekening meer mee gehouden omdat het bedrijf al meerdere servers en kennisbanken had. In de distributiefase bleek dat er geen NuGet server was die algemeen gebruikt werd. Omdat het opzetten van deze server te veel tijd zou kosten en ik mijn project moest afronden besloot ik hier geen energie meer in te stoppen.

Ik had ontdekt dat de bereikbaarheid van de package niet het enige grote probleem was voordat de ontwikkelaars de package zouden gebruiken. Ook de bekendheid bleek erg belangrijk. Er waren namelijk al veel stagiaires mij voor gegaan met het ontwikkelen van allerlei applicaties en packages maar weinig van deze applicaties werden ook maar daadwerkelijk gebruikt. Ik verwachte dat dit niet alleen lag aan de kwaliteit van de applicaties of de bereikbaarheid daarvan. Daarom had ik mijn package nog onder de aandacht gebracht bij de ontwikkelaars op de afdeling. Dit was niet alleen goed voor de bekendheid van mijn package maar was ook een leuke afsluiting van mijn afstudeerperiode bij het bedrijf Sogeti.

## Conclusie

In het kort dingen die ik achteraf anders had gedaan:

* Ik had een meer concrete opdracht gekozen.
* Ik had minder tijd gestopt in het onderzoeken naar het ontwikkelen van een relevante package. Ik zou de volgende keer ook eerst meer vooronderzoek doen naar de mogelijkheden (resources) die ik tot mijn beschikking had.
* Ik had eerder getest of mijn module/package zou werken in meerdere omgevingen, waarvoor de package bedoelt was.

Van alle goede keuzes die ik dit project gedaan heb, waren dit naar mijn mening de beste keuzes:

* Ontwikkelen in korte sprints, dit hielp mij goed om structuur te houden.
* De keus om van de initiële planning af te wijken en eerst een prototype te bouwen.
* De keuze welke tests ik uitvoerde. Hierdoor ontdekte ik de grote problemen van de package.

Achteraf gezien kijk ik erg positief terug naar hoe ik mijn project uitgevoerd had. Ik had gehoopt meer uit de onderzoeksfase te kunnen halen maar dit was helaas niet gelukt. Dit kwam onder andere doordat niet alle middelen beschikbaar waren. De rest van het project is goed verlopen ik heb mij erg veel in moeten vele nieuwe dingen geleerd (zoals Powershell). Ik vond dit erg leuk en interessant maar dit leed er wel toe dat ik weinig tijd over had om daadwerkelijk écht te ontwikkelen. Wat ik ontwikkeld heb, is van goede kwaliteit. Daarom kijk ik erg positief terug op mijn afstudeertraject bij de Microsoft afdeling van Sogeti.

# Evaluatie beroepstaken

Voor het begin van de het afstudeertraject had ik vier beroepstaken vastgelegd, waardoor ik mijn competenties zou tonen. Deze beroepstaken waren:

* Ontwerp Systeemdeel, Niveau 3
* Bouw Applicatie, Niveau 4
* Uitvoeren en rapporteren Testproces, Niveau 3
* Beheren en distribueren van software, Niveau 3

## Ontwerp Systeemdeel

Na het ontwikkelen van het prototype heb ik ontwerpen gemaakt van de applicatiestructuur (klassendiagram) en processen (sequentiediagram). Omdat er op dit punt al een prototype aanwezig was, zou het in eerste instantie kunnen lijken alsof, het maken van de diagrammen, simpelweg ‘overtekenen’ van huidige structuur zou zijn. De ontwerpfase is niet hiervoor gebruikt, maar is gebruikt om goed na te denken over de structuur en de kwaliteit van de applicatie. Tijdens het ontwerp proces heb ik doormiddel van de SOLID principes het ontwerp opgesteld en aangepast. Hierdoor is onder andere goed gekeken naar: Losse koppeling, abstracties en enkelvoudige verantwoordelijkheid (atomaire). Dit zorgde voor een goede kwaliteit applicatie. Dit bewijst zich vooral in de aanpassingen bij de tweede versie van het klassendiagram. Hierbij is de koppeling sterk verlaagd waardoor de inwisselbaarheid sterk verbeterd werd. Er is ook gekeken naar gebruik van ontwerppatronen (design patterns). Er waren geen ontwerppatronen nodig voor de onderdelen die ik ontwikkeld had in mijn package. Er zijn uiteindelijk geen ontwerppatronen gebruikt omdat er geen onderdelen waren waar dit nodig was, of omdat het onderdeel al een ontwerppatroon bevatte. Zo was het onderdeel dat ik schreef voor het autorisatieconfiguratie-bestand gebaseerd op het ‘composite pattern’[[5]](#footnote-5). Dit patroon zat al in het .Net framework en daarom hoefde ik hier alleen componenten en bladeren toe te voegen.

De diepgang in deze beroepstaak heeft zich niet bewezen in de kwantiteit van de diagrammen en gebruik van ‘design patterns’, maar in het gebruik van geteste principes bij het ontwerpen van de package en de hierdoor genomen keuzes.

## Bouw Applicatie

De meeste tijd in de afstudeerperiode heb ik besteed aan het bouwen van de package. Dit was opgedeeld in twee fases: het ontwikkelen van het prototype en de ontwikkelfase. Vergeleken bij mijn stageperiode heb ik veel minder regels code ‘gegenereerd’. Dit betekent niet dat ik minder hard gewerkt heb dan toen. Naast dat ik in mijn stageperiode met een (voor mij) bekende taal en framework gewerkt had, waarin ik direct aan de slag kon, heb ik deze keer ook zorgvuldiger gewerkt. Ik had mijn ontwikkelfases opgedeeld in sprints waardoor ik mijzelf dwong aan het eind van elke sprint een werkende versie op te leveren.

Het uitdagende van dit project was om de package naadloos te laten aansluiten op het .Net framework en het MVC Sjabloon en het leren van veel nieuwe technieken en tools. Zo heb ik voor het eerst in dit project uitgebreid gebruik gemaakt van Visual studio. Daarnaast heb ik de NuGet packager tool gebruik om NuGet packages te maken. Ik leerde nieuwe talen zoals Powershell en nieuwe technieken waaronder het .Net framework en de versiebeheer van Microsoft: Team server.

Door gebruik van de interface structuur, zoals ontworpen in de ontwerpfase, is er goed rekening gehouden met mogelijke inwisselbaarheid van code. Er is in de package ook rekening gehouden met het gebruik van de ‘virtual’ keyword. De ‘virtual’ methodes kunnen overschreven worden. Zo is er rekening mee gehouden dat een ontwikkelaar mijn package zou willen uitbreiden.

Uiteindelijk is na de testfase een werkende package opgeleverd en kunnen de ontwikkelaars de package gebruiken in hun ontwikkelprojecten.

## Uitvoeren testproces

Aan het eind van elke sprint werd de package functioneel getest en in de testfase is getest door middel van integratietests.

Aan het eind van elke sprint werden de nieuwe functionaliteiten getest door de gemaakte package te implementeren in een MVC omgeving en de nieuwe functionaliteiten te testen. Hiervoor zijn geen test rapporten geschreven omdat het ging over maar enkele nieuwe functionaliteiten die afgelopen week inde package gebouwd waren en het schrijven van deze testrapportage in verhouding te veel tijd in zou nemen en niet haalbaar zou zijn uit te voeren in de sprint. Tijdens de testfase is er wel een testrapport opgesteld om zo gestructureerd te kunnen testen.

Tijdens de eerste sprint van het ontwikkelen van de prototype had ik gekeken naar de mogelijkheid om den nieuwe functionaliteiten te testen door gebruik van unittests. Het voordeel van unit tests is, is dat ze één keer geschreven hoeven worden en daarna, oneindig vaak uitgevoerd kunnen worden zonder dat dit handmatig handelingen vereist. De onderdelen die ik had uitgebreid zaten ‘diep’ in het framework. Dit betekende dat wanneer ik deze onderdelen zou moeten testen er veel mock objecten geschreven moesten worden. Omdat dit veel meer tijd zou kosten dan handmatig testen, had ik besloten de functionaliteiten handmatig te testen.

Vooraf aan de testfase heb ik goed gekeken naar wat belangrijk zou zijn om te testen. Hierin had ik besloten dat vooral de integratie-test erg belangrijk was omdat hier de meest risico’s zaten en de grootste impact zouden hebben wanneer deze tests negatief resultaat zouden hebben. Dit kwam omdat ik geen applicatie ontwikkeld had, maar een package, die in veel omgevingen gebruikt zou moeten gaan worden.

Omdat er door de uitloop van het project minder tijd was voor de testfase had ik besloten geen uitgebreid gebruik te maken van testmethodieken zoals TMap (die ontwikkeld is door dit bedrijf), maar te gebruiken wat ik nodig had. Dit zorgde ervoor dat ik effectief en efficiënt de testperiode heb kunnen uitvoeren.

Wel heb ik gestructureerd gewerkt door eerst, van te voren te bepalen wat, hoe en in welke omgevingen er getest zou gaan worden voordat deze tests uitgevoerd werden. Dit werd samen met de testresultaten en een conclusie samengevoegd tot één rapport. Dit leverde een overzichtelijk rapport. Op deze manier maakte ik erg effectief gebruik van de gelimiteerde tijd die over was. Hierdoor heb ik de tests driemaal kunnen uitvoeren en tweemaal de package kunnen verbeteren om zo de ontdekte problemen op te lossen.

Er zijn geen scripts geschreven om de package te testen, maar er is wel een script geschreven die het opzetten van de testomgevingen versimpelde en versnelde, omdat hierdoor minder acties handmatig uitgevoerd hoefde te worden.

De behaalde competentie ligt hem niet in het schrijven van uitgebreide unittests of een ‘hoge dekkingsgraad’ maar in de keuze van de tests die ik heb uitgevoerd en het heirdoor effectief kunnen verbeteren van de applicatie.

## Beheren en distribueren van software

Het beheren en distribueren van de software is niet alleen gebeurd in de distributiefase van het project. Het beheren van de package begon al aan het begin bij het opzetten van de versiebeheer. Hierdoor werden verschillende versies van de broncode van de package bijgehouden.

Om ervoor te zorgen dat de package zo geïnstalleerd werd dat deze direct gebruikt kon worden, is er tijdens de ontwikkelfase een installatiescript geschreven in PowerShell voor in de NuGet Package. Dit script zorgde ervoor dat de package direct in gebruik genomen werd in het ontwikkelproject. Zo kon de ontwikkelaar de extra functionaliteiten die deze package toevoegde direct gebruiken.

Naast versiebeheer van de broncode had ik voor de NuGet package ook een versienummer bijgehouden. Door telkens het versienummer te verhogen van de NuGet package wanneer deze opnieuw gecompileerd werd, bleven de oudere versies behouden. Zo kon altijd een oudere versie van de package geïnstalleerd worden zou dit gewenst zijn. In totaal zijn er meer dan 50 versies van de package gecompileerd.

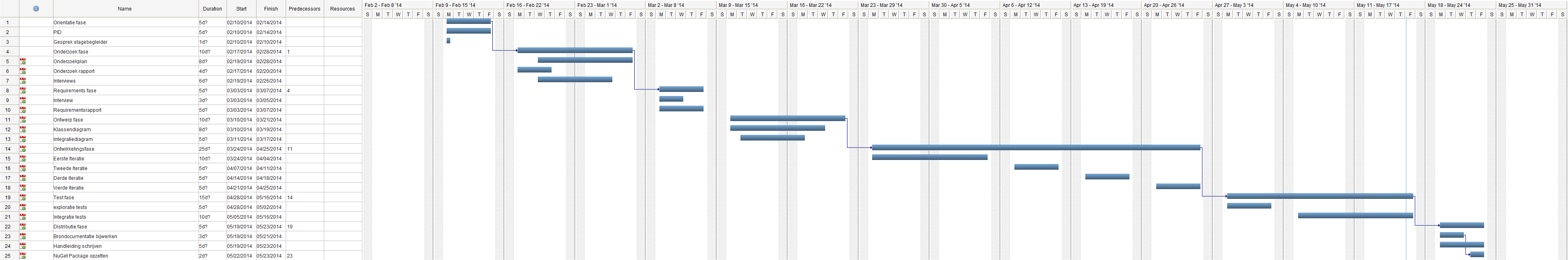
Als laatste is de package ook gedistribueerd. IK had gehoopt dit te kunnen doen via de structuur die Microsoft aanbied: een NuGet server. Er bleek op de afdeling helaas geen interne NuGet server te zijn, waarop de package geplaatst kon worden. Helaas was er geen tijd om deze nog op te zetten. Daarom is de package gedistribueerd via de opdrachtgever.

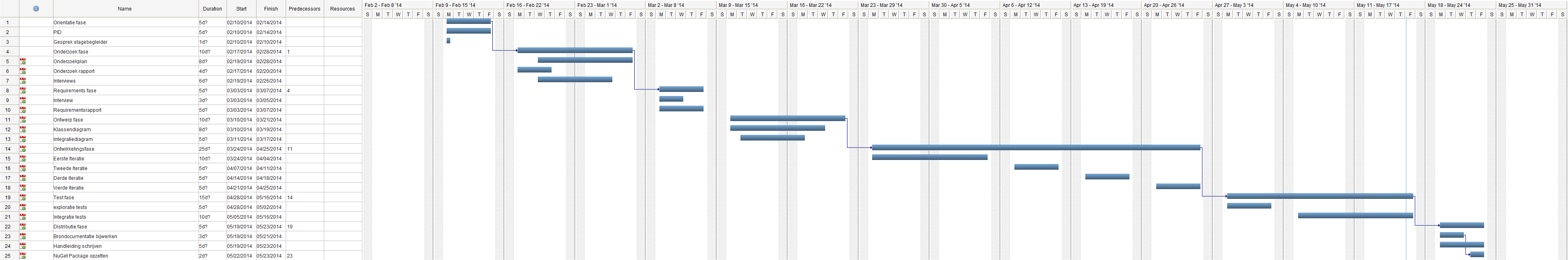
# Verklarende woordenlijst

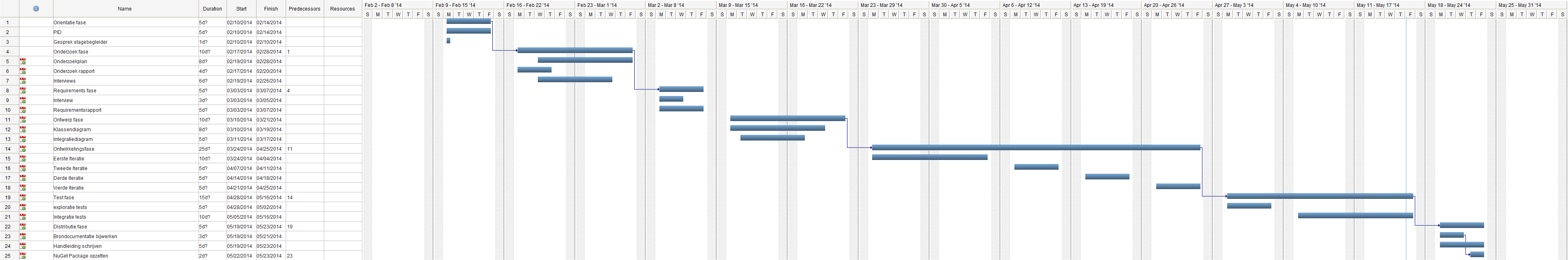
|  |  |  |
| --- | --- | --- |
| Woord | Vertaling | Betekenis |
| .NET |  | Een software framework ontwikkeld door Microsoft met o.a. ondersteuning voor programmeertaal interoperabiliteit. |
| Annotations | Annotaties, Aantekeningen, metadata | Syntactische Metadata die toegevoegd kan worden aan de broncode. |
| Autorisatie |  | De procedure waarmee bepaald wordt of (officiële) goedkeuring wordt toegekend. |
| C# |  | Programmeertaal ontwikkeld door Microsoft. |
| Framework | Raamwerk | Een structuur of schematisch plan dat de basis vormt voor een applicatie. |
| Hack |  | Een stuk computer code dat een snelle of onelegante oplossing biedt voor een bepaald probleem. |
| Highlight Report |  | In PRINCE2: Een rapport, gebruikt om het projectboard op de hoogte te houden van het project. |
| Library | Codebibliotheek | Een verzameling van programma's en softwarepakketten dat algemeen beschikbaar is gesteld. |
| MSDN (Microsoft developers network) | Microsoft-ontwikkelaars netwerk | Website met documentatie over gebruik van Microsoft producten. |
| MVC |  | Een sjabloon in het .Net framework. Het biedt een Model-View-Controller architectuur als basis voor web applicaties. |
| NuGet | Pakket manager | Een gratis en open source package managementsysteem voor het .NET Framework. |
| Package | Pakket, Module | Een samengesteld pakket van Bibliotheken, API’s en modules die geïnstalleerd kunnen worden in een ontwikkelproject. |
| powershell |  | Een op taken gebaseerde commandline en scripting taal die speciaal is ontworpen voor systeembeheer. |
| Refactoring |  | een gedisciplineerde techniek voor het herstructureren van bestaande programmeercode, het veranderen van de interne structuur zonder haar externe gedrag te veranderen. |
| Visual Studio |  | De programmeerontwikkelomgeving van Microsoft voor het ontwikkelen voor een Microsoft omgeving. |

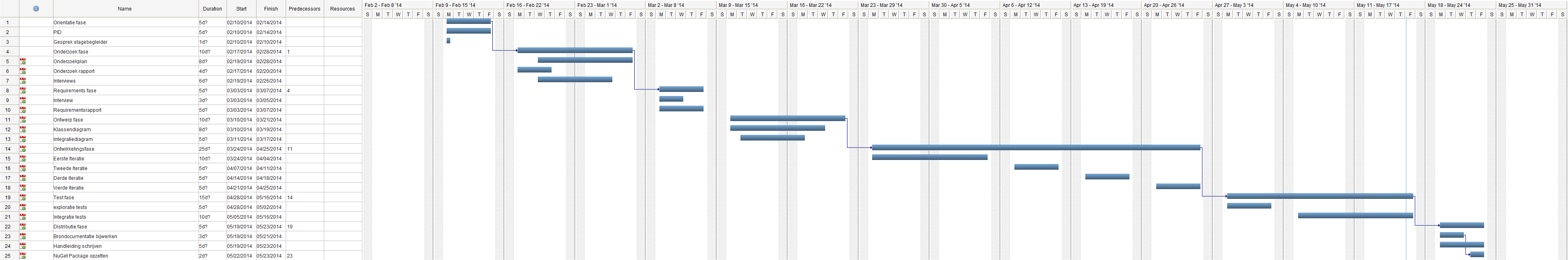
# Bijlage A Initiële Planning Ganttr

(Verspied over 4 pagina’s om de planning in leesbaar formaat te krijgen)









# Bijlage B Onderzoeksplan

Package gebruik en behoefte van .Net ontwikkelaars binnen Sogeti

## Aanleiding

Sogeti is een groot ICT detacheringsbedrijf. Er zijn veel ontwikkelaars die bij de klanten werken. Zij ontwikkelen software in onder andere .NET (C#). Bij het ontwikkelen van software komen vaak dezelfde onderdelen aan de orde zoals authenticatie, autorisatie, transacties etc. In het .NET framework zijn voor veel van deze systemonderdelen implementaties gebouwd. Over veel van deze implementaties is goed nagedacht en biedt dan ook voldoende functionaliteit of uitbreidbaarheid. Toch zal het voorkomen dat sommige onderdelen niet goed aansluiten bij hoe Sogeti werkt of missen er onderdelen in dit framework. Daarom heeft Microsoft de mogelijkheid toegevoegd om zelf packages te kunnen ontwikkelen. Deze mogelijkheid heet NuGet (packages).

Binnen het bedrijf wil ik een package gaan ontwikkelen, die aansluit bij de behoefte van de ontwikkelaars en die zij kunnen gebruiken in hun toekomstige projecten. Om een goede keuze te maken welke package ik ga ontwikkelen, wil ik allereerst onderzoek doen naar welke package er gebruikt worden door de ontwikkelaars en wat zij missen en/of welke ideeën zij hebben, voor een package. Op basis hiervan kan een goede keuze gemaakt worden welke package ontwikkeld gaat worden.

## Hoofdvraag

*Op welk onderdeel kan een package ontwikkeld worden waarmee de .NET ontwikkelaars van Sogeti het meeste tijd gaan besparen in bouwen van .Net web applicaties?*

## Deelvragen

1. Welke onderdelen/packages worden er gebruikt door de .Net ontwikkelaars van Sogeti?
2. Welke ideeën hebben de .Net ontwikkelaars voor het ontwikkelen van de package? (Welke onderdelen missen de .NET ontwikkelaars nu in het ontwikkelen van hun projecten?)
3. Welke alternatieve packages zijn er beschikbaar voor de aangedragen ideeën?
4. Hoeveel tijd gaat het ontwikkelen van een package voor dit onderdeel uiteindelijk besparen?
   * Hoe vaak gaat de package gebruikt worden?
   * Hoeveel tijd gaat het per keer besparen?
5. Hoeveel tijd kost het ontwikkelen van de ideeën tot een package?

## Werkwijze

Om de deelvragen, en uiteindelijk de hoofdvraag, te kunnen beantwoorden zullen er interviews gehouden worden met de ontwikkelaars van Sogeti op de Microsoft-afdeling. Naast de interviews zal er literatuur onderzoek gedaan worden naar de huidige gebruikte packages en naar mogelijke alternatieve of oplossingen voor de ideeën die aangedragen worden.

### Kwaliteit

Omdat het onderzoek gelimiteerd is tot drie weken zal niet voldaan kunnen worden aan alle kwaliteitseisen van onderzoek. Enkele kwaliteitseisen van onderzoek zijn:

* Het onderzoek is herhaalbaar.
* Het onderzoek is betrouwbaar.
* Resultaten uit het onderzoek zijn te generaliseren.
* Het onderzoek wordt onafhankelijk uitgevoerd.
* Er worden tijdens het onderzoek geen aannames gedaan.
* Interviews worden getranscribeerd en gecodeerd.

Omdat dit onderzoek gelimiteerd is aan een tijdsperiode van drie weken zal het niet mogelijk zijn al deze kwaliteitseisen na te streven. Bij kwalitatief onderzoek is het vaak gebruikelijk interviews te transcriberen, dit is het letterlijk uittypen van de interviews. Dit wordt gedaan om een betere analyse te kunnen doen op dat wat gezegd is. Omdat mijn onderzoek een klein onderzoek is en ik vooral van specifieke informatie wil vragen heb ik gekozen om niet te transcriberen. Ik heb wel gekozen om interviews te doen in plaats van enquêtes omdat hierdoor de gegeven antwoorden en ideeën beter toegelicht kunnen worden en ik kan controleren of de ideeën goed op mij zijn overgebracht. De gegeven antwoorden zullen wel genotuleerd worden, zodat deze gebruikt kunnen worden in het antwoorden van de deelvragen. Om ervoor te zorgen dat het onderzoek wel herhaalbaar is er structuur aangebracht in het interview door de hoofdvragen vast te stellen en is deze ook meegenomen in dit onderzoeksplan.

Een andere grens aan dit onderzoek is dat alleen de ontwikkelaars die momenteel niet op een opdracht zitten, beschikbaar zijn om geïnterviewd te worden. Het aantal mensen dat beschikbaar is en aanwezig is op de afdeling, is 10-20. Van de volledige afdeling van 120 man is dit ongeveer 10% van de volledige populatie. Om een correlatie aan te tonen binnen een populatie van 120 man met een foutmarge van 5% en betrouwbaarheidsniveau van 95% (en spreiding van 50%), zou de steekproef minimaal 92 personen moeten bevatten. Gelukkig hoeft er met dit onderzoek geen correlatie aangetoond te worden.

Wat jammer is van deze steekproef is dat het hierdoor geen volledige aselecte steekproef is. Alleen de mensen die momenteel niet op opdracht zitten, worden geïnterviewd. In principe heeft dit op zichzelf geen betekenis. De mensen uit de steekproef zouden net klaar kunnen zijn met een opdracht. Een mogelijk gevaar is wel dat alle mensen net van dezelfde opdracht afkomen of omdat ze met een andere rede momenteel beschikbaar zijn (bijvoorbeeld omdat ze minder ervaring hebben of slechter presteren dan degene die niet beschikbaar zijn). Tijdens de interviews en gesprekken met collega’s op de afdeling zal moeten blijken of hiervan enige sprake is. Omdat hier geen (directe) verbanden gelegd kunnen worden en zijn en omdat als dit wel zo zou zijn, mijn onderzoek geen fout resultaat zou opleveren, is dit niet verder onderzocht of rekening mee gehouden.

### Werkwijze Interviews

De interviews die gehouden gaan worden zijn bedoelt om de eerste twee deelvragen te kunnen beantwoorden. Niet alleen voor de herhaalbaarheid van de interviews, maar ook om de interviews voldoende structuur te geven is ervoor gekozen van te voren de gespreksvragen op te stellen. Deze vragen moeten de resultaten leveren waarmee de deelvragen beantwoord kunnen worden. Aan de hand van de onderstaande vragen zullen de interviews gehouden worden.

1. Van welke packages maakt u nu gebruik bij het opstarten van een project?
2. Welke packages heb je in het verleden gebruikt voor projecten?
3. Laten nadenken over de mogelijkheden van ontwikkelen van package:
   1. Hebt u wel eens gezocht naar een package die niet bleek te bestaan? (Zo ja, welke?)
   2. Welke functionaliteiten heeft u laatste tijd zelf ontwikkeld in .Net?
4. Waarvoor zou u nog een package willen hebben?
5. Hoe zou deze package eruit moeten komen te zien?
   1. Welke functionaliteiten moet de package hebben?
   2. Wat zou het voordeel zijn van deze package?
   3. In welke kader is deze package bruikbaar?

Om meetbare resultaten in te winnen zal bij vraag twee geturfd worden per project welke packages er zijn gebruikt. Om dit te doen zal gevraagd worden voor het laatste project de gebruikte packages op te noemen. Om te voorkomen dat hoofdzakelijke packages vergeten worden, zal tijdens het gesprek geverifieerd worden of ze de meeste gedownloade packages (volgens NuGet) ook gebruikt zijn. Verder zal er gevalideerd worden of de package die .Net automatisch meelevert ook gebruikt worden. De lijst meest gedownloade packages is vastgesteld door te kijken op nuget.org welke packages het meest gedownload worden hiervan zijn de volgende packages genomen die gevalideerd zullen worden tijdens de interviews: Automapper, Autofac, JQuery, Angular JS, Knockout JS, NHibernite, NUnit en Enterprise Library.

Bij de laatste drie vragen zal genotuleerd worden, wanneer de geïnterviewde een idee aandraagt en dit weet uit te werken of voldoende te omschrijven. Dit idee zal door literatuur onderzoek verder onderzocht worden.

# Bijlage C Onderzoeksrapport

Package gebruik en behoefte van .Net ontwikkelaars binnen Sogeti

## Aanleiding

Sogeti is een groot ICT detacheringsbedrijf. Er zijn veel ontwikkelaars die bij de klanten werken. Zij ontwikkelen software in onder andere .NET (C#). Bij het ontwikkelen van software komen vaak dezelfde onderdelen aan de orde zoals authenticatie, autorisatie, transacties etc. In het .NET framework zijn voor veel van deze systemonderdelen implementaties gebouwd. Over veel van deze implementaties is goed nagedacht en biedt dan ook voldoende functionaliteit of uitbreidbaarheid. Toch zal het voorkomen dat sommige onderdelen niet goed aansluiten bij hoe Sogeti werkt of missen er onderdelen in dit framework. Daarom heeft Microsoft de mogelijkheid toegevoegd om zelf packages te kunnen ontwikkelen. Deze mogelijkheid heet NuGet (packages).

Binnen het bedrijf wil ik een package gaan ontwikkelen, die aansluit bij de behoefte van de ontwikkelaars en die zij kunnen gebruiken in hun toekomstige projecten. Om een goede keuze te maken welke package ik ga ontwikkelen, wil ik allereerst onderzoek doen naar welke package er gebruikt worden door de ontwikkelaars en wat zij missen en/of welke ideeën zij hebben, voor een package. Op basis hiervan kan een goede keuze gemaakt worden welke package ontwikkeld gaat worden.

## Inleiding

Dit rapport beschrijft het onderzoek dat gedaan is. Dit onderzoek is gedaan om een keuze te kunnen maken voor het ontwikkelen van een package voor de afdeling Microsoft in Sogeti. Door middel van interviews is er gevraagd welke packages Sogeti ontwikkelaars veel gebruiken, welke packages zij graag nog zouden zien. Door literatuur onderzoek is verder gekeken hoe relevant deze ideeën zijn om te komen toto ontwikkeling van een NuGet package. Hiervoor is onder andere gekeken naar welke soort packages er gebruikt worden en welke alternatieve packages er zijn voor de ideeën die zijn aangedragen.

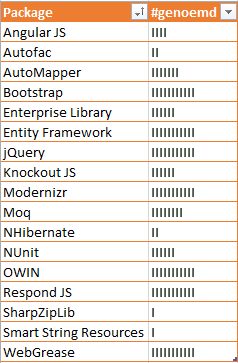
## Gebruik van packages

Visual Studio, de officiële ontwikkelomgeving van Microsoft, maakt gebruik van een pakketmanager om pakketten zoals packages, codebibliotheken en API’s te installeren in .Net projecten. Deze pakketmanager heet NuGet[[6]](#footnote-6). Bij het opstarten van nieuwe projecten, download Microsoft automatisch een aantal pakketten afhankelijke van welk sjabloon er gekozen wordt.

Bij het starten van een .Net MVC web-applicatie project worden de volgende packages gedownload:

* Die het schrijven van javascript ondersteunen: jQuery, Webgrease
* Die html5 en CSS3 features in oudere browers mogelijk maken: Modernizr, Respond JS
* Een standaard responsive css framework: Bootstrap
* De database abstractielaag: Entity framework.
* Voor beveiliging (autorisatie): OWIN

### Resultaten

In de interviews is gevraagd, de packages te benoemen die in het laatste project gebruikt zijn. Deze zijn geturfd en te zien in Figuur 1. Om te voorkomen dat packages vergeten werden is bij het interview gevalideerd of de meest bekende packages niet vergeten werden. Dit zijn alle packages die gedownload worden bij het starten van een MVC applicatie. Daarnaast is gevraagd naar Autofac, JQuery, Angular JS, Knockout JS, NHibernite, NUnit en Enterprise Library.

Figuur 1 Gebruik aantal package in laatste project

Uit de interviews kwam naar boven dat vooral deze packages veel gebruikt werden. Naast de genoemde packages werd ook Autofac, Moq, SharZipLib en Smart String Resources aangedragen als gebruikte packages. Uit de gesprekken kwam naar voren dat Moq gebruikt werd om unittests te schrijven. Deze werd niet zoveel gebruikt, omdat het schrijven van unittests niet meer zo gebruikelijk is. De SharpZipLib werd alleen gebruikt wanneer de applicatie bestanden moest kunnen zippen of unzippen.

Elke package voegt een ander onderdeel toe aan het project. Om duidelijk overzicht te kunnen creëren, is er voor elke package uitgezocht wat deze package inhoudt. Dit is te zien aan het onderstaande overzicht:

|  |  |  |
| --- | --- | --- |
| **Package** | **Categorie** | **Beschrijving** |
| Angular JS | Programmeerhulp | Een javascript library voor een dynamische webapplicatie |
| Autofac | Programmeerhulp | Een package die helpt bij inversie van controle |
| AutoMapper | Programmeerhulp | Een package die het mogelijk maakt viewmodellen volgens conversie automatisch te koppelen aan modellen. |
| Bootstrap | CSS Library | Een responsive front-end (css) framework |
| Enterprise Library | Programmeerhulp | Collectie van herbruikbare softwarwecomponenten voor o.a. logging, validatie, data-toegang |
| Entity Framework | Databaseabstractie | Database abstractielaag van Microsoft |
| jQuery | javascript Library | Een javascript library die het gebruik van javascript vereenvoudigt. |
| Knockout JS | javascript Library | een javascript library voor een dynamische (MVVM) webapplicatie |
| Modernizr | javascript Library | Een javascript library om te detecteren of HTML5 en CSS3 functionaliteiten werken in de browser |
| Moq | Programmeerhulp | Wordt gebruikt om mock objecten te maken van .Net klassen |
| NHibernate | Databaseabstractie | Alternatieve database abstractielaag. Kan ook gebruikt worden met niet Microsoft sql-databases. |
| NUnit | Programmeerhulp | Een alternatief unittest framework |
| OWIN | Anders | Een standaard interface tussen Webservers en applicaties. |
| Respond JS | javascript Library | Een javascript library die media css queries maakt in browers die dit niet ondersteunen. |
| SharpZipLib | Anders | Een zip library. |
| Smart String Resources | Programmeerhulp | Extensies op de core .NET functionaliteit en mogelijkheid om tekst buiten de code te noteren. |
| WebGrease | Optimalisatie | Kan gebruikt worden om javascript, css en image files te optimaliseren |

Om de packages Autofac en Automapper te verduidelijken is hiervoor een korte beschrijving te vinden onder het volgende kopje.

#### Autofac

Autofac[[7]](#footnote-7) is een pakket dat de ontwikkelaars helpt om te programmeren met een losse koppeling (inversie van controle). De package biedt de mogelijkheid specifieke klassen te koppelen aan een interface en de klassen op te halen via deze interface. Zo wordt er, los en eenmalig gedefinieerd welke implementatie van deze functionaliteit verwacht wordt. Het voordeel hiervan is dat, er hierdoor maar op één plaats de specifieke implementatie is gedefinieerd en deze dus snel is te wisselen.

Stel: Je hebt een interface ‘Output schrijver’. Hiervan kan de implementatie zijn dat de output weggeschreven wordt naar de console. Met Autofac kan een container gemaakt worden waarin de klasse die het wegschrijft naar console geregistreerd kan worden aan de interface ‘Output schrijver’. Hierna kan aan deze container een Output schrijver opgevraagd worden waarna deze container automatisch een Console schrijver terug geeft (omdat deze geregistreerd is als dé output schrijver). Dit zorgt ervoor dat er minder vaak specifieke implementaties gedefinieerd hoeven worden. Zo kan er dus sneller geswitcht worden van specifieke implementatie omdat deze maar op één plaats gedefinieerd is.

#### Automapper

Automapper[[8]](#footnote-8) is een pakket die helpt om modellen aan elkaar te koppelen aan de hand van een naamgeving regeling. Omdat .Net voorschrijft dat er altijd maar één model of lijst van modellen doorgegeven kan worden naar de view, wordt er door de ontwikkelaar vaak gebruik gemaakt van ViewModels. Dit is een model met alle gegevens die nodig zijn voor het bepaalde view. Door de automapper package hoeft deze viewmodel niet meer handmatig gevuld te worden maar gebeurt dit automatisch volgens een standaard naamgeving (een conventie).

Stel: Er is een model Blog en Auteur. Elke blog heeft een auteur. Een webpagina bevat een lijst met alle blogtitels en de naam van de auteur. Dan kan er een ViewModel gemaakt worden met een titel en naam van de auteur genaamd ‘AuteurNaam’. Nu kan met de automapper dit ViewModel gekoppeld worden aan de het Blog model. Wanneer dit gedaan is, weet de automapper dat wanneer de AuteurNaam van het viewmodel opgevraagd wordt de Naam van de Auteur van het Blog object moet teruggeven. Zo hoeft deze koppeling niet handmatig geprogrammeerd te worden.

## Behoefte van packages

Bij de interviews werden gevraagd naar onderdelen die de ontwikkelaars misten of graag zouden willen zien. Ook werd gevraagd waar, in de projecten, de meeste tijd in zit. Opvallend bij de beantwoording van deze vraag is, dat de meeste ontwikkelaars aangaven de meeste tijd niet zat in ontwikkeling van bepaalde onderdelen, maar vooral in het achterhalen en helder krijgen van de gewenste functionaliteit (requirements). Bij de interviews kwamen een aantal ideeën naar boven voor mogelijke packages.

### Autorisatie uitbreiding

.Net gebruikt ‘Roles’ die aan accounts gekoppeld kunnen worden om een gebruiker rechten te geven. Bij sommige applicaties wisselen niet alleen de Users van rol maar moet soms ook een rol van permissies kunnen veranderen. De standaard Microsoft implementatie biedt geen functionaliteit om de inhoud van rollen, runtime aan te kunnen passen. Daarnaast wordt het configureren van de rechten die rollen hebben, bij grotere applicaties exponentieel meer werk. Dit komt omdat grotere applicaties en meer rollen gedefinieerd hebben en meer acties bevatten. Daarnaast zijn er nog andere nadelen aan de huidige functionaliteiten van autorisatie. Er is bijvoorbeeld: geen overzicht van alle rechten-configuratie en geen mogelijkheid om uitzonderingen te definiëren of rechten terug te trekken.

Dit idee is door 3 personen aangedragen.

### Cache manager WCF

Microsoft WCF is een set API’s in het .Net framework dat gericht is op het bouwen van service georiënteerde applicaties. Binnen Sogeti loopt er momenteel een project waarbij WCF gebruikt wordt om een service te leveren, waarbij documenten gevalideerd moeten worden. Omdat dit een service georiënteerde applicatie is, betekent het dat performance heel belangrijk is. Om processen te versnellen, kan gebruik gemaakt worden van caching (in .Net webcache API). Om meer performance uit het WCF project te kunnen halen zal gekeken moeten worden naar de interne processen in het WCF en de mogelijkheid om onderdelen hierin te cachen. Hierbij kan bijvoorbeeld gedacht worden aan het cachen van de autorisatie zodat er bij de tweede aanvraag dit niet opnieuw gecontroleerd hoeft te worden.

Het doel van het bouwen van deze package is, om de performance van WCF applicaties te verhogen door gebruik van de cache manager.

Dit idee is door 1 persoon aangedragen.

### Html/Word/Pdf/Email templating

Een module of service waarmee Word of HTML templates beheerd kunnen worden. Deze zou bijvoorbeeld gebruikt kunnen worden om brieven en (html)e-mails te genereren. Naast het generen van documenten is het gewenst dat de service ook de mogelijkheid biedt tot samenvoegen van documenten. De module of service moet de mogelijkheid bieden aan de applicatiebeheerder om documenten te beheren in Word, HTML, PDF of platte tekst. De module (of service) moet ook iets bieden om in de templates gebruik te kunnen maken van condities. Aan de hand van deze condities kan een stuk tekst wel of niet worden opgenomen of herhaald worden.

Dit idee is door 1 persoon aangedragen.

## Alternatieve packages

Microsoft gebruikt een managementsysteem, genaamd NuGet om alle .Net packages te managen. Hierin kan dan ook gezocht worden op alle packages.

Voor elke package werd gezocht op een aantal steekwoorden vanuit die package omschrijving om te zoeken naar alternatieven packages. Voor deze packages is gekeken in hoeverre deze voldoende functionaliteit bieden om het ontwikkelen van het idee (dat door de ontwikkelaar aangedragen is) overbodig is.

### Autorisatie

Bij autorisatie is gezocht naar: Authorization en Roles.

Gevonden mogelijk alternatieven:

* <https://github.com/brockallen/BrockAllen.MembershipReboot>
* <https://www.nuget.org/packages/MvcAuthorization/>

De packages die gevonden werden, waren vaak redelijk specifiek. Bijvoorbeeld de package voor autorisatie in RavenDB. Er zijn twee mogelijke alternatieven die gevonden. Eén hiervan is een package die de autorisatie en authenticatie van .Net volledig vervangt. De andere package is een package die de functionaliteit toevoegt om de autorisatie te configureren in een configuratiebestand in plaats van in de code.

### WCF cachemanager

Dij deze package is hoofdzakelijk gezocht op de termen: WCF, performance en cache.

De packages die gevonden werden, waren onder andere profilers en andere soorten package om de performance van de applicatie te kunnen testen of alternatieven cachemanagers. Er werden geen packages gevonden die WCF een performance boost zouden geven.

### Templating

Bij deze package is hoofdzakelijk gezocht op: pdf, email, template, templating, word en html.

Enkele resultaten die een alternatief zouden kunnen bieden:

* <https://www.nuget.org/packages/TemplateEngine.Docx/>
* <https://www.nuget.org/packages/Spire.PDF/>
* <https://www.nuget.org/packages/EvoPDF/>

Bij deze zoekresultaten zaten veel html template engines als alternatief op Razor. Daarnaast kwamen bij zoekresultaten met de term ‘pdf’ veel pdf generators naar boven.

## Relevantie ontwikkeling

Om de relevantie van het bouwen van de package te bepalen, is gekeken naar een drietal categorieën:

* De tijd die het implementeren van de package zal besparen.
* Het aantal keren dat de package gebruikt zou worden.
* De mate waarin de kwaliteitskenmerken verwacht worden van de package.

Het was lastig om vast te stellen op welke basis de relevatie bepaald wordt. Allereerst is vanuit business oogpunt belangrijk dat er het meest winst mee behaald kan worden. Hoe meer een ontwikkelaar kan doen in de tijd die hij uitbetaald krijgt, hoe meer winst het bedrijf kan draaien. Van daaruit zijn de eerste twee relevanties bepaald. De tijd die het implementeren van de package zou besparen vermenigvuldigd met het aantal keer dat de package geïnstalleerd kan worden, levert de totale tijdswinst op.

Daarnaast koos ik ervoor om de relevantie te bepalen aan de hand van de ISO 9126 kwaliteitskenmerken. Ik heb hiervoor gekozen, omdat de kwaliteitskenmerken laten zien in welke mate kwaliteit verwacht wordt van de package/systeemdeel. Wordt er een lage kwaliteit van het systeemonderdeel verwacht, dan is het ontwikkelen van een package hiervoor van minder essentieel. Het voordeel van een package is dat het één keer (goed) ontwikkeld wordt en daarna vaak hergebruikt kan worden.

Naast de grootte van de package geeft de verwachte kwaliteit een indicatie van de hoeveelheid tijd die het zal kosten om de package te ontwikkelen en de tijd die het zal besparen bij gebruik van de package. Daarnaast biedt de package het voordeel dat, wanneer er een bug of probleem ontdekt wordt, deze één keer bijgewerkt kan worden in de package, waarna de applicaties die deze package gebruiken, gemakkelijk geüpdatet kunnen worden. Bij een systeemonderdeel waarvan minder kwaliteit wordt verwacht, is het minder essentieel dat deze in oude systemen geüpdatet wordt.

Als eerste heb ik de verwachte kwaliteit vastgesteld omdat deze ook invloed heeft op de verwachting van de hoeveelheid tijd die het gebruik van de package zal besparen.

### Kwaliteit

Ook Kwaliteit is meegenomen in dit onderzoek omdat de kwaliteit van de package aangeeft, hoe waardevol het onderdeel moet zijn. Wat hiermee bedoeld wordt, is dat wanneer er van een package/onderdeel een hoge kwaliteit (denk bijvoorbeeld aan fouttolerantie) geëist wordt, er ook meer getest zal moeten worden. Het zal dus uiteindelijk meer tijd besparen, wanneer er van een onderdeel waarvan een hoge kwaliteit verwacht wordt, ontwikkeld wordt. Om de verwachte kwaliteit te kunnen bepalen is voor alle voorgestelde packages in kaart gebracht, hoe belangrijk de kwaliteitskenmerken (ISO 9126) zijn.

In de tabel hieronder zijn de prioriteiten van de kwaliteitskenmerken bepaald voor de aangedragen ideeën. De mogelijkheid was er om dit zelf in te vullen vanuit de expertise die ik ontwikkeld had of door middel van een enquête de geïnterviewde dit te laten bepalen. Naast dat de geïnterviewde waarschijnlijk zijn eigen idee becijfert en het dus niet onafhankelijk in zou vullen, zou dit ook te veel tijd kosten. Om toch een goede indicatie te krijgen is deze met behulp van de bedrijfsbegeleider ingevuld. De resultaten zijn:

|  |  |  |  |
| --- | --- | --- | --- |
| Kwaliteitskenmerk | Autorisatie uitbreiding | WCF cachemanager | Templating service |
| 1. Functionaliteit:  * Juistheid * Geschiktheid * Naleving voorschriften * Beveiligbaarheid * Koppelbaarheid | **9.0**  10  8  4  10  8 | **7.2**  10  7  4  7  8 | **7.2**  10  7  4  6  9 |
| (2) Betrouwbaarheid:   * Bedrijfszekerheid * Foutbestendigheid * Herstelbaarheid | **8.3**  9  10  6 | **9.7**  9  10  10 | **6.3**  9  4  6 |
| (3) Bruikbaarheid   * Begrijpelijkheid * Leerbaarheid * Gebruiksgemak * Aantrekkelijkheid | **5.0**  6  3  8  3 | **1.0**  1  1  1  1 | **4.8**  8  6  4  1 |
| (4) Efficiency:   * Responssnelheid * Middelenbeslag | **7.5**  9  6 | **9.0**  10  8 | **6.0**  6  6 |
| (5) Onderhoudbaarheid:   * Wijzigbaarheid * Analyseerbaarheid * Stabiliteit * Testbaarheid | **7.8**  6  8  8  9 | **8.0**  6  8  10  8 | **7.5**  6  6  10  8 |
| (6) Overdraagbaarheid:   * Aanpasbaarheid * Installeerbaarheid * verdraagzaamheid * vervangbaarheid | **8.0**  4  9  9  10 | **7.3**  6  9  6  8 | **5.0**  8  4  4  4 |
| Totaal | 44.6 | 42.2 | 36.8 |

### Gebruik

Naast de kwaliteit die van de package verwacht wordt, is het gebruik van de package erg belangrijk voor het bepalen of het ontwikkelen van de package relevant is. Hoe vaker de package gebruikt kan worden, hoe meer rendement er uit het ontwikkelen van de package te halen valt. Om te bepalen of een package gebruikt wordt, is er gekeken naar hoe implementatie-specifiek de package is, om van daaruit te kunnen bepalen hoe vaak de package gebruikt zou kunnen worden.

Om te achterhalen hoe implementatie-specifiek de ideeën zijn, is voor elk idee gedefinieerd in welke context de package bruikbaar zou zijn en daarnaast in welke context de package een nuttige toevoeging zou leveren. Dit is te zien in de tabel hieronder:

|  |  |  |
| --- | --- | --- |
| Package | Bruikbaar | Nuttig |
| Autorisatie | Alle .Net applicaties. | Grotere web applicaties (vaak) |
| WCF cachemanager | Alleen de WCF .Net applicaties. | WCF applicaties waar performance boost nodig is. (zeldzaam) |
| Templating service | Applicaties waarin email of bestand mogelijkheden in zitten. | Applicaties die excessief gebruik maken van html e-mails of het maken van pdf’s. |

Om te bepalen hoe vaak deze context voorkomt in het bedrijf Sogeti wilde ik een overzicht opvragen van de projecten die het afgelopen jaar gedaan zijn op deze afdeling en in hoeveel procent van deze projecten deze context van toepassing zou zijn. Het bleek niet mogelijk hiervan een overzicht te krijgen. De drie andere opties die ik had waren:

* Zelf een schatting doen op eigen ervaring,
* Door gebruik van expertise van de ontwikkelaars op de afdeling een indicatie geven;
* Of door via de interne CV applicatie te kijken naar de gebruikte tags bij de projecten.

De CV applicatie is een interne web-applicatie binnen Sogeti waar iedere ontwikkelaar zijn CV op zet en bijhoudt. Hier voert de ontwikkelaar de projecten, die hij heeft uitgevoerd in en voegt hij hier bepaalde tags aan toe. Uit deze tags zou mogelijk opgemaakt kunnen worden, wat voor project uitgevoerd is. Op advies van de ontwikkelaars heb ik deze CV applicatie niet gebruikt. Dit kwam omdat bleek dat deze niet goed werd ingevuld. Er werd namelijk op basis van deze tags nieuwe opdrachten voor de ontwikkelaars gezocht. Hierdoor liet de ontwikkelaar vaak tags weg waarop hij niet wil ingezet wilde worden of andersom. Dat betekent dat deze tags geen goede representatie geven van de uitgevoerde opdracht.

Omdat ik zelf nog weinig ervaring heb met het bedrijf, leek het mijn geen optie om zelf percentages bepalen. Omdat ik geen tot zeer weinig ervaring heb met Sogeti zouden deze percentages een wilde gok zijn. Daarom heb ik besloten om de ervaring op de afdeling te gebruiken om een indicatie te kunnen krijgen van hoe vaak de context aanwezig is in de ontwikkelprojecten. Door vier ontwikkelaars te vragen hoe vaak zij zulke projecten deden en naar wat zij wisten wat er verder gebeurde op de afdeling, is er een indicatie gedaan. De schaal die genomen is, gaat als volgt (van hoog naar laag):

* Altijd
* Vaak
* Gemiddeld
* Soms
* Zeldzaam

De context van deze indicaties zijn: alle .Net opdracht die gedaan worden binnen de afdeling. In de tabel hieronder is te zien in welke verhouding de package bruikbaar en nuttig zou zijn (in alle .Net projecten die gedaan worden op de afdeling).

|  |  |  |
| --- | --- | --- |
| Package | Bruikbaar | Nuttig |
| Autorisatie | Altijd | Vaak |
| WCF cachemanager | Soms | Zeldzaam |
| Templating service | Vaak | Gemiddeld |

Uit deze indicatie kwam dat de autorisatie package het meest gebruikt zou kunnen gaan worden. Daarnaast zou de templating package ook vaak gebruikt kunnen worden. Bij de indicaties werd wel aangegeven, dat het zou uitmaken of de templating engine een module of service zou zijn. Wanneer dit een service zou zijn, in plaats van een package, zou het mogelijk kunnen zijn dat externe klanten er geen gebruik van willen maken (door een streng beveiligingsbeleid dat de bedrijven aanhouden).

### Tijdsbesparing

De tijdsbesparing bepalen is bijna onmogelijk voor een package die niet ontwikkeld is. Wel kan bepaald worden welke voordelen de package met zich mee brengt en op basis daarvan een indicatie gedaan worden hoeveel tijdsbesparing de package zou kunnen opleveren.

Allereerst wordt de hoeveelheid tijd bespaard, die in de package zit, wanneer precies dezelfde functionaliteit ontwikkeld moet worden. Hiervoor geldt dus dat de tijdsbesparing even groot is als de tijd die gestopt is in de ontwikkeling van de package. Daarnaast heeft elke package ook andere voor-, en mogelijk nadelen, wat impact heeft op de hoeveelheid tijdsbesparing.

#### Autorisatie

Niet alleen zou het ontwikkelen van deze package meer functionaliteiten opleveren voor de autorisatie, er zou ook veel tijd bespaart kunnen bij het configureren van de autorisatie rechten. Voor de huidige manier van configureren moet voor elke rol apart alle rechten gedefinieerd worden. Een uitbreiding op deze functionaliteit zou kunnen zijn dat rollen rechten van andere rollen zou kunnen erven. Dit zou betekenen dat rechten gegroepeerd kunnen worden in een aantal rollen en dat andere rollen rechten kunnen krijgen tot deze rollen, dus tot een groep rechten. Dit zou, vooral bij grote applicaties, veel tijd in configuratie kunnen schelen.

Er is wel een mogelijkheid dat de meeste ingebouwde functionaliteiten niet gebruikt zullen worden. Dit zou er dus voor zorgen dat er minder tijdsbesparing is dan de ontwikkeling van de package.

#### WCF cachemanager

Naast het zelf moeten ontwikkelen van de functionaliteiten die de package zou bieden, biedt het gebruik van deze package niet meer mogelijkheden tot tijdsbesparing.

Het hoofddoel van de package is om de performance te verhogen in WCF applicaties. Het nadeel van het ontwikkelen van deze package zou zijn, is dat het veel tijd kost om een kleine performance winst te behalen vergeleken tot het proberen verbeteren van een eigen applicatie. Microsoft heeft waarschijnlijk zelf ook al veel tijd besteed aan de performance van WCFT applicaties. De vraag is dus hoeveel performance er nog verbeterd kan worden en of het niet relevanter is te kijken naar het verbeteren van performance bij specifieke applicaties.

#### Templating

De templating service of package zou veel tijd kunnen besparen wanneer deze goed geïmplementeerd zou worden. Vooral de mogelijkheid om templates te kunnen generen op basis van condities en de mogelijkheid voor loops zou hierin een grote aanwinst zijn. Toch zijn vaak deze extra functionaliteiten niet nodig voor simpelere projecten. Er ontstaat hetzelfde gevaar als bij de autorisatie package, dat functionaliteiten niet nodig zijn en dus vaak niet gebruikt worden, waardoor het gebruik van de package minder tijd bespaart dan de ontwikkeling van de package kostte.

## Ontwikkelingstijd van de package

Om te kunnen bepalen of het mogelijk en relevant is (een eerste versie van) de package te ontwikkelen is er gekeken naar de ontwikkeltijd die die nodig is voor het ontwikkelen van de package. Om dit te kunnen bepalen is er gekeken naar de onderdelen die gebouwd moeten worden voor deze package.

### Autorisatie

Voor de autorisatie package zijn er twee mogelijkheden: of de huidige autorisatie wordt uitgebreid of het wordt volledig vervangen. Wanneer deze uitgebreid wordt is er al een basis aanwezig. Dit betekent dat op deze basis goed gebruikt kan worden voor eerste versie. Er zou dus snel een eerste versie ontwikkeld kunnen worden waarop extra uitbreidingen ontwikkeld kunnen worden. Deze uitbreidingen zullen van heel simpel tot een erg complex kunnen worden. Hierin kan gekozen worden wat haalbaar is te implementeren binnen de afstudeerperiode. Wanneer de autorisatie volledig vervangen moet worden ontstaat het gevaar, dat de opdracht te complex wordt of alleen een hele simpele implementatie ontwikkeld kan worden.

Voor deze package en waarschijnlijk voor alle andere packages geldt, dat er oneindig veel features verzonnen en ontwikkeld kunnen worden waardoor er geen maximale ontwikkelingstijd vast te stellen is. Aan de hand van de omschrijving verwacht ik dat de huidige tekortkomingen, door middel van uitbreidingen op de huidige code, in 10 weken kan ontwikkelen.

### WCF cachemanager

Voor het idee van de cachemanager zal er gekeken gaan worden naar de interne processen en stromen in een WCF applicatie. Omdat ik geen ervaring heb met WCF applicaties, is het onzeker hoeveel ontwikkeltijd het zal kosten deze package te ontwikkelen. Waarschijnlijk zal het mogelijk zijn één of enkele processen te verbeteren in 3 weken. Maar om echte performancewinst te krijgen zal volledig onderzoek gedaan moeten worden naar hoe WCF is opgebouwd. Dit zal waarschijnlijk veel meer tijd kosten en niet haalbaar zijn om uit te voeren in de afstudeerperiode.

### Templating

Vanuit mijn stageperiode heb ik ervaring met het ontwikkelen van een systeemonderdeel wat pdf’s kan genereren. Zo’n systeemonderdeel maken en goed laten functioneren kost op zichzelf al minimaal twee weken werk. Als daarnaast ook ondersteuning geboden moet worden voor Word en andere soort documenten zou het al snel mogelijk zijn dat, alleen de mogelijkheid tot genereren van documenten, al teveel tijd kost om te ontwikkelen (voor in de afstudeerperiode). Daarnaast werd in de omschrijving meegegeven, dat de package ook mogelijkheid zou moeten bevatten om condities en loops toe te voegen. Voor html bestaan er al veel implementaties van deze functionaliteiten. Dit wordt ook wel templating of template engine genoemd. Een voorbeeld hiervan is Razor in .Net of Twig in Symfony. Het ontwikkelen van zo’n template engine is niet alleen relatief moeilijk maar kost ook erg veel tijd. Dit zou niet haalbaar zijn om alleen te ontwikkelen in de afstudeerperiode. De kracht van deze package zou hem juist liggen in de mogelijkheden van deze template engine. Daarom is het afgeraden om deze package te ontwikkelen in een afstudeerperiode.

## Conclusie

Op basis van deze informatie heb ik gekozen de autorisatie package te gaan ontwikkelen. Deze optie sloot leek mij het meest haalbaar om uit te voeren in de afstudeerperiode en is daarnaast één van idee dat door de meeste ontwikkelaars is aangedragen. Hierdoor verwacht ik dat deze package ook het meest gebruikt zal worden.

De optie hierna het best geschikt, was het idee voor templating. Het nadeel van dit idee is, dat de afstudeerperiode niet voldoende beschikbaar zou zijn deze package goed uit te werken. Een mogelijkheid zou zijn om één onderdeel uit te werken zoals email of PDF. Het nadeel hiervan zou zijn dat hiervoor eigenlijk al veel packages beschikbaar zijn. Een andere mogelijkheid zou zijn om meerdere packages aan elkaar te koppelen alleen zou dit naar mijn verwachting niet haalbaar zijn om goed af te kunnen ronden in de afstudeerperiode.

Het idee om performance winst te ontwikkelen voor WCF applicaties leek mij het minst geschikt, omdat hiervoor erg onduidelijk was wat haalbaar zou zijn binnen de afstudeerperiode. Daarnaast zou dit veel tijd in onderzoek kosten. Dit zou betekenen dat er minder tijd aan andere dingen besteed zou kunnen worden. Ook sloot deze package niet goed aan op de gekozen beroepstaken (competenties vanuit de opleiding).

Het idee om de autorisatie uit te breiden is een idee wat haalbaar is om te ontwikkelen. Omdat er al een basis aanwezig is kan er in korte iteraties gewerkt worden en zo telkens meer functionaliteit toegevoegd worden. Zo kan er voorkomen worden dat de package aan het eind van de afstudeerperiode nog niet af is. Dit idee biedt ook genoeg mogelijkheden om met de beroepstaken bezig te gaan. Voor het ontwikkelen van de uitbreidingen zal ik wel meer expertise moeten verwerven over hoe de autorisatie nu geïmplementeerd is.

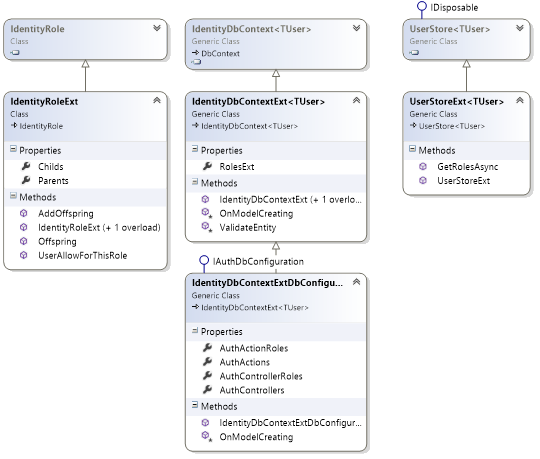
## Discussie en aanbevelingen

Dit onderzoek is een kleine opname van wat er momenteel binnen het bedrijf speelt. Waarschijnlijk zal het zo zijn, dat wanneer dit onderzoek een half jaar later uitgevoerd wordt, er andere resultaten naar boven komen, omdat het bedrijf voortdurend in ontwikkeling is.

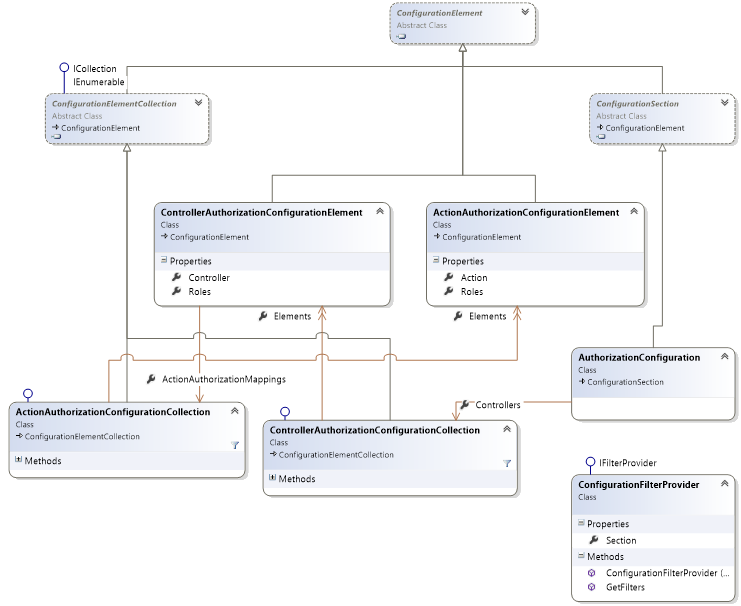
Om een goed en volledige beeld te krijgen waar binnen het bedrijf veel tijd bespaart kan worden in het ontwikkelproces, zou over een tijdsperiode meerdere projecten gemonitord kunnen worden. Op deze manier zou veel beter informatie ingezameld kunnen worden over welke systeemonderdelen herhaaldelijk zelf ontwikkeld worden. Zo zou ook onafhankelijk informatie ingezameld kunnen worden, in plaats van het achteraf interviewen van de ontwikkelaars.

# Bijlage D Klassendiagram versie1

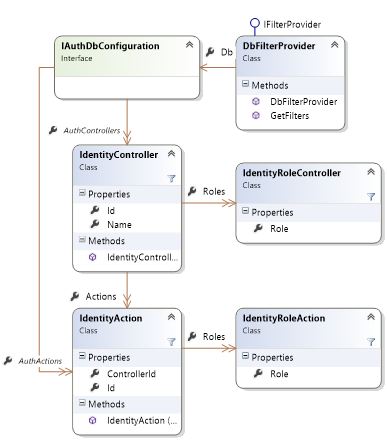
## Uitbreidingen op .Net



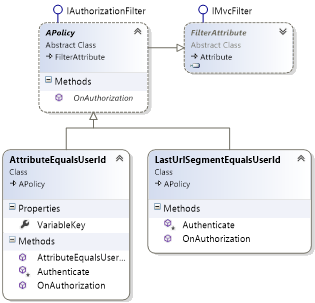
## Autorisatieconfiguratie in configuratiebestand



## Autorisatieconfiguratie in database

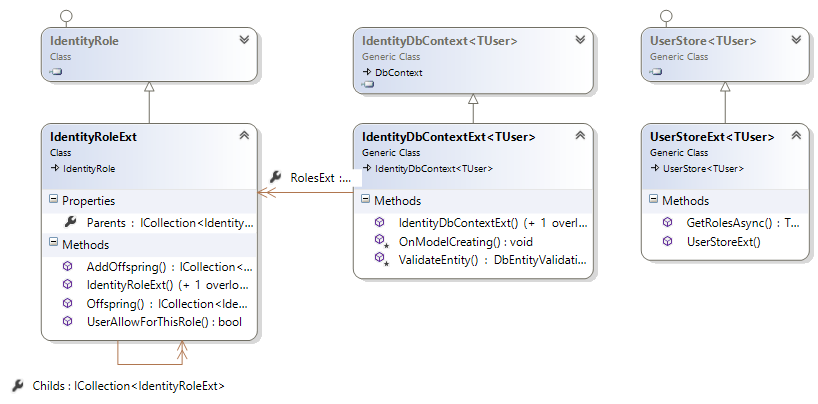


## Structuur Bedrijfsregels

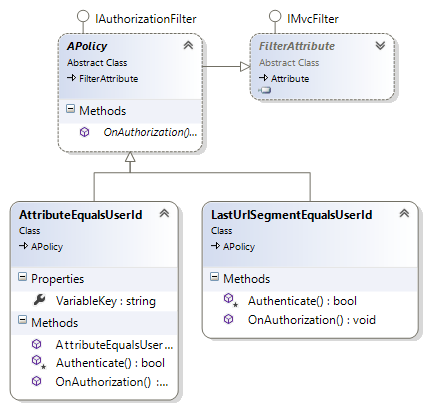


# Bijlage E Klassendiagram versie2

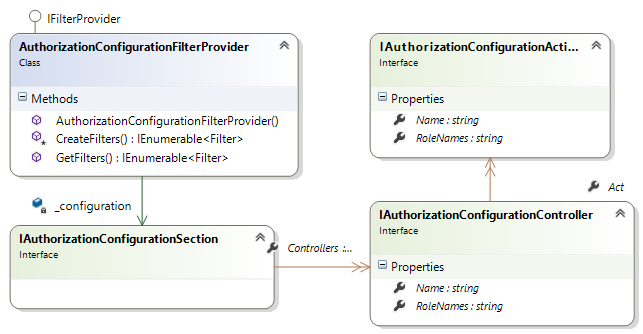
## Uitbreidingen op .Net



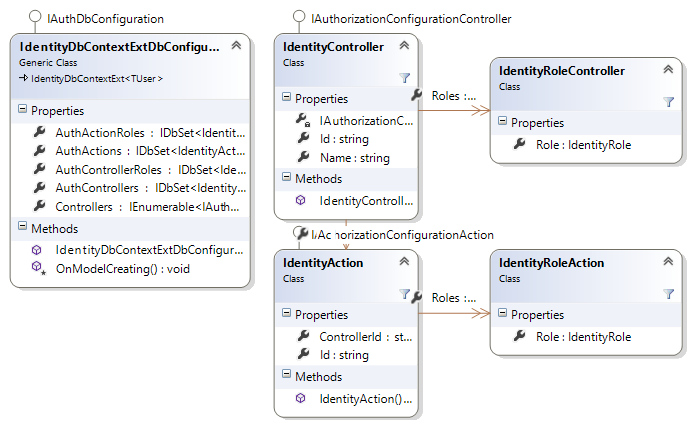
## Structuur bedrijfsregels



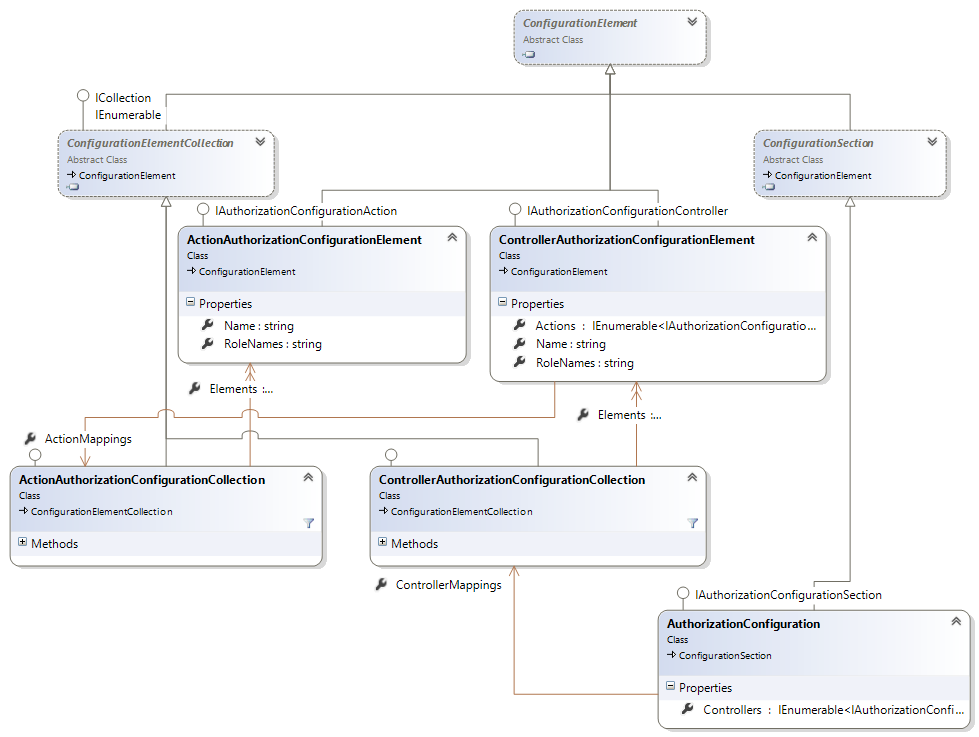
## Structuur autorisatieconfiguratie



## Autorisatieconfiguratie in database



## Autorisatieconfiguratie in configuratiebestand



# Bijlage F Sequentiediagram FilterProvider



# Bijlage G Testrapport derde iteratie

Integratietests - Iteratie 3

## Inleiding

Bij tweede testiteratie werd ontdekt dat de package niet alleen afhankelijk was van de versie van .Net framework, maar ook van alle afhankelijke packages. De package versies zijn zo veel mogelijk verlaagt en er is opnieuw een NuGet package gegenereerd. De NuGet package zal niet alleen getest worden in verschillende versies van .Net maar ook in verschillende versies van de afhankelijke packages. Hierbij zal gebruikt wordt gemaakt van de uiterste, dit betekent dat laagste en de hoogste versie van elke package getest wordt. Hierbij kan het mogelijk zijn dat een test omgeving niet werkend neergezet kan worden omdat alles achteruit teruggedraaid zal moeten worden. Komt dit voor dan zal voor deze versie niet getest kunnen worden of mijn package daarin werkt. Hiervoor is gekozen om de tijd die het testen kost in te perken. Omdat niet met 100% zekerheid gezegd kan worden dat wanneer de package werkt in de oudste en de nieuwste versie deze ook werkt in de versies ertussen, is er voor gekozen steekproefsgewijs nog een aantal versies te testen.

## Werkwijze

Om te bepalen of de package ‘werkt’ in een bepaalde omgeving, zal voor deze omgeving een project opgezet worden en de hoofdzakelijke functionaliteiten getest worden. Dit zal gedaan worden aan de hand van de volgende stappen:

1. Opzetten van project met de juiste omgeving.
2. Test1: Uitvoeren van het project.
   1. Draait de applicatie.
3. Installeren van de package.
4. Updaten van de database. Door de volgende commando’s uit te voeren in de packagemanager:
   1. Enable-migrations
   2. Add-migration Initial
   3. Update-database
5. Test2: Uitvoeren van het project.
   1. Draait de applicatie.
   2. Werken de huidige functionaliteiten nog.
6. Toevoegen van de controllers, acties en configuratie vanuit een eerder testproject.
7. Test3: Uitvoeren van het project.
   1. Draait de applicatie.
   2. Werkt de huidige functionaliteit.
   3. Werken de toegevoegde acties naar behoren.

Zijn de resultaten bij beide tests positief, dan is het testresultaat positief en is de package gemakkelijk te integreren. Zijn één van de testresultaten negatief, doordat bijvoorbeeld de toegevoegde functionaliteit niet werkt of er foutmeldingen teruggeven worden. Aan het eind van de testfase kunnen de negatieve resultaten bekeken worden en gezocht worden naar oplossingen.

Of de huidige functionaliteit nog ‘werkt’ zal getest worden door, door het menu van de web-applicatie te navigeren en eenmaal te registeren en in te loggen.

De test resultaten zullen als volgt genoteerd worden:

|  |  |  |
| --- | --- | --- |
| Actie | Result | Waarneming |
| Build (het compileren van de applicatie) | ‘Sucess’ of ‘Fail’ (‘-‘ wanneer niet uitgevoerd) | Hier worden de opvallende waarnemingen die gedaan worden tijdens het proces. Gebeurd alles volgens behoren dan kan dit veld leeggelaten worden. |
| Run (het draaien van de applicatie) |  |  |
| Functionality (of de applicatie werkt naar behoren) |  |  |
| Functionality of extension (werkt de package naar behoren) |  |  |

Komen er tijdens de tests foutmeldingen, dan zal gekeken worden of deze snel verholpen kunnen worden. De waarnemingen worden genoteerd en er wordt genoteerd welke handelingen zijn uitgevoerd om het probleem te verhelpen.

## Testomgevingen

De tests zullen uitgevoerd worden in de verschillende versies van het .Net, MVC en EF (Entity Framework). De verschillende omgevingen waarin getest zal worden zijn:

1. .Net framework 4.5.1 MVC5.1.2 –packages updated\*
2. .Net framework 4.5.1 MVC5.1.2
3. .Net framework 4.5.0 MVC5.1.2
4. .Net framework 4.5.0 MVC 4
5. .Net framework 4.5.1 MVC5.0.0
6. .Net framework 4.5.1 MVC5.1.2 EF2.0
7. .Net framework 4.5.1 MVC5.1.2 EF2.0.1
8. .Net framework 4.5.0 MVC5.1.0 EF 2.0.0

\*packages updated betekent dat alle packages in het project nog eenmaal geüpdatet worden naar de laatste versie. (Het sjabloon definieert versies van package die gedownload moeten worden, dit is niet voor alle packages de laatste versie)

## Resultaten

### .Net framework 4.5.1 MVC5.1.2 –packages updated

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 2 Project na installeren package

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 3 Project met testcontrollers

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |
| Functionality of extension | Success |  |

### .Net framework 4.5.1 MVC5.1.2

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 2 Project na installeren package

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 3 Project met testcontrollers

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |
| Functionality of extension | Success |  |

### .Net framework 4.5.0 MVC5.1.2

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 2 Project na installeren package

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 3 Project met testcontrollers

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |
| Functionality of extension | Success |  |

### .Net framework 4.5.0 MVC4

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Fail | Error over verkeerde configuratie of verkeerde versie van packages.  Error Razor package komt niet overeen met build nummer van system.webpages.razor. |
| Functionality | - |  |

Opmerkingen: Als eerste werd de error gegeven: method 'System.Web.WebPages.Razor.PreApplicationStartCode.Start()' failed. Door andere packages zoals de webpages package te verlagen kwam er een andere error dat het versie nummer niet meer overeen kwam met het systeem versie nummer van de package.

### .Net framework 4.5.1 MVC5.0.0

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success | Error couldn’t load version number MVC 5.1 |
| Run | Success |  |
| Functionality | Success |  |

Opmerkingen: er bleek in de web.config nog een versie 5.1 van de mvc package opgevraagd, dit is angepast naar 5.0 en toen was deze error verholpen.

#### Test 2 Project na installeren package

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 3 Project met testcontrollers

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |
| Functionality of extension | Success |  |

### .Net framework 4.5.1 MVC5.1.2 EF2.0

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 2 Project na installeren package

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 3 Project met testcontrollers

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |
| Functionality of extension | Success |  |

### .Net framework 4.5.1 MVC5.1.2 Mic EF2.0.1

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 2 Project na installeren package

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 3 Project met testcontrollers

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |
| Functionality of extension | Success |  |

### .Net framework 4.5.0 MVC5.1.0 EF 2.0.0

#### Test1 Basis project

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 2 Project na installeren package

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |

#### Test 3 Project met testcontrollers

|  |  |  |
| --- | --- | --- |
| Action | Result | Observations |
| Build | Success |  |
| Run | Success |  |
| Functionality | Success |  |
| Functionality of extension | Success |  |

## Conclusie

Op één test na werden alle tests succesvol voltooid. Deze test kon niet afgerond worden omdat het ontwikkelproject niet werkend gekregen kon worden met de oudere versie van MVC. Hierdoor kon ook niet getest worden of de NuGet package werkte in deze omgeving.

In alle tests die uitgevoerd konden worden werkte de package zonder problemen. Alleen voor de versie 4.0 en 4.4 van de MVC package is het onduidelijk of de package werkt. Voor deze versie van de package kon in de ondersteunde .Net versies geen werkende web-applicatie opgezet worden.

# Bijlage H Goedgekeurde afstudeeropdracht

Module ontwikkelen en testen voor in C# .NET web applicaties.

## Bedrijf

Sogeti is een dochteronderneming van de capgemini-groep. Het is een (profit) IT consultancy bedrijf gespecialiseerd in lokale professionele diensten. Het is een bedrijf met 20.000 mensen werkend over 15 landen waar Nederland er één van is. Ze heeft vestigingen in: Vianen, Amersfoort, Diemen, Groningen en Maastricht. Sogeti is opgericht in 1967 en is sinds 2002 in Nederland een BV. Het bedrijf in Nederland kende in 2012 2683 medewerkers en een winst van 3 miljoen.

Sogeti is een ICT detacheringsbedrijf wat betekent dat ze IT opdrachten doen voor klanten. De grootste diensten die zij leveren zijn ontwikkeling, testing, security en hightech (ontdekken en gebruiken van nieuwe technieken). De afstudeer opdracht zal gedaan worden in Amersfoort in de studenten-unit. Dit is een unit die zich bezig houdt met software ontwikkeling waarbij veel ruimte wordt geboden voor professionele begeleiding.

Meer informatie over het bedrijf is te vinden op wikipedia: <http://nl.wikipedia.org/wiki/Sogeti>

## Probleemstelling

Het bedrijf maakt gebruik van een zelfgebouwde web-applicatie als basis voor haar ontwikkelomgevingen. De zelfgebouwde software heet SARA en is zowel een framework op het MVC4 framework in .NET als een basis applicatie die meteen gerund kan worden. In praktijk blijkt SARA zo uitgebreid dat het te omslachtig is om te gebruiken en het daardoor niet gebruikt wordt.

Het bedrijf bouwt veel web applicaties en wil daarom veel gebruikte onderdelen standaardiseren en zo kunnen implementeren in toekomstige web applicaties omdat dit veel tijd kan schelen en de kwaliteit (in koppelbaarheid, foutbestendigheid en vervangbaarheid) verhoogt.

## Doelstellingen van de afstudeeropdracht

* Onderzoek doen naar veel gebruikte onderdelen van verschillende applicatieonderdelen en de noodzaak (voor Sogeti’s ontwikkelaars) en mogelijkheid voor een standaard module.
* Ontwikkelen en testen van de module die goed integreerbaar is in (in de structuur van) toekomstige webapplicaties van Sogeti.
* Opzetten van een omgeving waarin de applicatie doorontwikkeld, beheerd en gedistribueerd kan worden.

## Resultaat

Bij het afronden van het project is er een module waarvan de kwaliteit is vastgesteld en die door ontwikkelaars van Sogeti gebruikt kan worden in web applicaties.

## Planning

Week 1 – Oriëntatie (1 week)

* Oriëntatie, kennismaking collega’s en bedrijfsprocessen.
* Project opstart, Management door PRINCE2.
  + Schrijven van het PID (plan van aanpak)

Week 2 – Onderzoek naar noodzaak van applicatie onderdelen en eisen inzamelen (3 weken)

* Kwalitatief onderzoek (d.m.v. interviews) naar de noodzaak voor een module voor verschillende applicatie-onderdelen.
* Op basis van dit onderzoek wordt een onderdeel gekozen waar een module van geschreven wordt.
* De requirementsrapport opstellen:
  + wensen, eisen en relevante bedrijfsprocessen voor deze module worden opgehaald (door interviews) en gerapporteerd.

Week 6 – Ontwerp (1 week)

De modellen van de module in UML:

* Integratie model, Waar moet de module op aansluiten en hoe moet het in de applicatie geïntegreerd worden.
* Klassendiagram, structuur van de module. Het klassen diagram wordt ontworpen aan de hand van het SOLID[[9]](#footnote-9) principe.

Week 7 – Bouwen (6 weken)

Bouwen van de applicatie aan de hand van:

* Iteratief ontwikkelen op basis van Scrum sprints.
* Testgedreven. In elke sprint worden bijbehorende unittests geschreven.

Week 13 - Testen (3 weken)

In deze fase zal de applicatie getest worden met TMap en door de volgende tests:

* Een integratietest. Test of de module gemakkelijk in huidige applicaties geïntegreerd kan worden.
* Een exploratietest. Door de exploratie test zullen de belangrijkste en de meest foutgevoelige onderdelen getest worden.

Week 16 – Oplevering (1 week)

* Handleiding schrijven voor gebruik van de module.
* Distributie omgeving opzetten van waar de module gedownload kan worden als NuGet[[10]](#footnote-10) package.
* Ontwikkelomgeving laten aansluiten op de distributie omgeving zodat de module beheerbaar is.

## Op te leveren (tussen)producten

De producten die opgeleverd zullen worden zijn:

* PID (Plan van aanpak)
* Requirementsrapport
* Ontwerprapport met daarin klassendiagram en integratiediagram en toelichting op de ontwerpkeuzes.
* Test Resultaten in een testrapport.
* Broncode en Nuget package van de geschreven module.
* Distributie en beheer omgeving.

## Te demonstreren competenties en wijze waarop

Bij het uitvoeren van dit project zullen de volgende competenties op hoog niveau aangetoond worden.

|  |  |  |
| --- | --- | --- |
| *Beroepstaak* | *Niveau* | *Motivatie* |
| 3.2 Ontwerp systeemdeel | Niveau 3 | Voor een module die in meerdere projecten gebruikt gaat worden is het erge belangrijk dat de kwaliteit van het ontwerp goed is. Daarbij is het belangrijk dat gelet wordt op gebruik van de juiste design patterns en gebruik van SOLID principe (zoals uitgelegd bij de planning. |
| 3.3 Bouw applicatie | Niveau 4 | Het grootste deel van het project zal bestaan uit software ontwikkeling. Het ontworpen module zal uitgebouwd moeten worden. Hierbij zal ook goed gelet moeten worden op het SOLID principe wat moet resulteren en goed object georiënteerde code. |
| 3.5 Uitvoeren van en rapporteren over het testproces | Niveau 3 | Naast bouwen van de software, zal ook de kwaliteit van de software gewaarborgd moeten worden. Dit zal gebeuren door het schrijven van unit tests. Hierbij zal gezorgd moeten worden voor een zeer hoog percentage dekking. Daarnaast zal na het bouwen nog extra tijd besteedt worden aan het testen van de software zodat de kwaliteit van de module goed is vastgelegd. |
| 4.4 Beheren en distribueren van sotware | Niveau 3 | De gebouwde module moet na het testen ook gemakkelijk bruikbaar zijn. Hiervoor is het belangrijk dat de software via een distributie programma gemakkelijk gedistribueerd kan worden. |

# Bijlage I Formulier bespreking concept afstudeerdossier

|  |  |  |
| --- | --- | --- |
| **Bespreking concept** | **Tussentijds assessment** | **Eerste beoordeling** |

**Formulier bespreking concept afstudeerdossier**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student:** | J.I. Harkes, Joël | **Studentnummer:** | 10009477 |
| **Datum:** | 24 april 2014 |  |  |

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

**Verbeterpunten:**

Nederlands verbeteren, vooral grammaticaal.

Niet het resultaat van het onderzoek beschouwen als de uit te voeren opdracht maar het ingeleverde afstudeerplan.

De context ontbreekt bij het onderzoek. Sara wordt niet genoemd.

Planning ontbreekt; requirements ontbreken.

De grote lijn, het overzicht, ontbreekt.  
Eerst de grote lijn beschrijven, daarna pas de details/implementatie.

Doel van het prototype beter beschrijven.

Bij het beargumenteren van gemaakte keuzes ook eventuele alternatieven beschrijven.

Neem figuren op om je verhaal te ondersteunen.

Terugkoppelen naar de gebruikte SO-methode en/of Prince 2, zeker bij wijzigingen in de planning

**Opmerkingen:**

Het afstudeerdossier bevat wel het onderzoek maar nog geen requirements. Hierdoor en ook door de magere staat van het afstudeerverslag is het niet mogelijk om vast te stellen of de opdracht naar behoren is uitgevoerd.

**Naam begeleidend examinator:** A. Lousberg

**Datum:** 24 april 2014

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

# Bijlage J Formulier tussentijds assessment

|  |  |  |
| --- | --- | --- |
| **Bespreking concept** | **Tussentijds assessment** | **Eerste beoordeling** |

**Formulier tussentijds assessment**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student:** | Joel Harkes | **Studentnummer:** | 10009477 |
| **Datum:** | 15 mei 2014 | **eerste / tweede TTA:** | eerste |

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Aanpak** | **O** | **T** | **V** | **G** |
| *Passend* |  |  | X |  |
| *Theoretisch verantwoord* | X |  |  |  |
| *Samenhang uitvoering beroepstaken* |  | X |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Beroepstaken op afgesproken niveau uitgevoerd?** | | **O** | **T** | **V** | **G** |
| 1 | 3.2 Ontwerp systeemdeel |  | X |  |  |
| 2 | 3.3 Bouw applicatie |  | X |  |  |
| 3 | 3.5 Uitvoeren van en rapporteren over het testproces | X |  |  |  |
| 4 | 4.4 Beheren en distribueren van software | X |  |  |  |
| 5 |  |  |  |  |  |
| 6 |  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Producten** | **O** | **T** | **V** | **G** |
| *Tussenproducten* |  | X |  |  |
| *Eindproducten* | X |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Effectief communiceren** | **O** | **T** | **V** | **G** |
| *Binnen afstudeerbedrijf* | ontbreekt | | | |
| *Afstudeerdossier* |  | X |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Reflectie** | **O** | **T** | **V** | **G** |
| *Inzicht in eigen functioneren* | ontbreekt | | | |
| *Inzicht in eigen leerproces* |  |  | X |  |

**Toelichting per beoordelingscriterium**

|  |
| --- |
| **Aanpak** |
| De aanpak om te werken met interviews/onderzoek en het uitwerken naar een prototype is passend voor deze opdracht. Theorie klopt niet altijd, bijvoorbeeld opmerkingen over PRINCE2 en verwijzingen naar relevante bronnen ontbreken.  De probleemstelling en doelstelling voor het te ontwikkelen autorisatiepackage zijn onvoldoende concreet uitgewerkt. |

|  |
| --- |
| **Beroepstaken op afgesproken niveau uitgevoerd?** |
| **Het hoofdstuk Testen ontbreekt geheel in het verslag. Een ontwerp is aanwezig in de vorm van een aantal eenvoudige diagrammen.**  **Door onduidelijke probleem- en doelstelling is het niet vast te stellen of het ontwerp passend is.** |

|  |
| --- |
| **Producten** |
| Het eindproduct is niet zichtbaar in het verslag. De hoofdstukken 9.5 over Sprint 3 (Script nuget Package), 9.6 Conclusie, 10 Testfase en 11 Distributiefase zijn nog helemaal leeg. Dat maakt beoordeling niet mogelijk |

|  |
| --- |
| **Effectief communiceren** |
| Interviews vormden een belangrijk onderdeel van de communicatie binnen het bedrijf. Hoe goed dat verlopen is, is niet duidelijk op te maken uit het verslag. |

|  |
| --- |
| **Reflectie** |
| Door het verslag heen wordt er in de vorm van conclusies en discussies wel gereflecteerd op het gevolgde proces en de lering daaruit. Een afrondend hoofdstuk over het werk als geheel ontbreekt. |

**Advies**

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

**Besluit student**

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

|  |  |
| --- | --- |
|  | **Afstudeerdossier wordt op afgesproken datum ingeleverd** Inleverdatum: |
|  | **Afstudeerperiode wordt verlengd** Inleverdatum: |
|  | **Student stopt met afstudeeropdracht** |

**Naam begeleidend examinator:** A.M.J.J. Lousberg

**Naam tweede examinator:** G.M. Tuk

**Datum:** 15 mei 2014

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

1. <http://nl.wikipedia.org/wiki/ISO_9126> [↑](#footnote-ref-1)
2. MVC Authorization <https://mvcauthorization.codeplex.com/> [↑](#footnote-ref-2)
3. Meer informatie op <http://en.wikipedia.org/wiki/SOLID_(object-oriented_design)> [↑](#footnote-ref-3)
4. <http://stackoverflow.com/questions/23602948/rename-classes-with-powershell> [↑](#footnote-ref-4)
5. <http://en.wikipedia.org/wiki/Composite_pattern> [↑](#footnote-ref-5)
6. <http://www.nuget.org/> [↑](#footnote-ref-6)
7. <http://autofac.org/> [↑](#footnote-ref-7)
8. <http://automapper.org/> [↑](#footnote-ref-8)
9. In programmeer/ontwerp principe dat de kwaliteit van de code en ontwerp verhoogt vooral op gebied van beheerbaarheid en uitbreidbaarheid. <http://en.wikipedia.org/wiki/SOLID_(object-oriented_design)> [↑](#footnote-ref-9)
10. NuGet is een distributie applicatie voor C#, die de mogelijkheid biedt om pakketten te maken en te delen en te downloaden (en automatisch installeren). <http://www.nuget.org/> [↑](#footnote-ref-10)