Examinator:

Jos van Dongen

Opdrachtgevers:

Jochem Aubel
Stefan Op de Woerd

Organisatie: Perflectie

Een onderhoudbaar en uitbreidbaar online-platform voor Perflectie.

Afstudeerscriptie

Sebastiaan Alblas, 1605265
15-12-2014

I Versie beheer

Versie	Auteur	Status	Beschrijving	Datum
V1.0	Bas Alblas	Concept	-Eerste opzet document -Voorwoord toegevoegd -Opzet management samenvatting -Een onderhoudbaar en uitbreid online-platform voor Perflectie toegevoegd	15-10-2014
V1.1	Bas Alblas	Concept	-Programmeertaal onderzoek	04-11-2014
V1.2	Bas Alblas	Concept	-Architectuur onderzoek	20-11-2014
V1.3	Bas Alblas	Concept	-Herstructureren scriptie	08-12-2014
V1.4	Bas Alblas	Concept	-Persoonsgegevens verwerken onderzoek	09-12-2014
V1.5	Bas Alblas	Concept	-Iteratieve aanpak -Systeem architectur	10-12-2014
V1.6	Bas Alblas	Concept	-iteration plan -Moscow -De gebruikte software	11-12-2014
V1.7	Bas Alblas	Concept	-Het opzetten van de ontwikkelomgeving - De resultaat van de gebruikte methodes	12-12-2014
V1.8	Bas Alblas	Concept	-Implementeren van design oplossingen -Conclussie	13-12-2014
V1.9	Bas Alblas	Concept	-Eind conclussie -Aanbevelingingen -Toekomstige onderzoek -Evaluatie van de procesgang	13-12-2014
V1.10	Bas Alblas	Concept	-Controle op spelling en zinsopbouw	13-12-2014
V1.11	Bas Alblas	Concept	-Managementsamenvatting voltooid -Controle op spelling en zinsopbouw	14-12-2014

V1.12	Bas Alblas	Concept	-Inleiding voltooid -Controle op spelling en zinsopbouw	14-12-2014
V1.13	Bas Alblas	Definitieve versie	-Toevoegen van verklarende woordenlijst -Het reduceren van pagina's -Laatste controle op leesbaarheid	15-12-2014

II Voorwoord

Mijn naam is Sebastiaan Alblas, ik ben een vierdejaars student aan de Hogeschool Utrecht (HU). Ik volg de opleiding informatica, richting software engineer, met als specialisatie game development. Dit document bevat mijn afstudeerscriptie en behandeld de afstudeeropdracht die ik bij Perflectie heb mogen uitvoeren.

Ik heb de opdracht van Perflectie voor mijn afstudeerstage gekozen, omdat deze de elementen bevat waar ik de afgelopen 4 jaar voor gestudeerd heb. Gamification, het implementeren van complexe architectuur- en design patterns, het analyseren en verbeteren van een bestaand systeem en nog veel meer. Ik heb mijn kennis op een juiste manier in een hiervoor gepaste omgeving kunnen toepassen en kunnen bewijzen dat ik de juiste competenties bezit voor het uitvoeren van deze opdracht.

Tijdens de uitvoer van het project ben ik tegen veel uitdagingen aangelopen. Mijn initiële opdracht was het bestaande online-platform van Perflectie uitbreiden met een Partner module. Hiervoor heb ik eerst onderzoek gedaan om kennis op te doen over het huidige online-platform. De conclusie uit dit onderzoek was dat het niet verantwoord was om hier verder op te ontwikkelen. Hierdoor is mijn afstudeeropdracht aangepast naar; realiseer een nieuw onderhoudbaar en uitbreidbaar online-platform voor Perflectie inclusief de Partner module.

Tijdens mijn stage was ik onderdeel van een fantastisch team. Elk teamlid had uitgebreide kennis van zijn vakgebied en het was een geweldige ervaring om te kunnen leren van deze mensen.

Jochem Aubel wil ik bedanken voor zijn begeleiding. Jochem stond altijd klaar om mee te denken, op deze manier zijn we tot een geweldig product gekomen.

Dimitri Tholen wil ik bedanken omdat hij met geduld zijn kennis met mij heeft gedeeld, dit was heel leerzaam. Dimitri zorgde ervoor dat ik zonder problemen alle nieuwe software en programmeer methodes oppakte. Deze technieken en methodes zal ik in de toekomst zeker weer gebruiken.

Jesse Buitenhuis wil ik bedanken voor zijn kennis van de front-end. Jesse heeft in samenwerking met Dimitri een erg mooie architectuur neergezet binnen AngularJS waarbij een client-side is ontwikkeld met een hoge kwaliteit. Deze technieken en methodes waren erg leuk om te leren en ook weer bruikbaar in de toekomst.

Pascualle Vermeulen wil ik graag bedanken, zij heeft mij geholpen met het zoeken naar mijn persoonlijke competenties en daarnaast ook in het stellen van doelen waar ik mijzelf verder op kon ontwikkelen.

Serge Bekenkamp wil ik bedanken voor zijn inzet en hulp bij het ontwikkelen van het product. Als part-time student kan het lastig zijn als je in één keer in het diepe wordt gegooid in een project. Serge pakte de kennis en methodes snel op.

Stefan Op de Woerd wil ik graag bedanken voor zijn inzet om van Perflectie een groot succes te maken. Stefan gaf dagelijks de positieve ontwikkelingen van Perflectie door waarbij ik het gevoel kreeg dat wij een geweldig product neer gingen zetten welke door belangrijke coöperaties wordt gebruikt.

Mijn vader Bert Alblas en mijn zus Suzanne Alblas wil graag bedanken voor hun steun tijdens mijn stage en hun hulp bij het controleren van mijn documenten.

Als laatste wil ik Petra Belgraver bedanken voor haar geduld en geweldige inzet tijdens mijn afstudeerstage. Petra zorgde voor constructieve feedback waardoor ik mijn stage documenten succesvol heb kunnen afronden.

Tot slot wens ik u veel plezier met het lezen van mijn scriptie. Ik hoop dat u van dit document evenveel leert als ik heb gedaan tijdens mijn stageperiode.

III Managementsamenvatting

Dit document beschrijft het afstudeeronderzoek voor het afronden van de HBO opleiding Informatica aan de Hogeschool Utrecht (HU). De opdracht om voor Perflectie (Perflectie website, 2014) een onderhoudbaar en uitbreidbaar online platform te realiseren staat centraal in dit document.

Perflectie is een bedrijf die organisaties en mensen helpt succesvol te veranderen. Dit doen zij met een gelijknamig online platform waarin persoonlijke ontwikkelingen kunnen worden vastgelegd en gevolgd. Perflectie werkt met partners, die ook gebruik maken van het platform. Voor deze Partners is de partnermodule bedoeld zodat zij zelf functionaliteiten en personalisatie in het platform kunnen aanbrengen.

Mijn initiële opdracht was het toevoegen van de hierboven beschreven partner module aan het online platform van Perflectie. Tijdens het vooronderzoek is er uitgebreid onderzoek gedaan naar de eisen onderhoudbaarheid en uitbreidbaarheid en de inpasbaarheid van de Partner module in het huidige online-platform van Perflectie. De conclusie van dit onderzoek toonde aan dat het niet verantwoord is om in het huidige online-platform verder door te ontwikkelen omdat deze niet voldoet aan de eisen van onderhoudbaarheid en uitbreidbaarheid.

Hierop is de afstudeeropdracht aangepast naar

> *"Ontwerp en herbouw het Perflectie online-platform met de partner module inbegrepen, zodat het nieuwe systeem onderhoudbaar en uitbreidbaar is, voor 2 december."*

Deze scriptie richt zich dan ook op het onderwerp, op welke manier een nieuw online-platform voor Perflectie zo onderhoudbaar en uitbreidbaar mogelijk ontwikkeld kan worden binnen de eisen en wensen van Perflectie. Vervolgens is de opdracht opgesteld om een onderzoek uit te voeren om te komen tot een onderhoudbaar en uitbreidbaar systeem.

Binnen dit onderzoek, om Perflectie's nieuwe online-platform zo onderhoudbaar en uitbreidbaar mogelijk te ontwikkelen, staat de volgende hoofdvraag centraal:

> *"Hoe moet het nieuwe online-platform voor Perflectie worden ontworpen zodat, met de partner module inbegrepen, het systeem onderhoudbaar en uitbreidbaar is? "*

Om deze vraag te beantwoorden is de hoofdvraag opgedeeld in vier deelvragen:

1. *Wat maakt een systeem onderhoudbaar en uitbreidbaar en aan welke voorwaarden moet het systeem voldoen?*
2. *Wat zijn de eisen van Perflectie voor het nieuwe online-platform?*
3. *Welke structuur of structuren zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*
4. *Welke design oplossingen zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*

Om een geschikte projectmanagementmethode te vinden voor de realisatie van het nieuwe online-platform is er een vergelijkingsonderzoek uitgevoerd. Dit onderzoek heeft geleidt tot de Agile Scrum methode omdat deze het best past bij Perflectie, die volgens het LEAN startup principe producten ontwikkeld. Agile scrum is een iteratieve manier om projecten uit te voeren. Na iedere iteratie wordt een prototype gemaakt die getest kan worden, de functionaliteiten van dit prototype worden vooraf vastgesteld bij iedere iteratie. Binnen elke iteratie worden de functionele eisen, die benodigd zijn om een iteratie te voltooien, gedocumenteerd in het Iteration plan.

Om het nieuwe online-platform vooraf onderhoudbaar en uitbreidbaar op te zetten, is er een programmeertaal en architectuur vergelijkingsonderzoek gestart, dat gericht was op web applicaties.
Binnen het programmeertaal vergelijkingsonderzoek werden de actuele programmeertalen met elkaar vergeleken om zo te bepalen welke het best toepasbaar is voor het nieuwe online-platform. Hieruit kwam als resultaat het full stack web framework. Het full stack web framework dekt alle onderdelen binnen een web applicatie af. Op deze manier hoef je geen aparte backend programmeertaal binnen je applicatie te implementeren.

Vanuit het architectuur onderzoek werden de actuele architecturale style en patterns met elkaar vergeleken. Binnen dit onderzoek werden de voordelen van een architecturale style verzameld. Uit deze lijst werd door

Perflectie de meest belangrijke gekozen. Met dit als uitgangspunt werd een bijpassend advies gegeven aan Perflectie voor een combinatie van architecturale patterns. De Client/server architecturale style met een N-Tier structuur bleek het beste geschikt als architectuur voor het nieuwe online-platform van Perflectie.

Bij Agile development past Test Driven Development (TDD). Binnen TDD wordt een werkproces aangehouden genaamd red, green en refactor. De eisen, waar de gemaakte functionaliteiten aan het einde van dit werkproces aan moeten voldoen, zijn opgenomen in de definition of done. Het werkproces van red, green en refactor is door het team uitgebreid met meer (eigen) methodes en principes. Zo is het proces aangepast naar plan, red, green en refactor en zijn de volgende principes toegevoegd in de definition of done: Keep It Simple Stupid (KISS) principe, clean code principe en de S.O.L.I.D principes.

Om te meten of het systeem onderhoudbaar en uitbreidbaar is opgezet, is het op dezelfde manier beoordeeld als het oude online-platform beoordeeld is binnen het vooronderzoek. Om te meten of de gemaakte code en functionaliteiten onderhoudbaar zijn, zijn er interviews gehouden met de ontwikkelaars van Perflectie. Hierbij zijn dezelfde vragen gebruikt die ook in het vooronderzoek zijn gebruikt. Hier is uitgekomen dat het nieuwe platform op veel punten is verbeterd maar dat er nog diverse aandachtspunten zijn, voor deze aandachtspunten is een advies opgesteld.

Uit dit afstudeeronderzoek is af te leiden dat er geen standaard oplossing beschikbaar is voor het realiseren van een onderhoudbaar en uitbreidbaar systeem. Om hier te komen moet er een proces doorlopen worden om de gewoontes, de principes, de methodes en de technieken van het ontwikkelteam en het bedrijf op een iteratieve en leerzame manier aan te passen richting onderhoudbaarheid en uitbreidbaarheid. Het begint met bewustwording en verandering van gedrag, waarbij de principes, methodes en technieken ondersteunen om het gewenste resultaat te bereiken.

Inhoudsopgave

I Versie beheer 2
II Voorwoord 4
III Managementsamenvatting 5
1 Inleiding 9
2 Vooronderzoek 9
2.1 De initiële opdracht 9
2.2 Analyse van het huidige online-platform 10
2.3 Wat houdt een onderhoudbaar en uitbreidbaar online-platform voor Perflectie in? 11
2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen? 11
3 De opdracht 12
3.1 De opdracht 12
3.2 De eisen waar het systeem aan moet voldoen 13
3.3 De hoofdvraag met zijn deelvragen 13
3.4 Conceptueel model 13
3.5 Methode van onderzoek 16
3.6 Een beschrijving van het uitgangspunt 16
4 Theoretische onderzoek 16
4.1 Programmeertaal 16
4.1.1 Generaties programmeertalen 16
4.1.2 Structuur van een web applicatie 17
4.1.3 Backend programmeertalen 17
4.1.4 Client-side programmeertalen 22
4.1.5 Dataopslag 23
4.1.6 Conclusie te adviseren programmeertaal 24
4.2 Geschikte architecturale patterns voor het nieuwe online-platform van Perflectie 25
4.2.1 Wat is een architecturale pattern? 25
4.2.2 Design patterns 26
4.2.3 Hoofdvoordelen van een structuur gebaseerd op een architecturale pattern 26
4.2.4 Gekozen eisen van Perflectie 28
4.2.5 Definitie van architecturale patterns 28
4.2.6 Conclusie geschikte architecturale patterns 32
5 Iteratieve aanpak 36
5.1 De gebruikte methodes 36
5.1.1 Agile development 36
5.1.2 Principes, technieken en methodes 38
5.2 Systeem architectuur 40
5.2.1 Te realiseren eisen 41

5.2.2 Gekozen programmeertalen & architectuur 41
5.2.3 Architecture notebook 42
5.3 Iteration plan 43
5.4 MoScoW 43
6 De resultaten van de opdracht 44
6.1 Resultaat van gebruikte methodes 44
6.1.1 Projectmanagement 45
6.1.2 Toegepaste ontwikkel methodes 45
6.3 Implementeren van design oplossingen 48
6.4 Conclusie 48
6.4.1 Gemaakte code 48
6.4.2 Het nieuwe online-platform 48
6.4.3 Vergelijking 50
7 Eind Conclusie 50
7.1 *Hoe moet het nieuwe online-platform voor Perflectie worden ontworpen zodat, met de partner module inbegrepen, het systeem onderhoudbaar en uitbreidbaar is?* 50
8 Aanbevelingen 51
9 Toekomstige onderzoeken 52
10 Evaluatie van de procesgang 53
Verklarende woordenlijst 55
Geciteerde werken 56
Bijlage 59
A : Perflectie, Netwinst en positie van de student 59
B : Management methode: Agile Scrum 60
C : Verwerken van persoonsgegevens binnen de Nederlandse wet 70
D : Vooronderzoek 78
E : Concept architecture notebook 85
F : Nieuwe architecture notebook 92
G : Plan van aanpak 100
H : Iteration plan 113
I : Interviews : Het nieuwe online-platform & gemaakte functionaliteiten 151
J : DOMAIN UML DIAGRAM PERFLECTIE, 11-08-2014 154
K : Uitgewerkt concept Merge functionaliteit 155
L : De gebruikte software & extensies 158
M : Het opzetten van de ontwikkelomgeving 160

1 Inleiding

Perflectie helpt organisaties en mensen succesvol te veranderen. Dit doen zij met veel passie en overtuiging. Ze gaan hierbij altijd uit van de talenten van mensen en maken hen zelf verantwoordelijk voor de verandering. Zonder persoonlijke verantwoordelijkheid verandert er namelijk niets. (Perflectie website, 2014)

Perflectie is een klein bedrijf die, in het begin van dit project, samenwerkte met Netwinst om zijn huidige online-platform te realiseren en te onderhouden. De structuur en het organogram van Perflectie & Netwinst is te vinden in bijlage A : Perflectie, Netwinst en positie van de student.

Persoonlijke ontwikkeling is erg belangrijk maar wel complex te organiseren. Veel bedrijven willen hun personeel aanmoedigen om zichzelf verder te blijven ontwikkelen. Huidige trainingen en trajecten kosten veel geld en hebben vaak weinig tot geen resultaat. Binnen een paar weken valt een persoon alweer terug in zijn of haar oude gewoontes. Er is meer nodig om het gewenst gedrag te ontwikkelen en hiervoor is Perflectie de aangewezen partner.

Dit document beschrijft het afstudeeronderzoek voor het afronden van de HBO opleiding Informatica aan de Hogeschool Utrecht (HU), met als eindresultaat een advies voor het nieuwe online-platform van Perflectie. In dit document wordt ook verslag gedaan van de afstudeeropdracht bij Perflectie, waarin de volgende opdracht centraal staat

> *"Ontwerp en herbouw het Perflectie online-platform met de partner module inbegrepen, zodat het nieuwe systeem onderhoudbaar en uitbreidbaar is, voor 2 december."*

Voor een succesvol resultaat is het volgende onderzoek meegenomen, waarbij de vraag centraal staat

> *"Hoe moet het nieuwe online-platform voor Perflectie worden ontworpen zodat, met de partner module inbegrepen, het systeem onderhoudbaar en uitbreidbaar is? "*

Om de hoofdvraag te beantwoorden is deze opgedeeld in vier deelvragen, namelijk:

1. *Wat maakt een systeem onderhoudbaar en uitbreidbaar en aan welke voorwaarden moet het systeem voldoen?*
2. *Wat zijn de eisen van Perflectie voor het nieuwe online-platform?*
3. *Welke structuur of structuren zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*
4. *Welke design oplossingen zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*

2 Vooronderzoek

2.1 De initiële opdracht

Perflectie heeft een methodiek voor gedragsverandering die wordt ondersteund door een online platform (web applicatie). Dit platform wordt momenteel gebruikt door partners zoals Achmea, Independer, Reaal en Vodafone.

Voor Perflectie is een Partner module ontwikkeld. Deze module zorgt ervoor dat een Partner de omgeving van zijn online-platform naar eigen wens kan aanpassen. Bij het begin van mijn stage was de initiële opdracht de Partner module verder te ontwikkelen op het huidige online-platform van Perflectie.

De partner module geeft de functionaliteit dat vooraf gedefinieerde componenten toegevoegd kunnen worden door de Partner. Perflectie kan zelf ook de rol van Partner invullen en moet ook in staat zijn gedefinieerde componenten te kiezen. Een Partner kan op zijn beurt weer een Perflectie component als basis gebruiken. Dit component kan de Partner personaliseren zonder het originele component aan te tasten. Wanneer Perflectie een verandering doorvoert in dit component en de Partner heeft de attribuut niet gepersonaliseerd, dan krijgt hij de verandering doorgevoerd in zijn omgeving. Zo heeft de partner altijd de beschikking over de laatste versie van het gekozen component.

Een partner kan gebruikers uitnodigen in zijn omgeving. Een gebruiker kan een programma (Een programma kan bijvoorbeeld een trainingsprogramma zijn bestaande uit meerdere ontwikkelingsdoelen met bijbehorende stappen) volgen die de partner binnen zijn omgeving heeft gedefinieerd. De componenten binnen een programma moeten door een gebruiker te volgen zijn en de gebruiker moet deze ook kunnen personaliseren.

Dit alles vraagt om een duidelijk inzichtelijk systeem dat gemakkelijk onderhoudbaar, maar ook uitbreidbaar is.

Het uitgangspunt van de opdracht, die ik in augustus 2014 heb gekregen van Jochem Aubel, luidde als volgt:

> *Ontwikkel voor Perflectie een Partner module in het online-platform van Perflectie , met als doel dat een Partner de omgeving van zijn platform naar eigen wens kan aanpassen.*

2.2 Analyse van het huidige online-platform

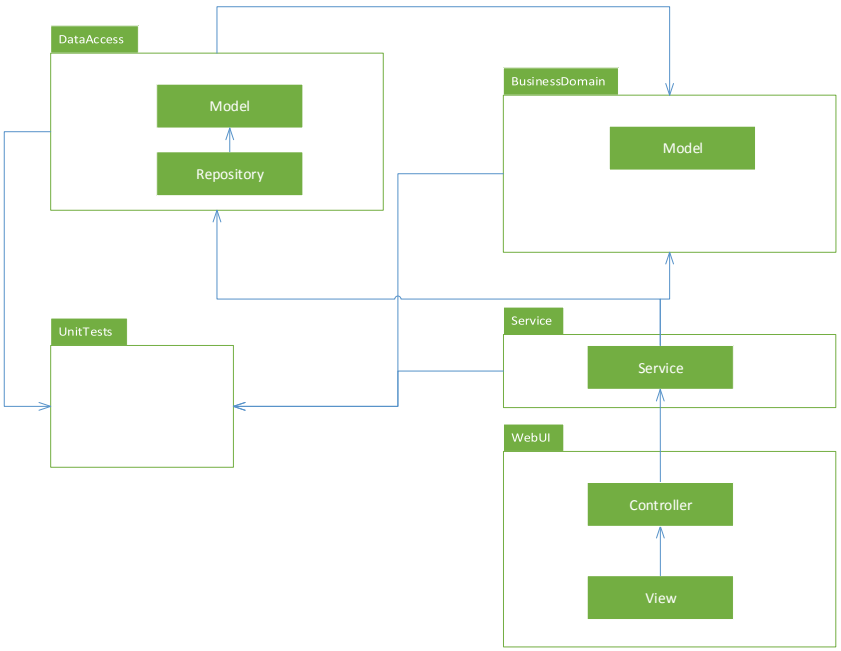
Voordat er aan de opdracht was begonnen is er eerst met Jochem Aubel (Eigenaar van Perflectie) afgesproken om vooraf een onderzoek te doen naar het huidige online-platform, waarin de volgende onderzoeksvraag centraal stond:

> *Is het verantwoord om, vanuit de optiek van onderhoudbaarheid en uitbreidbaarheid, het huidige online-platform uit te breiden met de partner module?*

Deelonderzoeksvragen afgeleid van de hoofdonderzoeksvraag:

1. *Wat maakt een systeem onderhoudbaar en uitbreidbaar?*
2. *Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?*
3. *Wat is het probleem/wat zijn de problemen binnen het huidige online-platform van Perflectie?*
4. *Wat is het structuur binnen het huidige online-platform van Perflectie?*
5. *Voldoet het huidige online-platform aan het resultaat en voorwaarden in deelonderzoeksvraag 2?*

Uit dit onderzoek (Zie bijlage D : Vooronderzoek) blijkt dat het huidige systeem van Perflectie, vanwege zijn grote omvang, op onderdelen niet te onderhouden is of dat er zich ongewenste afhankelijkheden voordoen. Dit heeft geresulteerd in een niet onderhoudbaar en niet uitbreidbaar systeem. Na een gesprek met Jochem Aubel is er tot de conclusie gekomen dat het beter is het systeem in zijn geheel opnieuw op te zetten, maar daarbij zoveel mogelijk functionaliteiten uit het oude systeem te gebruiken.



Figuur 1 Architectuur van het huidige online-platform

Om er voor te zorgen dat het nieuwe systeem niet op dezelfde manier wordt ontwikkeld als het huidige systeem heeft Perflectie de bevindingen van het vooronderzoek (zie bijlage D : Vooronderzoek) als uitgangspunt genomen voor het nieuwe project. Deze bevindingen bestaan uit een lijst van voorwaarden waaraan het systeem moet voldoen en controlevragen die een programmeur aan zichzelf en aan andere kan stellen. (zie bijlage D : Vooronderzoek - Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?)

2.3 Wat houdt een onderhoudbaar en uitbreidbaar online-platform voor Perflectie in?

Elk software systeem heeft zijn eigen specifieke functionele eisen, echter zijn er eisen die voor elk software systeem van toepassing zijn, deze eisen zijn onderhoudbaarheid en uitbreidbaarheid.

Zoals in het vooronderzoek (Zie bijlage D : Vooronderzoek) beschreven heeft een software systeem tijdens zijn gehele levensduur onderhoud nodig. Software kan gezien worden als een digitale levensvorm die wordt beïnvloed door zijn omgeving. Theoretisch kan het systeem zo ontworpen zijn dat het jarenlang zonder aanpassing kan draaien, in de praktijk is dit echter bijna nooit het geval. De software moet zich, net als elk andere levensvorm, aanpassen aan zijn omgeving om te overleven. (Vandegriend, 2014)

Als een systeem onderhoudbaar en uitbreidbaar is dan is er minimale inzet nodig van een programmeur om het systeem te begrijpen, onderhoud te plegen en aanpassingen binnen het systeem door te voeren. In de documentatie van een onderhoudbaar systeem zijn de richtlijnen waar de programmeur zich aan moet houden strikt vastgelegd. Door middel van unit tests, documentatie binnen de code en begrijpelijke code kan de programmeur zien of zijn implementatie of aanpassing binnen het systeem correct is. Dit heeft als voordeel dat de software langer gebruikt kan worden. De levensduur wordt verlengd.

Dit betekent echter niet dat een systeem immuun is voor inconsistente code van een programmeur. Als er geen tot nauwelijks overzicht is of als er geen controle wordt uitgevoerd binnen het systeem dan kan het voorkomen dat het systeem langzaam van zijn ontwerp gaat afwijken en zich ontwikkeld tot een niet onderhoudbaar en niet uitbreidbaar systeem. Daarom is het erg belangrijk, ook voor een onderhoudbaar en uitbreidbaar systeem, dat de documentatie up-to-date is en men zich hier strikt aan houdt. Wanneer de structuur van het systeem aangepast wordt moet het gehele project en documentatie in de aanpassing meegenomen worden.

Enkele voorbeelden waarom een systeem onderhouden moet zijn:

1. Bij het snel oplossen van een defect binnen het systeem
2. Bij het veranderen van business rules.
3. Door het veranderen van de omgeving waar het systeem zich in bevindt

2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?

Na het vooronderzoek is Perflectie tot de conclusie gekomen dat de Partner module in een nieuw te bouwen online-platform gerealiseerd gaat worden. Hierdoor krijg je de vrijheid om het systeem vooraf onderhoudbaar en uitbreidbaar op te zetten. Binnen het vooronderzoek (zie bijlage D : Vooronderzoek) zijn generieke methodes verzameld waarmee een onderhoudbaar systeem kan worden opgebouwd. Deze generieke methodes zijn als volgt:

- Ontwerp het systeem vanaf het begin vanuit het oogpunt van onderhoudbaarheid
- Ontwikkel en manage het systeem op een iteratieve manier
- Review regelmatig om de kwaliteit van het systeem te waarborgen
- Schrijf leesbare code zodat het eenvoudig te begrijpen is
- Refactor de code om de begrijpbaarheid te verbeteren
- Zorg voor relevante documentatie dat programmeurs helpt om het systeem te begrijpen
- Geautomatiseerde builds van het systeem zorgen voor gemakkelijk compileren van code
- Geautomatiseerde tests zorgen voor gemakkelijke validaties bij veranderingen binnen het systeem
- Continue integratie maakt de code gemakkelijk te builden en te testen
- Version control helpt om binnen het projectteam de code te testen en documentatie up to date te houden
- Verander, indien nodig, de manier van werken, met als doel onderhoudbaarheid

In het vooronderzoek zijn ook vragen vanuit de programmeur onderzocht (zie bijlage D : Vooronderzoek) om te testen of het systeem onderhoudbaar is. Dit om ervoor te zorgen dat de onderhoudbaarheid van het systeem wordt gewaarborgd tijdens (verdere) ontwikkeling. Deze vragen zijn als volgt:

- Kan ik code vinden dat gerelateerd is aan een specifiek probleem of verandering?
- Kan ik code begrijpen? Kan ik uitleggen wat de bedoeling is achter de functionaliteiten aan iemand anders?
- Is het gemakkelijk om de code te veranderen? Is het gemakkelijk voor mij om vast te stellen wat er verandert moet worden als consequentie van de vorige verandering? Zijn de hoeveelheden en omvang van deze consequentie klein?
- Kan ik snel een verandering binnen het systeem verifiëren?
- Kan ik een verandering doorvoeren met een laag risico op het breken van bestaande functionaliteiten?
- Als er een bug ontstaat, is het probleem snel en gemakkelijk te detecteren en te diagnoseren?

Deze vragen kunnen gesteld worden aan meerdere personen. Die persoon kan iemand zijn binnen het projectteam of iemand die geheel nieuw is binnen het systeem.

Het meenemen van deze methodes en vragen in het begin van het proces draagt bij aan een onderhoudbaar product. Een ander aspect dat Perflectie mee kan nemen is de keuze in het gebruik van programmeertalen en de architectuur van het systeem. Hierbij is van belang dat Perflectie zijn wensen en eisen duidelijk formuleert en dat deze meegenomen worden in het ontwerp van de architectuur voor het systeem. Dit allemaal om ervoor te zorgen dat de toekomstige applicatie optimaal, veilig en benaderbaar ingezet kan worden voor gebruikers en bedrijven.

3 De opdracht

3.1 De opdracht

Het uitgangspunt van de opdracht die in augustus 2014 verkregen is van Jochem Aubel luidde als volgt (zie hoofdstuk 2.1 De initiële opdracht voor verder details):

> *Ontwikkel voor Perflectie een Partner module in de omgeving van Perflectie zijn online-platform, met als doel dat een Partner de omgeving van zijn platform naar eigen wens kan aanpassen.*

Voordat er aan deze opdracht was begonnen is er samen met Jochem Aubel afgesproken om vooraf een onderzoek te doen naar het huidige online-platform, waarbij in het onderzoek de volgende onderzoeksvraag centraal stond:

> *Wat is de impact van de partner module op het huidige online-platform, en is dit verantwoord?*

Deelonderzoeksvragen afgeleid van de hoofdonderzoeksvraag:

1. *Wat maakt een systeem onderhoudbaar en uitbreidbaar?*
2. *Zijn de resultaten van deelvraag 1 toegepast op het huidige online-platform van Perflectie?*
3. *Wat zijn de voor- en/of nadelen als de onderzochte technieken worden geïmplementeerd?*

Uit dit onderzoek (zie bijlage D : Vooronderzoek) blijkt dat het huidige systeem van Perflectie, vanwege zijn grote omvang, op onderdelen niet te onderhouden is of er ongewenste afhankelijkheden aanwezig zijn. Dit heeft geresulteerd in een niet onderhoudbaar en niet uitbreidbaar systeem. Na een gesprek met Jochem Aubel is de conclusie getrokken dat het beter is het systeem in zijn geheel opnieuw op te zetten, maar daarbij wel zoveel mogelijk functionaliteiten te gebruiken vanuit het oude systeem.

Omdat na het onderzoek de opdracht enigszins is veranderd is hierop het uitgangspunt aangepast. Deze luidt nu als volgt:

> *Ontwerp en herbouw het Perflectie online-platform met de partner module inbegrepen, zodat het nieuwe systeem onderhoudbaar en uitbreidbaar is, voor 2 december.*

3.2 De eisen waar het systeem aan moet voldoen

Aan het eind van dit project (2 december, 2014) zal het online-platform van Perflectie herontworpen en herbouwd zijn, waarbij het nieuwe systeem voldoet aan de eisen van de opdrachtgever, Perflectie. De doelstellingen luiden als volgt:

Eisen

1. Het systeem is onderhoudbaar
2. Het systeem is uitbreidbaar
3. Het systeem houdt zich aan het vastgelegde ontwerp, waaronder:
    a. Architecture notebook
    b. Iteration plan
4. Het systeem heeft de functionaliteiten van het huidige online-platform van Perflectie
5. Het systeem heeft de functionaliteiten van de Partner module
6. Het systeem is testbaar
7. Het systeem is flexibel
8. Het systeem is veilig
9. Het systeem is toelaatbaarheid
10. Het systeem is schaalbaar

Testen

Om de eisen en daarmee de kwaliteit te waarborgen zal het systeem met behulp van Test Driven Development (TDD) getest worden. Hieruit kan afgeleid worden of het systeem aan de vastgestelde eisen voldoet.

3.3 De hoofdvraag met zijn deelvragen

Hoofdvraag productsopdracht

"Hoe moet het nieuwe online-platform voor Perflectie worden ontworpen zodat, met de partner module inbegrepen, het systeem onderhoudbaar en uitbreidbaar is? "

deelvragen

1. *Wat maakt een systeem onderhoudbaar en uitbreidbaar en aan welke voorwaarden moet het systeem voldoen?*
2. *Wat zijn de eisen van Perflectie voor het nieuwe online-platform?*
3. *Welke structuur of structuren zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*
4. *Welke design oplossingen zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*

3.4 Conceptueel model

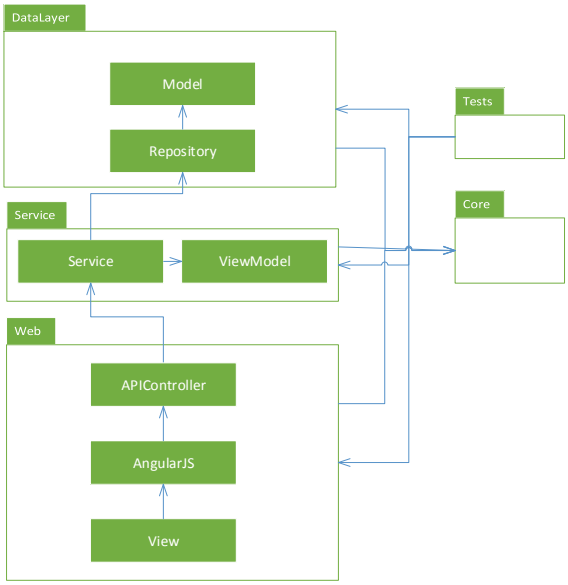
Aan het eind van dit project (2 december, 2014) is het nieuwe online-platform van Perflectie gerealiseerd vanuit het nieuwe ontwerp, waarbij het nieuwe systeem voldoet aan de eisen van de opdrachtgever, Perflectie, en aan de voorwaarden beschreven in het Vooronderzoek. (Zie bijlage D : Vooronderzoek)

De doelstellingen luiden als volgt:

Eisen

1. Het systeem is onderhoudbaar
2. Het systeem is uitbreidbaar
3. Het systeem houdt zich aan het, beschreven in de volgende documenten, ontwerp:
    a. Architecture notebook
    b. Technisch- en functioneel ontwerp
4. Functionaliteiten van het huidige online-platform van Perflectie
5. Functionaliteiten van de Partner module

Concept architectuur



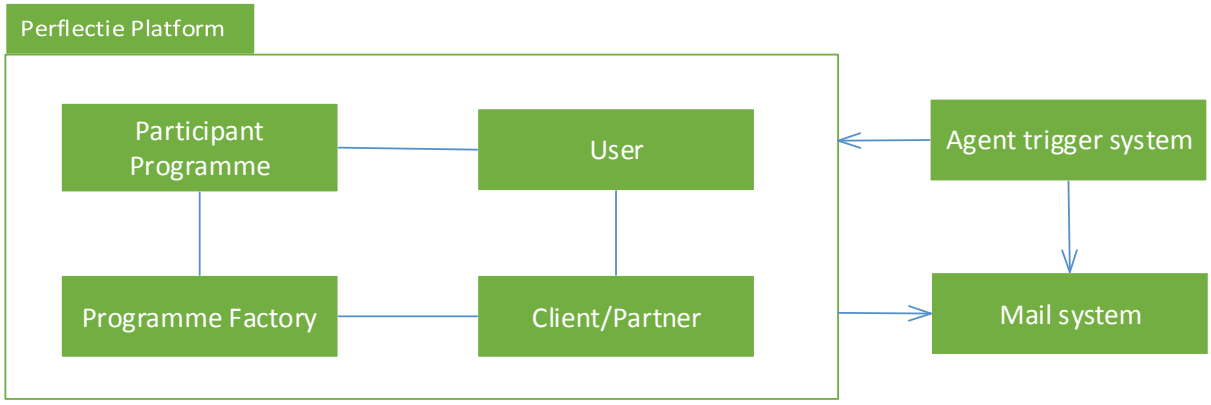
Figuur 2 Tier Architecture concept

Figuur 2 is een weergave van de nieuwe architectuur van het nieuwe online-platform.
Door de iteratieve aanpak kan de architectuur aangepast worden tijdens het project.
Er wordt gewerkt conform de scrum methodiek.

Testen
Om aan de eisen te voldoen en daarmee de kwaliteit te waarborgen zal het systeem met behulp van Test Driven Development (TDD) getest worden. Hieruit kan afgeleid worden of het systeem aan de afgesproken eisen voldoet.

Onderdelen van het nieuwe online-platform van Perflectie
Het nieuwe online-platform van Perflectie zal uit de volgende componenten bestaan:

- De programme factory
- Client/Partner
- Gebruikers
- Participants & feedbackmembers
- Mail systeem
- Agent trigger systeem



Figuur 3 Concept componenten voor het nieuwe platform

Deze componenten zullen zowel front-end, back-end als backoffice ontwikkeld worden en daarna worden geïntegreerd. Dit zal op een iteratieve wijze gebeuren, binnen de Scrum methode (zie bijlage B : Agile SCRUM), zodat er bij iedere iteratie verder gedetailleerd ontwikkeld kan worden.

Omdat na het vooronderzoek de opdracht enigszins is veranderd luidt deze nu als volgt:

> *Ontwerp en herbouw het Perflectie online-platform met de partner module inbegrepen, zodat het nieuwe systeem onderhoudbaar en uitbreidbaar is, met een opleverdatum van 2 december.*

De voorwaarden die hieronder staan genoteerd komen overeen met de voorwaarden van het vooronderzoek (Zie bijlage D : Vooronderzoek). Door invulling te geven aan iedere voorwaarde kan er op een correcte wijze een onderhoudbaar en uitbreidbaar systeem gebouwd worden.

Om te zorgen dat het nieuwe Perflectie online-platform onderhoudbaar is wordt, voor de bouw, een ontwerp opgesteld waarin de criteria van onderhoudbaarheid en uitbreidbaarheid zijn beschreven. Om het ontwerp begrijpelijk te houden zullen wij de UML (Unified Model Language) standaard aanhouden. Dit komt overeen met één van de voorwaarden genoteerd in hoofdstuk 2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?:

> *Ontwerp het systeem voor onderhoudbaarheid vanaf het begin*

Om de ontwikkeling van het nieuwe online-platform van Perflectie in goede banen te leiden zullen wij op een iteratieve manier het project managen en het product ontwikkelen. (zie bijlage H : Aanpak) Dit beantwoordt één van de voorwaarden genoteerd in hoofdstuk 2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?:

> *Het systeem op een iteratieve methode managen/ontwikkelen.*

Binnen Agile Scrum (Zie bijlage H : Aanpak & B : Agile SCRUM) zijn er meerdere methodes om het project regulier te reviewen en de kwaliteit van het systeem te waarborgen. Dit beantwoordt één van de voorwaardes genoteerd in hoofdstuk 2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?:

> *Reguliere reviews om de kwaliteit van het systeem te waarborgen.*

Binnen het scrumteam (Zie bijlage H : Aanpak & B : Agile SCRUM) zal de werkwijze aangepast worden om onderhoudbaar te werk te gaan. Deze werkwijze zal besproken worden binnen het scrumteam en zal meegenomen worden in de definition of done. (zie bijlage H : Definition of done) Wanneer de werkwijze juist wordt gevolgd worden de volgende voorwaarden, genoteerd in hoofdstuk 2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?, beantwoord:

1. *Leesbare code dat gemakkelijk te begrijpen is*
2. *Refactor de code om de begrijpbaarheid te verbeteren*
3. *Relevante documentatie dat programmeurs helpt om het systeem te begrijpen*
4. *Geautomatiseerde builds van het systeem zorgt voor gemakkelijk compileren van code*
5. *Continue integratie maakt de code gemakkelijk te builden en te testen*
6. *Version control helpt om de code, testen en documentatie up to date te houden binnen het projectteam*
7. *Verander, indien nodig, de manier van werken, met als doel onderhoudbaarheid*

Om de kwaliteit te waarborgen en om het systeem continu te kunnen verifiëren zal binnen het project Test Driven Development (TDD) gewerkt worden. Er zijn meerdere manieren om deze methode te volgen, maar uiteindelijk leidt dit tot hetzelfde resultaat. Een systeem dat onder test staat. Dit beantwoordt de volgende voorwaarde die genoteerd staat in hoofdstuk 2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?:

> *Geautomatiseerde tests zorgen voor gemakkelijke validaties bij veranderingen binnen het systeem.*

3.5 Methode van onderzoek

Voor Perflectie moest er technische informatie beschikbaar gesteld worden die in begrijpbare taal gelezen of gepresenteerd kon worden. Dit om zo de eigenaren van Perflectie mee te nemen in de ontwerpbeslissingen van het nieuwe online-platform. Om alle informatie te verzamelen zijn er twee onderzoeken opgestart.

Het eerste onderzoek is een vergelijkingsonderzoek van programmeertalen. Het doel van dit onderzoek is om voor Perflectie een geschikt programmeertaal of programmeertalen te adviseren. Het is belangrijk om een programmeertaal gedefinieerd te hebben voordat het systeem wordt ontworpen. Dit omdat een programmeertaal sterk kan verschillen van structuur en hier de ontwerp principes en methodes op aangepast moeten worden. Bijvoorbeeld als er object-georiënteerd (OO) gewerkt moet worden en de applicatie bestaat volledig uit javascript. Dan moet de OO pattern in javascript geïmplementeerd worden. Als C# gekozen wordt als programmeertaal, dan is de OO pattern al geïmplementeerd.

Het tweede onderzoek is een vergelijkingsonderzoek van architecturale patterns. Dit onderzoek was opgezet om voor Perflectie voordelen van architecturale patterns te presenteren. Uit deze voordelen heeft Perflectie een selectie gemaakt van waar het nieuwe online-platform aan moet voldoen. Op basis hiervan is een advies gegeven van een combinatie van architecturale patterns die het meest geschikt zijn voor het nieuwe online-platform.

3.6 Een beschrijving van het uitgangspunt

Om de groei van het aantal deelnemers in het platform te versnellen willen Jochem Aubel en Stefan Op de Woerd - de eigenaren van Perflectie - gaan werken met partners. Deze partners zijn bijvoorbeeld trainingsbureaus, coaches of consultants die medewerkers binnen bedrijven helpen met veranderprocessen. Het doel is dat dergelijke partners Perflectie kunnen inbedden in hun dienstverlening. Daarmee doen zij de feitelijke sales en implementatie van Perflectie, waardoor het business model schaalbaar wordt. Om dit interessant te maken voor partners, moet de applicatie naadloos inpasbaar zijn in hun dienstverlening.

De applicatie van het huidige online-platform blijkt niet flexibel te zijn. Feitelijk is er voor een partner niets instelbaar en is de partner dus niet in staat Perflectie zo in te richten dat het naadloos past binnen zijn dienstverlening. Om dit mogelijk te maken moet de applicatie flexibel gemaakt worden en moet er een beheeromgeving komen voor partners waar zij de applicatie naar hun eigen wensen kunnen inrichten.

Er is eerst een onderzoek gedaan naar de impact van een partner module op het huidige online-platform. Op basis van de resultaten van dit onderzoek zal er een ontwerp gemaakt worden dat de nieuwe functionaliteiten van de Partner module en de functionaliteiten van het huidige online-platform afdekt binnen een nieuw onderhoudbaar en uitbreidbaar systeem.

4 Theoretische onderzoek

4.1 Programmeertaal

Er zijn veel programmeertalen beschikbaar, ieder met zo zijn voor- en nadelen. Elke programmeertaal is van oorsprong opgericht met een doel. Moderne ontwikkeltalen zijn vooral gericht op het zo begrijpelijk, gemakkelijk, snel en juist programmeren van applicaties, waarbij oudere programmeertalen vaak complex maar wel erg efficiënt zijn. (zie hoofdstuk 4.1.1 Generaties programmeertalen)

4.1.1 Generaties programmeertalen

Programmeertalen worden ook wel gecategoriseerd in generaties.

Een eerste generatie (1GL) programmeertaal heeft geen compiler of assembler nodig om te kunnen functioneren. Hierdoor kan een machine direct de code gebruiken van een eerste generatie programmeertaal. (Generation languages, 2014)

Onder tweede generatie (2GL) programmeertalen vallen assembly talen. Het voordeel van deze generatie is dat de code leesbaar is voor een programmeur, maar voordat een machine de code kan draaien moet het eerst geassembleerd worden. Vandaar de term assembly. De programmeertaal is zo opgezet dat het alleen

geassembleerd kan worden voor specifieke hardware is, zoals de type van een processor. (Generation languages, 2014)

Een derde generatie (3GL) programmeertaal wordt ook wel high-level language genoemd. Een high-level programmeertaal is niet gelimiteerd aan hardware en is gemakkelijker te begrijpen. Aantal voorbeelden van een 3GL taal zijn C, Java en PASCAL. (Generation languages, 2014)

Vierde generatie (4GL) programmeertalen zijn zo geabstraheerd dat het beschrijvend van aard is, zoals in de reguliere taal. Dit wordt vooral ingezet bij het programmeren binnen databases en database scripts. (Generation languages, 2014)

Bij een vijfde generatie (5GL) programmeertaal helpen tools bij het programmeren van een applicatie. Een voorbeeld hiervan is Visual basic. (Generation languages, 2014)

4.1.2 Structuur van een web applicatie

Een web applicatie, dat bestaat uit geïmplementeerde business logica en data opslag, wordt meestal onderverdeelt in drie delen. Deze delen bestaan uit een backend voor logica, client-side voor representatie en server technologie voor data opslag. Om de architectuur en het ontwerp van het online-platform te bepalen zal eerst de programmeertaal voor ieder onderdeel gekozen worden. Deze keuzes zijn onderbouwd met een vergelijkingsonderzoek. (Designing web applications, 2014)

Het is nog niet duidelijk hoe zwaar het systeem van Perflectie belast zal worden, daarom zal er uit worden gegaan van een systeem dat snel groeit in aantal gebruikers. Omdat het online-platform van Perflectie vergelijkbaar is met een sociaal platform zijn er populaire sociale websites als vergelijking meegenomen. Ook zijn de meest bezochte websites meegenomen om als vergelijking te gebruiken. De reden hiervoor is dat deze websites veel actief verkeer hebben en de druk als gevolg hiervan moeten aankunnen.

De programmeertaal die geschikt is voor het online-platform van Perflectie moet aan de volgende eisen en wensen voldoen:

- Efficiënt
- Begrijpbaar
- Onderhoudbaar
- Overdraagbaar
- Betrouwbaar
- Business standard
- Ondersteuning van REST / API

Deze eisen en wensen zijn ontstaan vanuit de ervaring met het oude online-platform van Perflectie. Het oude platform had regelmatig problemen. De oorzaak van deze problemen en een snelle manier van oplossen werden nooit snel gevonden en niemand begreep waarom. Als doel bij het nieuwe online-platform is dit om deze problemen tegen te gaan. (zie bijlage D : Vooronderzoek)

4.1.3 Backend programmeertalen

De backend handelt de achterliggende logica van een applicatie af. Hier worden (betrouwbare) handelingen verricht wat je afgeschermd wilt houden voor de gebruikers, maar wel toegankelijk en aanpasbaar voor de developers. Deze handelingen bestaan onder andere uit het vertalen van de gebruikersinput naar het opslaan van data in de dataopslag. Om een duidelijk en begrijpelijke programmeeromgeving te realiseren vereist deze omgeving een programmeertaal dat globaal gebruikt wordt door de industrie van web applicaties. Om deze reden kiezen organisaties vaak voor C# of Java. (Google : Choosing a programming language, 2014) (Microsoft: Choosing a programming language, 2014)

Populaire programmeertalen voor webapplicaties zijn Ruby, Python, Java, ASP .NET en het full stack web framework, bijvoorbeeld de MEAN (MongoDb, Express, AngularJs, NodeJs) stack. De grote voordelen van de MEAN stack is Rapid development. Omdat Node.js binnen de stack ervoor zorgt dat Javascript ook gebruikt kan worden als backend is er geen aparte backend programmeertaal nodig. (Hall, 2013)

Daarnaast kan je voor de dataopslag kiezen voor een NoSQL database. Binnen een NoSQL database kun je gemakkelijk de structuur en data veranderen zonder afhankelijk te zijn van database migraties. De populariteit om op binnen deze methode applicaties te ontwikkelen is erg gegroeid. Het is daarom ook niet moeilijk om hulp of handleidingen te vinden. (MEAN stack tutorial, 2014)

Om de programmeertalen voor het nieuwe online-platform inzichtelijk te maken is er een vergelijkingstabel (zie **Tabel 1**) gemaakt. Hier zijn met de volgende factoren rekening gehouden:

Doel
Elke programmeertaal is van oorsprong opgericht met een doel. Vaak is dit het versimpelen (abstraheren) van een afgeleide programmeertaal. Zo is bijvoorbeeld C# afgeleid van C & C++ , maar binnen C# wordt de object-georiënteerde methode aangehouden waarbij C en C++ de functioneel georiënteerde methode wordt aangehouden. Door middel van de object-georiënteerde methode (zie hoofdstuk 4.2.5.6 Object-georiënteerd) kunnen objecten van andere objecten afleiden en zo een zelfvoorzienend en herbruikbaar object structuur vormen. De reden om het doel van een programmeertaal mee te laten wegen in de keuze voor een taal is omdat programmeertalen met vergelijkbare doelen sterk op elkaar lijken. Zo kan een programmeur met minder inspanning de programmeertaal leren.

Gebaseerd op
Waarop de programmeertaal is op gebaseerd.

Unique points
De Unique selling points van een programmeertaal. Elke programmeertaal heeft zo zijn extra extensie en/of attributen, zoals C# die Linq aanbiedt terwijl dit bij Java ontbreekt.

Websites die gebouwd zijn met
Populaire websites die als referentie en als vergelijkingsmateriaal gebruikt kunnen worden.

Documentatie
Kwaliteit en kwantiteit van documentatie om de programmeertaal te kunnen leren. Documentatie en goede voorbeelden zijn noodzakelijk voor het uitoefenen van een programmeertaal.

Professionals wereldwijd 2000
Aantal personen in percentage wereldwijd, in het jaar 2000, die de programmeertaal professioneel beheren.

Professionals wereldwijd 2014
Aantal personen in percentage wereldwijd, in het jaar 2014, die de programmeertaal professioneel beheren. Hiermee wordt de populariteit van de taal en de beschikbaarheid van werknemers gemeten.

Open vacatures
Het aantal vacatures voor de specifieke programmeertaal, om te meten of er in de Nederlandse industrie actief op wordt gezocht.

Tabel 1 Vergelijkingstabel backend programmeertalen

	C#	Java	PHP	Ruby	Python	NodeJS (MEAN STACK)
Doel	-Simpel -Modern -Generaal -Object georiënteerd	-Simpel -Modern -Generaal -Object georiënteerd	-HTML embedded programming language	-Dynamisch -Reflectief -Object georiënteerd -generaal	-Interactief object georiënteerd -Geïnterpreteerd	-Puur HTML/CSS/javaScript applicaties -Geen I/O blockade voor REST/API calls

Gebaseerd op	C, C++	C++	C, C++, Java, Perl, TCL	Ada, C++, CLU, DYLAN, EIFFEL, LISP, PERL, Python	ABC, ALGOL 68, C, C++, ICON, Java, LISP, Perl	
Unique points	-Gratis verkrijgbaar software -Linq & Lamda -Makkelijk installeerbare externe packages -Draait alleen systemen met windows OS en .Net	-Externe packages -Draait op alle besturing systemen	-Embedded programmeertaal -Draait op alle besturing systemen -Geen compiler	-Draait op alle besturing systemen	-Duck-typing -Gratis -Draait op alle besturing systemen	-Converteerbaar naar mobile applicatie -Gemakkelijk converterbaar naar een realtime applicatie -Draait op alle besturing systemen -Rapid development -Geen compiler
Website gebouwd met	Stackoverflow Microsoft	Oracle	Facebook (Facebook docs, 2014) Wikipedia	Twitter	Google (Google styleguide, 2014) Youtube	Walmart Ebay / PayPal LinkedIn
Documentatie	-Microsoft virtual academy -Je kan jezelf certificeren -MSDN Libr. + Samples -Externe tutorials	-Oracle Java tutorials -Je kan jezelf certificeren -Externe tutorials	-Externe tutorials	-Ruby documentation -Ruby guide -Externe tutorials	-Python documentation -Externe tutorials	-Externe en interne documenaties, tutorials en webinars.
Professionals Wereldwijd in het jaar 2000, (TIOBE Index october 2014, 2014)	0.358%	26.492%	1.897%	0.19%	1.25%	1.547%
Professionals wereldwijd oktober, 2014	4.748%	13.506%	2.942%	1.128%	2.333%	2.088%

(TIOBE Index october 2014, 2014)						
Open vacatures **(Zoekterm: <taal> developer)** **(Vacatures, 2014)**	377	379	329	25	57	**Javascript developer** 476 **Angular developer** 53 **Node developer** 33 **Frontend developer** 87

4.1.3.1 Object Georiënteerd programmeren

Uit de vergelijkingstabel (zie **Tabel 1**) is gebleken dat Java een veel gebruikte programmeertaal is onder professionals, maar waarbij de populariteit de afgelopen 14 jaar gehalveerd is. Dit is een aanduiding dat het niet verstandig is om Java te gebruiken bij het nieuwe project. De nu nog hoge percentage van Java professionals is er vooral omdat Java applicaties multiple-platform inzetbaar zijn. Maar talen zoals C# zijn gelimiteerd tot apparaten met windows besturing. (Google : Choosing a programming language, 2014)

Java is een Object georiënteerd-taal (OO). Voor vergelijkbare OO-talen zoals C#, Python en Ruby zijn daarentegen de afgelopen 14 jaar het aantal professionals gegroeid, met C# op kop.

Grote voordelen van een OO-taal zijn de overzichtelijkheid en helderheid. Door gebruik te maken van de OO middelen: objects, classes, data abstractie, encapsulatie, overerving en polymorfisme zorg je voor een onderhoudbare omgeving dat aangegeven is in business standards. Dit heeft als voordeel dat de applicatie gemakkelijk overgenomen kan worden door personen met kennis van deze standaarden. (Microsoft: Choosing a programming language, 2014)

4.1.3.2 De vrijheid van Embedded HTML code

Voor PHP als Embedded html code is het aantal professionals de afgelopen 14 jaar gegroeid. PHP is een open-source programmeertaal special voor web applicaties dat binnen HTML ge-embedded kan worden. Het onderscheidt van PHP ten opzichte van Javascript is dat de code afgehandeld wordt door de server, de server handelt functionaliteiten af en geeft HTML code terug. De client kan niet achterhalen hoe de achterliggende PHP code wordt afgehandeld maar ziet alleen de resulterende HTML code. (What is PHP?, 2014)

PHP is niet met types gedefinieerd en dit heeft zo zijn voor- en nadelen. Er kan op een OO manier met PHP gewerkt worden maar dit wordt niet afgedwongen. Dit kan voor foutgevoeligheid maar ook voor vrijheid zorgen. PHP is ook compileerbaar door middel van externe middelen. Dit zorgt voor een geoptimaliseerde uitvoerbare code zodat de server de applicatie ook optimaal in gebruik kan nemen. (What is PHP?, 2014)

4.1.3.3 Het builden van applicaties

Voor de bovengenoemde OO-programmeertalen is het compileren van de applicatie standaard en noodzakelijk voordat de applicatie kan worden uitgevoerd. Dit kan ervoor zorgen dat wanneer er bugs en/of problemen

voordoen binnen de applicatie dat oplossen tijdrovend is en ten koste van de productietijd gaat. Dit komt omdat er gewacht moet worden totdat de applicatie is gebuild na aanpassingen van de code (Compileren). Door middel van het werken met de TDD methode worden al veel bugs en problemen van te voren opgelost, maar het geheel uitsluiten is niet mogelijk. Zeker voor de client-side oplossingen is dit nog steeds een actueel aandachtspunt.

4.1.3.4 De full stack framework

Het full stack web framework is het afgelopen jaar een echt hype geworden. Een full stack web framework bezorgt de drie onderdelen(Client-side, backend en dataopslag) binnen een structuur van een web applicatie. De resultaten in Tabel 1 Vergelijkingstabel backend programmeertalen van Javascript professionals van het Jaar 2000 en het jaar 2014 hebben daarom geen betekenis. Door het kiezen van een full stack web framework zorg je voor een gehele web applicatie waarbij logica afgehandeld wordt door middel van Javascript. Dit heeft als voordeel dat je geen andere programmeertaal hoeft te gebruiken om backend logica af te handelen. Je volstaat met het toevoegen van een MySQL of een NoSQL database voor dataopslag. (Storz, 2014)

De afkorting van MEAN staat voor MongoDb wat een NoSQL database is, NodeJs dat door middel van javascript een server kan starten, Express dat de communicatie van de NodeJs versimpelt om taken aan te roepen om als voorbeeld data op te slaan of op te halen en als laatst AngularJs, om de structuur van de client-side af te handelen doormiddel van Javascript en de MVVM (Model, View, ViewModel) pattern.

MEAN stack is niet het enige full stack web framework die gebruikt kan worden voor een applicatie. Al deze vier onderdelen kunnen vervangen worden door een onderdeel met dezelfde functionaliteiten. Ook kan de NoSQL database vervangen worden door een MySQL database. (Full stack frameworks, 2014)

4.1.3.5 Conclusie backend programmeertaal

Als er gekeken wordt naar de eisen en wensen van Perflectie voor het nieuwe online-platform:

> *Een online-platform dat onderhoudbaar en betrouwbaar is, dat flexibel en dynamisch moet zijn en dat elke vorm en/of uitbreiding in de toekomst moet kunnen aannemen en/of implementeren.*

Voor Perflectie wordt een online-platform gemaakt waarbij de applicatie via REST API de functionaliteiten voor een client aanbiedt. Om de applicatie betrouwbaar te maken en te houden zal er via de TDD methode gewerkt worden. De methode van REST API en TDD kan geïmplementeerd worden op elke programmeertaal die in **Tabel 1** Vergelijkingstabel backend programmeertalen worden vergeleken.

Om de applicatie onderhoudbaar te maken en te houden is expertise vereist van de programmeur. Met elke programmeertaal kan een applicatie overzichtelijk gestructureerd worden, maar als de programmeur niet de expertise bezit kan de applicatie precies het tegenovergestelde worden. Om de applicatie toch onderhoudbaar en overzichtelijk te houden moet er een combinatie van programmeerwijze, structuur en de keuze van een programmeertaal worden afgesproken. OO talen zoals Java, C#, Ruby etc. eisen tijdens het builden van de code dat de juiste syntaxises zijn geïmplementeerd, wat bij PHP en de MEAN stack ontbreekt. PHP en de MEAN stack zorgen op hun beurt voor rapid development, maar vereisen meer expertise van de programmeur om een overzichtelijk business standaard structuur binnen de applicatie op te bouwen en vast te houden.

Mijn voorkeur gaat uit naar een full stack web framework. De redenen hiervoor zijn innovatie, snelheid en de mogelijkheden van het full stack web framework. Er kan gemakkelijk en snel een prototype gemaakt worden van een applicatie en deze kan worden gecompileerd naar een optimaal bruikbaar uitvoerbaar project dat door elk besturingssysteem gedraaid kan worden. Het full stack web framework is ontworpen om web applicaties zo snel en zo goed mogelijk op te bouwen. Doordat de applicatie volledig HTML5, CSS3 en Javascript is, komt de structuur overeen met clients en/of applicaties zoals een webbrowser of een mobiele applicatie. Doordat deze optie opengehouden wordt kan in de toekomst gemakkelijk en met een zelfde structuur een mobiele applicatie gemaakt worden.

4.1.4 Client-side programmeertalen

4.1.4.1 HTML & CSS

Client-side programmeren binnen web applicaties wordt vooral gedaan met (x-) HTML en CSS. Met behulp van Javascript worden pagina's (real time-) dynamisch en interactief gemaakt voor de gebruiker. Alternatieven zijn: Adobe Flash, Java Applets, PDF, SASS, SENCHA, Silverlight, XSL en SVG.

Het grootste probleem met het kiezen van een alternatief voor HTML en/of CSS is dat het vaak slecht tot nauwelijks herkenbaar is door zoekmachines zoals Google. Alleen al daarom is het aan te raden om HTML in combinatie met CSS en Javascript te gebruiken. De laatste versie van HTML is HTML5 en voor CSS is CSS3 de laatste versie. Elke nieuwe versie van HTML brengt wijzigingen in de syntax en structuur met zich mee.

4.1.4.2 CSS & LESS & Bootstrap 3

Binnen het huidige Perflectie systeem is er geen dynamisch CSS aanwezig. Het is mogelijk om met behulp van LESS of Sass dynamische variabel in te laden voor CSS attributen. CSS staat voor Cascading Style Sheet en zorgt ervoor dat een HTML element een stijl toegewezen krijgt.

Omdat een eis is dat het platform gepersonaliseerd moet kunnen worden door een partner en een gebruiker is LESS of Sass een oplossing om een dynamische stijl omgeving te creëren. LESS of Sass werkt met behulp van JavaScript en kan zo waardes doorgeven naar de customized stijl sheet. (lesCSS, 2014) (Fadeyev, 2014)

Waarom LESS?

LESS is een veelgebruikte library binnen organisaties, welke vergelijkbaar is met JQuery. Een andere library kan hetzelfde bereiken als LESS, maar omdat LESS één van de meest voorkomende is wordt deze het meest aangeraden.

Om het platform klaar te maken voor het snel groeiend aantal mobiele gebruikers is het verstandig om Bootstrap 3 te implementeren. De huidig geïntegreerde Bootstrap 2 is niet geoptimaliseerd voor de mobiele doelgroep. Het is meer gericht op de standaard voor gedefinieerde elementen voor een website.

Bootstrap 3 is speciaal gericht op mobiele gebruikers en zo kan er een online-platform opgezet worden dat gebruikt kan worden door een bredere groep van gebruikers. Het nadeel is wel dat Bootstrap 3 andere syntaxen gebruikt dan Bootstrap 2. Hierdoor moet er nieuwe styling en HTML elementen worden gemaakt. Dit geeft wel de mogelijkheid om het deze keer wel in één keer goed te doen, dit omdat in het huidige platform slecht tot nauwelijks onderhoudbare CSS aanwezig is onderverdeeld in veel nikszeggende bestanden. (Duchek, 2014)

4.1.4.3 JQuery VS AngularJS

JQuery is een erg populaire library om JavaScript binnen een website te implementeren. Het nadeel van JQuery is wel dat het zich niet houdt aan architecture en dat je dus als programmeur vrij bent hoe je JQuery implementeert. Dit is geen probleem als je onderling of persoonlijk regels opzet waardoor de code goed onderhoudbaar en leesbaar is, maar dit is in het huidige platform niet gebeurd.

Om dit tegen te gaan is het aan te raden om een javascript framework te gebruiken die zich aan de MVVM architectuur houdt, genaamd AngularJS. Dit heeft ook als voordeel dat het gehele backend los staat van de front-end en het zo ook niet meer mogelijk is om backend logica te implementeren in de front-end.

Door de afscheiding van de backend van de front-end kun je rijke en snelle web applicaties maken. Dit geeft voor de gebruiker een fijnere ervaring met het product omdat er haast geen tot weinig laadtijd is. Dit in tegenstelling tot het oude platform waarbij er pageloads bestaan van 3 á 4 seconden, deze tijd kan flink gereduceerd worden. In de toekomst kunnen er ook gespecialiseerde programmeurs worden ingehuurd of worden ingezet om de front-end en/of de backend te onderhouden en verder te ontwikkelen.

4.1.4.4 Conclusie client-side programmeertaal

Het is aan te raden om de client-side met HTML5 (zie hoofdstuk 4.1.4.1 HTML & CSS), CSS3 met LESS en Bootstrap 3 (zie hoofdstuk 4.1.4.2 CSS & LESS & Bootstrap 3) en AngularJS (zie hoofdstuk 4.1.4.3 JQuery VS AngularJS) te realiseren. Dit geeft als voordeel dat het systeem onderhoudbaar is en volgens business

standaard wordt gerealiseerd waarbij het gemakkelijk overgenomen kan worden door externe programmeurs. Als er niet gekozen wordt voor de MEAN-stack geeft deze structuur ook het voordeel dat de front-end zich houdt aan de MVVM architectuur dat gemakkelijk onderhouden kan worden los van de backend.

4.1.5 Dataopslag

4.1.5.1 SQL vs NoSql

Er zijn veel verschillende type databases in de markt, maar één nieuw opkomende is de NoSQL. Het verschil tussen SQL en NoSQL database is dat binnen een SQL database de relatie tussen model verloopt via tabels en tabel relaties via schema's. Een NoSQL database handelt de structuur en relatie op een genormaliseerde manier af. Dit maakt een NoSQL snel en gemakkelijk uitbreidbaar. Een NoSQL database heeft daarom vaak een hogere performance dan een SQL database.

4.1.5.1.1 NoSQL Types

Op de website van (Ibraheem, 2014) worden verschillende types NoSQL beschreven met hun doel en de voor en nadelen per type database.

De types van NoSQL op basis van (Ibraheem, 2014) zijn:

1. Key value store
    a. Om afbeeldingen op te slaan
    b. Bestand systeem gebaseerd op key's
    c. Object cache
    d. Systeem ontworpen op schaal
2. Column family store
    a. Web crawlers resultaten
    b. "Big data" problemen die op consistency regels rusten.
3. Graph store
    a. Sociale netwerken
    b. Relatie zware data
4. Document Store
    a. Hoog variabele data
    b. Documenten zoeken
    c. Integratie hubs
    d. CMS
    e. Publishen

De voordelen beschreven op (Ibraheem, 2014)

- Open source
- Horizontaal schaalbaar, maakt gebruik van meerdere processors
- Laag operationele kosten door Autosharding
- Geen object relationeel mapping
- Kan complexe datatypes opslaan
- Gemakkelijk om hoog variabele data op te slaan

En de nadelen beschreven op (Ibraheem, 2014)

- ACID transactie kan niet aan worden voldaan
    - ACID staat voor Atomair, consistent, geïsoleerd en duurzaam.
    - ACID zorgt ervoor dat twee transacties niet tegelijkertijd schrijfrechten op een enkel set aan model(-en) hebben binnen de database.
- Niet volwassen
- Staffs zijn nieuw voor een NoSQL database
- Geen indexes
- Slecht reportages van performance van de database

4.1.5.1.2 SQL Types

Een SQL database ook wel RDMS (Relational database management system) genoemd.

De voordelen beschreven op (Ibraheem, 2014)

- ACID transacties op elk database niveau maakt het gemakkelijker om mee te programmeren.
- Door doorontwikkelde beveiliging op kolommen en rijen worden ongeautoriseerde acties op de database tegengehouden.
- De meeste SQL code kan overgezet worden naar een andere SQL database
- RDMB bestaat al lang en kan beschouwd worden als volwassen
- Er is voldoende personeel beschikbaar die op een goede manier met SQL om kan gaan

En de nadelen beschreven op (Ibraheem, 2014)

- Object relatie mapping kan complex zijn
- RDBM schalen niet mee met procesoren als er table joins worden gebruikt
- Het kan complex zijn om hoog variabele data op te slaan in tabels

4.1.5.2 Conclusie SQL of NoSQl

Als er gekozen wordt voor een full stack web framework dan is NoSQL geschikt zijn voor dataopslag. Dit omdat er genoeg documentatie bestaat waardoor het gemakkelijk is om het structuur te implementeren maar ook omdat de performance gunstig kan zijn voor in de toekomst. Perflectie verwacht veel gebruikers met veel data, de performance kan door middel van een NoSQL fors verbeterd worden. Het risico wat dit met zich meebrengt is dat een NoSQL database een jong concept is dat veel problemen met zich mee kan brengen. Hierdoor is de betrouwbaarheid niet groot. Op het forum Ycombinator (Don't use MongoDb, 2014) stelt mongoDb mensen gerust maar geeft zelf ook duidelijk aan dat het een nieuw systeem is:

> *"MongoDB is still a new product, there are definitely rough edges, and a seemingly infinite list of things to do."*

Als er niet gekozen wordt voor een full web framework dan is een SQL database aan te raden. Dit omdat er meer support en tools beschikbaar zijn die geschikt zijn voor het snel en gestructureerd werken met zo'n type database. Ook kunnen de gegevens uit de oude database van het huidige platform overgezet worden naar de nieuwe database.

4.1.6 Conclusie te adviseren programmeertaal

Het advies is gegeven om de partner module te ontwikkelen binnen een nieuw project omdat de partner module op deze manier sneller gerealiseerd kan worden. Het volgende is geadviseerd om mee te nemen in dit proces:

Om al de drie onderdelen van een web applicatie af te dekken is het aan te raden om een full stack web framework te gebruiken.(zie hoofdstuk 4.1.3.5 Conclusie backend programmeertaal)

De reden hiervoor is het innovatieve karakter, de snelheid en de mogelijkheden van het full stack web framework. Er kan gemakkelijk en snel een prototype gemaakt worden van een applicatie en deze kan worden gecompileerd naar een optimaal bruikbaar uitvoerbaar project dat op elk besturingssysteem uitgevoerd kan worden. De full stack web framework is opgezet om web applicaties zo snel en goed mogelijk op te bouwen. Doordat de applicatie volledig opgebouwd is uit HTML5, CSS3 en Javascript kunnen er gemakkelijk clients gemaakt worden voor zowel de web browser als voor een mobiele applicatie.

Het oude online-platform van Perflectie is gebouwd met behulp van C# binnen de MVC structuur en een Microsoft SQL database. Hierdoor is de kans groot dat het nieuwe online-platform ook uit deze onderdelen zal bestaan. Als hier weer voor wordt gekozen is het aan te raden om voor de client-side HTML, CSS en AngularJs te gebruiken. (zie hoofdstuk 4.1.4.4 Conclusie client-side programmeertaal) Dit zorgt ervoor dat de client-side losgekoppeld wordt van de backend en zo een eigen architectuur kan behouden. Dit geeft als voordeel een onderhoudbare en overzichtelijke MVVM Architectuur. Voor de backend zal door middel van REST API de backend logica worden afgehandeld door middel van C#.

4.2 Geschikte architecturale patterns voor het nieuwe online-platform van Perflectie

Dit onderzoek is opgezet om een geschikte softwarestructuur voor het nieuwe online-platform te ontwerpen. Binnen dit onderzoek worden de voordelen van architecture patterns in kaart gebracht. Van deze voordelen worden de belangrijkste, waaraan het nieuwe online-platform moet voldoen, gekozen door Perflectie. Door middel van een vergelijkingsonderzoek worden de hieronder gekozen patterns vergeleken met hun primaire voordelen. Dit onderzoek biedt informatie bij het vinden van een geschikte architectuur voor het nieuwe online-platform van Perflectie.

De gekozen architecture patterns

De architecturale patterns die in het vergelijkingsonderzoek is onderzocht zijn in Tabel 2 Overzicht architecturale patterns geformuleerd. Deze patterns zijn verkregen uit (Architectural patterns and styles, 2014).

Tabel 2 Overzicht architecturale patterns

Architecture pattern	Beschrijving
Client / Server	Hierbij wordt het systeem verdeelt in twee onderdelen, de client en de server. De client maakt calls naar de server, waarbij door de server de applicatie logica wordt afgehandeld.
N-Tier / 3-Tier	Verdeelt het systeem in meerdere lagen, wat vergelijkbaar is met de layered architecture. Maar binnen de N-Tier bevinden de lagen zich fysiek gescheiden op een computer.
Component based	Verdeelt het systeem in meerdere functionele of logische componenten die herbruikbaar zijn. De communicatie met deze component verloopt via een gedefinieerde interface.
Object oriented	Een design principe gebaseerd op het herbruikbaar stellen van zelfvoorzienende objecten, waarbij elk object data en gedrag bevat die relevant is voor het object.
Layered Architecture	Verdeelt het systeem in een gelaagde hiërarchie.
Service Oriented architecture	Een systeem dat services aanbied en hier gebruik van maakt.
Message bus	De Message Bus zorgt voor een voor gedefinieerde communicatie standaard bestaande uit een set aan calls die via de bus kunnen worden ontvangen en verstuurd zonder dat de betreffende applicatie specifieke details nodig heeft.
Domain driven design	Een object georiënteerde pattern die focust op het gebruikmaken van een gedefinieerde business domain om modellen te definiëren.

4.2.1 Wat is een architecturale pattern?

Een architecturale pattern is binnen de informatica een business standaard oplossing welke hergebruikt kan worden bij het ontwerpen van een systeem. Een architecture style wordt gebruikt in combinatie met architecturale pattern en vormen gezamenlijk een systeem. Er zijn meerdere soorten patterns, bijvoorbeeld: een architecturale pattern die gebruik maakt van componenten en van deze componenten de relaties definieert tot een structuur. (Avram, 2014) Hierbij lost een design pattern een specifiek flow van business requirements binnen een systeem op. (Design patterns, 2014)

Dat een pattern gebruikt kan worden bij het ontwerpen van een systeem maakt het niet een op zichzelf staand ontwerp. Een pattern is een concept wat als basis gebruikt kan worden en waar aanpassingen en/of toevoegingen voor nodig zijn om het systeem te laten werken. Het voordeel van het gebruik van een pattern is dat het systeem de karakteristieken van de pattern behoudt. (Design patterns, 2014)

Architecturale patterns kunnen gecategoriseerd worden in hun hoofddoel. De Tabel 3 Architecture categorieën verkregen van (Architectural patterns and styles, 2014) geeft weer hoe architecturale patterns onderverdeeld kunnen worden in categorieën.

Tabel 3 Architecture categorieën

Categorie	Architecture pattern
Communicatie	Server Oriented architecture Message bus
Deployment	Client/Server N-Tier 3-Tier
Domein	Domain driven design
Structuur	Component based Object oriented Layered Architecture

4.2.2 Design patterns

4.2.2.1 Separated presentation patterns

Seperation of concerns

Door seperation of concern zorg je ervoor dat componenten een duidelijke rol krijgen. Bijvoorbeeld de MVC pattern. De MVC pattern heeft drie rollen: model, view en de controller. Het model representeert de data, de view representeert de userinterface en de controller wordt gebruikt voor het afhandelen van input vanuit de gebruiker, manipuleren van het model en uitvoeren van operaties. (Baley, Belcham, & Kovacs, 2014)

Event-based notification

Door de event-based notifaction pattern is het mogelijk voor componenten om te reageren op specifieke gebeurtenissen vanuit andere componenten, zonder dat de component kennis heeft over deze componenten. Bijvoorbeeld de observer pattern die notificaties toont naar de view als het model veranderd. (Gupta, Hartkopf, & Ramaswamy, 2014)

Delegated event handling

De controller handelt gebeurtenissen af die getriggerd worden vanuit de userinterface. Deze gebeurtenissen worden delegate afgehandeld. Er kan een delegate object aangemaakt worden dat een referentie naar een methode kan encapsuleren. Andere objecten binnen het systeem kunnen daarna gebruik maken van de delegate methode van het object, zonder te hoeven weten welk methode precies wordt aangeroepen. (Mahfouzi, 2014) (Delegates and Events in C# / .NEt, 2014)

4.2.2.2 Software design

Dependency injection

Door middel van dependency injection kunnen klassen los gekoppeld worden. Dit wordt gedaan door interfaces die aanduiden welke methodes en/of properties een klas moet bevatten. Het meest bekende voorbeeld is een schakelaar voor een lamp. Hierbij heb je twee klasses: de lamp en de schakelaar. De lamp krijgt een interface genaamd IAanUit. IAanUit bevat de definitie van twee methodes: Aan en Uit. De logica van de methodes aan en uit zijn binnen de lamp klasse vastgesteld en voldoet zo aan de interface IAanUit. De schakelaar kan zo elk object die de interface IAanUit erft aanroepen en is zo niet direct afhankelijk van de klasse lamp. (Caprio, 2014)

4.2.3 Hoofdvoordelen van een structuur gebaseerd op een architecturale pattern

1. **Abstractie**
   Abstractie is het weglaten van niet-essentiële informatie. Zo kan een systeem in grote lijnen getoond worden wat gemakkelijker is voor onderhoud.
2. **Isolatie**
   Het afschermen van invloeden van buitenaf. Bijvoorbeeld een component zijn structuur is geïsoleerd en is alleen bereikbaar via zijn gedefinieerde interface.
3. **Beheerbaar**
   Dit geeft aan dat het systeem gemakkelijk beheerbaar is.
4. **Prestatie**
   Dit geeft aan dat het systeem goed presenteert onder druk.
5. **Testbaar**

Dit geeft aan dat het systeem gemakkelijk testbaar is

6. **Uitbreidbaar**
   Het systeem is gemakkelijk uit te breiden met meer functionaliteiten/componenten/lagen.
7. **Laag complexiteit**
   Het systeem is zo opgezet dat de complexiteit laag is en daardoor gemakkelijk te begrijpen.
8. **Flexibel**
   Het systeem is flexibel inzetbaar in meerdere situaties
9. **Loose coupling** (zie Verklarende woordenlijst)
10. **Schaalbaar**
    Het systeem kan gemakkelijk opgeschaald worden mocht dit nodig zijn. Situaties waar dit in kan voorkomen is dat de gebruikersgroep groter wordt waardoor de datbaseserver moet opschalen naar een cluster servers. Zo kan de data onderverdeelt worden. Dit wordt vaak gedaan in een cloud omgeving.
11. **Applicatie simpliciteit**
    Het systeem is simpel opgezet, hierdoor duidelijk en gemakkelijk aan te passen.
12. **Beschikbaarheid**
    Beschikbaarheid van een systeem is hoeveelheid uptime en zo dus de hoeveel tijd de gebruiker het kan gebruiken.
13. **Begrijpelijk**
    Het systeem is zo opgebouwd dat objecten, methodes en algoritmes zo gedefinieerd zijn als gewone taal, waardoor het systeem begrijpelijk wordt.
14. **Herbruikbaarheid**
    Het systeem is zo opgebouwd dat onderdelen gemakkelijk hergebruikt kan worden in andere systemen en/of andere situaties.
15. **High cohesion** (zie Verklarende woordenlijst)
16. **Domain alignment**
    Het reduceren van kosten en tijd door middel van het hergebruik van functionaliteiten zoals het gebruik van services en interfaces.
17. **Toelaatbaarheid**
    Het systeem biedt service functionaliteiten en beschrijvingen aan andere applicaties door middel van voorgedefinieerde communicatie middelen, schema's en of protocollen.
18. **Interoperability**
    Door gebruik te maken van industry standaarden, kan het systeem gemakkelijk overgezet worden naar verschillende platformen.
19. **Gerationaliseerd**
    Het gebruik van specifieke enkelvoudige functionaliteiten in plaats van dupliceren van de functionaliteiten in het systeem zorgt ervoor dat er geen dubbele functionaliteiten ontstaan.
20. **Hoge beveiliging**
    Door gebruik te maken van verschillende methodes kan zo de beveiliging van het systeem verhoogd worden. Bijvoorbeeld het afschermen van de business en service logica, waardoor het niet door het publieke netwerk toegankelijk is maar wel door een lokaal client applicatie die het representeert voor de gebruiker.
21. **Centraal data access**
    Data wordt opgeslagen op de server, hierdoor is datamanagement gemakkelijk.
22. **Gemak bij onderhoud**
    Rollen en verantwoordelijkheden zijn gescheiden en ondergebracht in meerdere systemen. Hierdoor is de eindgebruiker zich niet bewust wanneer er een onderdeel wordt gerepareerd of onderhouden.
23. **Communicatie**
    Alle onderdelen binnen het systeem kunnen gebruik maken van een onderdeel, zoals het business domain.
24. **Gemak bij deployment**
    Een systeem is gemakkelijk in deployment wanner een nieuwe versie en/of verandering kan worden doorgezet zonder andere componenten of het systeem in geheel te veranderen.
25. **Gemak bij development**
    Het systeem is opgebouwd uit onderdelen die alleen benaderd kunnen worden vanuit één punt. Zo kan het systeem ontwikkeld worden zonder andere delen van het systeem in de weg te zitten.
26. **Matiging van de technische complexiteit**

Door het systeem in delen onder te verdelen en benaderbaar te maken via één punt zorg je voor een heldere service voor de rest van het systeem.

27. **Lage kosten**
    Door gebruik te maken van componenten binnen een systeem, die gemakkelijk herbruikbaar of toegankelijk is voor andere systemen, verdeel je de kosten van development en onderhoud.

4.2.4 Gekozen eisen van Perflectie

Na het inventariseren van de hoofdvoordelen (zie hoofdstuk 4.2.3 Hoofdvoordelen van een structuur gebaseerd op een architecturale pattern) die een architecturale pattern kan bieden heeft Perflectie een selectie gemaakt dat voor hun online-platform belangrijk is.
Deze selectie is:

- Beheerbaar
- Testbaar
- Flexibel
- Uitbreidbaar
- Veilig
- Toelaatbaarheid
- Schaalbaar

4.2.5 Definitie van architecturale patterns

De architectuur van een software systeem is meestal niet opgezet met één architecture pattern. Het bestaat meestal uit combinaties van meerdere patterns die samen één software systeem vormen. Hier is een voorbeeld gegeven binnen de website van (Architectural patterns and styles, 2014): Een systeem kan een SOA pattern gebruiken voor services die intern worden afgehandeld met een gelaagde architectuur en een object-georiënteerd domain.

4.2.5.1 Layered architecture

Een combinatie van architecturale patterns komt in het algemeen voor bij het bouwen van web applicaties. Web applicaties met een gelaagde architectuur worden in de basis opgebouwd in drie lagen. Deze lagen worden op elkaar gestapeld. Zo'n structuur valt onder de categorie deployment pattern. De communicatie die via de lagen verloopt zijn expliciet en loosely coupled. Door gebruik te maken van een gelaagde structuur zorg je voor een separation of concern waardoor de flexibiliteit en onderhoudbaarheid van het systeem wordt verhoogd.

De communicatie van een gelaagde structuur verloopt van boven naar beneden (Startend bij de presentatie laag) waarbij geen laag overgeslagen mag worden. Ook mag een laag niet communiceren met een bovenliggende laag. Bij relaxed layering mogen componenten in een laag communiceren met zijn bovenliggende laag. (Layered application, 2014)

De lagen van een systeem kunnen zich op dezelfde fysieke computer of op andere fysieke computers bevinden. Dit word een N-tier genoemd. Bij een N-tier kunnen de componenten in elke laag communiceren met andere lagen door middel van gedefinieerde interfaces. (Architectural patterns and styles, 2014)

De principes die een gelaagde architectuur volgt zijn: (Verkregen uit (Architectural patterns and styles, 2014))

- **Abstractie**
    - Door gebruik te maken van een gelaagde architectuur zorg je voor een abstract systeem. Dit komt omdat elke laag een rol toegewezen krijgt. Een rol bevat verantwoordelijkheden en relaties binnen het systeem.
- **Encapsulatie**
    - Door middel van encapsulatie zorg je ervoor dat data types, methodes, properties of implementaties alleen worden getoond binnen zijn laag en/of binnen de grenzen van zijn component.
- **Gedefinieerde functionele lagen**
    - Door het scheiden van logica van een systeem binnen de lagen zorg je voor een helder systeem. Bovenliggende laag communiceert met de onderliggende laag en zorgt zo voor een flow van data.
- **High cohesion**

    - Binnen lagen worden duidelijke afhankelijkheden gesteld. Dit zorgt voor een High cohesion binnen het systeem.
- **Herbruikbaarheid**
    - De lagere gelegen lagen hebben geen onafhankelijkheid op de bovenliggende lagen. Hierdoor kunnen lagen hergebruikt worden voor andere systemen en/of scenario's.
- **Loose coupling**
    - Communicatie tussen lagen is gebaseerd op abstractie en zorgt voor loose coupling tussen lagen.

(Layered application, 2014) De basis van een gelaagde architectuur kan onderverdeeld worden in drie onderdelen:

Presentatie laag
De presentatie laag is verantwoordelijk voor het afhandelen van de interactie met de gebruiker. Deze laag kan zo verder ontworpen worden met een andere architecture pattern, bijvoorbeeld met de MVC pattern. De MVC pattern is een interactie model wat vergelijkbaar is met een gelaagde architectuur, of opgezet worden met een SOA architecture pattern. Deze pattern zorgt ervoor dat via messages de communicatie verloopt tussen de server en de client applicatie. (Architectural patterns and styles, 2014)

Business laag
De business laag of ook wel de service laag genoemd is verantwoordelijk voor het afhandelen van de business rules/logica van de applicatie. Door de afscheiding van de business laag worden hier ook de tests afgehandeld van de applicatie. Dit kan omdat alle communicatie en alle logica afgehandeld wordt binnen de business laag.

Data laag
Binnen de data laag wordt de code gedefinieerd welke data binnen de database kan beheren.

4.2.5.2 Client/Server
De client/server architectural pattern, verkregen uit (Architectural patterns and styles, 2014), bestaat uit een server die benaderd wordt door één of meerdere clients. De server is opgezet door middel van een 2-Tier structuur waarbij de client alleen zorgt voor de presentatie logica.

Een voorbeeld hiervan is een web applicatie die de gebruiker data toont en gebruikersinput afhandelt door middel van calls naar de server.

Andere variaties op de client/server zijn:

Client-Queue-client-system
Doormiddel van deze aanpak zorg je ervoor dat clients kunnen communiceren met andere clients via de server. Een client kan data lezen en versturen vanuit de server, de server slaat de data op in een (wacht-)rij. Dit zorgt ervoor dat clients bestanden en informatie met elkaar worden gesynchroniseerd.

Peer-to-peer application
Gebaseerd op het client queue client systeem, Peer to peer zorgt ervoor dat de server en de client hun rol kunnen veranderen en zo de files en informatie synchroniseren over de clients. Dus elke client heeft de rol van zowel server als een client, ook wel een peer genoemd.

Application servers
Een gespecialiseerde architectural pattern waarbij de server een applicatie host. Een voorbeeld hiervan is een client die een applicatie draait van de server doormiddel van een framework.

De grootste voordelen van een client/based architectuur zijn:
- Hoge veiligheid
    - Omdat alle data opgeslagen wordt op de server, zorgt dit voor volledige controle over de beveiliging. Bij clients heb je hier geen controle over.
- Centraal data toegang

    - De data wordt alleen opgeslagen op de server. Hierdoor heb je gemakkelijk toegang om de data te beheren.
- Gemak bij onderhoud
    - Rollen en verantwoordelijkheden zijn onderverdeelt over meerdere servers die communiceren met elkaar via een netwerk. Dit zorgt er voor dat een client niet afhankelijk is van één server en er zo gemakkelijk onderhoud gepleegd kan worden.

4.2.5.3 Component-based

Een systeem dat gestructureerd is met een component-based architecture heeft intern functionele en logische componenten. Deze componenten communiceren met elkaar via heldere interfaces die methodes, properties en instanties bevatten. Een component based structuur zorgt voor een hoger level van abstractie dan de object-georiënteerde pattern.

De principes die een component-based architectuur volgt zijn: (Verkregen uit (Architectural patterns and styles, 2014))

- Herbruikbaar
    - Componenten horen zo ontworpen te zijn dat deze hergebruikt kunnen worden in verschillende applicaties. Het kan wel zo zijn dat componenten voor specifieke taken zijn ontworpen.
- Vervangbaar
    - Componenten kunnen vervangen worden door vergelijkbare componenten
- Niet context specifiek
    - Componenten zijn ontworpen zodat ze in andere omgevingen en contexten kunnen werken. Specifieke data moet naar het component worden gestuurd in plaats van dat een component hier direct toegang toe heeft.
- Uitbreidbaar
    - Componenten kunnen tot het systeem toegevoegd worden om zo de functionaliteiten uit te breiden.
- Encapsulatie
    - Componenten geven alleen zijn interface weer dat door andere componenten kan worden aangeroepen. Het component toont verder geen details van interne functionaliteiten of variabels.
- Onafhankelijk
    - Componenten horen ontworpen te zijn met geen tot minimale afhankelijkheden met andere componenten. Zo kunnen componenten ingezet worden in andere systemen zonder dat het systeem en/of componenten worden aangetast.

4.2.5.4 Domain-driven design

Een domain-driven design is een systeem dat ontworpen is vanuit zijn business domain, zijn elementen, gedrag en relaties onderling. Er wordt een domain model gemaakt vanuit het business domain. Het domain model heeft zo hetzelfde gedrag als een framework en kan zo gebruikt worden binnen het systeem.

Domain-driven design wordt meestal toegepast binnen een development team waarbij communicatie belangrijk is om tot overeenstemming te komen, of waarbij het ontwerp of applicatie in normale taal uitgelegd moet worden aan belanghebbende. (Laribee, 2014)

4.2.5.5 Message bus

De message bus architecture zorgt ervoor dat het systeem data kan verzenden en ontvangen via één of meerdere communicatie kanalen. Hierdoor kunnen applicaties communiceren zonder specifieke gedetailleerde kennis te hebben. Een message bus wordt ingezet bij individuele applicaties, die onderling communiceren via schema's en overeengekomen infrastructuren, voor het verzenden en ontvangen van calls.

Een message bus architecture zorgt voor volgende functionaliteiten: (Verkregen uit (Architectural patterns and styles, 2014))

- Message-oriented communications
    - Alle communicatie tussen applicaties zijn gebaseerd op calls met hetzelfde schema's/structuur.

- Complex processing logic
    - Complexe methodes kunnen uitgevoerd worden met een set aan kleinere methodes.
- Modificeren om logica af te handelen
    - Doordat de interactie met de bus via algemene schema's en methodes verloopt, kun je gemakkelijk methodes toevoegen en verwijderen.
- Integratie binnen verschillende omgevingen
    - Door gebruik te maken van een message-based communication model gebaseerd op algemene standaarden, kun je met applicaties communiceren die ontwikkeld zijn in verschillende omgevingen, zoals C# en Java.

Er zijn meerdere variaties van de message bus ontworpen om complexere situaties te ondersteunen.

Enterprise service bus
Gebaseerd op de message bus architectuur, een Enterprise Service Bus (ESB) gebruikt services voor het communiceren tussen de bus en componenten toegevoegd aan de bus.

Internet service bus
Vergelijkbaar met een ESB, een Internet service bus (ISB) heeft applicaties gehost in de cloud in plaats van een enterprise netwerk.

4.2.5.6 Object-georiënteerd
Een object-georiënteerd (OO) architectuur is een design principe die gebruik maakt van individuele herbruikbare en zelfvoorzienende objecten, waarbij elk object data bevat en relevante functionaliteiten die het object toebehoren. Een OO systeem maakt gebruik van samenwerkende objecten in plaats van routines en procedures. De objecten zijn discreet, onafhankelijk en loosely coupled.

De hoofd principes van een object-georiënteerd architectuur is als volgt: (Verkregen uit (Architectural patterns and styles, 2014))

- **Abstractie**
    - Door een OO structuur aan te houden zorg je dat complexe operaties gegeneraliseerd worden. Bijvoorbeeld een abstracte interface. Deze kan de definitie geven voor alle Create, Read, Update en Get methodes van objecten.
- **Compositie**
    - Objecten kunnen van andere objecten gestructureerd worden en er kan gekozen worden om interne properties en functionaliteiten te verbergen.
- **Inheritance**
    - Objecten kunnen van andere objecten overerven. Een object kan de geërfde methodes en properties gebruiken of overschrijven. Overerving maakt een applicatie onderhoudbaar en overzichtelijk.
- **Encapsulatie**
    - Objecten laten alleen zijn functionaliteiten zien door methodes, properties en events. Interne details zoals zijn staat en variables van andere objecten worden verborgen. Dit maakt het gemakkelijk om objecten te updaten of te vervangen, zolang hun interface overeenkomen zonder de objecten en code aan te tasten.
- **Polymorfisme**
    - Het overschrijven van een basis object waardoor het nieuwe object via dezelfde methodes een ander gedrag en/of vorm toont.
- **Decoupling**
    - Een object kan decoupled worden door middel van een abstracte interface.

Voorbeelden van een geïmplementeerde OO architectuur zijn de programmeertalen C# en Java.

4.2.5.7 Service-oriented architecture
Een service-oriented architecture (SOA) geeft een applicatie zijn functionaliteiten en een set aan services. Deze services kunnen gebruikt worden door externe applicaties door te communiceren met een schema via

message-based interactie. De SOA pattern vereist wel dat de applicatie door middel van interfaces een scope wordt gedefinieerd, en niet via componenten of via een OO structuur.

De hoofd principes van een service-oriented architectuur zijn als volgt: (Verkregen uit (Architectural patterns and styles, 2014))

- **Services zijn autonoom**
  - Elk service wordt onafhankelijk ontwikkeld, onderhouden en ingezet. Hierdoor kan elke service onafhankelijk van andere services worden beheerd.
- **Services zijn bereikbaar**
  - Services zijn bereikbaar via het netwerk, locaal of via internet zolang het zich aan het communicatie protocol houdt.
- **Services zijn loosely coupled**
  - Services zijn loosely coupled. Elke service is onafhankelijk van andere en kan zo vervangen en/of ge-update worden zonder de applicatie te breken.
- **Services delen een schema en een contract, niet een klasse**
  - Services delen contracten en schema's om te communiceren.
- **Compatible gebaseerd op beleid**
  - Beleid betekend in dit geval het definiëren van features, zoals transport, protocol of beveiliging.

4.2.5.8 N-Tier

N-Tier is een deployment pattern die het systeem onderverdeeld in onderdelen zoals een gelaagde architectuur. Elk onderdeel kan geplaatst worden op aparte fysieke computers. De N-tier is doorontwikkeld waarbij de component-oriented aanpak als basis is genomen. Hierdoor loopt de communicatie niet via messages maar via specifieke methodes.

Elke tier binnen een N-tier is onafhankelijk van alle andere tiers. Een tier hoeft alleen te weten hoe het de calls moet afhandelen van een tier onder zich of als het een bovenliggende tier heeft deze call doorgeven.

Een voorbeeld van een N-Tier/3-Tier architectuur is een web applicatie waarbij beveiliging belangrijk is. De business layer met daarin de logica zit achter een firewall, waarbij de presentation layer zich op een aparte tier in het netwerk bevindt.

4.2.6 Conclusie geschikte architecturale patterns

De architectuur van een software systeem is meestal niet opgezet met één architecture pattern. Een systeem is meestal opgebouwd uit combinaties van meerdere patterns dat het systeem structuur geeft. Een voorbeeld hiervan is gegeven op (Architectural patterns and styles, 2014): Een systeem kan een SOA pattern gebruiken voor services die intern worden afgehandeld met een gelaagde architectuur en een object-georiënteerd domain.

Om voor het nieuwe online-platform van Perflectie een geschikte architectuur te ontwerpen zijn er acht architecture patterns gekozen. (Zie hoofdstuk 4.2 Geschikte architecturale patterns voor het nieuwe online-platform) Van deze acht gekozen patterns is er een vergelijkingstabel opgezet. Dit geeft inzicht in welke primair behoefte de pattern voldoet. Zo kunnen er patterns gekozen en gecombineerd worden totdat het voldoet aan de eisen genoteerd in hoofdstuk 4.2.4 Gekozen eisen van Perflectie.

De vergelijkingstabel is onderverdeelt in 2 delen. Tabel 3 Deel 1 vergelijkingstabel architecture patterns en Tabel 4 Deel 2 vergelijkingstabel architecture patterns. De behoeftes waar Perflectie's nieuwe online-platform aan moet voldoen is oranje/rood gearceerd. De behoeftes waar een pattern aan voldoet is groen gearceerd.

Tabel 3 Deel 1 vergelijkingstabel architecture patterns

Primaire voordelen	Gelaagde architectuur	Client/Server	Component-based	Domain driven
Abstractie				

Isolatie				
Beheerbaar				
Prestatie				
Testbaar				
Uitbreidbaar				
Laag complexiteit				
Flexibel				
Louse coupling				
Schaalbaar				
Applicatie simpliciteit				
Beschikbaarheid				
Begrijpelijk				
Herbruikbaar				
High cohesion				
Domain enlignment				
Toelaatbaar				
Interoperable				
Gerationaliseerd				
Hoge beveiliging				
Centraal data access				
Gemak bij onderhoud				
Communicatie				
Gemak bij deployment				
Gemak bij development				

Matiging van de technische complexiteit				
Lage kosten				

Tabel 4 Deel 2 vergelijkingstabel architecture patterns

Primaire voordelen	Message bus	Object Georienteerd	Service-Oriented	N-Tier
Abstractie				
Isolatie				
Beheerbaar				
Prestatie				
Testbaar				
Uitbreidbaar				
Laag complexiteit				
Flexibel				
Louse coupling				
Schaalbaar				
Applicatie simpliciteit				
Beschikbaarheid				
Begrijpelijk				
Herbruikbaar				
High cohesion				
Domain enlignment				
Toelaatbaar				
Interoperable				
Gerationaliseerd				
Hoge beveiliging				

Centraal data access				
Gemak bij onderhoud				
Communicatie				
Gemak bij deployment				
Gemak bij development				
Matiging van de technische complexiteit				
Lage kosten				

4.2.6.1 Aanbeveling: Client/Server met een N-Tier structuur

Structuur

Het nieuwe online-platform zal binnen een gelaagde N-Tier structuur gebouwd worden, waarbij het aan te raden is dat de logica binnen de lagen object-georiënteerd en via componenten worden afgehandeld. Dit omdat er veel complexe methodes binnen de applicatie gerealiseerd moeten worden en door middel van de OO methode en componenten de complexiteit wordt ge-encapsuleerd. Dit heeft als voordeel dat een component van het systeem geen details hoeft te weten van andere componenten. Dit verhoogt de onderhoudbaarheid en gaat aannames tegen van programmeurs in het gebruik van methodes.

Het werken met de OO-Methode verhoogt ook de testbaarheid en de uitbreidbaarheid van de applicatie. Dit omdat er gebruik wordt gemaakt van herbruikbare klassen. Hierdoor hoef je alleen de blauwdruk van de klasse te testen en/of te verwijderen, toe te voegen of aan te passen.

Domein

Perflectie heeft geen software-ontwikkelafdeling en daardoor ook geen professionals die kunnen assisteren bij het ontwerpen van het nieuwe online-platform. Hiervoor is de domain-driven pattern geschikt om een platform te ontwerpen en te ontwikkelen waarbij er weinig tot geen kennis van het platform nodig is. Door het implementeren van de domain-driven design kan zo de eigenaren van Perflectie meebeslissen in het ontwerp en ontwikkel proces van de applicatie.

Deployment

Om de veiligheid van het nieuwe online-platform van Perflectie te waarborgen is de client/server pattern gekozen. Dit in combinatie met een 3-Tier (Drie lagen N-Tier) structuur waarbij de flexibiliteit en de schaalbaarheid van de applicatie gewaarborgd blijft. De nadeel van een N-Tier over een gelaagde architectuur is de beheerbaarheid. Dit komt omdat de lagen verdeelt zijn over verschillende fysieke platformen, hierdoor is het lastiger om deze te bereiken en te beheren.

Communicatie

Omdat het nieuwe online-platform van Perflectie ook toegankelijk moet zijn voor partners en/of toekomstige andere (mobile) clients is een flexibele communicatie pattern nodig waarbij het systeem ook toelaatbaar moet zijn voor externe bronnen. Hiervoor is een service-georiënteerde pattern het meest geschikt.

5 Iteratieve aanpak

5.1 De gebruikte methodes

De uitgevoerde onderzoeken en de resulterende principes, methodes en technieken geven een antwoord op de deelvraag: welke design oplossingen zijn geschikt in het maken van het nieuwe online-platform van Perflectie? Door een geschikte managementmethode, principes, technieken en methodes te gebruiken kan een systeem en de conversie van functionele eisen naar functionaliteiten iteratief onderhoudbaar ontwikkeld worden.

5.1.1 Agile development

Managementmethode vooronderzoek

Voordat er met dit project begonnen was, was er eerst een onderzoek (zie bijlage B : Management methode: Agile Scrum) gestart welk projectmanagement methode het best past bij het ontwikkelen van het nieuwe online-platform. Wekelijks vonden gesprekken plaats met één van de eigenaren van Perflectie, Jochem Aubel, waarbij Perflectie als bedrijf en de Partner module aan bod kwamen. Tijdens deze gesprekken is de LEAN startup (zie bijlage B : The LEAN startup) aan bod gekomen. Dit is de methodiek wat Perflectie volgt en is vervolgens ook meegenomen binnen de onderzoek naar een geschikte projectmanagement methode. De LEAN startup is ontwikkeld om (beginnende) bedrijven te behoeden voor problemen die zich voordoen in de hedendaagse bedrijfswereld. Deze methode heeft drie onderdelen: “Visie”, “Sturen” en “Versnellen”. (Ries, 2011) De LEAN startup heeft de methode zo ontworpen dat je leert om te rijden met je onderneming. Dus in plaats van complexe situaties uit te denken en daarvoor plannen te maken dat vooral gebaseerd zijn op aannames, maak je constante aanpassingen binnen het sturen naar jouw einddoel. Dit heet “Build-Measure-Learn”. (Ries, 2011)

Uit de vergelijkingsonderzoek voor management methodes binnen bijlage B : Management methode: Agile Scrum, is tabel 4 als resultaat gekomen.

Tabel 2 Vergelijkingstabel project managementmethodes

Tabel 1	Eisen waaraan te voldoen (Bijlage 17.3)		
Methodes	Agile Scrum	Waterval methode	RUP
Volgt de methode Iteraties?	Sprint planning Sprint review Daily meeting Twee tot vier weken durende sprint	Volgt aflopende fases in plaats van iteraties	Het ontwikkel van het product verloopt in iteraties met behulp van een storyboard, use cases en taken.
Kunnen deze iteraties binnen een Milestone lopen? Of kan dat gedefinieerd worden?	Agile Scrum is flexibel, waarbij je zelf milestones, statussen etc. kunt definiëren.	Waterval methode houdt zich aan de fase wat definieert staat en wat opvolgt. Als je dit aanpast dan val je weg van de waterval methode	RUP is een niet zo generieke methode waarbij je vast zit aan voor gedefinieerde methodes.
Kan in deze methode ook het product eigenaar in betrokken worden?	Hier is een speciale rol, product owner, voor benoemd.	Voordat er begonnen wordt aan de ontwikkel fase en nadat de ontwikkel fase is afgerond zal er met de belanghebbende worden gesproken over wensen en eisen.	Voordat er begonnen wordt aan de ontwikkel fase en nadat de ontwikkel fase is afgerond zal er met de belanghebbende worden gesproken over wensen en eisen.
Kan in deze methode ook de klanten in betrokken worden?			
Heeft deze methode korte iteratieve reminders/meetings	Sprint planning Sprint review Daily meeting		Met tests binnen de software zal het project worden gevalideerd en

om het project in de juiste baan te sturen?	Twee tot vier weken durende sprint		kan het project management hierop bijsturen.
Heeft deze methode een efficiënte en effectieve overzicht om een zo goed mogelijke acceleratie te maken?	Als het werkend product van een sprint gedefinieerd is dan kan aan de sprint begonnen worden.	Als de gehele software vooraf gedefinieerd is dan kan er aan de sprint begonnen worden..	Als de gehele software vooraf gedefinieerd is dan kan er aan de sprint begonnen worden..
Wordt deze methode ondersteund binnen team foundation server?			

In de vergelijkingstabel Tabel 2 Vergelijkingstabel project managementmethodes is te zien dat Agile Scrum het meest voldoet aan de eisen en wensen van de organisatie en het project. Ook omdat het ondersteund wordt door team foundation server, hierdoor is het gemakkelijk te integreren met het project dat ontwikkeld wordt binnen Visual studio.

Van RUP is er de best practices overgenomen om de architectuur van het systeem in kaart te brengen. Hierbij wordt de architectuur van het systeem eerst gevisualiseerd en voor elke sprint worden de use cases, requirements en modellen ontworpen en gedocumenteerd.

Het systeem wordt met Agile Scrum en onderdelen van RUP ontwikkeld.

Agile development
Binnen Agile software development wordt er iteratief te werk gegaan. Dit project volgt de Agile methodiek conform Agile Scrum. (zie bijlage B : Agile SCRUM) Agile Scrum erkent iteraties als sprints, waarbij voor iedere sprint een sprint planning plaatsvindt. Binnen een sprint planning wordt door het scrum team afgesproken welke taken afgeleid kunnen worden van de sprint backlog. Voor iedere taak wordt een schatting gemaakt van de hoeveelheid felocity die er nodig is om taken te realiseren. Als laatste wordt samen met het scrum team tot een ontwerp oplossing gekomen door middel van domain-driven design.

Dit proces en de afspraken die tijdens de interviews worden gemaakt voor elke sprint worden gedocumenteerd in het Iteration plan.(zie bijlage H : Iteration plan) Daarbij dient het Iteration plan als een document met de verzamelde functionele eisen met een daarbij behorende ontworpen domain model.

Sprint backlog
De lijst van alle taken en backlog items binnen een sprint noemen we de sprint backlog. Om voor iedere sprint planning alle beschikbare backlog items te verkrijgen en bij elke item de eisen, zijn interviews gehouden met de Product Owner. De Product Owner in dit project was Jochem Aubel. Binnen iedere interview werd eerst de scope bepaald van de sprint, daarna werd van de scope alle gewenste functionele eisen verzamelt via Jochem Aubel. Van deze eisen werden backlog items gemaakt en later binnen de sprint planning omgezet naar taken met een geschatte waarde van inspanning.

Om Jochem Aubel mee te nemen in het design proces is er de domain-driven design methode aangehouden, zie hoofdstuk 4.2.5.4 Domain-driven design, tijdens de sprint backlog interviews.

Domain-driven design
De Product owner was binnen dit project ook een lid binnen ons Scrum Team, ondanks geen tot weinig technische programmeer kennis. Om toch zoveel mogelijk informatie en tot een overeenstemming te komen bij ontwerp keuzes binnen het online platform is er met de domain-driven design methodiek gewerkt. Hierbij is er voor elk sprint het domain model aangepast en/of uitgebreid op basis van de functionele eisen binnen een sprint. Dit ontwerp werd vervolgens uitgebreid met een Use Case diagram om zo de onderlinge relaties en verantwoordelijkheden weer te geven.

Sprint planning
Voordat er binnen dit project met een sprint begonnen werd, werden er interviews gehouden met de Product Owner. Als eerst werd de scope van de sprint bepaald door de product owner. Daarna werden binnen de sprint interviews, samen met de product owner en het scrum team, de functionele eisen vastgelegd. Deze functionele eisen worden direct overgenomen van de Product owner en worden daarna naar taken omgezet voor de developer. Deze taken kregen ieder een felocity toegewezen. Een felocity is de geschatte waarde die nodig is om de taak te voltooien. Zo kan na elke sprint een gemiddelde felocity berekend worden wat het team elke sprint kan realiseren. Hoe hoger de felocity, des te hoger is de toegevoegde waarde die het team per sprint oplevert.

5.1.2 Principes, technieken en methodes

Principes, technieken en methodes
Het huidige online-platform van Perflectie, die binnen dit project vernieuwd zal worden, is beoordeeld op een vooronderzoek (zie bijlage D : Vooronderzoek) naar onderhoudbare en uitbreidbare systemen. Uit dit onderzoek zijn generieke principes, technieken en methodes gekomen die ingezet kunnen worden om een systeem te beoordelen tijdens of na productie. Voor deze principes , technieken en methodes zie hoofdstuk 2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?

Hoe gaan we de resultaten meten?
Voor elke generieke methode is het belangrijk om een specifiek passende meetmethode toe te passen. Dit om te kunnen beoordelen of het systeem aan de voorwaarden voldoet. Dit is onderzocht in het Vooronderzoek en deze resultaten zullen hieronder gepresenteerd worden.

Wat is een onderhoudbaar systeem?
Binnen de vooronderzoek kwam als resultaat wat een onderhoudbaar en onderhoudbaar systeem inhoud voor Perflectie. Als een systeem onderhoudbaar en uitbreidbaar is dan is er nauwelijks inzet nodig van een programmeur om het systeem te begrijpen, onderhoud te plegen en toevoegingen binnen het systeem uit te voeren. (zie hoofdstuk 2.3 Wat houdt een onderhoudbaar en uitbreidbaar online-platform voor Perflectie in?)

Binnen hoofdstuk 2.4 Wat voor mogelijkheden zijn er om een onderhoudbaar en uitbreidbaar systeem te ontwikkelen?, is er een vragenlijst verkregen van (Crouch, 2014). Door middel van deze vragen kan je beoordelen of een systeem onderhoudbaar is.
Om het nieuwe online-platform, na voltooiing, te beoordelen net als het huidig online-platform zijn de volgende vragen afgeleid van de verzamelde principes, technieken en methodes:

- Is het systeem vanaf het begin ontworpen met onderhoudbaar voor ogen?
- Wordt het systeem op een iteratieve methode gemanaged/ontwikkeld?
- Waren er of bevinden zich er reguliere reviews om de kwaliteit van het systeem te waarborgen?
- Is er binnen het systeem leesbare code dat gemakkelijk te begrijpen is?
- Is de code gerefactored in een proces om de begrijpbaarheid te verbeteren?
- Is er relevante documentatie voor programmeurs om het systeem te begrijpen?
- Kan het systeem automatisch gebuild worden zodat het eenvoudig is om de code te compileren?
- Zijn er (geautomiseerde) tests aanwezig zodat het gemakkelijk is om bij veranderingen het systeem te valideren?
- Is er continue integratie binnen de code zodat builden en testen gemakkelijk gemaakt wordt?
- Wordt version control binnen het systeem gebruikt zodat ieder teamlid de laatste versie heeft van de code, testen en documentatie?
- Heeft ieder teamlid binnen het systeem de werkwijze aangehouden om zo onderhoudbaar werk te produceren.

De gebruikte methodes
De methodes die binnen dit project gebruikt gaan worden om zo de voorwaardes die hierboven genoteerd zijn te beantwoorden en om zo een onderhoudbaar en uitbreidbaar systeem te bereiken wijkt niet af van de al gedefinieerde methodes in hoofdstuk 3.4 Conceptueel model. Deze methodes moeten ervoor zorgen dat het nieuwe online-platform onderhoudbaar en uitbreidbaar ontworpen wordt. Deze methodes opgesomd zijn:

Door het bijhouden van een Iteration plan (zie bijlage H : Iteration plan), wordt het systeem iteratief gedocumenteerd. Het iteration plan bevat de functionele ontwerp van alle sprints. De ontwerpen worden gemaakt door middel van de unified model language (UML) standaard. Door deze methode wordt de volgende voorwaarde (zie bijlage D : Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?) binnen het nieuwe online-platform gerealiseerd.

Ontwerp het systeem voor onderhoudbaarheid vanaf het begin

Om de ontwikkeling van het nieuwe online-platform van Perflectie in goede banen te leiden zal er op een iteratieve manier het project gemanaged en het product ontwikkeld worden. (Zie bijlage G : Aanpak) Dit beantwoordt één van de voorwaardes genoteerd in de bijlage D : Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?:

Het systeem op een iteratieve methode managen/ontwikkelen.

Binnen Agile Scrum (Zie bijlage G : Aanpak & B : Agile SCRUM) zijn er meerdere methodes om het project regulier te reviewen en de kwaliteit van het systeem te waarborgen. Dit beantwoordt één van de voorwaardes genoteerd in de bijlage D : Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?:

Reguliere reviews om de kwaliteit van het systeem te waarborgen.

Binnen het scrumteam (Zie bijlage G : Aanpak & B : Agile SCRUM) zal de werkwijze aangepast worden om onderhoudbaar te werk te gaan. Deze werkwijze zal besproken worden binnen het scrumteam en zal meegenomen worden in de definition of done. (Zie bijlage H : Definition of done)

De definition of done bevat een procedure van werkzaamheden voor een programmeur afgestemd binnen een ontwikkel omgeving. Kort samengevat bestaat de definition of done uit de:

- Red, Green en Refactor, deze methode wordt gehanteerd binnen de Test driven development methode (Palermo, Guidelines for Test-Driven Development, 2014)
    - Refactor
        - Clean code
        - Om bij de Refactor fase te assisteren bij het maken van “clean code” zijn de volgende materialen gedeeld met het ontwikkel team. Binnen deze materialen wordt gedetailleerd uitgelegd hoe je clean code maakt.
            - http://www.slideshare.net/mariosangiorgio/clean-code-and-code-smells?related=2
            - http://www.slideshare.net/arturoherrero/clean-code-8036914?related=3
            - http://not-at-school.blogspot.nl/2011/03/coding-rules-for-clean-and-robust-c.html
    - Plan
        - Plan is voor de red stap geïmplementeerd. Dit heeft Perflectieintern zo afgesproken omdat er fouten ontstonden doordat taken werden gerealiseerd zonder kennisdeling of zelfs taken die al gerealiseerd waren nogmaals werden uitgevoerd. Door middel van de plan stap kun je eerst kritisch de taak bekijken en analyseer je zo het proces tot de realisatie van deze taak binnen het team.
- Code smells
    - Verzameling van code smells, als je code hier niet aan voldoet heb je de Refactor fase niet goed afgerond. Hierdoor kan een taak nooit klaar zijn voordat alle code smells opgelost zijn.
- S.O.L.I.D
  De code houdt zich aan de S.O.L.I.D principes
    - Single responsibility principle (SRP)
        - SRP komt overeen met de term Cohesion. (Zie Verklarende Woordenlijst)
          Bij de SRP is het de bedoeling dat een class maar één verantwoordelijk binnen het systeem heeft. Zo blijft het systeem overzichtelijk en is het gemakkelijker onderhoudbaar. (Martin, 2003)
    - Open/Closed principle principle (OCP)
        - Binnen de OCP heeft ieder module binnen het systeem twee primaire attributen. Open voor uitbreiding en gesloten voor aanpassing. Binnen dit principe wordt er

nieuwe code toegevoegd in plaats van oude werkende code te vervangen of aan te passen. Dit wordt bereikt door middel van abstractie. Het is niet de bedoeling dat dit principe overal in het systeem wordt toegepast. Het is alleen bedoeld voor delen binnen het systeem wat vaak veranderd. Wanneer de OCP correct wordt toegepast binnen een systeem dan wordt het systeem onderhoudbaar, herbruikbaar en betrouwbaar. (Martin, 2003)

- Open voor uitbreidingen
  - De functionaliteiten van een module kan worden uitgebreid. Zo kan het systeem evolueren om zich aan te passen aan andere situaties.
- Gesloten voor aanpassingen
  - De code binnen de module mag niet aangepast worden.

- Liskov subsitution principle (LSP)
  - De LSP helpt om de OCP te forceren. Door middel van het toepassen van hiërarchie en overerving is het mogelijk om een module uit te breiden zonder dat daar de bestaande source code op aangepast hoeft te worden.
- Interface segregation principle (ISP)
  - "De ISP helpt tegen "fat" interfaces. Een interface met veel methodes kan gegroepeerd worden. Dit helpt bij het inzetten van de groepen onder meerdere clients. Zo kunnen clients één groep van methodes gebruiken en andere clients de andere groepen. " (Martin, 2003)
- Dependency inversion principle (DIP)
  - Door middel van dependency inversion verhelp je het probleem dat er afhankelijkheden tussen procedures met details bestaan binnen het systeem. Als er veranderingen plaatsvinden binnen deze details kan diegene die afhankelijk hiervan zijn ongewenst aangepast worden. Om dit tegen te gaan creëer je door middel van de DIP een structuur waarbij de procedure en de details afhankelijk zijn van abstractie. (Martin, 2003)

Wanneer de hierboven genoemde werkwijze juist wordt gevolgd worden de volgende voorwaarden (genoteerd in de bijlage D : Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?) binnen het nieuwe online-platform voldaan:

1. *Leesbare code dat gemakkelijk te begrijpen is*
2. *Refactor de code om de begrijpbaarheid te verbeteren*
3. *Relevante documentatie dat programmeurs helpt om het systeem te begrijpen*
4. *Geautomatiseerde builds van het systeem zorgt voor gemakkelijk compileren van code*
5. *Continue integratie maakt de code gemakkelijk te builden en te testen*
6. *Version control helpt om de code, testen en documentatie up to date te houden binnen het projectteam*
7. *Verander de manier van werken, met als doel onderhoudbaarheid*

Om de kwaliteit te waarborgen en om het systeem continu te kunnen verifiëren zal binnen het project Test Driven Development (TDD) gewerkt worden. Er zijn meerdere manieren om deze methode te volgen, maar uiteindelijk zorgt het voor een dezelfde doel: een systeem dat onder test staat. Dit beantwoordt de volgende voorwaarde die genoteerd staat in de bijlage D : Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?:

Geautomatiseerde tests zorgen voor gemakkelijke validaties bij veranderingen binnen het systeem.

5.2 Systeem architectuur

Als uitgangspunt van het vergelijkingsonderzoek over programmeertalen en architecturen is er antwoord gegeven op de vraag: welk structuur of structuren zijn geschikt voor het nieuwe online-platform van Perflectie? De uitkomst van de geschikte architectuur kon dan in een architecture notebook geïmplementeerd worden waar het nieuwe online-platform zich aan moet houden.

Het conceptueel ontwerp, in hoofdstuk 3.4 Conceptueel model, en het conceptueel architecture notebook (zie bijlage E : Concept architecture notebook) is in het begin van dit project opgezet zodat Perflectie een beeld kon krijgen van de applicatie en zo al kon beginnen met ontwikkelen. Op basis van de keuzes van Perflectie, in hoofdstuk 5.2.2 Gekozen programmeertalen & architectuur, is het concept aangepast tot een officieel ontwerp welke is bijgestuurd tijdens iteraties.

5.2.1 Te realiseren eisen

De globale doelstellingen van het nieuwe online-platform luiden nu als volgt:

Eisen

- Het systeem is onderhoudbaar
- Het systeem is uitbreidbaar
- Het systeem houdt zich aan het, beschreven in de volgende documenten, ontwerp:
    - Architecture notebook
    - Iteration plan
- Functionaliteiten van het huidige online-platform van Perflectie
- Functionaliteiten van de Partner module
- Het systeem is testbaar
- Het systeem is flexibel
- Het systeem is veilig
- Het systeem is toelaatbaar
- Het systeem is schaalbaar

De wijziging ten opzichte van het conceptueel model is dat er geen specifiek functioneel- of technisch document wordt gemaakt. Dit komt omdat het iterationplan de functionele en technische eisen per sprint afdekt. Zo kan het systeem iteratief ontwikkelt worden, waarbij voor elke sprint het iterationplan bijgewerkt wordt met functionele en technische eisen.

Ook zijn de eisen gekozen door Perflectie, die gedefinieerd zijn in hoofdstuk 4.2.4 Gekozen eisen van Perflectie, toegevoegd aan de globale doelstellingen.

5.2.2 Gekozen programmeertalen & architectuur

Backend

Het advies voor een full web stack framework, in hoofdstuk 4.1.6 Conclusie te adviseren programmeertaal, is door Perflectie afgekeurd. De reden hieroor is dat het huidige online-platform ontwikkeld is in C# onder de MVC structuur. Perflectie wil graag zoveel mogelijk modellen en functionaliteiten hergebruiken. Dit kan niet gerealiseerd worden als er voor het full web stack framework wordt gekozen. Om deze reden zal de backend van het nieuwe online-platform worden ontwikkeld in C#.

Client

Om een rijkere client-side te creëren welke gemakkelijker te onderhouden is, wil Perflectie het advies meenemen gegeven in hoofdstuk 4.1.4.4 Conclusie client-side programmeertaal. Als AngularJs ingezet wordt in combinatie met LESS en Bootstrap 3, dan kan door middel van REST API communicatie plaats vinden tussen de client en de server.

Database server

Omdat C# wordt gekozen als backend taal is er een Windows server operating systeem verreist. Dit omdat binnen de server .Net geïnstalleerd moet worden om een C# applicatie te kunnen uitvoeren. Een groot voordeel van het gebruik van alleen Microsoft software is dat het op elkaar is afgestemd. Zo kan vanuit Visual studio het project gemakkelijk gepubliceerd worden in een Windows server. Maar er kan ook gebruik worden gemaakt van de gratis Microsoft SQL server. Via Microsoft SQL server kun je meerdere DBSM databases creëren waarin het nieuwe online-platform zijn data kan opslaan.

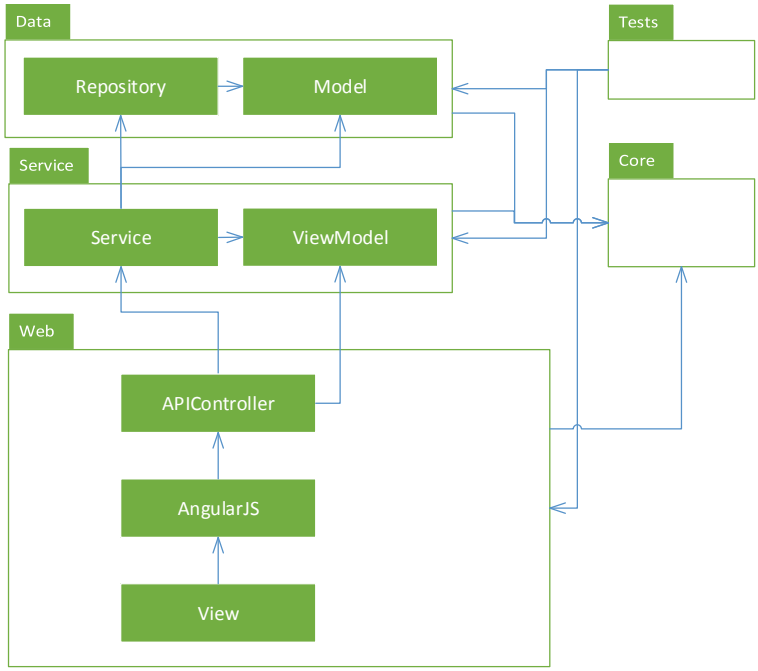
Structuur

Het advies dat gegeven wordt in hoofdstuk 4.2.6.1 Aanbeveling: Client/Server met een N-Tier structuur wordt deels door Perflectie overgenomen. Door tijdsdruk zal er niet component-based gewerkt worden en zal de eerste versie van het online-platform niet draaien op meerdere fysieke platformen. Ook zal de service-oriented

pattern niet meegenomen worden in het systeem, maar wordt er op basis van de REST architecture style gewerkt. Door middel van REST kan een applicatie eenvoudig via http, requests naar de server sturen om zo functionaliteiten en/of data op te vragen. (Haddad, 2014)

5.2.3 Architecture notebook

Logical layers model



Figuur 5 Tier Architecture concept

Figuur 5 is een weergave van de lagen structuur van het nieuwe online-platform. De lagen zijn gelijk aan het conceptueel model maar de rol van enkele lagen zijn gewijzigd. Zo kan de ApiController binnen de client gebruik maken van zowel services als de viewmodels. Dit moet omdat de controller platte data van een model moet doorgeven naar de client, een viewmodel genoemd. De conversie van een model naar een viewmodel gebeurd meestal al in de service, maar de controller moet wel kennis hebben van dit viewmodel.

De service maakt gebruik van viewmodels in zijn eigen tier en de models vanuit de data tier. Er is gekozen om het model naar de data tier te verplaatsen omdat alle models en relaties onderling de context van de database vormt. Vanuit de context wordt de datastructuur van het systeem gedefinieerd. Zo kan een Repository, welke door een service aangeroepen kan worden, CRUD(Create, Read, Update en Delete) functionaliteiten naar de database versturen.

Door de iteratieve aanpak kunnen de tier structuur en de rollen aangepast worden tijdens het project. Wanneer dit gebeurd wordt ook de documentatie hierop aangepast.

Testen

Om aan de eis te voldoen dat het systeem testbaar is wordt Test Driven Development (TDD) toegepast. Deze methode, dat vooral ingezet wordt bij agile development, is niet veranderd ten opzichte van het voorgestelde concept in hoofdstuk 3.4 Conceptueel model. Vanuit TDD kan afgeleid worden of het systeem aan de afgesproken eisen voldoet. Door middel van externe middelen kan het systeem ook real-time worden getest. (Palermo, Guidelines for Test-Driven Development, 2014)

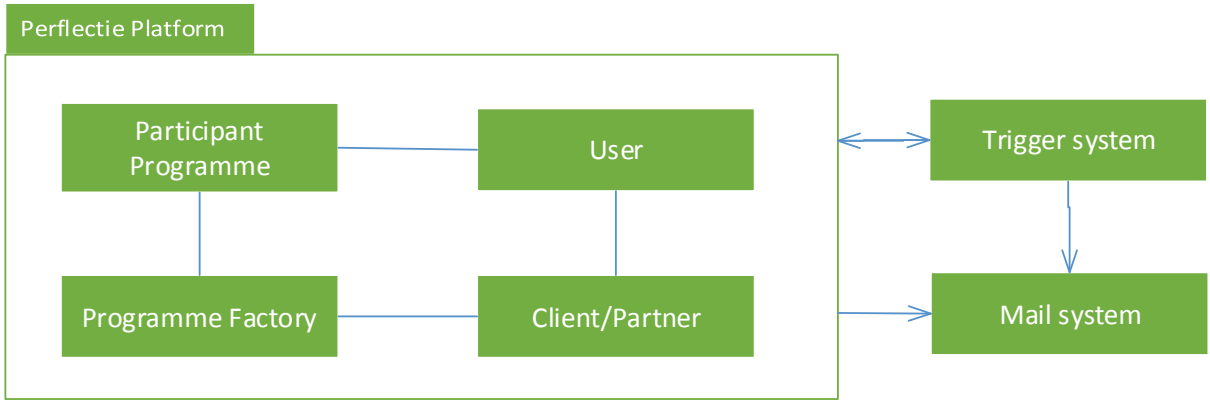
Onderdelen van het nieuwe online-platform van Perflectie

Het nieuwe online-platform van Perflectie zal uit de volgende componenten bestaan:

- De programme factory
- Client/Partner
- Gebruikers

- Participants & feedbackmembers
- Mail systeem
- Trigger systeem

Als onderdeel van het nieuwe online-platform was ook een agent trigger systeem ingepland. Dit is geschrapt nadat bekend werd dat er te weinig tijd is om dit agent trigger systeem te realiseren voor de deadline. Om het nieuwe online-platform wel te voorzien van een systeem dat functionaliteiten kan uitvoeren op basis van gebeurtenissen, zal er een extern product gebruikt worden die aan de trigger functionaliteiten voldoet. Hierbij mis je de vrijheid dat iemand met weinig tot geen kennis van programmeren triggers kan instellen, maar ook dat de applicatie direct in zijn of haar code de trigger systeem activeert.



Figuur 6 Concept componenten voor het nieuwe platform

Binnen de componenten in Figuur 6 Concept componenten voor het nieuwe platform, is alleen de agent component gewijzigd. Die wordt vervangen door een extern component die acties kan uitvoeren in een wachtrij of op een bepaald tijdstip. Deze componenten worden zowel front-end, back-end als backoffice ontwikkeld en worden daarna geïntegreerd. Dit zal op een iteratieve wijze gebeuren, binnen de Scrum methode (zie B : Management methode: Agile Scrum), zodat er bij iedere iteratie verder gedetailleerd ontwikkeld kan worden.

5.3 Iteration plan

Het iteration plan (zie bijlage H : Iteration plan) is een document waarvan de inhoud onderling is afgesproken binnen Perflectie. Binnen dit document worden alle eisen vanuit de Product Owner geregistreerd en later omgezet naar backlog items en taken. Het document wordt voor en na elke sprint bijgewerkt. Zo worden de functionele eisen aangepast en/of toegevoegd of wordt het domain model ge-update.

Door middel van Agile development en Domain-driven design, in hoofdstuk 5.1.1 Agile development, zijn alle functionele eisen van het nieuwe online-platform in kaart gebracht. Voor ieder sprint is er door middel van interviews met de Product Owner de wensen en eisen van het nieuwe platform in kaart gebracht. Deze wensen werden vervolgens verwerkt tot backlog items en taken om deze functionele eis te realiseren. De overzet van concept naar functionaliteiten staan genoteerd in het Iteration plan bij elke sprint. (zie bijlage H : Iteration plan)

5.4 MoScoW

De MoScow-methode is een prioriteiten lijst welke gebruikt kan worden binnen software development. De MoScow methode is een afkorting voor:

- Must haves
  - De requirements die in het eindresultaat aanwezig moeten zijn
- should haves
  - Deze eisen zijn gewenst maar zonder is het product ook gerealiseerd en bruikbaar
- could have
  - Deze eisen worden gerealiseerd als daar tijd voor over is
- won’t haves

- Deze eisen zullen in dit project niet aan bod komen maar voor toekomstige aanbeveling zou dit interessant zijn

Voordat er met dit project begonnen was is er een MoScow gedefinieerd. Deze staat gedocumenteerd binnen het plan van aanpak (zie bijlage G : Plan van aanpak). Gedurende de ontwikkeling is de MoScow aangepast omdat een onderdeel verviel door tijdgebrek. Het agent trigger systeem kon niet gerealiseerd worden. Hiervoor moest vervanging geregeld worden die wel binnen het tijdschema gerealiseerd kon worden en die dezelfde functionaliteiten kon invullen. Daarom zal er een extern trigger systeem geimplementeerd worden die functies binnen de applicatie kan aanroepen.

Wanneer voor dit project alle must haves zijn geïmplementeerd volgens de principes, technieken en methodes die onderling zijn vastgesteld dan is dit project succesvol gerealiseerd.

Must haves (zie hoofdstuk 3.4 Conceptueel model)

- De programme factory
- Client/Partner
- Gebruikers
- Participants & feedbackmembers
- Mail systeem
- Trigger systeem
- De mogelijkheid binnen het systeem om e-mails te versturen op basis van triggers welke afgehandeld wordt door een trigger systeem.
- De mogelijkheid binnen het systeem om functionaliteiten uit te voeren op basis van triggers welke afgehandeld wordt door een trigger systeem.
- Bestaande functionaliteiten van het huidige online-platform

Should haves

- Real-time feedback van het systeem naar de gebruiker
- De mogelijkheid om via CRUD functionaliteiten triggers te onderhouden binnen het systeem

Could haves

- Het implementeren van de mogelijkheid om meerdere soorten persoonlijkheidstesten voor een programma te kunnen kiezen.

Won't haves

- De gebruiker kan op basis van een iteratieve wijze zijn competenties en levels ontwikkelen.

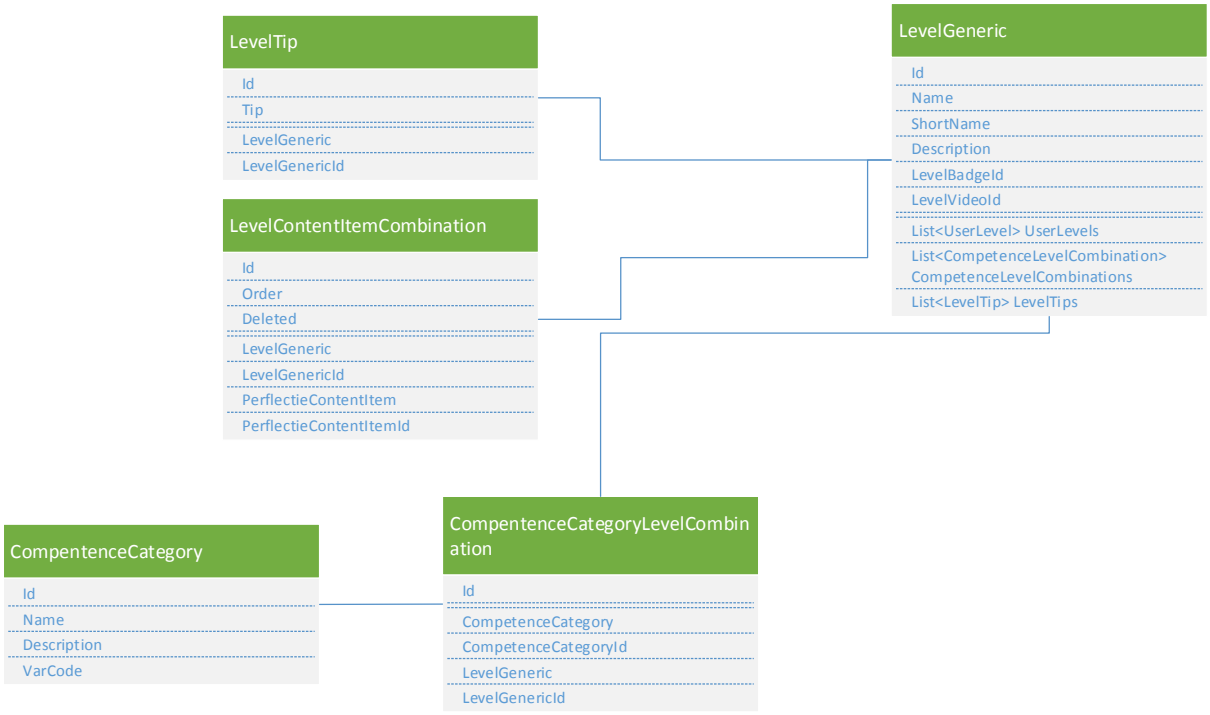
6 De resultaten van de opdracht

6.1 Resultaat van gebruikte methodes

Om een voorbeeld te geven van de uitkomst van de gebruikte methodes, om een onderhoudbaar systeem te bereiken, is het volgende scenario (zie volledige uitwerking concept bijlage K : Uitgewerkt concept Merge functionaliteit) als voorbeeld genomen.

Scenario

The programme factory – Binnen sprint 1 wilde Perflectie het onderdeel Programme factory realiseren. De programme factory houdt in dat Perflectie of een Partner zijn eigen programma kan samenstellen uit modules, levels, spannende momenten en content items. Binnen het huidige systeem was dit slordig en inconsistent opgezet. Om een impressie te geven van de implementatie van de programme factory in het huidige platform is er een gedeelte van de domain model in UML gebracht. (zie Figuur 14 Huidige online platform Programme factory)



Figuur 14 Huidige online platform Programme factory – Level naar competentie (Module)

Elke onderdeel binnen de programme factory , zoals het level in figuur 14, was aangeduid als Generic en bestonden er combinatie classes naar andere classes. Zo was de gehele structuur van een programma en zijn onderdelen en de relaties onderling hard geprogrammeerd in de applicatie. Dit maakte de applicatie onoverzichtelijk en niet onderhoudbaar.

6.1.1 Projectmanagement

Agile Scrum meetings

Voordat er begonnen werd met sprint 1 en het realiseren van de Programme factory, werden er eerst interviews gehouden om alle functionele eisen van sprint 1 te verzamelen. Deze zijn gedocumenteerd binnen het Iteration plan (zie bijlage H : Iteration plan- sprint 1 - User Stories). Om tot deze resultaat te komen zijn er kritische vragen gesteld over de programme factory en waar deze aan moet voldoen. Voor het interview is geen vragenlijst gebruikt, de informatie is vrij door middel van een gesprek met de Product owner verkregen.

Er zijn wel extra functionaliteiten bijgekomen ten opzichte van het huidig online-platform. Zo moet het voor een partner mogelijk zijn om een programma of een onderdeel van een programma te kunnen baseren op bestaande data van hetzelfde type binnen zijn eigen omgeving of die van Perflectie. Dit houdt in dat een programma (een child programme) data overneemt van zijn afgeleide programma (een base programme). Deze data kan hij op zijn beurt ook overschrijven, maar als de partner dit doet krijgt hij geen updates meer door van zijn base van de specifieke overschreven data.

Het voordeel hiervan is dat een partner altijd de laatste versie van een programma van Perflectie kan gebruiken maar toch zijn eigen persoonlijke toevoegingen of aanpassingen kan doorvoeren. Hiervan is als eerst het concept uitgetekend voordat er begonnen werd met het modeleren van het domain model.

6.1.2 Toegepaste ontwikkel methodes

Om een voorbeeld te geven van de onderzochte en gekozen methodes (zie hoofdstuk 5.1 De gebruikte methodes) wordt hier de scenario (zie bijlage K : Uitgewerkt concept Merge functionaliteit) gebruikt om een aantal functionaliteiten uit te werken. Als uitkomst wordt er onderhoudbaar code verwacht dat gemakkelijk te begrijpen is. Dit wordt getest door de onderzochte meetmethode(zie hoofdstuk 5.1.2 Principes, technieken en methodes). Deze methode bestaat uit vragen wat een programmeur aan zichzelf en andere kan stellen en deze vragen zullen afgenomen worden door de ingehuurde ontwikkelaars van Perflectie.

Test-driven development
De functionele eisen vanuit de product backlog die worden gebruikt zijn:

Ontwikkelprogramma (Child Programme). Daarbinnen:

1. basisprogramma componenten kunnen zien

Plan
Voordat er begonnen werd met het uitwerken van de backlog items is het Merge concept uitgewerkt. (zie bijlage K : Uitgewerkt concept Merge functionaliteit) En hiervoor een backlog item met taken aangemaakt. Dit was de plan fase.

```
[Fact]
public void I_Want_To_See_My_BaseAttributes_Which_Is_Not_Overriden()
{
    //Assign
    var baseProgramme = new Programme
    {
        Name = "BaseProgramme",
        Description = "Dit is een base programme"
    };
    var childProgramme = new Programme
    {
        Name = "Mijn partner programme",
        Parent = baseProgramme
    };

    var mergedObject = childProgramme;

    //Act

    //Assert
    Assert.Equal("Mijn partner programme", mergedObject.Name);
    Assert.Equal("Dit is een base programme", mergedObject.Description);
}
```

Figuur 21 Red test

Red
De red fase is er om ervoor te zorgen dat de test opgezet wordt zodat de gewenste data verwacht wordt, maar de test zelf en zijn functionaliteiten werken nog niet. Om een red test te implementeren word er een When_A_Child classe aangemaakt in de Tests laag. Deze classe bevat een methode genaamd I_Want_To_See_My_BaseAttributes_Which_Is_Not_Overridden(). Figuur 21 is een schermafbeelding van de uitgewerkte test in Red fase.

```
var currentObject = childProgramme;

if (currentObject != null)
{
    do
    {
        //geef alle properties van de class dat gemerged kunnen worden
        List<PropertyInfo> programmeProperties = mergedObject.GetType().GetProperties().Where(x => !Attribute.IsDefined(x, typeof(NotMergableAttribute))).ToList();

        //loop door properties en vervang de null value's van het meegestuurde object
        foreach (PropertyInfo programmeProperty in programmeProperties)
        {
            object value = programmeProperty.GetValue(currentObject, null);

            if (value == null)
            {
                continue;
            }

            object oldValue = programmeProperty.GetValue(mergedObject, null);

            if (oldValue == null)
            {
                programmeProperty.SetValue(mergedObject, value);
            }

        }

        currentObject = currentObject.Parent;

    } while (currentObject != null);

}

//Assert
Assert.Equal("Mijn partner programme", mergedObject.Name);
Assert.Equal("Dit is een base programme", mergedObject.Description);
```

Figuur 22 Green test

```
[Fact]
public void I_Want_To_See_My_BaseAttributes_Which_Is_Not_Overriden()
{
    //Assign
    var baseProgramme = new Programme
    {
        Name = "BaseProgramme",
        Description = "Dit is een base programme"
    };
    var childProgramme = new Programme
    {
        Name = "Mijn partner programme",
        Parent = baseProgramme
    };

    var mergedObject = childProgramme.MergeWithParents();

    //Act

    //Assert
    Assert.Equal("Mijn partner programme", mergedObject.Name);
    Assert.Equal("Dit is een base programme", mergedObject.Description);
}
```

Figuur 23 Refactor test

Green
Binnen de green fase wordt alles gedaan om de test groen te krijgen. In
dit scenario is het erven van attributen
die de child niet heeft Overridden. (zie figuur 21 voor de green test)

Refactor
Verwerk de logica binnen het systeem en optimaliseer de code.

Binnen figuur 23 is te zien dat de logica in de green test figuur 22 is verwerkt binnen één methode. Deze methode is genoemd MergeWithParents().

De code die gebruikt wordt moet nog eerst geoptimaliseerd worden en vervolgens het S.O.L.I.D principe volgen. Als een programmeur terugkijkt naar de code of er volgt een bug waardoor het systeem niet werkt is het niet vanuit figuur 22 af te lezen waar het probleem ligt of wat de methode precies doet.

Daarom moet de code geoptimaliseerd worden. Omdat de Merge functionaliteit toepasbaar moet zijn voor meerdere classes binnen het systeem is de MergeWithParents() methode een generic static extension method geworden die een IMergable verwacht. Hierdoor kunnen alle klassen die de Interface IMergable implementeren deze methode gebruiken. De klasse moet dan wel tenminste voldoen aan de Parent property.

Clean
Om de methode onderhoudbaar en duidelijk te houden, ook al is het generic opgezet, is er gekozen voor mergable-, merged- en currentobject. Dit om de benaming van het object een zo duidelijk mogelijk betekenis te geven.

Ook de nesting en if statements die te zien zijn in figuur 22 zijn niet meer terug te vinden in figuur 23. Van deze nesting en if methodes zijn er specifieke methodes aangemaakt met maar één taak.

S.O.L.I.D
Single responsible
IMergable interface bezorgt voor de classes die de interface implementeren de Merge functionaliteiten. Hierdoor is de IMergable alleen verantwoordelijk voor deze functionaliteiten.

Open/Closed
Het open/closed principe wordt niet toegepast op de MergeWithParents methode omdat er niet verwacht wordt dat de functionaliteit in de toekomst continu gewijzigd hoeft te worden.

Liskov
Omdat open/closed vervalt, vervalt ook de Liskov.

Interface segregation
Er is een IMergable interface aangemaakt dat specifiek de groep moet bereiken van objecten die moet kunnen mergen met parents. Daarom is de interface specifiek ingezet en wordt er geen gebruik gemaakt van een fat interface.

Dependency Inversion
Door middel van de IMergable interface, wordt er tegen een interface aan geprogrammeerd en niet tegen een klasse. Hierdoor is de methode niet afhankelijk van één of meerdere classe.

```
public static T MergeWithParents<T>(this T mergableObject)
    where T : IMergable<T>
{
    // gebruik een apart object om de verschillen te tracken
    var mergedObject = mergableObject.ReturnDefaultObject();

    //Zet de huidig object met de waardes van de meegegevenobject dat gemerged moet worden
    var currentObject = (T)mergedObject.InjectFrom(mergableObject);

    do
    {
        //geef alle properties van de class dat gemerged kunnen worden
        IEnumerable<PropertyInfo> mergedObjectProperties = currentObject.GetPropertiesThatAreMergable();

        //loop door properties en vervang de null value's van het meegestuurde object
        foreach (PropertyInfo property in mergedObjectProperties)
        {
            mergedObject.MergeValueOfPropertyIfNull(currentObject, property);
        }

        currentObject = currentObject.Parent;

    } while (currentObject.IsNotNull());

    return mergedObject;
}
```

Figuur 24 Refactored Clean & SOLID

6.3 Implementeren van design oplossingen

Design patterns

Om een probleem binnen het systeem snel op te lossen werd er gewerkt met een design pattern. Een design pattern is een blauwdruk om een bepaald probleem in het systeem op te lossen. De pattern is geen volledig ontwerp en heeft aanpassingen en toevoegingen nodig om te kunnen functioneren binnen een systeem. Het voordeel van het gebruik van patterns is dat het zijn karakteristieken behoudt.

Binnen Perflectie is er veel verzet tegen het implementeren van design patterns. Perflectie refereert naar geschreven artikelen, zo zouden design patterns:

- Tegen het principe van Keep It Simple, Stupid (KISS) ingaan

Dit is om tegen te gaan dat je niet je code forceert om in patterns te werken. Code moet eerst gerealiseerd worden met de KISS principe en later als het nog nodig is, patterns op toepassen.

- Zich richten op het oplossen van verkeerde problemen

Een concept zou niet gekopieerd moeten worden, maar enkel gerefereerd. Maar als er alleen gerefereerd wordt en niet gekopieerd dan is er niet officieel een pattern gebruikt. (Graham, 2014)

- Onbewust maak je al gebruik van design patterns binnen developing tools zoals Visual studio

Veel developingtools maken al gebruik van patterns zonder dat de gebruiker hier van bewust is. Hierdoor kan herhaling in code voorkomen worden. (Pierry, 2014)

Om bovenstaande redenen is er voor het nieuwe online-platform nog geen onderzoek gedaan naar te implementeren design patterns.

6.4 Conclusie

Om te testen of het nieuwe online-platform voldoet aan de voorwaarden waar een onderhoudbaar systeem aan moet voldoen, beschreven in de bijlage D : Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?, zijn de afgeleide vragen beantwoordt via gesprekken met Jochem Aubel. Zie hoofdstuk 5.1.2 Principes, technieken en methodes voor deze vragen.

Omdat de genoemde vragen in hoofdstuk 5.1.2 Principes, technieken en methodes gericht zijn op gemaakte code, zal aan de hand hiervan de eerder uitgewerkte concept van de Merge functionaliteit, in bijlage K : Uitgewerkt concept Merge functionaliteit, beoordeeld worden.

Om het nieuwe online-platform, na voltooiing, te beoordelen net als het huidig online-platform zijn er naast de vragen in hoofdstuk 5.1.2 Principes, technieken en methodes ook vragen opgenomen om het nieuwe online-platform te beoordelen.

Deze vragen zijn gesteld aan Jochem Aubel, in het begin van dit project om het huidige online-platform te beoordelen en aan het eind van dit project om het nieuwe online-platform te beoordelen.

6.4.1 Gemaakte code

Om te beoordelen of de gemaakte functionaliteiten vanuit de functionele eisen onderhoudbaar zijn, zijn de vragen, die genoteerd staan in de introductie van hoofdstuk 6.4 Conclusie, gesteld aan de developers van Perflectie. De antwoorden hierop zijn terug te vinden in bijlage I : Interviews : Het nieuwe online-platform & gemaakte functionaliteiten.

Uit het resultaat van de interviews is af te leiden dat de vastgestelde methodes (zie hoofdstuk 5.1 De gebruikte methodes) onderhoudbare code heeft gegenereerd. Bij hogere complexiteit of bij het niet meer opvolgen van de methodes door drukte, door bijvoorbeeld een deadline, is uit de interviews op te maken dat de kwaliteit van onderhoudbaarheid op zo'n moment te wensen over laat.

6.4.2 Het nieuwe online-platform

- Is het systeem vanaf het begin zo ontworpen dat het onderhoudbaar is?

Het nieuwe online-platform is vanaf het begin zo ontworpen, ondersteund door voorgaande onderzoeken (zie bijlage D : Vooronderzoek), dat het onderhoudbaar is . Deze onderzoeken hadden als resultaat technieken, methodes en principes om een onderhoudbaar systeem te creëren.

Daarom voldoet het nieuwe online-platform aan deze eis.

- Wordt het systeem op een iteratieve methode gemanaged/ontwikkeld?

Ja, de ontwikkeling van het nieuwe online-platform is met de Agile Scrum methodiek (zie bijlage B : Agile Scrum) gerealiseerd. Binnen deze methode wordt het project iteratief ontwikkeld door middel van sprints.

- Waren er / of bevinden zich reguliere reviews om de kwaliteit van het systeem te waarborgen?

Nee, er vindt geen controle plaats om het systeem op kwaliteit te waarborgen. Er wordt op dit moment blind vertrouwd op het werk van de ontwikkelaars, en dat ze zich aan de vastgelegde technieken, methodes en principes houden (zie hoofdstuk 5.1.2 Principes, technieken en methodes).

- Is er binnen het systeem leesbare code die gemakkelijk te begrijpen is?

Ja, de gehouden interviews onder de programmeurs van Perflectie zijn positief over het onderhoud en de begrijpbaarheid van de code. (zie bijlage I : Interviews : Het nieuwe online-platform & gemaakte functionaliteiten)

- Is de code gerefactored in een proces om de begrijpbaarheid te verbeteren?

Ja, doormiddel van Test Driven Development (TDD) is er een Refactor fase waar de ontwikkelaar zich aan moet houden. Deze fase is door het Scrum team van Perflectie uitgebreid met de clean code, KISS en het SOLID principe. (zie hoofdstuk 5.1.2 Principes, technieken en methodes).

- Is er relevante documentatie beschikbaar voor programmeurs om het systeem te begrijpen?

Nee, er is een iteration plan (zie bijlage H : Iteration plan) en een architecture notebook(zie bijlage F : Nieuwe architecture notebook). In het architecture notebook staat de structuur van het nieuwe online-platform in grote lijnen aangegeven. In het iteration plan staan de functionele eisen, met daarbij het domain model. Het iteration plan is echter na de derde sprint niet tot nauwelijks meer bijgewerkt. Om deze reden is de documentatie niet voldoende up-to-date.

- Kan het systeem automatisch gebuild worden zodat het eenvoudig is om de code te compileren?

Ja, omdat het systeem met behulp van Visual studio (zie bijlage L : De gebruikte software & extensies) is opgezet kan de code gemakkelijk gecompileerd en gebuild worden naar een externe omgeving.

- Zijn er (geautomatiseerde) tests aanwezig zodat het eenvoudig is om bij veranderingen het systeem te valideren?

Ja, door middel van test driven development (TDD), nCrunch en xUnit zijn er real-time tests aanwezig in het nieuwe online-platform. (zie bijlage M : Het opzetten van de ontwikkelomgeving)

- Is er continue integratie binnen de code zodat builden en testen gemakkelijk gemaakt wordt?

Ja, er is een perflectiedev omgeving waar de applicatie live getest kan worden voordat deze naar de live omgeving gepubliceerd wordt. Ook zijn de servers direct toegankelijk vanuit Visual studio, waardoor het builden naar deze omgeving eenvoudig en snel gaat.

- Wordt binnen het systeem version control gebruikt zodat ieder teamlid de laatste versie heeft van de code, testen en documentatie?

Ja, via GIT kan iedere teamlid binnen teamfoundation server de laatste versie van het project opvragen.

- Heeft ieder teamlid binnen het systeem de werkwijze aangehouden om zo onderhoudbaar mogelijk werk te produceren.

Nee, er is niet tot nauwelijks gecontroleerd of de werkwijze echt werd nageleefd door elke ontwikkelaar van Perflectie. Hierdoor is het niet meetbaar of aan te tonen of iedereen zich aan de vastgestelde werkwijze heeft gehouden. (zie hoofdstuk 5.1 De gebruikte methodes)

6.4.3 Vergelijking

Het huidige online-platform voldoet niet aan: (zie bijlage D : Vooronderzoek– Voldoet het huidige online-platform aan de resultaat aan voowaarden in deelvraag 2?)

- Ontwerp het systeem voor onderhoudbaarheid van het begin af
- Het systeem op een iteratieve methode managen/ontwikkelen
- Reguliere reviews om de kwaliteit van het systeem te waarborgen
- Leesbare code dat makkelijk te begrijpen is
- Refactor de code om de begrijpbaarheid te verbeteren
- Relevante documentatie dat programmeurs helpt om het systeem te begrijpen
- Geautomatiseerde tests zorgen voor gemakkelijke validaties bij veranderingen binnen het systeem
- Version control helpt om de code, testen en documentatie up to date te houden binnen het projectteam
- Verander de manier van werken, met als doel onderhoudbaarheid

Het nieuwe online-platform voldoet niet aan:

- Reguliere reviews om de kwaliteit van het systeem te waarborgen
- Relevante documentatie dat programmeurs helpt om het systeem te begrijpen
- Verander de manier van werken, met als doel onderhoudbaarheid

Het nieuwe online-platform

Zoals te zien is het nieuwe online-platform fors verbeterd ten opzichte van het huidige online-platform. Door het invoeren van de projectmanagementmethode van Agile scrum wordt ervoor gezorgd dat de meeste problemen tijdens het project zichtbaar worden gemaakt en worden opgelost. Dit komt omdat er iteratief te werk wordt gegaan en zo snel wordt ingespeeld op veranderingen vanuit de omgeving.

Naast iteratief ontwikkelen is het ook belangrijk dat de denkwijze van het ontwikkelteam verandert richting onderhoudbaarheid. Hiervoor zijn binnen de definition of done principes, technieken en methodes geplaatst waar een gemaakte code aan moet voldoen. Door middel van deze methodes en ook omdat de methodes worden afgestemd met andere ontwikkelaars worden de gemaakte functionaliteiten begrijpelijk en onderhoudbaar.

Er moet wel zorgvuldig met deze methodes worden omgegaan. Door onzorgvuldigheid heeft het nieuwe online-platform een aantal voorwaarde waaraan niet wordt voldaan. Door haastig werk om bijvoorbeeld een deadline te halen of door slechte communicatie wordt de integriteit van het systeem aangetast zonder dat iemand zich dit realiseert. Om deze reden is het belangrijk om regelmatig controles uit te voeren op gemaakte functionaliteiten en te beoordelen of deze voldoen aan de vooraf vastgestelde voorwaarden.

Het iteration plan is binnen dit project in het begin met domain driven design goed bijgehouden. Na sprint 3 is dit gedeelte niet verder bijgewerkt en zijn er alleen functionele eisen vastgesteld. Dit heeft geresulteerd in het probleem dat er geen betrouwbare en up-to-date gedetailleerde documentatie beschikbaar zijn van een functioneel ontwerp van het nieuwe online-platform.

7 Eind Conclusie

7.1 *Hoe moet het nieuwe online-platform voor Perflectie worden ontworpen zodat, met de partner module inbegrepen, het systeem onderhoudbaar en uitbreidbaar is?*

Er is geen standaard oplossing beschikbaar voor het realiseren van een onderhoudbaar en uitbreidbaar systeem. Om hier te komen moet er een proces doorlopen worden om de gewoontes, de principes, de methodes en de technieken van het ontwikkelteam en het bedrijf op een iteratieve en leerzame manier aan te

passen richting onderhoudbaarheid en uitbreidbaarheid. Het begint met bewustwording en verandering van gedrag, waarbij de principes, methodes en technieken ondersteunen om het gewenste resultaat te bereiken.

De best passende projectmanagement methode voor Perflectie waarin de medewerkers kunnen leren en hun gedrag en werkwijze kort cyclisch kunnen aanpassen is de Scrum Agile methode. Door de dynamiek van het online-platform is continue ontwikkelen noodzakelijk, om dit onderhoudbaar en uitbreidbaar te houden is een kort cyclische projectmethodiek een voorwaarde.

Om het platform maximaal onderhoudbaar en uitbreidbaar te maken is het belangrijk om de voorwaarden van de architectuur niet alleen af te stemmen op de wensen en eisen van Perflectie en haar gebruikers maar ook zeker op de voorwaarden van onderhoudbaarheid en uitbreidbaarheid. De client server N-Tier structuur bleek voor Perflectie de meest geschikte architectuur.

Om continue het systeem te valideren is de Test Drive Development (TDD) methode noodzakelijk, door het automatisch testen wordt veel tijd bespaard en de kwaliteit gewaarborgd. Hierbij is het van belang strakke richtlijnen op te stellen en je hier ook aan te houden.

Ondanks dat niet alle eisen zijn ingevuld zoals in dit document geadviseerd kan het systeem in grote lijnen voldoen aan de technische en functionele eisen van een systeem die onderhoudbaar en uitbreidbaar is.

Als tijdens het ontwikkelen en het door ontwikkelen van het nieuwe online-platform rekening wordt gehouden met de in dit document beschreven aanbevelingen dan wordt het systeem maximaal onderhoudbaar en uitbreidbaar opgebouwd. Door de snelle ontwikkeling van de omgeving van een systeem is het noodzakelijk om de software principes, methodes en technieken continue aan te passen zodat het systeem ook onderhoudbaar en uitbreidbaar blijft in de toekomst.

8 Aanbevelingen

Reguliere controles

Het nieuwe online-platform heeft nog enkele voorwaarden waar het niet aan voldoet. Dit is vooral veroorzaakt door haast en onzorgvuldigheid tijdens het project. Om te waarborgen dat de vastgestelde methodes binnen de ontwikkeling van Perflectie gevolgd worden, is het aan te bevelen om hier reguliere controles op uit te voeren. Dit kan door elk gereed gemelde backlog item samen met een collega te controleren conform de Definiton of Done.

Reguliere onderzoeken

Doordat de technieken achter het maken van software snel ontwikkelen is het belangrijk om up-to-date te blijven. Om deze methodes, principes en technieken en de actualiteit ervan te waarborgen is het aan te bevelen om wekelijks of maandelijks bevindingen met elkaar te delen door middel van een presentatie. Hierbij presenteert elk lid van het scrum team een methode, principe en/of techniek die in de ontwikkeling meegenomen kan worden.

Façade pattern

Er bevinden zich enkele complexe functionaliteiten binnen het systeem, de Merge functionaliteiten is daar een voorbeeld van. Hierdoor is het niet duidelijk voor een programmeer welke methode hij of zij moet gebruiken om een bepaalde functionaliteit te realiseren. Bij dit soort complexe problemen is het aan te raden om een façade pattern te implementeren. Een façade pattern zorgt ervoor dat de communicatie van een sub-systeem/component via één interface verloopt. Dit zorgt voor abstractie waardoor de complexiteit wordt verminderd.

Het verwerken van persoonsgegevens

Door Perflectie is er nog geen rekening gehouden met de wetgeving bij het verwerken van persoonsgegevens. Hiervoor is er een onderzoek gestart om informatie te verzamelen wat de wet rondom persoonsgegevens precies inhoudt en op welke manier Perflectie hier rekening mee moet houden. (zie bijlage C : Verwerken van persoonsgegevens binnen de Nederlandse wet)

Perflectie heeft in het kader van dit onderzoek drie vragen opgesteld die zij graag beantwoord krijgen:

- Welke data mag Perflectie opslaan?

Perflectie slaat persoonsgegevens op van zijn gebruikers die kunnen terugleiden tot een natuurlijk persoon. Hierdoor moet Perflectie zich inschrijven bij de CBP en zich houden aan de WBP wet. Daarnaast mag Perflectie alleen de persoonsgegevens opslaan voor een doel waarvoor de gebruiker vooraf duidelijk is ingelicht. De regels en inhoud van deze wet is gedocumenteerd in bijlage Verwerken van persoonsgegevens binnen de Nederlandse wet – WBP.

- Hoe moet deze data beveiligd worden?

Er worden geen duidelijke eisen gesteld vanuit de WBP, CBP of NCSN, wel worden richtlijnen aangegeven waar de hard- en software van een organisatie aan moet voldoen en waar de organisatie op beoordeeld kan worden. (Zie bijlage C : ICT-beveiligingsrichtlijnen voor web applicaties van het Nationaal Cyber Security Centrum) Hiervoor heeft de NCSN een lijst beschikbaar gesteld met daarin de actuele beveiligingsrisico's voor (web-)applicaties. Deze risico's worden bepaald door de inschalingmatrix welke gedocumenteerd staan in de bijlage NCSN Inschalingsmatrix. Als laatste is hier een cyclus gedocumenteerd waarbij een organisatie blijvend een passend beveiligingsniveau kan behouden. De plan-do-check-act-cyclus. Hier kunt u meer over lezen in bijlage Plan-do-check-act.

- Wie mag bij deze data komen?

Persoonsgegevens mogen alleen verwerkt worden voor een bepaald doel en op basis van dit doel mogen geautoriseerde gebruikers bij deze data komen.

De gebruiker
Iedereen heeft recht op inzage in zijn of haar persoonsgegevens. Bij buitensporig (vaak) verzoek hoeft hier geen gehoor aan gegeven worden. Bij een verzoek moet de organisatie binnen vier weken handelen, waarbij het antwoord schriftelijk moet zijn. Hierbij moet de organisatie aan de gebruiker:

- Een volledig overzicht van de door de organisatie verwerkte gegevens leveren
- Een omschrijving van
    - Het doel of de doeleinden van de gegevensverwerking
    - De categorieën van gegevens waarop uw verwerking betrekking heeft
    - De ontvangers of categorieën van ontvangers
- Alle beschikbare informatie over de herkomst van de gegevens afgeven

Als er naar wordt gevraagd is de organisatie verplicht om informatie over de systematiek van de geautomatiseerde gegevensverwerking te verstrekken, dit hoeft alleen als daarbij geen bedrijfsgeheimen worden prijsgegeven.

Nadat de onderzoek voltooid was wilde Perflectie ook weten welke stappen Perflectie moet ondernemen om persoonsgegevens te mogen verwerken. Hier kan op basis van het vooronderzoek geen eenduidig antwoord op worden gegeven en moet om deze reden meegenomen worden in toekomstige onderzoeken.

9 Toekomstige onderzoeken

Reguliere controles Het systeem is niet in detail gedocumenteerd, daarom is het aan te raden een onderzoek uit te voeren op welke manier het systeem en/of documentatie gecontroleerd kan worden. Zodat in de toekomst reguliere controles op een juiste manier uitgevoerd kunnen worden.

Reguliere onderzoeken
Reguliere onderzoek naar ontwikkelingen van buitenaf is een methode dat bij ieder softwareontwikkelingsbedrijf hoort. Ontwikkelingen gaan snel en door ontwikkelingen van buitenaf te volgen blijven de medewerkers elkaar stimuleren om te leren van nieuwe bruikbare technieken en deze uit te proberen.

Abstractie

Op dit moment staat er een nieuw online-platform met geïmplementeerde functionaliteiten die zich aan de KISS en SOLID principes houdt. Om overzichtelijkheid te houden in het platform is het aan te raden om patterns en/of components te implementeren om de abstractie van het systeem te verhogen. Hiervoor zou een onderzoek gestart kunnen worden om de impact te bepalen op het huidige online-platform en de manier waarop dit gedaan kan worden.

N-Tier
Het nieuwe online-platform is opgebouwd uit meerdere lagen die fysiek gescheiden kunnen worden. Op dit moment is dit nog niet gedaan en staan alle lagen op dezelfde server. Dit kan beveiligingsrisico's met zich meebrengen, daarnaast is het systeem niet gemakkelijk uit te breiden. Het is aan te raden om een onderzoek te starten op welke manier dit voor het nieuwe online-platform gerealiseerd kan worden en wat de impact hiervan is.

Beveiliging
Het nieuwe online-platform is niet opgezet met als hoofddoel beveiliging. Wanneer Perflectie steeds meer bekendheid krijgt dan wordt de dreiging van buitenaf ook steeds groter. Om deze reden is het belangrijk dat Perflectie een onderzoek start welke dreigingen verwacht kunnen worden en hoe het platform hiertegen te beveiligen is.

Het verwerken van persoonsgegevens
Het verwerken van persoonsgegevens moet zorgvuldig en volgens de wet verlopen. Dit voor zowel Perflectie als zijn gebruikers van het online-platform. Daarom is het belangrijk om dieper in te gaan op de resultaten van het al uitgevoerde onderzoek en deze aan te vullen (zie bijlage C : Verwerken van persoonsgegevens binnen de Nederlandse wet).

10 Evaluatie van de procesgang

Start-up Perflectie
Perflectie is een start-up bedrijf gericht op het stap voor stap helpen bij het ontwikkelen van competenties van personen. Dit doen zij door middel van een online-platform. Mijn initiële taak was om het huidige online-platform van Perflectie uit te breiden met functionaliteiten van een Partner module. Maar na een onderzoek gedaan te hebben naar het huidige online-platform is deze opdracht fors veranderd.

Geen kans op falen
Zekerheid staat bij Perflectie voorop. Hierdoor was het erg lastig om adviezen en veranderingen binnen Perflectie en het online-platform door te voeren. De eenvoudige reden hiervoor was dat Perflectie geen geld en tijd meer heeft als het project faalt. Mocht dit gebeuren dan zou het bedrijf Perflectie direct ophouden met bestaan.

Vooronderzoek
Om Perflectie toch van gedachte te laten veranderen is er een vooronderzoek gestart om in kaart te brengen waarom het huidige online-platform niet onderhoudbaar is. Hiervoor werd de definitie van onderhoudbaar in kaart gebracht en hoe je een systeem kunt beoordelen of deze onderhoudbaar is. Uit het resultaat van dit onderzoek werd als snel duidelijk dat het huidige online-platform niet geschikt was om verder op te ontwikkelen. Het huidig online-platform, gebouwd door Netwinst, is door veel kleine veranderingen en updates niet onderhoudbaar en niet uitbreidbaar geworden.

Komst Dimitri Tholen
Dimitri was een grote aanwinst in dit project. Dimitri Tholen heeft veel kennis van zijn vak als software developer. Dit hielp bij het overtuigen van Perflectie dat het beter was om een nieuw project op te zetten voor het online-platform. Dit omdat nieuwbouw zelfs minder tijd kost dan het huidige online-platform uit te breiden met nieuwe functionaliteiten.

Afvallen van Netwinst
Perflectie heeft, nadat het heef gekozen voor nieuw bouwen van het online-platform, besloten om Netwinst niet meer bij de bouw te betrekken. Netwinst was verantwoordelijk voor de staat van het huidige online-

platform.

Komst Serge Bekenkamp
Serge Bekenkamp kwam als parttime developer en assisteerde bij het ontwikkelen van het nieuwe online-platform.

Procesgang van het Scrum Team
Tijdens het samenwerken in het Scrum team dacht ik bij de eerste twee sprints, dus de eerste 4 weken van dit project, dat alles soepel verliep en goed ging. Ieder lid van het scrum team was bezig met zijn eigen gedeelte van een feature die vanuit de backlog was vastgesteld.

Alleen bij zowel sprint 1 als sprint 2 werd er geen prototype gerealiseerd. Na sprint 3, toen we een prototype wilden publisheren liepen we tegen problemen aan. Hierna moest de snelheid van ontwikkelen verhoogt worden om de deadline te halen. Ook de procesgang binnen het project moest veranderen.

Nadat ieder lid van het scrum teamverantwoordelijk werd voor het afmaken van een hele feature, in plaats van een gedeelte, kwam er voortgang in de ontwikkeling.

Om deze reden is na sprint 3 ook nauwelijks tot geen documentatie bijgehouden omdat het realiseren van het product voorop stond.

Vaststellen van methodes, principes en technieken
Het onderzoeken van de juiste managementmethode, programmeertaal en architectuur vond ik erg lastig. Dit komt omdat het ontwerp erg breed is. Ik ben erg tevreden met de resultaten van elk onderzoek, waarbij Perflectie en ikzelf veel van hebben geleerd.

Afronden van het nieuwe online-platform
Het koste veel effort en tijd van ieder teamlid om het product binnen de deadline af te kunnen krijgen. Dankzij de geweldige inzet van iedereen is dit toch gelukt. Helaas is door de haast de integriteit van het nieuwe online-platform aangetast. Dit kwam omdat men zich niet altijd aan de vastgestelde methodes hield.

Evaluatie van mijn stage
Ik vond het een erg leerzame stage waarin ik veel geleerd heb van elke aspect in het maken van een web-applicatie. Het was echter vaak wel een moeizaam proces om veranderingen en/of adviezen binnen Perflectie door te voeren, maar hiervan heb ik wel geleerd om op een correcte manier een ander te overtuigen van mijn overtuigingen en keuzes.

Tijdens het ontwikkelen van het nieuwe online-platform is er niet tot nauwelijks gekeken naar gemaakt ontwerpen, dit werd ook niet afgedwongen. Wat ik hier jammer aan vond is dat er functionaliteiten gemaakt werden binnen het platform die niet correct of nodig zijn. Dit kon tegengegaan worden door eerst de documentatie te lezen.

Agile development is een bruikbare methodiek bij het ontwikkelen van een software product. Door continu feedback en ontwikkeling van het team zelf, evolueert het team langzaam naar een effectief en efficiënt productie team.

Als ik kijk wat er gerealiseerd is na drie maanden hard werken ben ik erg trots op het resultaat en bewonder ik de inzet van iedereen die betrokken is bij dit project.

Verklarende woordenlijst

1. **Encapsulatie**
   Door middel van encapsulatie worden aannames van ontwikkelaars bij het gebruik van methodes en functionaliteiten binnen het systeem tegen gegaan. Zo zorg je ervoor dat data types, methodes, properties of implementaties alleen worden getoond binnen zijn eigen laag of component. Hierdoor loopt de communicatie via één punt, de Interface die de laag of component definieert.
   Op deze manier kan er alleen gebruik gemaakt worden van objecten binnen deze laag of component, via de toegang die voor deze laag of component is gedefinieerd. (Java - Encapsulation, 2014)

2. **Gelaagde architectuur**
   Doordat lagen de systeem functionaliteiten afscheiden van een andere laag is het systeem helder en inzichtelijk. Een laag communiceert met de laag onder zich, en kan zo reageren op data die de laag terugkrijgt. Zo verloopt de flow van data en functionaliteiten tussen de lagen. (Architectural patterns and styles, 2014)

3. **High cohesion vs Low Cohesion**
   Bij high cohesion zorg je ervoor dat een klasse of functie een duidelijke taak krijgt. Low cohesion is wanneer de klasse of functie veel taken krijgt die weinig of niets met elkaar gemeen hebben. Door ervoor te zorgen dat componenten en/of lagen binnen het systeem duidelijke rollen toegewezen krijgen zorg je zo voor een duidelijk en onderhoudbaar systeem. Low cohesion wordt ook gezien als een code smell dat bad coding aanduidt. (Miller, Cohesion And Coupling, 2014)

4. **Loosely coupling vs tight coupling**
   Door middel van loosely coupling zorg je ervoor dat elk component en laag binnen het systeem gebruik maakt van andere componenten en/of lagen met minimale kennis van het component of laag. Dit zorgt ervoor dat het component en/of laag gebruikt kan worden in een ander scenario. Tight coupling is precies het tegenovergestelde en wordt ook gezien als een code smell. (Miller, Cohesion And Coupling, 2014)

Geciteerde werken

Architectural patterns and styles. (2014, 12 03). Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/ee658117.aspx

Avram, A. (2014, 12 08). *A quick look at Architectural styles and pattersn*. Retrieved from InfoQ: http://www.infoq.com/news/2009/02/Architectural-Styles-Patterns

Baley, K., Belcham, D., & Kovacs, J. (2014, 12 08). *Separation of concerns*. Retrieved from MSDN magazine: http://msdn.microsoft.com/en-us/magazine/ekstremalna-przerobka-asp-net--czesc6-podzial-obowiazkow.aspx

Basisprincipes PRINCE2. (2014, 9 20). Retrieved from van-bommel: http://www.van-bommel.org/Van_Bommel_professionalisering/Basis_PRINCE2.html

Beveilingsadviezen. (2014, 12 02). Retrieved from NCSC: https://www.ncsc.nl/dienstverlening/response-op-dreigingen-en-incidenten/beveiligingsadviezen

Caprio, G. (2014, 12 08). *Dependency Injection*. Retrieved from MSDN magazine: http://msdn.microsoft.com/en-us/magazine/cc163739.aspx

Crouch, S. (2014, 10 22). *Developing maintainable software*. Retrieved from Software sustainability Institute: http://software.ac.uk/resources/guides/developing-maintainable-software

Delegates and Events in C# / .NEt. (2014, 12 08). Retrieved from Akadia: http://www.akadia.com/services/dotnet_delegates_and_events.html

Design patterns. (2014, 12 08). Retrieved from Sourcemaking: http://sourcemaking.com/design_patterns

Design Patterns. (2014, 9 21). Retrieved from Source Making: http://sourcemaking.com/design_patterns

Designing web applications. (2014, 12 08). Retrieved from MSDN : http://msdn.microsoft.com/en-us/library/ee658099.aspx

Developer's Guide to Dependency Injection Using Unity. (2014, 9 21). Retrieved from MSDN Microsoft: http://msdn.microsoft.com/en-us/library/dn223671(v=pandp.30).aspx

Don't use MongoDb. (2014, 12 01). Retrieved from YCombinator: https://news.ycombinator.com/item?id=3202081

Duchek, M. (2014, 12 08). *Differences Between Bootstrap v2.3 and v3.0 (RC2)*. Retrieved from Mattduchek: http://mattduchek.com/differences-between-bootstrap-v2-3-and-v3-0/

Facebook docs. (2014, 11 06). Retrieved from Facebook: https://developers.facebook.com/docs/reference/

Fadeyev, D. (2014, 12 08). *Using the LESS CSS Preprocessor for smarter style sheets*. Retrieved from SmashingMagazine: http://www.smashingmagazine.com/2010/12/06/using-the-less-css-preprocessor-for-smarter-style-sheets/

Full stack frameworks. (2014, 12 01). Retrieved from Nodeframework: http://nodeframework.com/index.html#mvc

Generation languages. (2014, 12 08). Retrieved from Computer Hope: http://www.computerhope.com/jargon/num/1gl.htm

Google : Choosing a programming language. (2014, 12 08). Retrieved from google: https://sites.google.com/site/choosingaprogramminglanguage/home/organisational-policies

Google styleguide. (2014, 11 04). Retrieved from Google: https://code.google.com/p/google-styleguide/

Graham, P. (2014, 12 12). *Revenge of the Nerds*. Retrieved from paulgraham.com: http://sourcemaking.com/design_patterns

Gupta, S., Hartkopf, J., & Ramaswamy, S. (2014, 12 08). *Event Notifier, a Pattern for Event Notification*. Retrieved from Marco panizza: http://www.marco.panizza.name/dispenseTM/slides/exerc/eventNotifier/eventNotifier.html

Haddad, C. (2014, 12 09). *REST Easy: API Design, Evolution, and Connection*. Retrieved from WSO2: http://wso2.com/whitepapers/rest-easy-api-design-evolution-and-connection/?utm_source=infoq&utm_medium=whitepaper&utm_campaign=infoqresteasy

Hall, A. (2013, 12 08). *How to pick the right programming language*. Retrieved from Mashable: http://mashable.com/2012/07/11/developer-programming-languages/

Ibraheem, B. (2014, 12 01). *Making sense of nosql*. Retrieved from Hebrayeem: http://hebrayeem.blogspot.nl/2014/01/making-sense-of-nosql.html

Java - Encapsulation. (2014, 12 08). Retrieved from Tutorialspoint: http://www.tutorialspoint.com/java/java_encapsulation.htm

Katcherovski, V. (2014, 9 19). *project-management-methodologies*. Retrieved from Easy Projects: http://www.easyprojects.net/blog/2012/08/23/project-management-methodologies/

Lanen, R. v., & Solingen, R. v. (2014). *SCRUM voor managers.* Management boek.

Laribee, D. (2014, 12 08). *An Introduction To Domain-Driven Design*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/magazine/dd419654.aspx

Layered application. (2014, 12 04). Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/ff650258.aspx

lesCSS. (2014, 9 26). Retrieved from lesCSS: http://lesscss.org/

Mahfouzi, M. (2014, 12 08). *Events and delegates simplified*. Retrieved from CodeProject: http://www.codeproject.com/Articles/4773/Events-and-Delegates-Simplified

Martin, R. C. (2003). *Agile Software Development: Principles, Patterns, and Practices.* Upper Saddle River, NJ, USA: Prentice Hall. Retrieved from http://www.objectmentor.com/resources/articles/srp.pdf

MEAN stack tutorial. (2014, 11 28). Retrieved from Thinkster: https://thinkster.io/angulartutorial/mean-stack-tutorial/

Microsoft: Choosing a programming language. (2014, 12 08). Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/cc168615.aspx

Miller, J. (2014, 10 22). *Code better*. Retrieved from Creating a Maintainable Software Ecosystem: http://codebetter.com/jeremymiller/2006/08/14/creating-a-maintainable-software-ecosystem/

Miller, J. (2014, 12 08). *Cohesion And Coupling*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/magazine/cc947917.aspx

Netwinst Home. (2014, 08 19). Retrieved from Netwinst: http://www.netwinst.nl/

O'Neill, S., Clarke, E., & van Gend, C. (2007). *A guide to Project Management.* Kaapstad, Zuid Afrike: Mercury Crescent.

Palermo, J. (2014, 12 09). *Guidelines for Test-Driven Development*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/aa730844%28v=vs.80%29.aspx

Palermo, J. (2014, 12 10). *Guidelines for Test-Driven Development*. Retrieved from MSDN: http://msdn.microsoft.com/en-us/library/aa730844(v=vs.80).aspx

Perflectie website. (2014, 06 22). Opgehaald van Perflectie: http://www.perflectie.nl

Pierry, R. (2014, 12 14). *Discover the Design Patterns You're Already Using in the .NET Framework*. Retrieved from MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/cc188707.aspx

PRINCE2: Managen van Succesvolle Projecten met PRINCE2. (2010). Crown Copyright, TSO. Retrieved from http://www.van-bommel.org/Van_Bommel_professionalisering/Basis_PRINCE2.html

Projects, E. (2014, 9 19). *5-effective-project-management-methodologies*. Retrieved from Easy Projects: http://www.easyprojects.net/blog/2014/06/06/5-effective-project-management-methodologies/

Radatz, J. (1990). *IEEE Standard Glossary of Software Engineering Terminology.* Universidad Nacional de Colombia.

richtsnoeren beveiliging persoonsgegevens. (2014, 12 02). Retrieved from Cbp web: http://www.cbpweb.nl/Pages/pb_20130219_richtsnoeren-beveiliging-persoonsgegevens.aspx

Ries, E. (2011). *The LEAN startup.* Verenigde staten: Crown Business.

Rijksoverheid. (2014, 12 01). *Cookiewet: regels en richtlijnen*. Retrieved from Rijksoverheid: http://www.rijksoverheid.nl/onderwerpen/internet/bescherming-privacy-op-internet/cookiewet-regels-en-richtlijnen

Rijksoverheid. (2014, 12 01). *Handleiding wet bescherming persoonsgevens*. Retrieved from Rijksoverheid: http://www.rijksoverheid.nl/documenten-en-publicaties/brochures/2006/07/13/handleiding-wet-bescherming-persoonsgegevens.html

Royce, W. (1970). *Managing the development of large software system.* TRW. Retrieved from Wa: http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf

Solingen, R. v., & Rustenburg, E. (2010). *De kracht van SCRUM.* Prentice Hall.

Storz, E. (2014, 12 08). *An introduction to full-stack Javascript*. Retrieved from SmashingMagazine: http://www.smashingmagazine.com/2013/11/21/introduction-to-full-stack-javascript/

TIOBE Index october 2014. (2014, 11 06). Retrieved from TIOBE Programming Community: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

Vacatures. (2014, 11 06). Retrieved from Monsterboard: http://jobsearch.monsterboard.nl/

Vandegriend, B. (2014, 10 22). *The Importance of Maintainable Software*. Retrieved from Basil Vandegriend: Professional Software Development: http://www.basilv.com/psd/blog/2006/the-importance-of-maintainable-software

Wesley, A. (2004). *The Rational Unified Process An Introduction (3de editie).* Verenigde staten: Pearson Education.

Wetten Wbp. (2014, 12 01). Retrieved from CPB: http://www.cbpweb.nl/Pages/ind_wetten_wbp.aspx

What is PHP? (2014, 12 08). Retrieved from PHP: http://php.net/manual/en/intro-whatis.php

W-mdr90. (2014, 9 20). *De watervalmethode*. Retrieved from InfoNu: http://pc-en-internet.infonu.nl/diversen/97898-de-watervalmethode.html

Bijlage

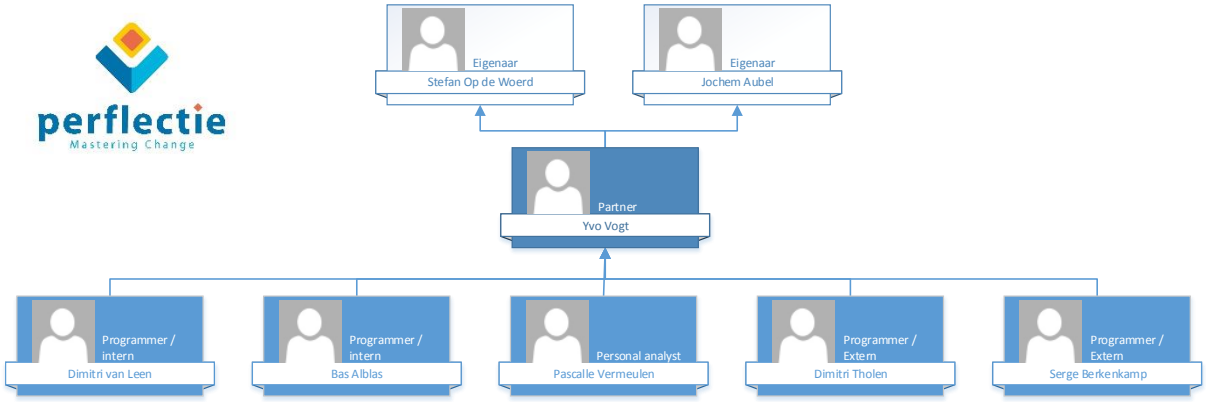
A : Perflectie, Netwinst en positie van de student

De opdrachtgever

Jochem Aubel zal mij als opdrachtgever begeleiden tijdens mijn afstudeerstage.

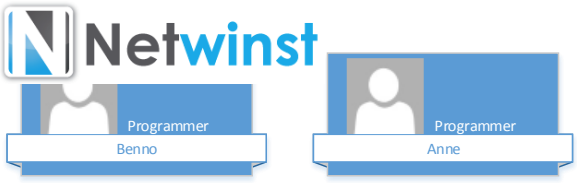
Organisatie binnen het project

Perflectie (Perflectie website, 2014) is de organisatie die initiatief onderneemt. Perflectie is een klein bedrijf dat een online reflectie platform heeft ontwikkeld in samenwerking met Netwinst (Netwinst Home, 2014). Perflectie, die zich als bedrijf bezighoudt met de persoonlijk ontwikkeling van veel klanten, heeft behoefte aan een betrouwbaar product.



Figuur 1 Perflectie Organisatie chart

Voor de opdracht om een betrouwbaar online-platform te creëren heeft Perflectie een extern bedrijf ingehuurd. Het bedrijf, genaamd Netwinst (Netwinst Home, 2014), is ervaren in het maken van online applicaties. Tijdens de ontwikkeling van de partner-module en het verbeteren van het huidige platform zal Netwinst betrokken zijn in het assisteren van programmeren.



Figuur 2 Netwinst Organisatie chart

Positie van de student

Ik zal een onderbouwd voorstel indienen aan de opdrachtgever. Binnen dit voorstel staat hoe het huidige online-platform verbeterd en uitgebreid kan worden.

Hiervoor zal eerst een literatuuronderzoek uitgevoerd worden met als hoofdvraag: In hoeverre en op welke manier kan het Perflectie platform worden verbeterd qua dynamiek voor klanten, zowel technisch als functioneel, binnen een projectteam dat op de juiste methode en manier geleid wordt?

Aan de hand hiervan zal er een functioneel en een technisch ontwerp, een plan van aanpak en een voorstel worden gemaakt en gepresenteerd worden om het huidig platform met zijn structuur eventueel te verbeteren. Dit voorstel wordt na volledige goedkeuring van de opdrachtgever uitgevoerd door mij en 3 technische collega's in samenwerking met een extern bedrijf genaamd Netwinst. (Netwinst Home, 2014)

Het team

Omdat Perflectie zich als bedrijf niet bezighoudt met de ontwikkeling van applicaties zal er extern personeel ingehuurd worden. Netwinst wordt als bedrijf ingehuurd waarbij Netwinst Anne Boon en Benno van Dijk inzet om hulp te bieden met programmeren. Netwinst is ook de organisatie die het huidige online-platform heeft ontwikkeld.

Dimitri Tholen wordt als ZZP-er ingehuurd en brengt uitgebreide kennis en programmeer ervaring met zich mee. Dimitri Tholen zal de contact persoon zijn voor vragen over de technieken en methoden die wij in het platform zullen doorvoeren.

Jesse Buitenhuis is ingehuurd voor de front-end ontwikkeling. Hij zal zich, met hulp van Dimitri Tholen, verdiepen in de technieken die gebruikt worden in de front-end. Vragen over de front-end en/of opdrachten die in de front-end doorgevoerd moet worden met Jesse Buitenhuis besproken.

Op 2 Oktober is Serge Bergenkamp als stagiair aangenomen. Hij zal hulp bieden bij het goed laten verlopen van de communicatie tussen de backend en front-end.

Dimitri Tholen, Jesse Buitenhuis, Serge Bergenkamp, Anne Boon, Benno van Dijk, Jochem Aubel en ikzelf vormen samen het scrumteam.

B : Management methode: Agile Scrum

Voordat ik aan de slag ben gegaan met het onderzoeken van methodes om een projectteam te leiden heb ik mij zelf eerst verder verdiept in Perflectie en in de partner module. Dit heb ik gedaan om zo eerst in kaart te brengen wat de methodes zijn van Perflectie, hun principes en het uiteindelijk doel van het eindproduct.

The LEAN startup

Voor het uitwerken van dit hoofdstuk zijn er wekelijks gesprekken geweest met Jochem Aubel (Eigenaar van Perflectie) , waarbij Perflectie als bedrijf en de partner module uitgebreid aan bod kwamen. Tijdens het gesprek is mij verteld dat zij de LEAN startup methode toepassen op hun onderneming maar ook op de ontwikkeling van hun producten. The LEAN Startup methode is ontworpen om (beginnende) bedrijven te behoeden voor de problemen die zich voordoen in de hedendaagse bedrijfswereld.

Het eerste probleem is dat in deze tijd beginnende bedrijven weinig tot geen zekerheid hebben. Ze weten niet wie hun klant is en wat hun product precies moet worden. Ook omdat de wereld zich steeds sneller ontwikkeld wordt het steeds moeilijker om zekerheid te krijgen.

Het tweede probleem is dat er vaak niet projectmatig wordt gewerkt, de zogeheten “Just do it” mentaliteit. Na veel uren in een project gestopt te hebben, is de kans uiteindelijk groot dat het geen succes wordt.

Als oplossing is de LEAN startup methodiek ontwikkeld. Deze methode bestaat uit drie onderdelen: “Visie”, “Sturen” en “Versnellen”. (Ries, 2011) Met de LEAN startup methode leer je je onderneming en je product naar een succes brengen.

Visie

Om iets te bouwen, of het nu een project of een onderneming is, heb je management nodig. Management wordt vaak gezien als een niet creatief en inflexibel binnen een beginnend bedrijf. Daarom houden veel personen een “Just do it” mentaliteit aan, waarbij elke vorm van management, proces en discipline wordt ontweken. Helaas kan deze aanpak leiden tot chaos en mislukking.

Hieruit komt de overtuiging dat elk proces een modern managementprincipe moet volgen. Dit heeft als effect dat de productiviteit verhoogt wordt, met zo min mogelijk middelen. Hiermee wordt maximaal gebruik gemaakt van de tijd, passie en vaardigheden van medewerkers.

Maar hoe kan een project gemanaged worden wanneer het te realiseren doel niet 100% duidelijk is?
Er moet worden gestreefd naar voorspelbaar en testbaar (tussen)resultaat. Je kunt bijvoorbeeld Milestones binnen het project inplannen waarbij duidelijk is beschreven waaraan het product op dat moment moet voldoen.

Het principe van Visie is mooi beschreven in (Ries, 2011). Hierin geeft hij als voorbeeld dat elke onderneming een eigen motor ontwerpt om te groeien. Elke toevoeging of vernieuwing, bijvoorbeeld een nieuw product, een nieuwe feature etc. is om de motor verder te ontwikkelen. Echter blijkt niet elke toevoeging of vernieuwing een verbetering te zijn van de motor. De meeste tijd van een onderneming zal daarom ook gaan zitten in het continue corrigeren en verder verbeteren van de motor.

Hierbij komt een tweede, maar ook belangrijk feedback punt in het proces en dat is het sturen van het voertuig waarin de motor zich bevindt. Sturen spreekt meestal zo voor zich dat er niet meer bij wordt nagedacht. Hierbij wordt een raket als voorbeeld gebruikt. Mocht er één fout zitten in de besturing dan kan dit een catastrofaal gevolg hebben. Daarom wordt het proces van start tot het einde en elke handeling in het sturen nauw en op een iteratieve manier in kaart gebracht, afhankelijk in welke situatie de raket zich op dat moment bevindt.

Het bijsturen wordt helaas ook vaak in startende onderneming en projecten vergeten. Er wordt dan op een zodanig complex niveau gepland en stap voor stap tot het uiterste detail uitgedacht. Dit kan leiden tot het voorspellen van miljoenen risico's en hiervoor moet dan architectuur en middelen voor worden ingezet wat later niet nodig blijkt te zijn om de risico's af te dekken. Hierdoor wordt er teveel geld en tijd in onderhoud en/of in het opzetten van het product geïnvesteerd. Dit kan een bedrijf ten gronde richten.

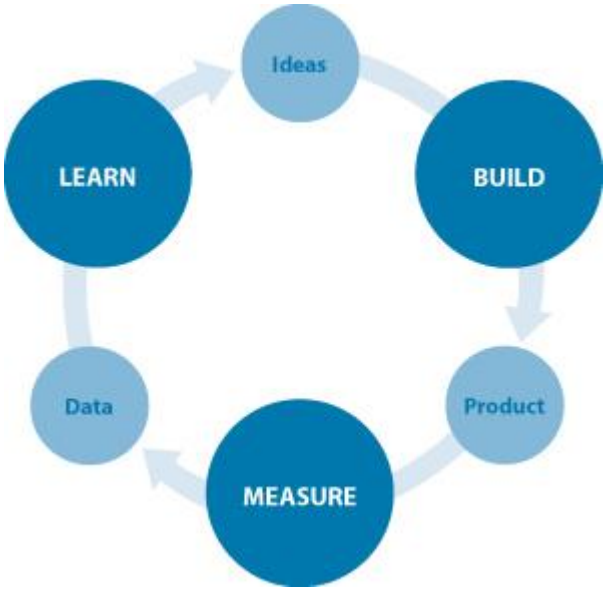
De LEAN startup heeft de methode zo ontworpen dat je leert om te rijden met je onderneming. Dus in plaats van complexe situaties uit te denken en daarvan plannen te maken dat vooral gebaseerd zijn op aannames, maak je constante aanpassingen binnen het sturen naar jouw einddoel. Dit heet "Build-Measure-Learn". (Ries, 2011)

Tijdens deze methode van sturen kun je leren wanneer het tijd is om aan je stuur te trekken ("Pivot") of op je pad te blijven.("Persevere") Ook is het belangrijk dat je duidelijk weet waar je heen gaat. Dit met als doel dat als je tijdens je werk een omweg of een verkeerde weg inslaat je gefocust blijft op je eindbestemming en daar naar toe blijft sturen.

Een onderneming heeft ook een ware eindbestemming in gedachte. Dit is meestal het creëren van een succesvolle en wereld-veranderende onderneming.
Dit wordt ook wel de onderneming Visie genoemd. (Ries, 2011)

Sturen

Basis van elke onderneming is dat het ideeën vormt in producten. Klanten gebruiken het product, waarbij ze feedback en data genereren. Deze feedback is kwalitatief gezien veel belangrijker voor de onderneming dan bijvoorbeeld geld, prijzen of marketing. Om deze feedback te verrijken gebruikt de LEAN startup een Build-Measure-Learn loop. (Figuur 1)
Het meest riskante van een plan bij een onderneming is aannames doen. Bij het produceren in de LEAN methode leer je wanneer je een hoeveelheid energy, materiaal, tijd en geld in een project moet stoppen. Dit creëert een product wat op dat moment aan alle eisen en wensen voldoet, genaamd een minimum viable product (MVP).



Figuur 1 Build-Measure-Learn, verkregen uit (Ries, 2011)

Een MVP is een versie van het product waarbij je met behulp van het Build-Measure-Learn model een minimaal hoeveelheid materiaal en tijd kan inschatten. Deze versie mist een hoeveelheid features dat later belangrijk kunnen zijn. Maar een MVP kan ook voor extra werk zorgen, daarom moet er eerst gemeten worden wat de impact is van een MVP. (Ries, 2011)

Om een dergelijk MVP te meten wordt het ook wel een “Measure phase” genoemd. Binnen deze fase is de grootste uitdagingen om vast te stellen of het product development wel leidt tot vooruitgang. Als uitgangspunt van deze fase is de belangrijkste keuze om verder het pad te bewandelen(“Persevere”) of een ommekeer te maken. (“Pivot”) Dit heeft als voordeel dat als je op tijd een Pivot uit kan voeren dit tijd en geld zal schelen. (Ries, 2011)

Versnellen

“De meeste projecten die een onderneming uitvoert zijn niet inzichtelijk. Hoe vaak zou een product uitgebracht worden? Is er een specifieke reden om wekelijks, dan wel dagelijks of één keer per kwartaal uit te brengen? Door continu een nieuwe versie uit te brengen kan men ervoor zorgen dat risico’s direct zichtbaar worden. Als je te lang wacht kan het zijn dat product niet meer gewild is.
Hoeveel tijd en energy moeten bedrijven dan investeren in infrastructuren en in het plannen van het anticiperen op succes? Te snel opleveren en je loopt het risico te veel tijd te spenderen die je ook ergens anders voor kon gebruiken. Te langzaam opleveren en je bent je voordeel naar de markt kwijt (Ries, 2011)
De hierboven genoemde zinnen staan genoteerd in het boek van (Ries, 2011). In dit boek staat duidelijk welk risico een jong bedrijf loopt als ze niet of juist alleen maar versnellen. Om hierin een goede balans te vinden zijn er enkele regels en principes waar je je aan kunt houden en zijn er vragen die je aan jezelf kunt stellen.

De belangrijkste vraag: “welke activiteiten creëren waarde en welke zijn alleen waardeloze tijd besteding?”. (Ries, 2011)

Daarna volgen vragen die beantwoord moeten worden door elke startende onderneming om succesvol te kunnen zijn: (Ries, 2011)

1. Wat voor producten willen de klanten echt?
2. Hoe zal het bedrijf groeien?
3. Wie is onze klant?
4. Naar welke klanten zullen wij luisteren en naar welke niet?

Wanneer met deze vragen rekening wordt gehouden kan een onderneming of projectteam op een efficiënte en effectieve wijze een iteratie inplannen. De duur van deze iteratie zou dan zo kort mogelijk moeten zijn waarbij het maximale resultaat behaald zal worden. Binnen deze periode zal dus versneld worden.

Samenvatting

The LEAN startup geeft een richtlijn om een product te ontwikkelen en hierin Milestones te definiëren. Deze Milestones moeten zo kort, efficiënt en effectief mogelijk gerealiseerd worden. Om te testen of een Milestone gehaald is moet bij het einde van een Milestone een testbaar product zijn gerealiseerd. Dit is van te voren vastgesteld binnen de betreffende Milestone. Hierbij moet rekening gehouden worden met het feit dat wanneer er ontwikkeld wordt om een Milestone te bereiken het proces wordt opgedeeld in iteraties. Dan kan aan elk begin van een nieuw iteratie het Build-Measure-Learn (Zie figuur 1) principe worden doorlopen. Hier kunnen voortijdig risico’s in kaart gebracht worden en indien nodig een Pivot of een Persevere worden uitgevoerd.

Projectmanagement methodes

Na mijn analyse van het LEAN startup principe heb ik als vervolg een onderzoek gedaan naar wat voor project management methode hier het best bij past. Hiervoor heb ik meetpunten gebruikt waar de methode zoveel mogelijk aan zal moeten voldoen.

1. Volgt de methode Iteraties?
2. Kunnen deze iteraties binnen een Milestone lopen? Of kan dat gedefinieerd worden?
3. Kan in deze methode ook de product eigenaar betrokken worden?
4. Kunnen in deze methode ook de klanten betrokken worden?
5. Heeft deze methode korte iteratieve reminders/meetings om het project in de juiste baan te sturen?
6. Heeft deze methode een efficiënt en effectief overzicht om een zo hoog mogelijke acceleratie te maken?
7. Kan de methode toegepast worden op een team van 8 personen?

Het projectmanagement methodes die in consideratie heb zijn: (Projects, 2014) (Katcherovski, 2014)

- Agile SCRUM
- Watervalmethode
- PRINCE2
- RUP

De hier bovenstaande methodes zijn de meest gebruikte binnen IT softwareontwikkeling (Katcherovski, 2014) (Projects, 2014)

Agile SCRUM

Scrum bewijst zich al 20 jaar in organisaties als een bewezen methode om projecten te managen. In 2006 begon Scrum in Nederland pas echt te groeien. Nu is het één van de meest gebruikte aanpakken voor softwareontwikkeling. (Solingen & Rustenburg, 2010) De negen voordelen van Scrum: (Lanen & Solingen, 2014)

1. Meer grip op het eindresultaat
2. Ruimte om met voortschrijdend inzicht om te gaan
3. Blijere klanten en gebruikers
4. Meer waarde tegen lagere kosten
5. Kortere doorlooptijden
6. Hogere productiviteit
7. Betere kwaliteit
8. Stoppen is een optie
9. Blijere medewerkers

Beschreven in (Lanen & Solingen, 2014), is er een situatie waarvoor Scrum wel en waarvoor het niet geschikt is.

Wanneer heb je iets aan Scrum?
Je hebt pas iets aan Scrum wanneer je in een omgeving bevindt waar je resultaat moet leveren. Waar klanten geholpen worden of waar producten gemaakt worden. Als je in een omgeving zit waar meer geld wordt uitgegeven dan gebudgetteerd, of waar je onvoldoende kwaliteit wordt geleverd of onvoldoende aan verwachtingen wordt voldaan, daar gaat Scrum je helpen. (Lanen & Solingen, 2014)

Wanneer moet je Scrum niet gebruiken?
Scrum zal je niet kunnen helpen in eenvoudige situaties. Wanneer resultaten sterk voorspelbaar zijn en er weinig dynamiek is, dan voegt Scrum weinig tot niks toe. (Lanen & Solingen, 2014)

Het Scrumproces

Scrum is een simpel methodiek. Wanneer een Scrumproject is opgestart bestaat het uit één resultaat, twee lijsten, drie rollen en vier meetings. Hierbij richt je een stabiel Scrum team op die in een vast ritme (meestal twee weken) een werkend resultaat opleveren. Om het resultaat te behalen zal eerst de meest waardevolle onderdelen afgemaakt worden, dat zo snel mogelijk laten zien aan wie daar belang van heeft. Hierdoor kan het scrumteam leren wat er gemaakt moet worden en hoe het dat op de beste manier kan doen. De prioriteit ligt

op waarde en het met een team kort-cyclisch toewerken naar direct bruikbare resultaten. (Solingen & Rustenburg, 2010)

Het scrumproces bestaat uit 12 stappen. (Lanen & Solingen, 2014)

Het product
De eerste stap is om een product neer te zetten, dat werkt en getest is. Dit wordt ook wel een bruikbaar product genoemd. Dit is een tussenresultaat wat nog niet bedoeld is voor je klant of eindgebruiker, maar om feedback te krijgen. Elke iteratie is het de bedoeling dat je een product levert wat af is. In de Definition-of-done staat wanneer een product officieel klaar is.

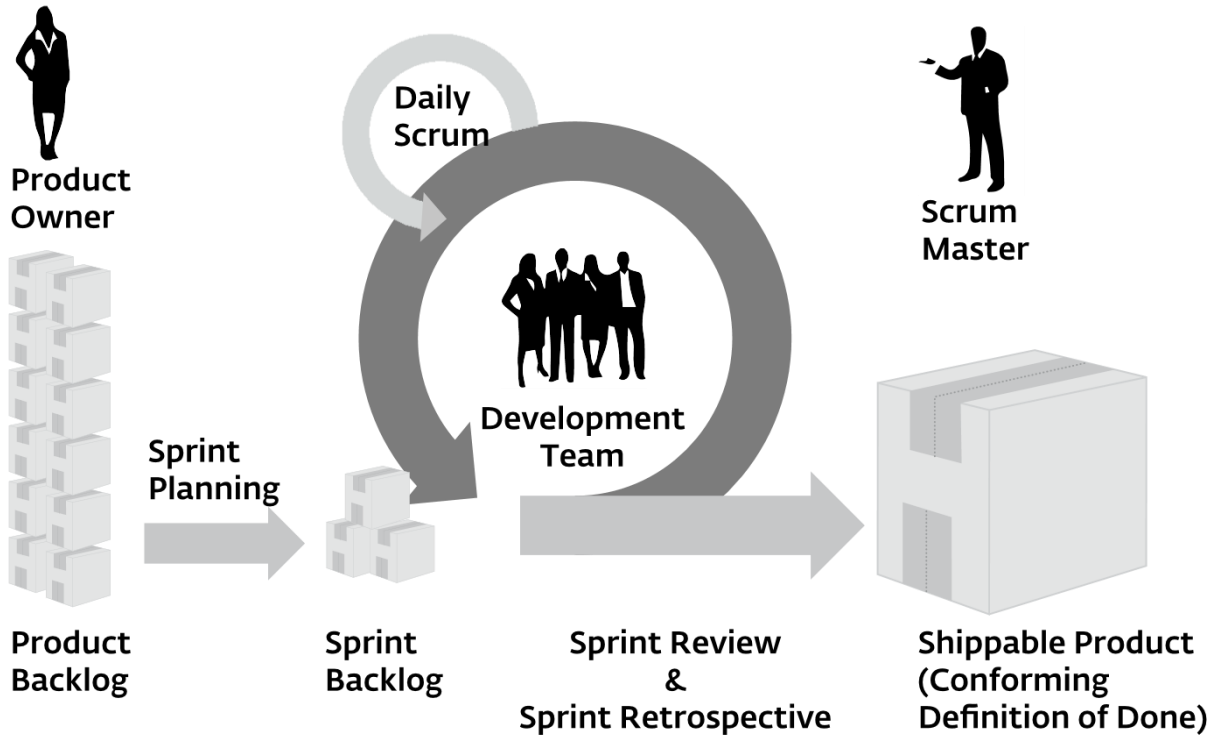
Eerste lijst: de Product Backlog
De wensen van het product eigenaar om het resultaat te behalen staan beschreven in de product backlog. In een product backlog is de meeste belangrijkste item als eerst geordend. Dus datgene wat de grootste waarde heeft of dat het grootste risico wegneemt, pak je als eerste op binnen je scrumteam

Eerste rol: de Product eigenaar/owner
De product eigenaar is verantwoordelijk voor invulling van de product backlog en de vertalingnaar de buitenwereld en het team

Tweede rol: het Development team
Het development team is zo ingericht dat het zelf organiserend is binnen een project. Een dergelijk team brengt betrouwbaar informatie, zoals hoeveel iets gaat kosten, de inschattingen van hoeveel uur eraan gewerkt moet worden etc. In dit multifunctionele team zitten mensen die samen alle competenties bezitten, om een idee op de product backlog binnen een sprint om te zetten in een concreet en werkend resultaat. Het development team bestaat maximaal uit negen leden en minimaal uit drie. Vanaf tien leden is het effectiever het team te splitsen in twee teams.

De derde rol: De Scrum Master
De scrum master zijn taak lijkt op die van een projectleider maar is dit niet. De scrum master is een coach die ervoor zorgt dat de Scrum-wijze correct wordt toegepast en goed verloopt. Daarvoor heeft de coach het recht om een interventie te plegen binnen het project. Deze interventies kunnen zowel leidend als adviserend zijn.

Het Scrum-team
De eerder drie genoemde rollen: de scrum master, de product owner en het development team samen wordt het scrum-team genoemd. Binnen een scrum-team zijn enkele regels vastgesteld welke rollen met elkaar samengevoegd kunnen worden. Een product owner kan een lid van het development team zijn. Een scrum master kan ook lid van het development team zijn. Maar een scrum master kan nooit de product owner zijn.



Figuur 2 Het Scrum proces, verkregen uit (Lanen & Solingen, 2014)

De sprint planning meeting
Omdat het development team zelf organiserend is, moet het team ook kunnen inschatten hoeveel werk gedaan kan worden binnen een sprint. Tijdens de eerste sprint planning meeting maakt het team een gedetailleerd plan voor die sprint. Onderdeel van een plan is een lijst van taken afgeleid van de product backlog.

De tweede lijst: de sprint backlog
De lijst van alle taken binnen een sprint noemen we de sprint backlog. Een sprint backlog wordt meestal op een plek getoond waar elk teamlid gemakkelijk toegang toe heeft, als voorbeeld: een lijst op een bord aan de muur. Een taak kan een bepaalde status hebben met ToDo, Doing en Done. Deze status is dynamisch en kan zelf gedefinieerd worden. Om een status aan een taak te geven krijg je een helder overzicht van wat nog gedaan moet worden, wat is ontwikkeling is en wat klaar is.

De daily Scrum meeting
Om ervoor te zorgen dat iedereen zich aan zijn verantwoordelijkheid houdet binnen een sprint wordt er dagelijks een daily scrum meeting gehouden. Een deze meeting staan drie vragen centraal: (Solingen & Rustenburg, 2010)

1. Wat heb ik gisteren bereikt voor het team?
2. Wat ga ik vandaag bereiken met het team?
3. Waar heb ik hulp bij nodig van het team?

Als blijkt dat het resultaat van de sprint niet meer haalbaar is dan moet het plan aangepast worden.

De Sprint review meeting
Na elke sprint is het de bedoeling dat het development team een nieuwe werkende versie van het product heeft gemaakt. Dit laten zij zien aan diegene dat er belang bij heeft en ze vergaren zoveel mogelijk feedback. Dit wordt gedaan in de sprint review meeting. Ook wordt de product backlog besproken. Dit om er ervoor te zorgen dat iedereen de product backlog kent en zo een samenwerking en overzicht bestaat met het gehele scrum team.

De Sprint retrospective
De bedoeling is om van elke sprint te leren. Vooral leren hoe het scrum team het steeds beter kan doen. Daarom gaat het Scrum team na elke sprint review bij elkaar zitten om te kijken hoe de volgende sprint beter

kan verlopen. Deze meeting wordt ook wel de sprint retrospective genoemd. Het resultaat moet leiden tot verbetering van de volgende sprint.

Product Backlog Refinement
Omdat de Product Backlog ook binnen het proces verbeterd kan worden bestaat hier geen officieel meeting voor. Om dit op te lossen kan je hiervoor zelf een meeting inplannen om de week. Tijdens deze meeting ga je samen met het scrum team het product backlog in taken verdelen, met daarbij een schatting van kosten om het te voltooien.

Agile binnen Scrum
Agile en Scrum hebben allebei een aparte betekenis. Agile staat voor flexibiliteit en wendbaarheid binnen een project. Dit kan gedaan worden door het bijsturen van een project.
Scrum is een leerprogramma waarmee je na elke iteratie verbeterd. Een plan dat bruikbaar is door de meeste mensen. Scrum houdt een werkwijze aan waar je op een Agile manier te werk gaat. Op een iteratieve manier bekijk je het resultaat na een sprint en zo kan je zo lerende wijs verbetering doorvoeren.

Watervalmethode
De watervalmethode is een methode dat vaak gebruikt wordt bij softwareontwikkeling. Vanuit de watervalmethode zijn er verschillende vernieuwende modellen gemaakt, waarbij de nadelen van het traditionele watervalmodel grotendeels verdwijnt. (W-mdr90, 2014)

Deze nieuwe modellen bestaan uit:

- Model van Royce
- Het “Sashimi”-model
- Aorta lifecycle model
- V-model

Het watervalmethode proces
De watervalmethode moet je zien als een loop van ontwikkeling dat in fases verloopt, zoals een waterval. De fases die daarbij worden doorlopen bestaan uit:

1. De definitiestudie/analyse
2. Basisontwerp
3. Technisch ontwerp/detailontwerp
4. Bouw
5. Testen
6. Integratie
7. Beheer en onderhoud

Elk van deze fases wordt één voor één doorlopen waarbij een fase teruggaan niet mogelijk is. Elk fase dient afgesloten te worden, voordat je met een nieuwe fase mag beginnen. Als er een fout binnen een fase is ontstaan dan dient men de fout eerst op te lossen. (W-mdr90, 2014)

Het grote nadeel van deze methode is dat het niet mogelijk is om tijdens het project van richting te veranderen omdat elk fase geheel afgesloten moet worden. Het voordeel hiervan is wel dat elk fase geheel goed afgerond is voordat de volgende fase gestart wordt.

Model van Royce
Binnen het model van Royce staat beschreven dat het verkeerd is dat het niet mogelijk is om een fase terug te gaan. Dit omdat het testen na de development komt. Mocht het product niet aan de testen voldoen, dan moet het proces van de waterval opnieuw worden doorlopen. Royce heeft hiervoor de waterval methode aangepast om zo het probleem aan te pakken en dat je als projectteam niet vast zit aan een fase. (Royce, 1970)

"Sashimi" model
Het Sashimi model is opgezet door Peter Degrace. Binnen dit model is het mogelijk dat fases elkaar overlappen. Zo kan er tijdens het analyseren ook een basisontwerp worden opgezet en/of een analyse plaatsvinden tijdens een basisontwerp. Hierdoor zal er zo min mogelijk tijd, geld en personeel worden verspild tijdens faseringen. (O'Neill, Clarke, & van Gend, 2007)

Aorta lifecycle model
Binnen elke nieuw model van een waterval methode is te zien dat het proces steeds meer verkort wordt. Ook binnen het Aorta Model is dit te zien. Binnen de "normale" watervalmethode ziet de klant pas na alle fases, de oplevering, het product. Als er gebruik wordt gemaakt van het Aorta lifecycle model, dan wordt na elke fase feedback gevraagd van de klant.

V-model
Binnen een V-model wordt de nadruk gelegd op het verifiëren en ontwikkelen van het softwareontwerp. Dit proces is opgedeeld in een aantal fasen die elk een product opleveren. Wanneer alle producten van een fase zijn opgeleverd, dan start de volgende fase met het opgeleverde product als basis. Dit proces herhaalt zich tot elk wens en eis binnen het systeem is getest en ontwikkeld is en zo het vertrouwen in het systeem groeit.

PRINCE2
Projects in controlled environments (PRINCE) is een structuur met generieke methodes om effectief projectmanagement te realiseren van projecten met een duidelijke doelstelling. PRINCE is opgestart als een standaard binnen de Britse overheid om projecten te managen. Zeven jaar later werd PRINCE2 geïntroduceerd als een meer generieke standaard om elk type project te ondersteunen.

Principes
Omdat PRINCE2 staat voor een generiek mogelijk methode zijn er zeven principes beschreven in (PRINCE2: Managen van Succesvolle Projecten met PRINCE2, 2010) van welk omvang ook op elk project te gebruiken.

Als eerste principe komt de Voortdurende zakelijke rechtvaardiging. Een PRINCE2 project vindt zijn rechtvaardiging in gedefinieerde business case. Hierin wordt duidelijk genoteerd: (Basisprincipes PRINCE2, 2014)

> "wat de toegevoegde waarde is voor de omgeving, organisatie of situatie waarin het projectresultaat wordt gebruikt;
> waarom is gekozen voor deze oplossing;
> welke voordelen deze biedt;
> wat de baten en kosten van het project zijn, zowel kwalitatief als kwantitatief;
> en welke aannames hierbij zijn gehanteerd.
> Een Business Case bevat liefst een goede investeringsanalyse en een risicoanalyse.
> De Business Case wordt gedurende het project geëvalueerd en bijgesteld. Beslissingen worden altijd afgemeten aan het effect op de Business Case.
> Na afloop van het project vindt een evaluatie plaats van deze uitgangspunten."

Als tweede principe komt leren van ervaringen. Het team gaat actief op zoek naar leerervaring wat hij of zij is tegengekomen tijdens de projectaanpak. Deze ervaringen worden tijdens het project verzameld en bij de afsluiting gedocumenteerd voor toekomstig gebruik.

Derde principe: heldere rollen en verantwoordelijkheden. Binnen een project met de PRINCE2 methode staat de samenwerking tussen de drie belangrijke partijen: de organisatie, de leverancier en de gebruikers. De leverancier staat voor de partij die de kennis en mankracht levert voor het project. Dit kan zowel intern als extern zijn. In PRINCE 2 wordt een project board gebruikt waarop alle belangrijke besluiten worden genomen. Dit wordt ook de BUS-principe genoemd. (PRINCE2: Managen van Succesvolle Projecten met PRINCE2, 2010)

Vierde principe: besluitvorming per fase. Binnen een fase kan de opdrachtgever van het project tijdig beslissingen nemen met een stuurgroep. (NoGo) Daarbij geldt dat een team pas aan een project mag werken wanneer het project is goedgekeurd door het project board.(Go!) Na goedkeuring mag het team aan de slag gaan.

Vijfde principe: management by exception. Wanneer een fase wordt goedgekeurd tijdens een project moet er een besluit worden gemaakt. Dit om te bepalen of een fase is afgerond of waar er nog fouten plaatsvinden. Wanneer een fase volledig is afgerond staat dit in de randvoorwaarden wat voor het project is vastgelegd. Wanneer er dreigt dat één van deze voorwaarden gebroken wordt is er sprake van een 'exception'. Dan moet het team het project bijsturen in de goede richting.
Zesde principe: productgerichte aanpak. Om het project zo gestructureerd op een PRINCE 2 manier te hanteren is er een productgerichte werkwijze. Het product based planning techniek definieert alle op te leveren eindproducten en tussentijdse producten. Ook worden hierbij de kosten in rekening gebracht door het team om het product te realiseren. De kosten kan variëren van uren, materiaal, kwaliteit etc. Met deze productgerichte aanpak kan een projectmanager duidelijk meten wanneer het project bijgestuurd moet worden.

Zevende principe: projectbesturing op maat. PRINCE2 heeft, zoals eerder geschreven, generieke methodes om elk type project te leiden. In de loop van de tijd zijn er verschillende documenttemplates ontwikkeld om elk type project gerichter te managen. Hiervoor is in iedere organisatie een vraag naar een manier om gerichte complexe projecten aan te pakken en de risico's zo goed mogelijk te vermijden. Daarom is PRINCE2 een volledig schaalbare methode om aan elk situatie te kunnen voldoen. SCRUM is ook afgeleid van de PRINCE2 methode. (Solingen & Rustenburg, 2010)

RUP

Rational Unified Process (RUP) is een iteratief softwareontwikkelingsproces dat ontwikkeld is door Rational Software en later door ontwikkeld door IBM. RUP is een management methode om gestructureerd taken en verantwoordelijkheden in een projectteam te verdelen. Het doel hiervan is om de productie proces, binnen een deadline en budget, hoog kwaliteit product te leveren dat aan de eisen en wensen voldoet. (Wesley, 2004)

Principes

Het proces van RUP is een verzameling van best practices binnen software development. In totaal zijn er zes principes om het proces van RUP door te lopen. (Wesley, 2004)
Het eerste principe van RUP is dat je software op een iteratieve wijze ontwikkelt. Dit om ervoor te zorgen dat je in de snelle omloop van ontwikkeling, het resultaat van fouten bij het ontwikkelen van het product etc. kan inspelen om zo je product op tijd aan te passen om een zo maximaal kwalitatief product te leveren.

Als tweede principe staat het beheer van eisen en wensen. Beheer van eisen en wensen is een proces dat ervoor zorgt dat er beter controle kan plaatsvinden van het project, verbeterde software kwaliteit van de eisen en wensen van de klant, verminderde kosten en vertragingen van het project en verbetering van de communicatie binnen het team.

Gebruik van architectuur dat op componenten is gebaseerd, staat als derde principe. Het ontwerp van het product van het project staat bij RUP centraal bij architectuur. De architectuur is de vorm waar het systeem uiteindelijk uit moet bestaan. Dit geeft structuurl binnen het systeem, regels en eisen. Daarvan is een component binnen het product een los staande subsysteem binnen het software systeem. Dit component kan uit zichzelf handelingen verrichten of binnen een geheel systeem deze handeling uitvoeren.

Dan volgt het vierde principe, visueel gebruik maken van modellen om de software in beeld te brengen. Visueel gebruik maken van modellen helpt je om gemakkelijk een indruk te krijgen hoe het systeem ontwikkeld moet worden voordat hij überhaupt nog bestaat. Zo kan je visueel prototypes maken waar ook Use Cases, Use Case Diagrammen, klassen diagrammen etc. worden gemaakt.
Als vijfde principe moet je de software kwaliteit continu verifiëren. Het verifiëren van de software kwaliteit kan alleen gedaan worden met duidelijke testen binnen het software systeem. De belanghebbende en de betrokkenen kunnen dan zo het systeem testen op zijn kwaliteit. Vooraf wordt duidelijk besproken met de belanghebbenden waar het systeem aan moet voldoen.
Als zesde en laatste principe is het beheer van veranderingen binnen de software. Net zoals alle vorige project management methodes is verandering van het ontwikkelen van de software een onvermijdelijke stap. De architectuurkeuze welke vooraf vastgelegd is moet worden gehandhaafd ondanks dat deze veranderingen moeten worden doorgevoerd. Als dit niet kan dan moet het gehele systeem aangepast worden.

Fases van een RUP proces

RUP kent vier hoofdfases binnen een project:

1. Inceptie fase

Inceptiefase beschrijft de scope van het project. Binnen een scope staat vastgelegd de inhoud en de begrenzing hiervan. Ook de kosten en de verwachte baten worden ingeschat. Het uiteindelijke resultaat is een business case voor het project, op basis van de business case wordt er een “Go” of “NoGo” gegeven.

2. Elaboratie fase

Binnen de elaboratie fase wordt de functionele en non-functionele requirements in kaart gebracht. Deze requirements worden ook wel Use Cases genoemd. Ook het systeem architectuur wordt ontworpen. Het uiteindelijke resultaat van deze fase is het leveren van een projectplan met daarin vastgesteld of het project haalbaar is. Onderdelen van een dergelijk plan bestaat uit de inhoud, tijdschema en de kosten hiervan.

3. Constructie fase

Binnen de constructie fase wordt het product ontwikkeld van de architectuur dat vastgesteld staat binnen de Elaboratie fase tot een compleet testbaar geheel.

4. Transitie fase

Binnen de transitie fase wordt de overdracht geregeld. Hierin zijn de activiteiten om het product de valideren, voorbereiden, nazorg en overdracht van de verantwoordelijkheden. Als afsluiting wordt er een lessons learned document gemaakt met daarin allen bevinden wat met het volgende project meegenomen kan worden.

Conclusie

Omdat PRINCE 2 een te generieke methode is en ook omdat Scrum afgeleid is van PRINCE 2 voor software ontwikkeling, heb ik deze keuze niet meer meegenomen. Daarna is er van agile scrum, waterval methode en RUP een vergelijkingstabel gemaakt (zie hieronder) met daarin de voor- en nadelen wat in hoofdstuk Projectmanagement methodes staat vastgesteld.

Omdat de waterval methode breed toepasbaar is met verschillende modellen, worden deze modellen in het vergelijk meegenomen. Mijn interpretatie van het resultaat van de waterval methode is dat het een verouderde manier is welke niet geschikt is voor een productontwikkeling in een snel ontwikkelende omgeving . Dit omdat het niet tot nauwelijks iteraties ondersteund waar een proces/fase herhaalt kan worden, en er geen leer mogelijkheid is na elke iteratie.

Tabel 1	**Eisen waaraan te voldoen (Bijlage 17.3)**		
Methodes	Agile Scrum	Waterval methode	RUP
Volgt de methode Iteraties?	Sprint planning Sprint review Daily meeting Twee week durende sprint	Volgt aflopende fases in plaats van iteraties	Het ontwikkel van het product verloopt in iteraties met behulp van een storyboard, use cases en taken.
Kunnen deze iteraties binnen een Milestone lopen? Of kan dat gedefinieerd worden?	Agile Scrum is flexibel, waarbij je zelf milestones, statussen etc. kan definiëren.	Waterval methode houdt zich aan de fase wat definieert staat en wat opvolgt. Als je dit aanpast dan val je weg van de waterval methode	RUP is een niet zo generieke methode waarbij je vast zit aan voor gedefinieerde methodes.
Kan in deze methode ook het product eigenaar in betrokken worden?	Hiervoor is een speciale rol, product owner, voor gereserveerd.	Voordat er begonnen wordt tijdens het ontwikkel fase en nadat de ontwikkel fase is afgerond zal er met de belanghebbende worden gesproken over wensen en eisen.	Voordat er begonnen wordt tijdens het ontwikkel fase en nadat de ontwikkel fase is afgerond zal er met de belanghebbende worden gesproken over wensen en eisen.

Kan in deze methode ook de klanten in betrokken worden?			
Heeft deze methode korte iteratieve reminders/meetings om het project in de juiste baan te sturen?	Sprint planning Sprint review Daily meeting Twee week durende sprint		Met tests binnen de software zal het project worden gevalideerd en kan het project management hierop bijsturen.
Heeft deze methode een efficiënte en effectieve overzicht om een zo goed mogelijke acceleratie te maken?	Als het werken product van een sprint gedefinieerd is dan kan er een snelle sprint worden gemaakt.	Als de gehele software vooraf gedefinieerd is dan kan er een snelle sprint worden gemaakt.	Als de gehele software vooraf gedefinieerd is dan kan er een snelle sprint worden gemaakt.
Wordt deze methode ondersteund binnen team foundation server?			

In de vergelijkingstabel (tabel 1) is te zien dat Agile Scrum het meest voldoet aan de eisen en wensen van de organisatie en het project. Ook omdat het gesteund wordt door team foundation server is het gemakkelijk te integreren binnen het project dat ontwikkeld wordt binnen Visual studio.
Van RUP zullen wij de best practices overnemen om de architectuur van het systeem in kaart te brengen. Hiervan zullen wij de architectuur van het systeem als eerst visualiseren en voor elke sprint daarvan de use cases, requirements en modellen ontwerpen en documenteren.
Met Agile Scrum en onderdelen van RUP zullen wij het systeem ontwikkelen.

C : Verwerken van persoonsgegevens binnen de Nederlandse wet

Binnen het huidige online-platform van Perflectie wordt veel omgegaan met persoonlijke gegevens van klanten en gebruikers. In het nieuwe systeem zal dit ook zo zijn. Daarom is het belangrijk dat met deze informatie verantwoordelijk en volgens de Nederlandse wet omgegaan wordt.

Perflectie heeft zich tot nu toe niet actief beziggehouden met het onderzoeken en volgen van de regels die binnen de Nederlandse wet zijn vastgesteld in het verwerken van persoonsgegevens. Daarom is dit onderzoek opgezet om eventuele risico's in kaart te brengen. De vragen voor dit onderzoek vanuit Perflectie, die door Jochem Aubel (Eigenaar Perflectie) zijn geformuleerd, zijn:

1. Welke data mag Perflectie opslaan?
2. Hoe moet deze data beveiligd worden?
3. Wie mag bij deze data komen?

Om deze vragen te beantwoorden zal ik gebruik maken van externe bronnen van de overheid en websites. Dit onderzoek is dan ook alleen bedoeld als een document met samengevatte informatie verkregen van deze bronnen, dat antwoord geeft op de bovenstaande vragen.

De Wet bescherming persoonsgegevens
Er zijn regels binnen de Nederlandse wet vastgesteld in het kader van het verwerken van persoonsgegevens. De Wet bescherming persoonsgegevens (Wbp) is de Nederlandse uitwerking van Europese richtlijn rondom bescherming van persoonsgegevens. (Wetten Wbp, 2014) Binnen de handleiding van de Wbp is een stroom diagram dat aangeeft of je de Wbp wel of niet binnen je organisatie moet toepassen. (Rijksoverheid, 2014) Een samengevatte inhoud van deze wet is te lezen in hoofdstuk WBP.

De cookie wet. (Rijksoverheid, 2014)

Websites moeten bezoekers informeren als zij cookies willen plaatsen. De bezoeker moet daarvoor toestemming geven. Dat geldt alleen voor cookies die surfgedrag bijhouden. Een samengevatte inhoud van deze wet is te lezen in hoofdstuk Cookiewet.

Het recht om vergeten te worden.
In de toekomst moet het mogelijk zijn voor bezoekers om aan te geven dat een organisatie ze moet 'vergeten'. De organisatie moet dan alle persoonsgegevens van de bezoeker verwijderen. Ook kunnen bezoekers een kopie opvragen van alle opgeslagen persoonsgegevens.

Het College Bescherming Persoonsgegevens
Het College Bescherming Persoonsgegevens (CBP) houdt toezicht op de verwerking van persoonsgegevens. Het kan eventueel boetes opleggen als organisaties de Wbp overtreden. Deze bevoegdheid wil het kabinet verder uitbreiden.

Ook vraagt het kabinet of het CBP bedrijven kan ondersteunen bij het ontwikkelen van nieuwe producten of bedrijfsprocessen. Bijvoorbeeld bij het opzetten van een abonnement systeem gekoppeld aan een website. Doordat het CPB eerder kritisch meekijkt naar de privacy, worden onnodige kosten en boetes achteraf voorkomen.

Digitale persoonsgegevens beveiligen
De CBP heeft beveiligingsnormen vastgesteld die zij gebruiken bij het onderzoeken en beoordelen van beveiliging van persoonsgegevens. Hierin staat dat de informatiebeveiliging moet voldoen in samenhang met de algemeen geaccepteerde beveiligingsstandaarden binnen de praktijk van de informatiebeveiliging. Hier wordt als voorbeeld gegeven de code voor informatiebeveiliging of de ICT-beveiligingsrichtlijnen voor webapplicaties van het Nationaal Cyber Security Centrum. Ook is hier een cyclus gedocumenteerd waarbij een organisatie blijvend een passend beveiligingsniveau kan behouden. De plan-do-check-act-cyclus. Hier kunt u meer over lezen in hoofdstuk Richtlijnen beveiliging van persoonsgegevens. (richtsnoeren beveiliging persoonsgegevens, 2014)

Wat zijn persoonsgegevens?

Gegevens zijn persoonsgegevens als:

- De gegevens informatie bevatten over een natuurlijk persoon
- Die persoon identificeerbaar is
- Gegevens die naar hun aard feitelijke informatie over een persoon geven(Bijv. NAW gegevens, maar ook als voorbeeld de persoon zijn IQ)

Als de persoonsgegevens kunnen leiden tot identificeren van een natuurlijk persoon:

- Gegevens over ondernemingen of organisaties
- Gegevens over voorwerpen of objecten

Wat houdt verwerken van persoonsgegevens in?

Verwerken van persoonsgegevens doe je als je:

- Verzamelen, vastleggen en ordenen
- Bewaren, bijwerken en wijzigen
- Opvragen, raadplegen, gebruiken
- Verstrekken door middel van doorzending
- Verspreiding of enige andere vorm van terbeschikkingstelling
- Samenbrengen, met elkaar in verband brengen
- Afschermen, uitwissen of vernietigen van gegevens

Elke handeling met betrekking tot persoonsgegevens is een verwerking van persoonsgegevens.

Cookiewet

Functionele cookies

Onder functionele cookies wordt verstaan cookies die een dienst of webshop moeten laten functioneren. Dit zijn bijvoorbeeld cookies die bijhouden wat er in een winkelwagen zit.

Versoepeling voor analytische cookies

Door de geplande versoepeling hoeft een website alleen om toestemming te vragen als dit ook echt nodig is om de privacy te beschermen. Bijvoorbeeld als de verzamelde statische gegevens ook worden gebruikt voor het opbouwen van bezoekersprofielen.

Tracking cookies: altijd toestemming vereist

Tracking cookies worden gebruikt om individueel surfgedrag bij te houden en om profielen op te stellen. Hiervoor moet je altijd toestemming vragen.

WBP

Moet Perflectie aan de Wbp voldoen?

Ja, Perflectie verwerkt persoonsgegevens binnen een geautomatiseerd systeem. Hierdoor is Perflectie, beschreven in (Rijksoverheid, 2014), de verantwoordelijke die met deze gegevens aan de voorwaarde van Wbp en bijbehorende wetten moet voldoen.

Als je aan de Wbp moet houden aan welke eisen moet je voldoen?

De Wbp vereist dat het verwerken van persoonsgegevens:

- Op behoorlijke en zorgvuldige wijze wordt uitgevoerd
- In overeenstemming met de wet is
- Verzamelen van persoonsgegevens met een duidelijke doel
    - Welbepaald
    - Uitdrukkelijk omschreven
    - Gerechtvaardigd zijn
- Duidelijk bepaald
    - Er mogen geen gegevens verzameld worden zonder dat daar een duidelijke doel is vastgesteld. Er mogen wel gegevens voor meerdere doeleinden verzameld worden. Deze doeleindes hoeven geen verband met elkaar te hebben.
- Omschrijven
    - Het doel of doeleinde moet omschreven worden voordat er begonnen wordt met het gebruik van de applicatie.(Het verzamelen van data) Deze doel of doeleinde mag niet uitgebreid worden of verandert worden. Er mag wel een nieuw doel of doeleinde gesteld worden wat gebruikt maakt van eerder verzamelde data van andere doelen of doeleindes.
- Noodzakelijk
    - Is het noodzakelijk voor het doel om deze gegevens te verwerken?

Gegevensverwerking moet steeds gebaseerd kunnen worden op één van de zes in de Wbp genoemde grondslagen. Als dit niet kan, dan is het verwerken van deze persoonsgegevens niet toegestaan.

1. Ondubbelzinnige toestemming
    a. De betrokkene moet zijn wil in vrijheid hebben geuit
        i. De betrokkende mag niet onder druk of dwang instemmen met het verwerken van zijn gegevens.
    b. De toestemming van de betrokkene moet gericht zijn op bepaalde gegevensverwerking(-en)
        i. De betrokkene moet zo geïnformeerd zijn voordat hij goedkeuring geeft voor het verwerken van zijn persoonsgegevens dat hij begrijpt waarvoor hij toestemming geeft.
    c. De toestemming moet ondubbelzinnig zijn
        i. Er mag geen twijfel ontstaan over de toestemming van de betrokkene. Die twijfel kan voorkomen worden door een verzoek om toestemming zo in te richten dat de

toestemming ondubbelzinnig blijkt. De last van dit bewijs ligt bij Perflectie. Perflectie moet kunnen bewijzen dat de overeenkomsten die zijn opgesteld bewijsbaar ondubbelzinnig zijn.

2. Noodzakelijk voor de uitvoering van een overeenkomst
    a. Als u met iemand een overeenkomst hebt gesloten, mag u de persoonsgegevens van diegene verwerken voor zover dat noodzakelijk is om de overeenkomst uit te kunnen voeren.
    b. Er mag op basis van deze grondslag ook gegevens verwerkt worden in de fase voor het sluiten van de overeenkomst
        i. De betrokkene verzoekt om de handelingen
        ii. De handelingen noodzakelijk zijn om de overeenkomst te kunnen sluiten
3. De verwerking is noodzakelijk ter uitvoering van een wettelijke plicht van de verantwoordelijke
    a. Het moet redelijkerwijs niet goed mogelijk zijn de wettelijke plicht uit te voeren zonder het verwerken van persoonsgegevens
4. Het verwerken van persoonsgegevens is noodzakelijk ter vrijwaring van een vitaal belang van de betrokkene
    a. Als voorbeeld: Voor medisch belang
5. Het verwerken van gegevens is noodzakelijk voor de goede vervulling van een publiekrechtelijke taak
6. Het verwerken van persoonsgegevens is noodzakelijk voor de behartiging van een gerechtvaardigd belang
    a. Als er geen activiteiten uitgeoefend kan worden zonder het verwerken van persoonsgegevens.
    b. Dit geldt ook voor het gericht tonen van reclame dat afwijkt van het hoofddoel van de organisatie.
    c. Noodzakelijk
    d. De rechten van de betrokkenen mogen niet gebroken worden om als voorbeeld geen inbreuk te maken op zijn privacy
7. De gegevensverwerking mag niet onverenigbaar zijn met het doel waarvoor de gegevens zijn verzameld.
    a. De verdere verwerking mag niet onverenigbaar zijn, de betekenis hiervan hangt af van omstandigheden
    b. De factoren waar rekening mee gehouden moet worden
        i. De mate van verwantschap tussen het oorspronkelijke doel en het doel van de verdere verwerking
        ii. De aard van de gegevens
        iii. De gevolgen van de beoogde (verdere) verwerking voor de betrokkene
        iv. De wijze waarop de gegevens zijn verkregen en de mate waarin passende waarborgen voor de betrokkene zijn genomen

Aan welke kwaliteitseisen moet mijn gegevensverwerking voldoen?

- Niet bovenmatig
    - Er moet voor gezorgd worden dat voldoende en adequate gegevens verwerkt worden volgens het doel of de doelen van Perflectie. Er mogen geen gedetailleerde gegevens verwerkt worden als dat voor het doel niet noodzakelijk is.
- Toereikend
    - Er mogen niet te weinig gegevens verwerkt worden. Dat wil zeggen dat alle gegevens verwerkt moeten worden die voor het betreffende doel noodzakelijk is.
- Ter zake dienend zijn
    - Er mogen alleen gegevens verwerkt worden die noodzakelijk zijn voor het doel. Er mogen geen overbodige gegevens verwerkt worden.
- De verwerkte gegevens moeten verder juist en nauwkeurig zijn
    - Deze verplichting is niet absoluut. Er hoeft niet altijd door ons na te worden gegaan of de door ons verwerkte gegevens juist zijn. Het is alleen wel verstandig om periodiek te controleren of de gegevens accuraat zijn.

Aanmelden als verantwoordelijke

U moet als verantwoordelijke uw gegevensverwerking melden bij:

- Het College bescherming persoonsgegevens
- Als door u of uw brancheorganisatie een functionaris voor de gegevensbescherming is benoemd.

Welke gegevensverwerking moet gemeld worden?

- Uw geheel of gedeeltelijk geautomatiseerde gegevensverwerking
- Uw handmatige gegevensverwerking mits deze aan een voorafgaand onderzoek is onderworpen

Welke gegevensverwerking hoef ik niet te melden?

- De doeleinden van de vrijgestelde verwerkingen
- De verwerkte gegevens of categorieën van gegevens
- De categorieën van betrokkenen
- De ontvangers of categorieën van ontvangers aan wie gegevens worden verstrekt

Wat moet ik doen bij een melding?

- Naam van de verantwoordelijke (Perflectie)
- Adres van de verantwoordelijke
- Doel of doeleinden van de gegevensverwerking
- (Categorieën van) betrokkenen
- (Categorieën van) gegevens van deze betrokkenen
- (Categorieën van) ontvangers
- De voorgenomen doorgiften van persoonsgegevens aan landen buiten de Europese unie
- Een omschrijving van de door u te nemen beveiligingsmaatregelen

Wat moet ik doen als ik mijn gegevensverwerking wijzig?

- Het doel of de doeleinden van de gegevensverwerking doorgeven aan de CBP
- De (Categorieën van) betrokkenen en ontvangers formuleren van deze wijzigingen
- De beveiligingsmaatregelen behouden1.4.4 De bewerker

De eisen die Wbp stelt aan de bewerker van persoonsgegevens zijn als volgt vastgesteld:

- De bewerker biedt voldoende waarborging met betrekking tot de technische en organisatorische beveiliging
- Er moet een overeenkomst met de bewerker bestaan welke de afdwingbare verbintenissen tussen de organisatie en de bewerker uitsluit.
- In overeenkomst dat de bewerker de persoonsgegevens die worden verzameld vanuit uw applicatie uitsluitend verwerkt in uw systeem
- De bewerker de beveilingsverplichtingen nakomt
    - Passende technische en organisatorische maatregelen nemen om het verlies van gegevens of onrechtmatig verwerking tegen te gaan
    - De (technische en organisatorische) maatregelen die de bewerker neemt, moeten een passend beveiligingsniveau garanderen.
        - De risico's van de verwerking en de aard van de te beschermen gegevens.
        - U moet rekening houden met de stand van de techniek en de kosten van de maatregelen
    - Onnodige verzameling en/of verwerking voorkomen
    - De beveiliging moet adequaat zijn, dit betekend dat het systeem periodiek gecontroleerd moet worden op beveiligingsrisico's en/of verbeterende beveiligingsimplementaties, door bijvoorbeeld technische ontwikkelingen.
- De bewerker daadwerkelijk toezien op naleving van deze beveiligingsverplichtingen, door middel van controles en/of aangewezen persoon.

Richtlijnen beveiliging van persoonsgegevens

ICT-beveiligingsrichtlijnen voor web applicaties van het Nationaal Cyber Security Centrum

Het NCSC (Nationaal Cyber Security Centrum) scant regelmatig honderden bronnen op het internet om actuele bedreigingen in beeld te brengen. Hiervoor stellen zij beveiligingsadviezen op, hierin is te lezen wat de kwetsbaarheden waren of zijn en hoe dit verholpen kan worden. (Beveilingsadviezen, 2014)

Een organisatie kan het NCSC inhuren voor hulp bij incidenten. Dit hulpmiddel is bedoelt voor rijksoverheidsorganisaties maar er zijn plannen om dit uit te breiden naar vitale sectoren.

Ook behandelt het NCSC uitgangspunten bij het bepalen van de kans en schade, deze punten zijn:

- Firewall
    - Gebruik een firewall om alleen de poorten beschikbaar te stellen waar een door beleid gedefinieerde en gedocumenteerde noodzaak voor bestaat.
- Filtering op segment
    - Maak gebruik van verschillende netwerksegmenten (Bijvoorbeeld: Beheersegment, productiesegment en een serversegment) afgescheiden door firewalls
        - Verwijzingen
        - http://www.govcert.nl/dienstverlening/Kennis+en+publicaties/factsheets/achtergrondinformatie-over-storm-worm.html
        - http://www.govcert.nl/dienstverlening/Kennis+en+publicaties/whitepapers/raamwerk-beveiliging-webapplicaties.html
- Beveiliging van infrastructuurcomponenten
    - Beveilig infrastructuurcomponenten zoals routers, switches en servers
- Gebruik van een server
    - Een server wordt alleen gebruikt om voorgedefinieerde diensten aan te bieden. De server wordt niet als werkstation gebruikt.
- Anti-virus software
    - Voorzie elke host van antivirussoftware. Dit geldt voor werkstations en servers, zoals file- en mailservers.
- Content filtering
    - Controleer netwerkverkeer van en naar het internet zoveel mogelijk op kwaadaardige inhoud.
- Versleuteling
    - Gebruik de versleuteling van veilige protocollen (SFTP, SSH, HTTPS)
- Geen standaard configuratie
    - Maak zo min mogelijk gebruik van standaard configuratie van systemen.
- Gebruikers hebben geen administrator rechten
    - Gebruikers hebben geen Administrator of Root rechten op een werkstation
- Beheeraccounts
    - Gebruik aparte beheeraccounts voor beheertaken
- Patchen
    - Volg de beveiligingsadviezen op en installeer beschikbare en relevante patches of work-arounds.
- Gebruikers bewustzijn
    - Maak eindgebruikers bewust van de meest voorkomende beveiligingsrisico's. Zoals het klikken van willekeurige links en/of mail attachments.
- Mobiele apparatuur is voorzien van beveiligingsmaatregelen
    - Mobiele apparaten, zoals laptops, zijn op het hetzelfde beveiligingsniveau als de werkstation binnen de organisatie.
- Belangrijke systemen zijn fysiek beveiligd
    - Fysieke beveiliging voor belangrijke systemen, zoals servers en infrastructuurcomponenten
- Geen vreemde hardware op het netwerk

- Zorg ervoor dat er alleen hardware aanwezig is op het netwerk welke is aangekocht en wordt beheerd door de organisatie.

Plan-do-check-act

1. Beoordeel de risico's
    a. Inventariseer de dreigingen die kunnen leiden tot een beveiligingsincident, de gevolgen die dit beveiligingsincident kan hebben en de kans dat deze zich voor kan doen. Tref op basis hiervan gericht beveiligingsmaatregelen die het gewenste beveiligingsniveau waarborgen.
2. Maak gebruik van algemeen geaccepteerde beveiligingsstandaarden
    a. Binnen het vakgebied informatiebeveiligingen zijn er veel beveiligingsmethoden, -standaarden en maatregelen. Maak hier gebruik van als het van toepassing is.
3. Controleer en evalueer regelmatig
    a. Controleer met regelmaat of de beveiligingsmaatregelen daadwerkelijk zijn getroffen en worden nageleefd. Beoordeel periodiek op risico's en of de bestaande risico's nog voldoen aan de eisen.(Zie punt 1) Pas waar nodig de beveiligingsmaatregelen aan.

NCSN Inschalingsmatrix

De beveiligingsadviezen van NCSC bevatten een inschaling van de beschreven kwetsbaarheid. De mogelijke waarden per onderdeel zijn Low, Medium of High. Voor zowel de kans als schade wordt een set vragen beantwoord die leiden tot een waarden. Bron: (Beveilingsadviezen, 2014) (Zie download InschalingsMatrix)

Kans

Door de onderstaande vragen te beantwoorden en de waarde toe te kennen kan je zo de kans bepalen.

Tabel 3 Kansen matrix

Vraag	Optie 1		Optie 2		Optie 3	
Is de kwetsbaarheid aanwezig in de standaard configuratie/installatie?	Nee	1	Onduidelijk/Ja	3		
Is er Exploitcode beschikbaar?	Geen	1	Proof of concept	4	Exploit	6
Zijn er technische details beschikbaar	Geen	1	Enigszins	2	volledig	3
Vereiste toegang	Fysiek	1	LAN/Directe omgeving	4	Internet	6
Vereiste credentials?	Admin	1	User	2	Geen	4
Hoe complex is het technisch gezien om de kwetsbaarheid uit te buiten?	Complex	1	Gemiddeld	2	Eenvoudig	3
Is er gebruikersinteractie nodig?	Complex	1	Eenvoudig	3	geen	4

Wordt de kwetsbaarheid in het wild uitgebuit?	Nee	1	Beperkt	2	Grootschalig	3
Wordt de kwetsbaarheid, naar verwachting, op korte termijn misbruikt of verschijnt er een exploit?	Nee	1	Ja	3		
Beschikbaarheid oplossing?	Ouder dan 2 maanden	1	Tot 2 maanden oud	2	Geen	3

Schade

De schade wordt bepaald door een van de onderstaande schadeomschrijvingen te kiezen.

Tabel 4 Schade matrix

Vraag	Optie 1		Optie 2		Optie 3	
Denial of Service (DoS)	Nee	Low	Ja, Client	Low	Ja, infrastructuur dienst	High
Uitvoeren van willekeurige code	Nee	Low	Ja, Gebruikers rechten	Medium	Ja, Root / adminstrator rechten	High
Rechten op afstand	Nee	Low	Ja, remote shell	Medium	Ja, remote root- shell	High
Verwerven lokale admin/root rechten	Nee	Low	Ja	Medium		
Lekkage informatie	Nee	Low	Ja, systeem informatie	Medium	Ja, data	High

Regels voor Perflectie om persoonsgegevens te verwerken

Welke data mag Perflectie opslaan?

Perflectie slaat persoonsgegevens op van zijn gebruikers die kunnen terugleiden tot een natuurlijk persoon. Hierdoor moet Perflectie zich inschrijven bij de CBP en zich houden aan de WBP wet. Daarnaast mag Perflectie alleen de persoonsgegevens opslaan voor een doel waarvoor de gebruiker vooraf duidelijk is ingelicht. De regels en inhoud van deze wet is gedocumenteerd in hoofdstuk WBP.

Hoe moet deze data beveiligd worden?

Er worden geen duidelijke eisen gesteld binnen de WBP, CBP of NCSN, wel worden richtlijnen aangegeven waar de hard- en software van een organisatie aan moet voldoen en waar de organisatie op beoordeeld kan worden. (Zie bijlage C : ICT-beveiligingsrichtlijnen voor web applicaties van het Nationaal Cyber Security Centrum) Hiervoor stelt de NCSN ook een lijst beschikbaar met daarin de actuele beveiligingsrisico's voor (web-)applicaties. Deze risico's worden bepaald door de inschalingmatrix welke gedocumenteerd staat in NCSN

Inschalingsmatrix. Als laatste is hier een cyclus gedocumenteerd waarbij een organisatie blijvend een passend beveiligingsniveau kan behouden. De plan-do-check-act-cyclus. Hier kunt u meer over lezen in bijlage C : Plan-do-check-act.

Wie mag bij deze data komen?

Persoonsgegevens mogen alleen verwerkt worden voor een bepaald doel en op basis van dit doel mogen geautoriseerde gebruikers bij deze data komen.

De gebruiker
Iedereen heeft recht op inzage in zijn of haar persoonsgegevens. Bij buitensporig (vaak) verzoek hoeft hier geen gehoor aan gegeven worden. Bij een verzoek moet de organisatie binnen vier weken handelen, waarbij het antwoord schriftelijk moet zijn. Hierbij moet de organisatie aan de gebruiker:

- Een volledig overzicht van de door de organisatie verwerkte gegevens leveren
- Een omschrijving van
    - Het doel of de doeleinden van de gegevensverwerking
    - De categorieën van gegevens waarop uw verwerking betrekking heeft
    - De ontvangers of categorieën van ontvangers
- Alle beschikbare informatie over de herkomst van de gegevens afgeven

Als er naar wordt gevraagd is de organisatie verplicht om informatie over de systematiek van de geautomatiseerde gegevensverwerking te verstrekken, dit alleen als daarbij geen bedrijfsgeheimen worden prijsgegeven.

D : Vooronderzoek

Om antwoord te bied en op de hoofdvraag en de deelvragen (zie hoofdstuk 3.3 De hoofdvraag met zijn deelvragen), zullen de antwoorden ondersteund worden door wetenschappelijke artikelen, boeken, bestaande code en / of bewezen ontwerp principes.

Wat maakt een systeem onderhoudbaar en uitbreidbaar?

Elke software systeem heeft zo zijn eigen eisen toegewezen gekregen. Waar systemen gezamenlijk wel aan moet voldoen is onderhoudbaarheid.

Waarom is een onderhoudbaar en uitbreidbaar systeem belangrijk?

Een software systeem heeft geheel zijn levensduur onderhoud nodig. Software bestaat uit een digitale vorm, dat bloot staat aan verval. In theorie kan software jarenlang draaiend blijven maar in praktijk blijkt dit nooit te gebeuren. Software kan vergeleken worden als een organisme dat bestaat in een bepaalde omgeving en die zich in deze omgeving moet aanpassen om te overleven. (Vandegriend, 2014)
Daarom moet binnen een software systeem nieuwe features doorgevoerd worden en bugs moeten opgelost worden. Aantal oorzaken om een systeem te onderhouden zijn:

1. Het oplossen van een defect binnen het systeem
2. Het veranderen van business rules.
3. Het veranderen van de omgeving waar het systeem in zich bevindt

Een onderhoudbaar systeem is gemakkelijk uit te breiden met features en gemakkelijk om bugs op te lossen. Dit heeft als voordeel dat de software langer gebruikt kan worden.

Wat is een onderhoudbaar systeem?

Een onderhoudbaar systeem zorgt ervoor dat je gemakkelijk de volgende acties uit kan voeren: (Crouch, 2014)

- Bugs oplossen
- Nieuwe features toevoegen
- Het verbeteren van gebruik
- Het verbeteren van performance

- Het maken van een oplossing om toekomstige bugs tegen te gaan
- Het veranderen of vervangen van de omgeving van de code
- Dat het gemakkelijk is dat andere programmeurs het systeem kan onderhouden

Het Software & Systems Engineering Standards Committee beschrijft een onderhoudbaar system binnen (Radatz, 1990) als volgt:

> *"Het gemak waarmee een software module of onderdelen kunnen worden gemodificeerd om fouten te corrigeren, de prestaties te verbeteren of andere kenmerken of aanpassing aan een veranderende omgeving ."*

Waarom is een onderhoudbaar systeem gewenst?

Het is gebruikelijk om aan te nemen dat het gewoonlijk is dat als de programmeur het systeem begrijpt en kan onderhouden dat dat voldoende is. Deze situatie kan binnen een periode veranderen met als vervolg dat de programmeur niet meer benaderbaar is. Dan wordt het lastig voor de organisatie om de vaardigheden en kennis te vervangen.

In dit geval en in andere gevallen zal de nieuwe programmeur zich in code bevinden wat niet tot nauwelijks te begrijpen is. Dit kost kostbare tijd. Een onderhoudbaar systeem is gemakkelijk te begrijpen. Dit is een systeem dat begrepen kan worden door iemand nieuw, of iemand dit al een tijd is weggeweest, met minimale moeite.

De gevolgen van een niet onderhoudbaar systeem

Wanneer de koste laag gehouden moeten worden en/of wanneer materialen en personeel minimaal is gebeurt het snel dat een systeem zo snel mogelijk opgezet wordt. Hier wordt vooral de methode aangehouden “als het maar werkt”. Bij de methode mis je documentatie, testing en refactoring. Meestal worden deze gemiste onderdelen beloofd om na het project op te pakken, dit wordt vaak niet gedaan.

De “als het maar werkt” methode kan in het begin veel tijd schelen tijdens de ontwikkeling van een systeem, maar later door onleesbare code, bugs, geen documentatie en tests wordt de voorsprong van tijd ingehaald door langdurige onderhoud.

Beschreven in (Crouch, 2014) kan deze schade ook vergeleken worden met schulden die je opstapelt binnen een systeem, over tijd. Deze schulden zullen later geïncasseerd worden als onderhoud.
Een systeem dat niet onderhoudbaar is opgezet kost ongeveer 4 keer meer tijd om veranderingen en verbeteringen door te voeren, ter vergelijking hoe lang het heeft geduurd om het onderdeel te ontwikkelen. (Crouch, 2014) Voor deze reden wordt vaak een systeem geheel opnieuw opgezet.

Hoe kan een onderhoudbaar systeem opgezet worden?

Er zijn enkele principes, methodes en technieken wat aangehouden kan worden om een onderhoudbaar systeem op te zetten. Veel van deze zijn generiek inzetbaar bij elk software systeem. Op de website (Crouch, 2014) en (Miller, Code better, 2014) zijn de generieke methodes opgesomd als:

- Ontwerp het systeem voor onderhoudbaarheid van het begin af
- Het systeem op een iteratieve methode managen/ontwikkelen.
- Reguliere reviews om de kwaliteit van het systeem te waarborgen
- Leesbare code dat makkelijk te begrijpen is
- Refactor de code om de begrijpbaarheid te verbeteren
- Relevante documentatie dat programmeurs helpt om het systeem te begrijpen
- Geautomatiseerde builds van het systeem zorgt voor gemakkelijk compileren van code
- Geautomatiseerde tests zorgen voor gemakkelijke validaties bij veranderingen binnen het systeem
- Continue integratie maakt de code gemakkelijk te builden en te testen
- Version control helpt om de code, testen en documentatie up to date te houden binnen het projectteam

- Verander de manier van werken, met als doel onderhoudbaarheid

Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?

Vragen voor een programmeur

Er zijn enkele vragen wat je kunt stellen om te kunnen beoordelen of een systeem onderhoudbaar is. (Crouch, 2014)

- Kan ik code vinden dat gerelateerd is aan een specifiek probleem of verandering?
- Kan ik code begrijpen? Kan ik uitleggen wat de bedoeling is achter de functionaliteiten aan iemand anders?
- Is het gemakkelijk om de code te veranderen? Is het gemakkelijk voor mij om vast te stellen wat er verandert moet worden als consequentie van de vorige verandering? Zijn de hoeveelheden en omvang van deze consequentie klein?
- Kan ik snel een verandering binnen het systeem verifiëren?
- Kan ik een verandering doorvoeren met een laag risico dat er bestaande functionaliteiten breken?
- Als er een bug ontstaat, is het probleem snel en gemakkelijk te detecteren en te diagnoseren?

Deze vragen kunnen herhaald worden en dan als perspectief van iemand anders van jouw projectteam gebruiken en iemand die geheel nieuw is binnen het systeem.

De voorwaarden

De methodes en principes die beschreven staan in hoofdstuk 5.1 De gebruikte methodes kunnen gebruikt worden als voorwaarden waar een onderhoudbaar en uitbreidbaar systeem aan moet voldoen. Deze voorwaarden zijn als volgt:

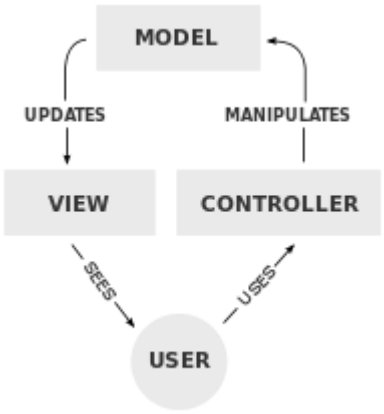
- Is het systeem ontworpen van het begin af om onderhoudbaar te zijn?
- Wordt het systeem op een iteratieve methode gemanaged/ontwikkeld?
- Waren er of bevinden zich er reguliere reviews om de kwaliteit van het systeem te waarborgen?
- Is er binnen het systeem leesbare code dat gemakkelijk te begrijpen is?
- Is de code gerefactored in een proces om de begrijpbaarheid te verbeteren?
- Is er beschikking tot relevante documentatie voor programmeurs om het systeem te begrijpen?
- Kan het systeem automatisch gebuild worden zodat het gemakkelijk is om de code te compileren?
- Zijn er (geautomiseerde) tests aanwezig zodat het gemakkelijk is om bij veranderingen het systeem te valideren?
- Is er continue integratie binnen de code zodat builden en testen gemakkelijk gemaakt wordt?
- Wordt version control binnen het systeem gebruikt zodat ieder teamlid de laatste versie heeft van de code, testen en documentatie?
- Houdt of heeft ieder teamlid binnen het systeem de werkwijze aangehouden om zo onderhoudbaar mogelijk werk te produceren.

Wat is het structuur binnen het huidige online-platform van Perflectie?

De huidige online-platform van Perflectie maakt gebruik van MVC 4. MVC4 Verzorgd voor een systeem een standaard collectie van packages. Deze packages hebben als inhoud subsystemen wat verder geïmplementeerd kan worden binnen het MVC project. Dit helpt om snel en eenvoudig een solide web applicaties op te bouwen.

Gebruikte software

Het online-platform van Perflectie, dat huidig gebruikt word, is opgebouwd in Visual studio 2012. Met Visual studio kan er voorgemaakte Microsoft C# projecten of lege projecten opgezet worden. Perflectie heeft de MVC 4 project als basis gebruikt, dit creëert een programmeer omgeving met de architecture pattern: Model View Controller. Nummer vier binnen MVC staat voor de vierde versie van MVC van Visual studio.



Figuur 2 MVC Design pattern

Door het gebruiken van MVC 4 word er een standaard collectie van packages geïnstalleerd. Deze packages hebben als inhoud subsystemen wat verder geïmplementeerd kan worden binnen het MVC project. Dit helpt om snel en eenvoudig een solide web applicaties op te bouwen.
De meest belangrijkste packages onder deze verzameling zijn:

- Microsoft authentication (Identity 2.0)

Microsoft heeft voor zijn web applicaties een eigen authenticatie systeem ontwikkeld. Dit systeem is vrij om te gebruiken en zorgt voor de ontwikkelaar een basis van enkele gebruikers systeem wat gemakkelijk uitgebreid kan worden. Zulke uitbreidingen kunnen bestaan uit: gebruikers rollen, eigen gebruikers gegevens en het refereren naar andere classes binnen het systeem.

- Entity Framework

Bij het implementeren van een Entity Framework, kan er code-first programming te werk gaan. Deze methode heeft als inhoud dat een Context class automatisch geconverteerd word naar een database structuur en data.

- JSON

JSON is een techniek om vanaf de backend naar de front-end, maar ook vanaf de front-end naar de backend functionaliteiten en data te communiceren. Door JSON objecten te maken kan er zo binnen javascript aan de gebruiker de data getoond worden. Via JSON calls naar de backend kunnen er functionaliteiten worden opgeroepen.

- JQuery

JQuery is een verzameling van functionaliteiten binnen javascript. Ook wel een library genoemd. Deze methodes kunnen verder weer worden gebruikt binnen je javascript om zo met min mogelijk code complexe functionaliteiten uit te voeren. Ook kan JQuery uitgebreid worden met plug-ins.

- Bootstrap

Bootstrap is een verzameling van functionaliteiten binnen CSS en Javascript. Ook wel een library genoemd. Deze methodes kunnen verder worden gebruikt binnen je CSS en HTML om zo met min mogelijke code complexe functionaliteiten en visuele structuren uit te voeren. Ook bootstrap kan worden uitgebreid met plug-ins.

Het hoofd solution van het project van Perflectie bestaat huidig uit 5 projecten, genaamd:

1. Business Domain
2. Data Access
3. Services
4. Unit Test
5. Web UI
6. Mockup

Backend

Business Domain

Binnen elk goed gestructureerd softwaresysteem is er een business domain gedefinieerd. Hierin staat de classes met hun attributen en relaties met andere domain classes. Deze classes zijn de blauwdrukken om objecten te instantiëren wat de mogelijkheid heeft om opgeslagen te worden in de database. Deze objecten worden uiteindelijk gepresenteerd aan de gebruiker via de client.
Verschillende manieren kan worden gebruikt om hetzelfde doel te bereiken binnen een software systeem, maar niet elke weg is de juiste weg. Sommige wegen leiden tot niet onderhoudbare code, complexe en repetitieve functionaliteiten wat niet hoort bij het systeem.

Huidige staat van de business domain

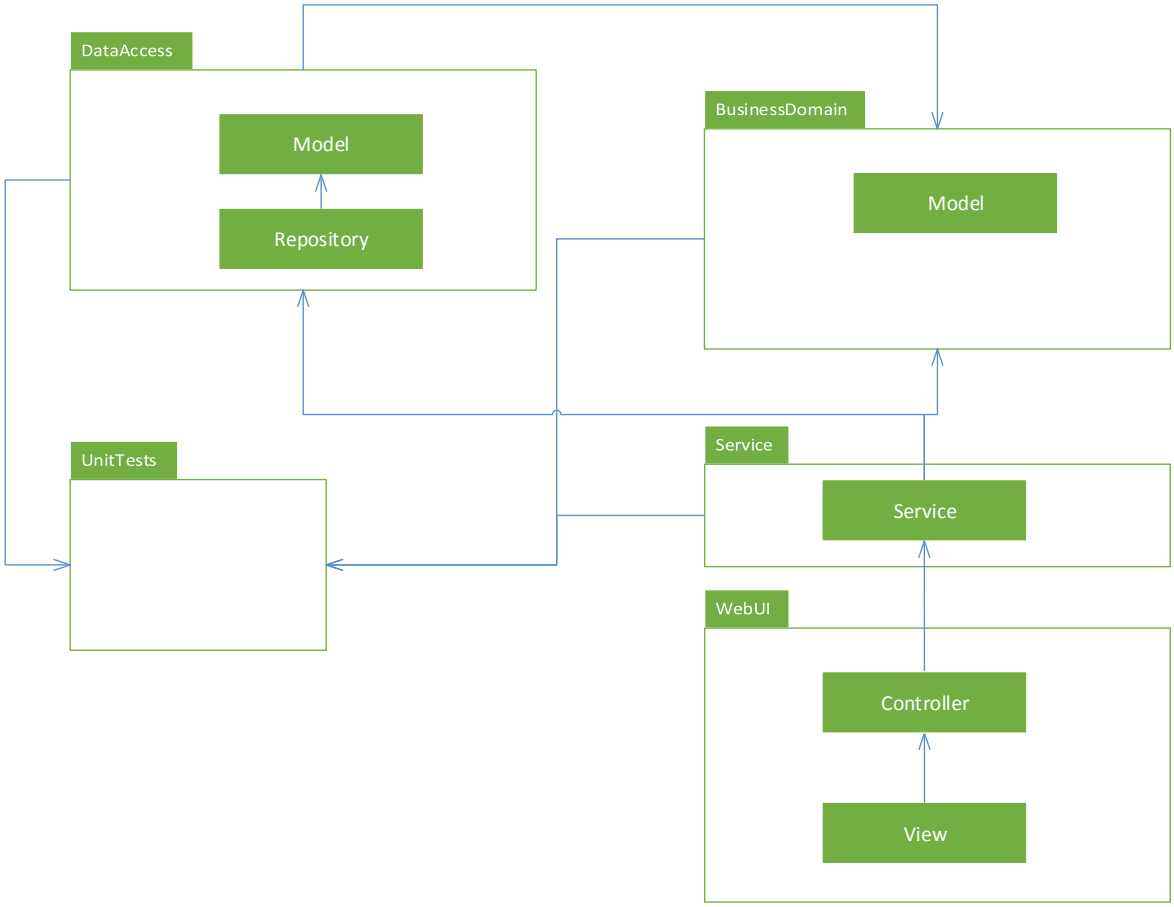
De code dat is opgehaald op 11-08-2014, heeft als inhoud een business domain, dat geanalyseerd is door mij. Deze analyse heeft uiteindelijke geleid naar een UML class diagram. (Zie bijlage J) De reden waarom ik voor deze aanpak heb gekozen is omdat via een UML diagram het gemakkelijk is om een duidelijke beeld te krijgen van een software structuur zonder daar veel tijd aan te spenderen.
Eerste impressie
Mijn eerste impressie van het UML class diagram (zie bijlage J) is dat er geen architectuur en design-patterns zijn geïmplenteerd, slechte benaming, repetitieve data, inconsistente data, losstaande classes, geen unit-tests en onnodige relaties.

- **Geen design patterns**
  Er zijn geen design patterns toegepast om code problemen op te lossen, inplaats van design patterns is er repetitieve data toegevoegd. Dit kan verbeterd en/of aangepast worden met een duidelijker en consistente data structuur. Door gebruik te maken van de website van SourceMaking (Design Patterns, 2014), kan er een heldere presentatie gegeven worden welke design patterns er beschikbaar zijn en op welke probleem en omgeving deze toegepast kan worden.
- **Repetitieve & inconsistente data**
  Repetitieve en inconsistente classes en attributen zijn gedefinieerd binnen de business domain. Deze data kan worden genormaliseerd en toebedeeld worden met betere benaming. Als oplossing moet er afgesproken worden welke conventie binnen het project aangehouden wordt en de design principes moet in kaart worden gebracht.
- **Losstaande/zwevende classes**
  De classes die losstaan en totaal geen relatie met iets heeft binnen het domain classes zijn losstaande classes. Dit betekend dat de klasse niet binnen het systeem word gebruikt of voor een andere bedoeling. Na verder analyse blijkt dat deze classes niet gebruikt worden en verwijdert kan worden.
- **Onnodige relaties**
  Domain classes hebben veel en/of teveel relaties naar andere domain classes. Dit leid naar een niet onderhoudbare software systeem en database structuur. Om dit probleem op te lossen moeten er generieke classes en methodes aangemaakt worden met behulp van design patterns.

Huidige lagen structuur van het online-platform

In het huidige project is de MVC architectuur toegepast, hier is alleen niet aan de regels gehouden. De verantwoordelijkheden en functionaliteiten in de huidige applicatie is niet aan de regels gebonden wat in een

MVC is vastgesteld. (Zie figuur 6) Zonder te houden aan vastgestelde regels van lagen heb je als resultaat een project met code, dat groot afhankelijk is van elkaar. Het grootste nadeel hiervan is dat de MVC pattern niet meer onderhoudbaar, vervangbaar of uitbreidbaar is. Als je vooraf een duidelijke lagen architectuur aanbrengt kan je een laag vervangen, aanpassen of verwijderen zonder dat onderliggende laag word aangetast.



Figuur 6 Structuur van het online-platform van Perflectie

Het gebruik van (real-time) unit testing en dependency injection

Op dit moment is er geen tests aanwezig binnen het systeem. Dit blijkt ook na elke nieuw release van het online-platform van Perflectie. Na elke release geeft Perflectie aan dat er duidelijke bugs aanwezig zijn binnen het systeem. Deze bugs worden ook weer ontdekt door de gebruikers die een eindproduct krijgen geleverd. Om deze bugs zo vroeg mogelijk en liefst niet live tegen te komen moeten er unit tests geïmplementeerd worden. Met behulp van Dependency Injection (Developer's Guide to Dependency Injection Using Unity, 2014) kan een system lousely coupled opgezet worden. Dit in combinatie met Unit testing kan je zowel test data opzetten voor de unit tests zowel ook real-time database aankoppelen.

Client-side

Views & JavaScript

Binnen het huidige online-platform van Perflectie is er niet zorgvuldig omgegaan met de MVC conventies en regels. Dit heeft als effect dat er logica word afgehandeld binnen de controllers en binnen de .cshtml views. Dit is algemeen vastgesteld als bad code design. Om duidelijk de logica te uit de client-side te halen zijn er strenge en vaste regels nodig voor de programmeurs en voor de toekomstige programmeurs. Dit document zal het architecture notebook heten. In dit document zal onder andere te lezen zijn in welke laag onder welke verantwoordelijkheid ligt.

Ook is er inline scripts gebruikt van javascript binnen de views. Dit maakt een view onoverzichtelijk en niet onderhoudbaar. Een stuk javascript code kan veel regels bevatten en de logica binnen dit script in een view maakt dit niet herbruikbaar of overzichtelijk.

Wat is het probleem/wat zijn de problemen binnen het huidige online-platform van Perflectie?

Binnen het project heb ik vooral met Jochem Aubel contact gehad over het huidige online-platform en welke rol de partner module aan moet voldoen. Tijdens deze gesprekken kwamen wij achter dat gedurende jaar het huidige online-platform bestaat altijd bugs opgelost of nieuwe features toevoegt moest worden. Deze taken duurde lang en het was voor de programmeur een lange tijd niet duidelijk hoe het systeem precies werkt. Documentatie over het systeem bestaat niet en er zijn geen tests geïmplementeerd. Die heeft als effect dat na elke release van een nieuwe versie van het online-platform er veel bugs bestaan die door de gebruiker gevonden wordt.

1. Geen documentatie
2. Geen tests, geen vertrouwen
3. Verouderde versies van software binnen het project
4. Veranderingen binnen het systeem kunnen niet gemakkelijk gevalideerd worden
5. Architectuur & Design patterns worden niet binnen het systeem aangehouden, dit heeft geleidt tot:
    a. Veel afhankelijkheden en onduidelijkheden binnen het systeem
    b. God classes (een class binnen een systeem die (bijna) alles uitvoert
    c. Geen tot weinig design patterns, dit heeft geleidt tot:
        i. Code dat herhaald wordt

Voldoet het huidige online-platform aan de resultaat aan voorwaarden in deelvraag 2?

Om te testen of het huidige online-platform voldoet aan de voorwaarden waar een onderhoudbaar systeem aan moet voldoen beschreven in hoofdstuk Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?, heb ik de vragen gesteld die daar afgeleid van zijn. Deze vragen zullen beantwoordt worden aan de hand van het huidig online-platform en via gesprekken met Jochem Aubel.

- Is het systeem ontworpen van het begin af om onderhoudbaar te zijn?

Na een gesprek met Jochem Aubel heb ik vernomen dat het systeem initieel wel onderhoudbaar ontworpen is. Dat is ook Netwinst haar motto: om applicaties te leveren die onderhoudbaar en uitbreidbaar zijn, waarbij gewerkt wordt met modules die gemakkelijk vervangbaar zijn. De documentatie met daarbij ook de ontwerpen zijn niet door Netwinst vrijgegeven aan Perflectie.
Aan de hand van het systeem en de gemaakte UML-diagram (hfst. 6.1) is het overduidelijk dat het systeem niet onderhoudbaar of door middel van modules is opgezet. Daarom voldoet het systeem niet aan deze voorwaarde.

- Wordt het systeem op een iteratieve methode gemanaged/ontwikkeld?

Nee, de ontwikkeling van het huidige online-platform was zo verlopen waarbij als eerst het platform is opgezet en daarna gedurende periode van een jaar steeds nieuwe features zijn toegevoegd. Dit is niet iteratief gemanaged of op een iteratieve manier ontwikkeld.

- Waren er / of bevinden zich reguliere reviews om de kwaliteit van het systeem te waarborgen?

Er wordt elke dag een daily stand-up gehouden om taken te bespreken. Hier werd echter nooit rekening gehouden met het systeem in het geheel, maar alleen dat de feature moest werken.
De Agile Scrum is later ingevoerd waarbij het huidige systeem niet aan deze voorwaarden kan voldoen.

- Is er binnen het systeem leesbare code dat gemakkelijk te begrijpen is?

Nee, na een gesprek met Jochem Aubel en een vorige collega, een programmeur, van Perflectie was het duidelijk dat bugs oplossen of überhaupt het systeem te begrijpen en uit te breiden een hele taak is. Hier komt teveel tijd in te zitten waarbij Jochem Aubel er ook geen overzicht over heeft wat er precies wordt gemaakt.

- Is de code gerefactored in een proces om de begrijpbaarheid te verbeteren?

Er zijn pogingen gedaan om de code te refactoren, maar na een jaar van code corrosie is dit een te grote taak en is daarom ook nog niet compleet uitgevoerd. Daarom voldoet het systeem nog niet aan deze voorwaarde.

- Is er beschikking tot relevante documentatie voor programmeurs om het systeem te begrijpen?

De documentatie van het huidig systeem met daarbij ook de ontwerpen zijn niet door Netwinst vrijgegeven aan Perflectie. Daarom voldoet het systeem niet aan deze voorwaarden.

- Kan het systeem automatisch gebuild worden zodat het gemakkelijk is om de code te compileren?

Ja, omdat het systeem met behulp van visual studio is opgezet kan er binnen visual studio gemakkelijk de code compileren en builden naar een externe omgeving.

- Zijn er (geautomiseerde) tests aanwezig zodat het gemakkelijk is om bij veranderingen het systeem te valideren?

Er zijn geen tests aanwezig in het huidige online-platform project.

- Is er continue integratie binnen de code zodat builden en testen gemakkelijk gemaakt wordt?

Er zijn test omgevingen opgezet, deze bevinden zich echter bij Netwinst. Hierdoor duurde het meestal een dag om het project in een live of test omgeving te builden en testen.

- Wordt version control binnen het systeem gebruikt zodat ieder teamlid de laatste versie heeft van de code, testen en documentatie?

Ja, via SVN en Ankh had iedere teamlid de laatste versie van het project.

- Houdt of heeft ieder teamlid binnen het systeem de werkwijze aangehouden om zo onderhoudbaar mogelijk werk te produceren.

Nee, de teamleden die aan het huidige online-platform gewerkt heeft voldeden niet aan de eisen van een programmeur. Dit komt omdat zij niet in bezit waren van vaardigheden om een onderhoudbaar systeem neer te zetten.

Conclusie

Wat is de impact van de Partner module op de huidige online-platform en is dit verantwoordelijk?

Tijdens het analyseren van het huidig online-platform, wat gedocumenteerd staat in hoofdstuk Voldoet het huidige online-platform aan de resultaat aan voorwaarden in deelvraag 2?, ben ik al snel tot conclusie gekomen dat het een zwaar niet onderhoudbaar product is.

Er is heel wat mis met het online-platform wat huidig bestaat voor Perflectie. Het huidige online-platform voldoet aan geen één van de onderzochte voorwaarden voor een onderhoudbaar systeem.(Zie hoofdstuk Voldoet het huidige online-platform aan de resultaat aan voorwaarden in deelvraag 2?) Daarom raadt ik aan om niet verder te ontwikkelen binnen het huidige online-platform. Code smells en bad architecture zorgt ervoor dat het platform onoverzichtelijk is en ook niet meer te onderhouden is.

Mocht Perflectie wel aangeven om verder te gaan op het huidige platform dan geeft dit de moeilijkheid dat er een moeizaam proces bij komt van het vinden en oplossen van functionaliteiten, bugs etc. Daarom geef ik het advies om de partner module te ontwikkelen binnen een nieuw project, dit omdat het proces sneller is om de partner module te realiseren.

Mocht al mijn raad meegenomen worden om een nieuw project op te starten geeft dit het voordeel dat de voorwaarden, binnen hoofdstuk Voldoet het huidige online-platform aan de resultaat aan voorwaarden in deelvraag 2?, meegenomen kunnen worden.

E : Concept architecture notebook

Het softwarepakket Perflectie van het gelijknamige bedrijf helpt werkgevers hun werknemers te stimuleren in het informele leren en borgt nieuw gedrag aantoonbaar in de praktijk.

Voor Perflectie moet er een Partner module ontwikkeld worden, deze module zorgt er voor dat een Partner de omgeving van zijn online-platform naar zijn eigen wens kan aanpassen. Mijn initiële opdracht was het verder ontwikkelen op het bestaande platform. Na uitgebreid onderzoek kwam ik tot de conclusie dat het huidige platform onvoldoende geschikt is om op door te ontwikkelen. Om de gewenste functionaliteiten voor klanten beschikbaar te maken en om er voor te zorgen dat het platform in de toekomst uitbreidbaar blijft is er voor gekozen het platform opnieuw op te zetten met inzet van herbruikbare bestaande elementen uit het huidige systeem.

Dit alles vraagt om een duidelijk inzichtelijk systeem dat gemakkelijk onderhoudbaar is, maar ook dat dynamisch uitbreidbaar is.

Architectural Significant Requirements

Functional Requirements

De volgende functional requirements zijn van invloed op de architectuur:

Regelnummer	Beschrijving
FR 1	De mogelijkheid tot het maken en Costumizen van een component
FR 2	Van verschillende componenten een programma kunnen maken
FR 3	De mogelijkheid van toewijzen van meerdere rollen van een User
FR 4	De rol van een User bepaald waar hij wel en niet toegang tot heeft binnen het systeem
FR 5	De mogelijkheid van een User om een programma van een Partner te kiezen
FR 6	De mogelijkheid van een Partner om voorgedefinieerde componenten of eigen componenten te gebruiken.
FR 7	Een partner mag pas van een andere partner zijn componenten gebruiken als dit toegestaan is door de eigenaar van het component
FR 8	Een User die de rol van Programmabeheer is toegewezen kan een programma van een Partner CRUD. (Create,Read, Update, Delete)
FR 9	Een User die de rol van Partnerbeheer is toegewezen kan een Partner van het systeem CRUD. (Create, Read, Update, Delete)

Non-functional requirements

De volgende non-functional requirements zijn van invloed op de architectuur:

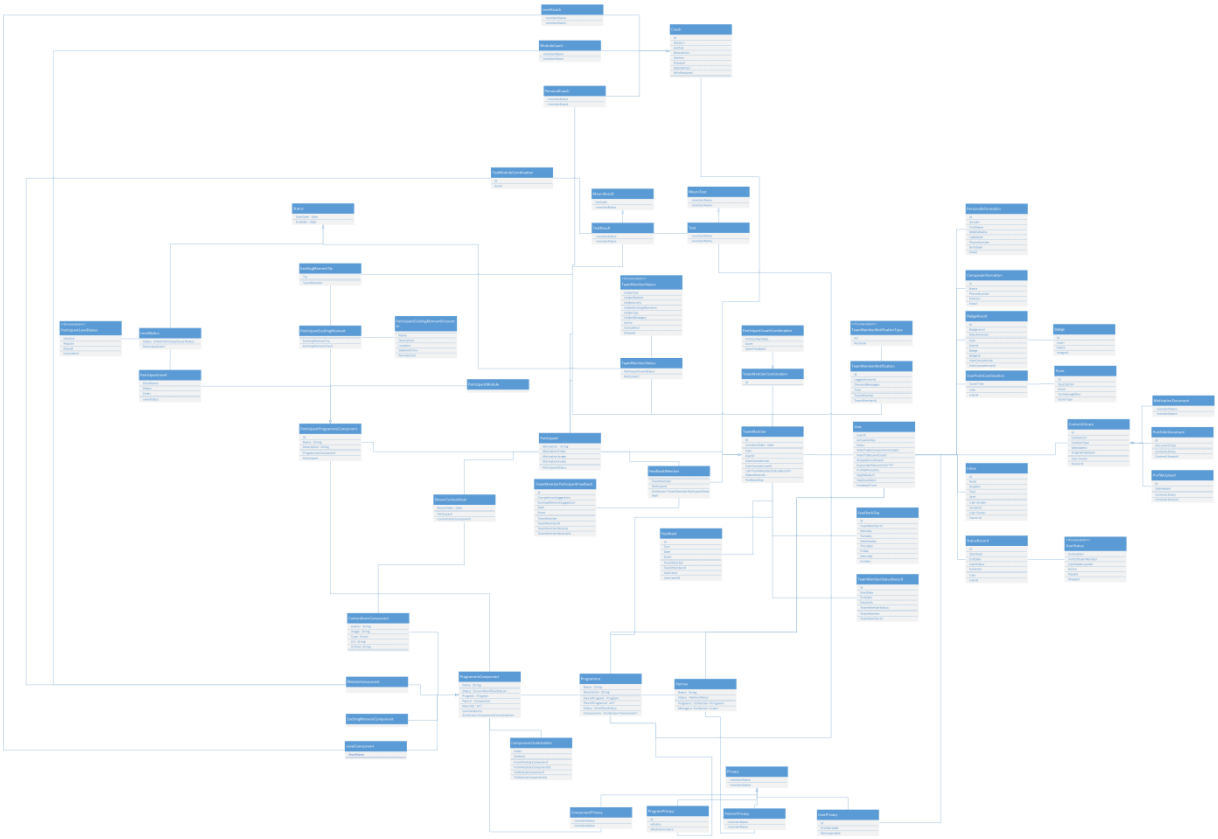
Regelnummer	Beschrijving	Type
NFR 1	Bij het maken en customizen van programma componenten zal de gebruiker met verschillende mogelijkheden, authorizaties en restricties te maken hebben. Voorbeeld: Gebruikers van een partner kunnen alleen de samengestelde partner programma's kiezen.	Functionaliteit/Juistheid

NFR 2	Het systeem zal in eerste instantie uit verschillende componenten en lagen bestaan, het moet echter de volgende aanpassingen kunnen doen: - Doordat perflectie voorgedefineerde programma's kan maken moeten deze gebruikt maar niet aangepast kunnen worden door een partner. Wel kan een partner dit programma uitbreiden met zijn eigen componenten - Het systeem moet in staat zijn verschillende soorten componenten aan te kunnen bieden. - Het systeem moet gemakkelijk uitbreidbaar zijn om verschillende soorten componenten toe te voegen en aan te kunnen bieden binnen het platform	Onderhoudbaarheid, uitbreidbaarheid en Portabiliteit
NFR 3	Omdat er gebruik zal worden gemaakt van AngularJS moet de aangegeven data beschikbaar zijn via een Controller of via een APIController.	Functionaliteit
NFR 4	De back-end zal ontwikkeld worden met de .NET MVC 5 C# van Visual Studio en de SOLID principe. Dit houdt in dat er lously coupled wordt tussen classes en voor elk functionaliteit zullen we test driven werken. Hiervoor maken we gebruik van Dependency Injection en Unit Testing voor elk business rule gedefinieerd door de product owner.	Constraint
NFR 5	Er zal code-first gewerkt worden met de EntityFramework waardoor een database structuur automatisch genereert worden.	Functionaliteit
NFR 6	Het systeem moet gebruik maken van de business standards en patterns	Functionaliteit/Juistheid
NFR 7	Naast de geleverde Microsoft authenticatie zal er ook rollen gebaseerd worden gewerkt binnen de code en classe definitie.	Beveiliging
NFR 8	Het systeem moest juist gedocumenteerd worden	Documentatie
NFR 9	Test driven development	Functionaliteit/Juistheid
NFR 10	Het systeem zal gebouwd worden in een MVC 5 omgeving van Visual Studio 2012+	Functionaliteit
NFR 11	Het gebruik van Dependency Injection binnnen het MVC 5 project	Juistheid
NFR 12	Het gebruik van AngularJS voor de Client-side	Functionaliteit
NFR 13	AutoMapper om van het model automatisch een viewmodel te maken	Functionaliteit / Juistheid

Onderbouwing van de Requirements

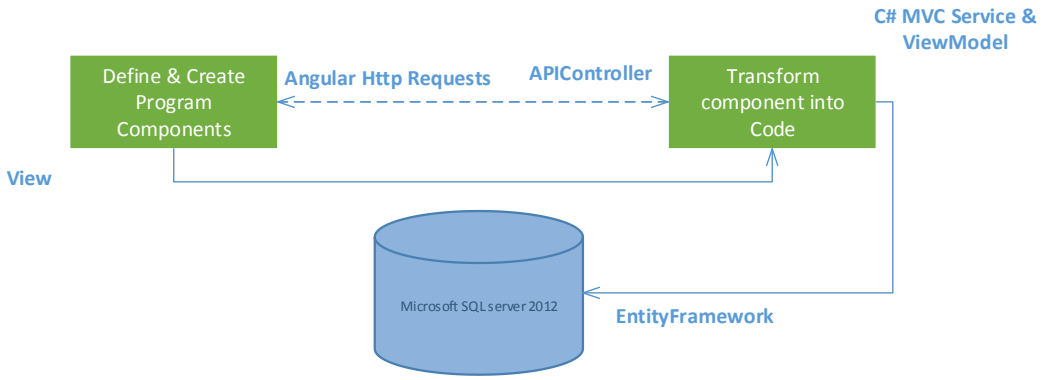
Beslissing	Requirement
Er moeten autorisatie niveaus toegepast worden om de juiste content en functionaliteitenbeschikbaar te stellen aan de gebruiker.	NFR 1
Wij hebben gekozen voor een generieke program component factory om zo het systeem uitbreidbaar en zo dynamisch mogelijk te maken. De partner kan op deze manier up to date blijven met de nieuwste ontwikkelingen en veranderingen van Perflectie maar kan ook zelf zijn componenten definiëren.	NFR 2
Omdat er verschillende manieren zijn hoe C# zijn routing en data oplost en AngularJS moet er vooraf afgesproken worden hoe data beschikbaar moet worden gesteld. De communicatie kan via een Controller of via een APIController verlopen.	NFR 3
Een nieuwe werkwijze zal toegepast worden om de integriteit van de code te behouden. Dit door toekomstige problemen, waar nu het huidige systeem mee kampt, niet weer te laten gebeuren. Dit zal zekerheid geven, minder bugs en het systeem zal gemakkelijker uitbreidbaar zijn.	NFR 4
Gemakkelijker werken binnen het MVC project omdat een grote taak voor je wordt geautomatiseerd.	NFR 5, NFR 6
Autorisatie moet op meerder niveaus binnen het systeem geborgd zijn.	NFR 7
Documentatie moet bestaan zodat de product owener en/of programmeurs gemakkelijk in het systeem kunnen kijken en oppakken om het verder te ontwikkelen.	NFR 8
Voor elk business rule waar een bepaald Service zich aan moet houden zal door de product owner worden genoteerd, deze zal dan doorgevoerd worden als Unit Test waarna geprogrammeerd kan worden.	NFR 9

Business Domain Model

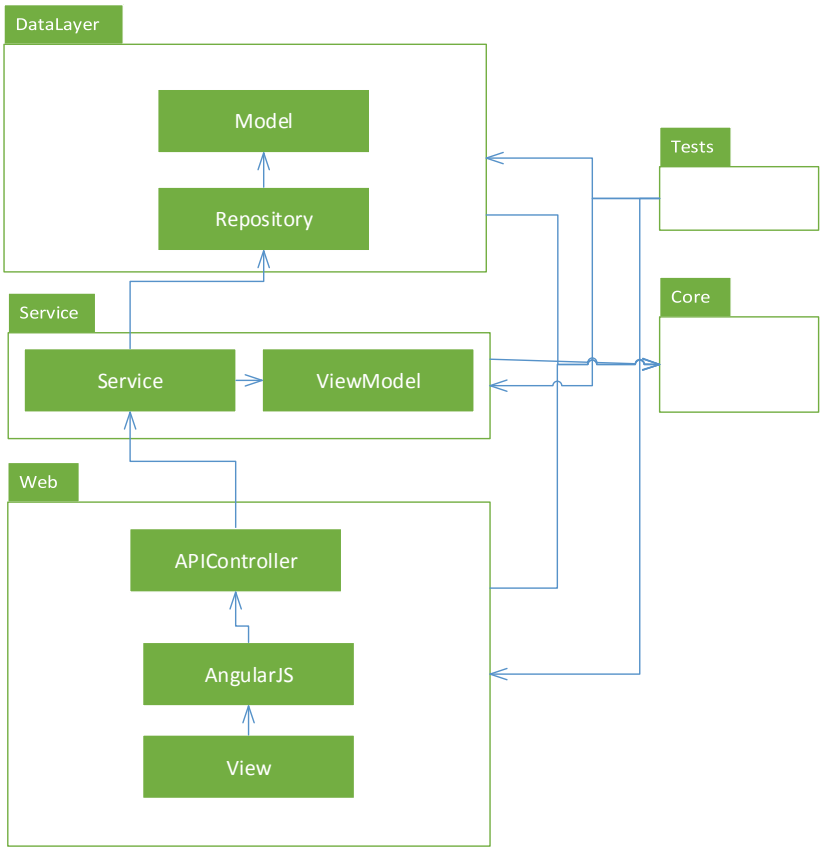


Software Partitioning

Logical component model



Logical Layers

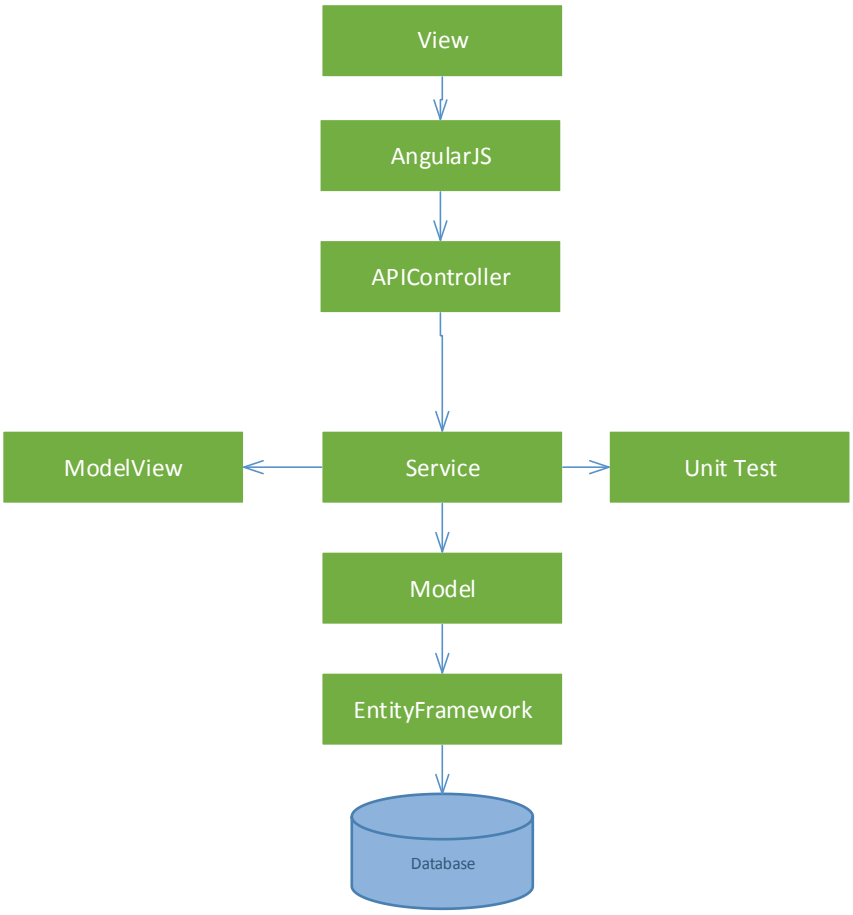


3 – Tier Architecture

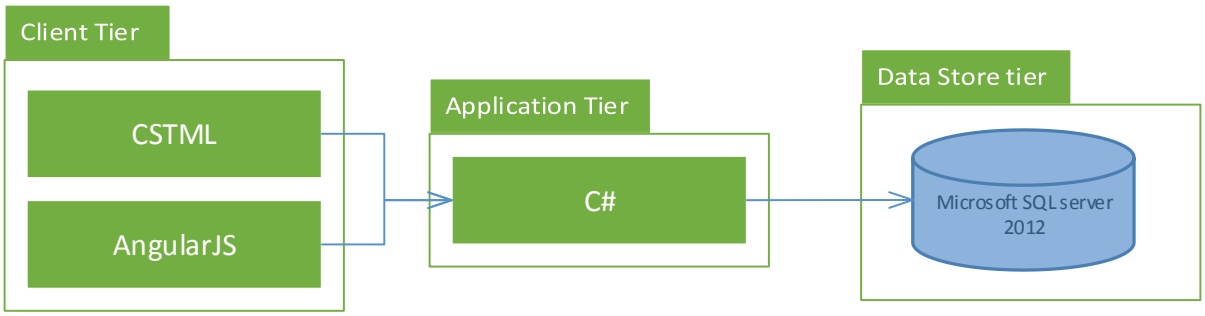
De C# MVC applicatie wordt in de volgende lagenstructuur verdeeld:

Laag	Inhoud
Web	1. Bestaat uit het eenvoudig tonen van de views 2. Client side interaction met AngularJS
Service	1. Hierin zit de controller en services van de Domain gespecifieerde classes 2. Verantwoordelijk voor het doorverwijzen van de routing van de gebruiker naar zijn gewenste view 3. Verantwoordelijk voor het juist afhandelen van een request tussen een controller en business domain 4. Op services zullen Unit tests worden gedefinieerd 5. Unit tests zullen op de service losgelaten worden waarbij de integriteit en de regels van het systeem waaraan hij zich moet houden worden gecheckt.
DataLayer	1. Hierin staan de business domeinklassen die benodigd zijn voor het realiseren van het project. 2. Implementatie van EntityFramework met zijn Context. Deze laag bestaat uit de communicatie met de database van Microsoft SQL en de business domain.

Fysieke Componenten

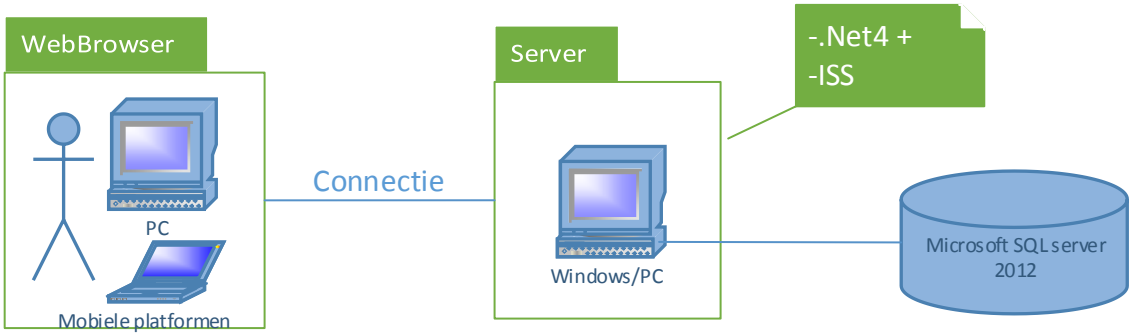


Tier Model



Deployment model

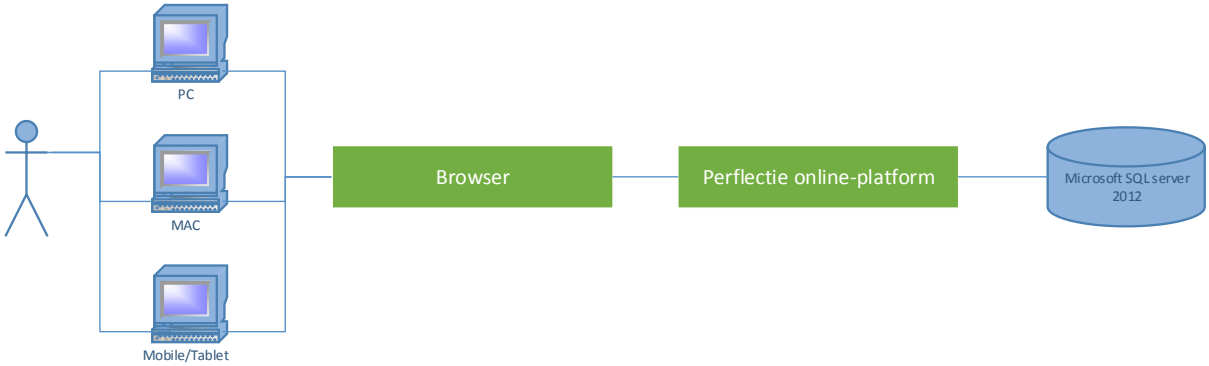
Dit is een voorbeeld hoe de omgeving eruit kan zien waarop de applicatie moet gaan draaien.



De Windows computer kan niet vervangen worden door elke andere computer, bijvoorbeeld Mac of Linux. Het is van belang dat op de computer Microsoft SQL server 2012+ draait omdat anders de webapplicatie geen gegevens kan opslaan.

Use Model

Dit is een voorbeeld hoe de omgeving eruit kan zien waarop de applicatie gebruikt gaat worden.



Er moet rekening mee gehouden worden dat een gebruiker met elk mogelijk elektronisch apparaat met een browser inlogt in de Perflectie online-platform. Het is van belang dat de gebruiker het platform geheel kan gebruiken en overzien.

F : Nieuwe architecture notebook

Het softwarepakket Perflectie van het gelijknamige bedrijf helpt werkgevers hun werknemers te stimuleren in het informele leren en borgt nieuw gedrag aantoonbaar in de praktijk.

Voor Perflectie moet er een Partner module ontwikkeld worden, deze module zorgt er voor dat een Partner de omgeving van zijn online-platform naar zijn eigen wens kan aanpassen. Mijn initiële opdracht was het verder ontwikkelen op het bestaande platform. Na uitgebreid onderzoek kwam ik tot de conclusie dat het huidige platform onvoldoende geschikt is om op door te ontwikkelen. Om de gewenste functionaliteiten voor klanten

beschikbaar te maken en om er voor te zorgen dat het platform in de toekomst uitbreidbaar blijft is er voor gekozen het platform opnieuw op te zetten met inzet van herbruikbare bestaande elementen uit het huidige systeem.

Dit alles vraagt om een duidelijk inzichtelijk systeem dat gemakkelijk onderhoudbaar is, maar ook dat dynamisch uitbreidbaar is.

Architectural Significant Requirements

Functional Requirements

De volgende functional requirements zijn van invloed op de architectuur:

Regelnummer	Beschrijving
FR 1	De mogelijkheid tot het maken en Costumizen van een component
FR 2	Van verschillende componenten een programma kunnen maken
FR 3	De mogelijkheid van toewijzen van meerdere rollen van een User
FR 4	De rol van een User bepaald waar hij wel en niet toegang tot heeft binnen het systeem
FR 5	De mogelijkheid van een User om een programma van een Partner te kiezen
FR 6	De mogelijkheid van een Partner om voorgedefinieerde componenten of eigen componenten te gebruiken.
FR 7	Een partner mag pas van een andere partner zijn componenten gebruiken als dit toegestaan is door de eigenaar van het component
FR 8	Een User die de rol van Programmabeheer is toegewezen kan een programma van een Partner CRUD. (Create,Read, Update, Delete)
FR 9	Een User die de rol van Partnerbeheer is toegewezen kan een Partner van het systeem CRUD. (Create, Read, Update, Delete)

Non-functional requirements

De volgende non-functional requirements zijn van invloed op de architectuur:

Regelnummer	Beschrijving	Type
NFR 1	Bij het maken en customizen van programma componenten zal de gebruiker met verschillende mogelijkheden, authorizaties en restricties te maken hebben. Voorbeeld: Gebruikers van een partner kunnen alleen de samengestelde partner programma's kiezen.	Functionaliteit/Juistheid
NFR 2	Het systeem zal in eerste instantie uit verschillende componenten en lagen bestaan, het moet echter de volgende aanpassingen kunnen doen: - Doordat perflectie voorgedefineerde programma's kan maken moeten deze gebruikt maar niet aangepast kunnen worden door een partner. Wel kan een	Onderhoudbaarheid, uitbreidbaarheid en Portabiliteit

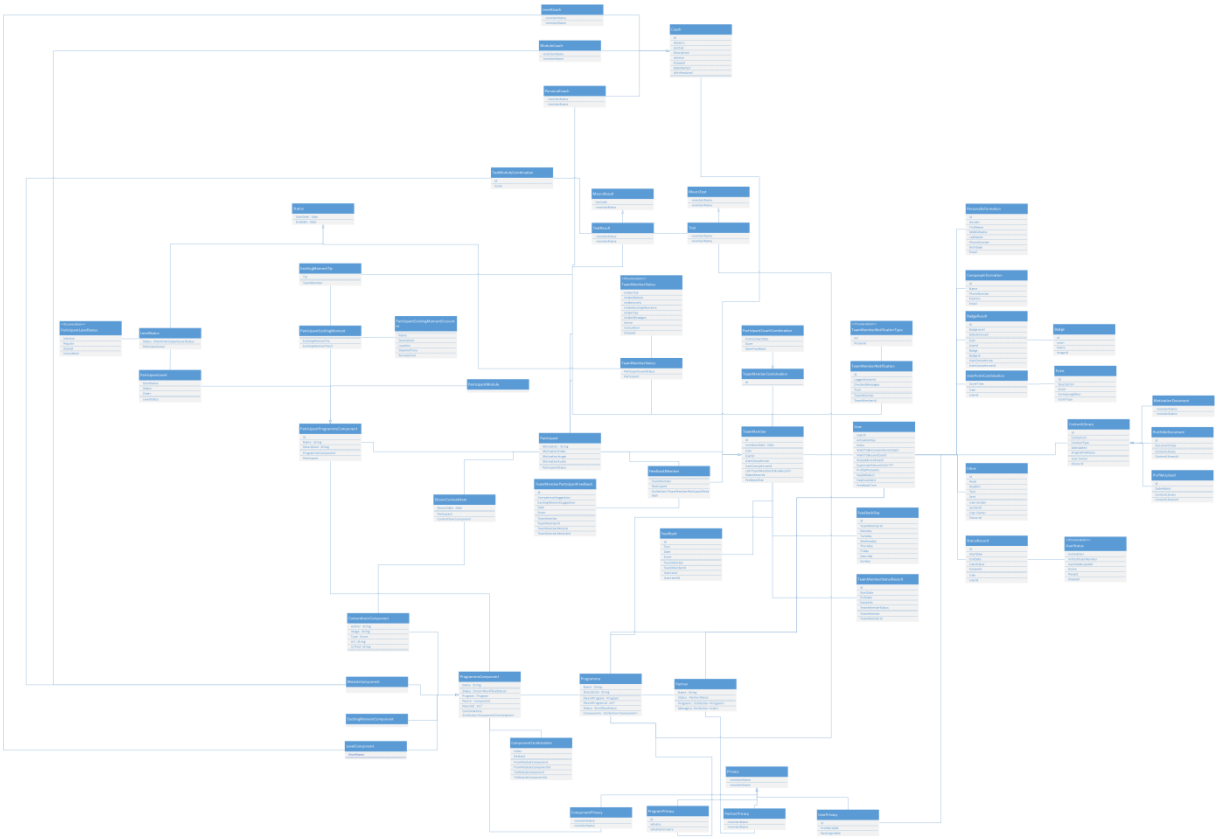
	partner dit programma uitbreiden met zijn eigen componenten - Het systeem moet in staat zijn verschillende soorten componenten aan te kunnen bieden. - Het systeem moet gemakkelijk uitbreidbaar zijn om verschillende soorten componenten toe te voegen en aan te kunnen bieden binnen het platform	
NFR 3	Omdat er gebruik zal worden gemaakt van AngularJS moet de aangegeven data beschikbaar zijn via een Controller of via een APIController.	Functionaliteit
NFR 4	De back-end zal ontwikkeld worden met de .NET MVC 5 C# van Visual Studio en de SOLID principe. Dit houdt in dat er lously coupled wordt tussen classes en voor elk functionaliteit zullen we test driven werken. Hiervoor maken we gebruik van Dependency Injection en Unit Testing voor elk business rule gedefinieerd door de product owner.	Constraint
NFR 5	Er zal code-first gewerkt worden met de EntityFramework waardoor een database structuur automatisch genereert worden.	Functionaliteit
NFR 6	Het systeem moet gebruik maken van de business standards en patterns	Functionaliteit/Juistheid
NFR 7	Naast de geleverde Microsoft authenticatie zal er ook rollen gebaseerd worden gewerkt binnen de code en classe definitie.	Beveiliging
NFR 8	Het systeem moest juist gedocumenteerd worden	Documentatie
NFR 9	Test driven development	Functionaliteit/Juistheid
NFR 10	Het systeem zal gebouwd worden in een MVC 5 omgeving van Visual Studio 2012+	Functionaliteit
NFR 11	Het gebruik van Dependency Injection binnnen het MVC 5 project	Juistheid
NFR 12	Het gebruik van AngularJS voor de Client-side	Functionaliteit
NFR 13	AutoMapper om van het model automatisch een viewmodel te maken	Functionaliteit / Juistheid

Onderbouwing van de Requirements

Beslissing	Requirement
Er moeten autorisatie niveaus toegepast worden om de juiste content en functionaliteitenbeschikbaar te stellen aan de gebruiker.	NFR 1
Wij hebben gekozen voor een generieke program component factory om zo het systeem uitbreidbaar en zo dynamisch mogelijk te maken. De partner kan op deze manier up to	NFR 2

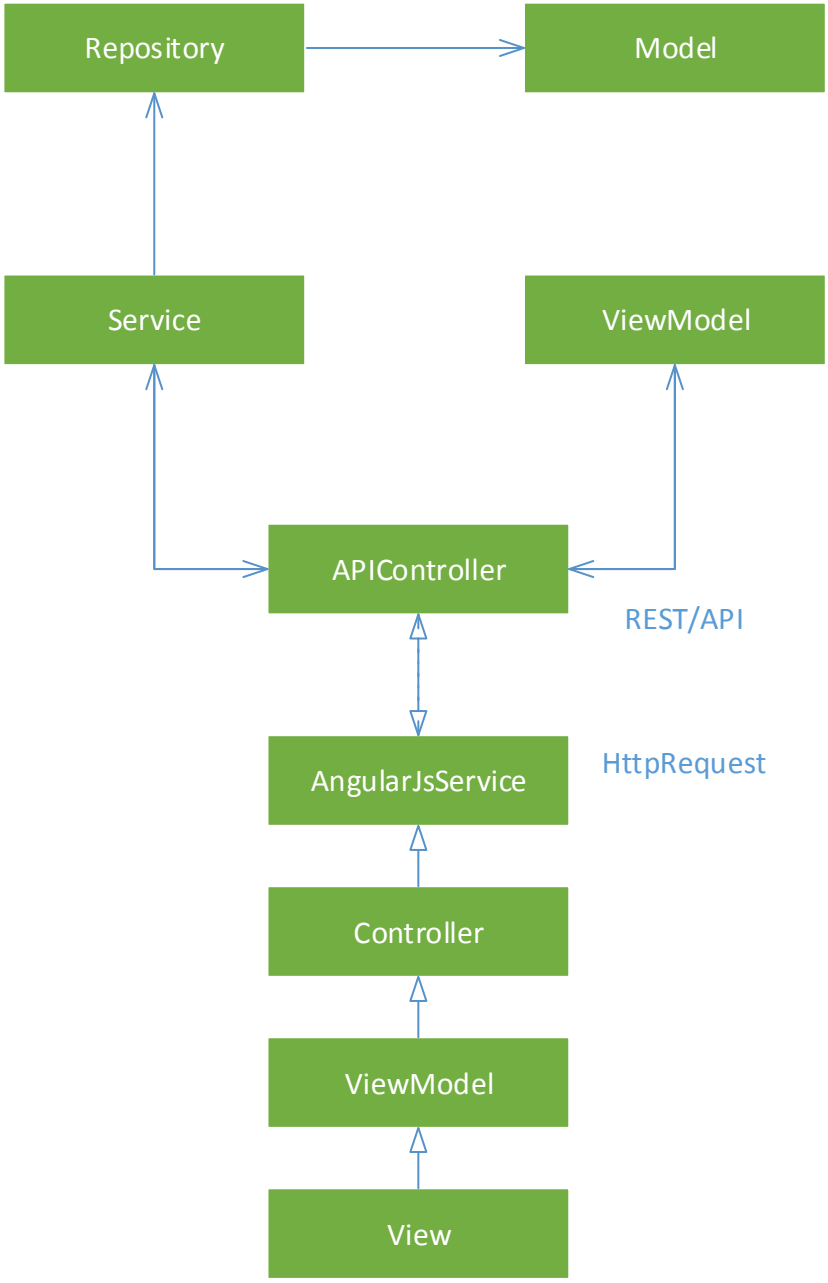
date blijven met de nieuwste ontwikkelingen en veranderingen van Perflectie maar kan ook zelf zijn componenten definiëren.	
Omdat er verschillende manieren zijn hoe C# zijn routing en data oplost en AngularJS moet er vooraf afgesproken worden hoe data beschikbaar moet worden gesteld. De communicatie kan via een Controller of via een APIController verlopen.	NFR 3
Een nieuwe werkwijze zal toegepast worden om de integriteit van de code te behouden. Dit door toekomstige problemen, waar nu het huidige systeem mee kampt, niet weer te laten gebeuren. Dit zal zekerheid geven, minder bugs en het systeem zal gemakkelijker uitbreidbaar zijn.	NFR 4
Gemakkelijker werken binnen het MVC project omdat een grote taak voor je wordt geautomatiseerd.	NFR 5, NFR 6
Autorisatie moet op meerder niveaus binnen het systeem geborgd zijn.	NFR 7
Documentatie moet bestaan zodat de product owener en/of programmeurs gemakkelijk in het systeem kunnen kijken en oppakken om het verder te ontwikkelen.	NFR 8
Voor elk business rule waar een bepaald Service zich aan moet houden zal door de product owner worden genoteerd, deze zal dan doorgevoerd worden als Unit Test waarna geprogrammeerd kan worden.	NFR 9

Business Domain Model

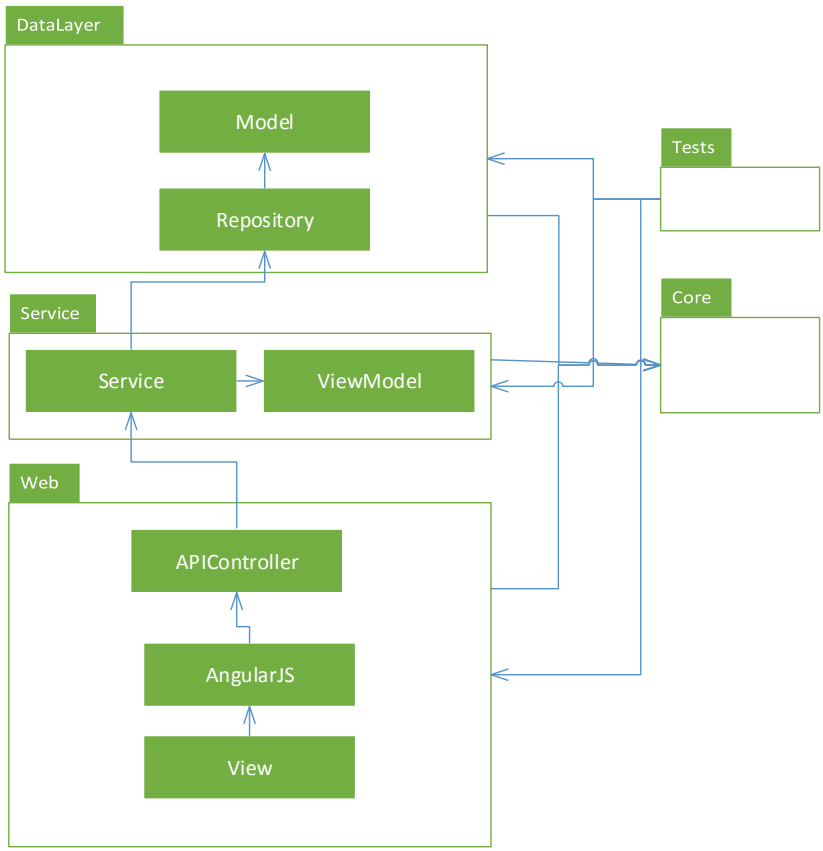


Software Partitioning

Logical component model



Logical Layers

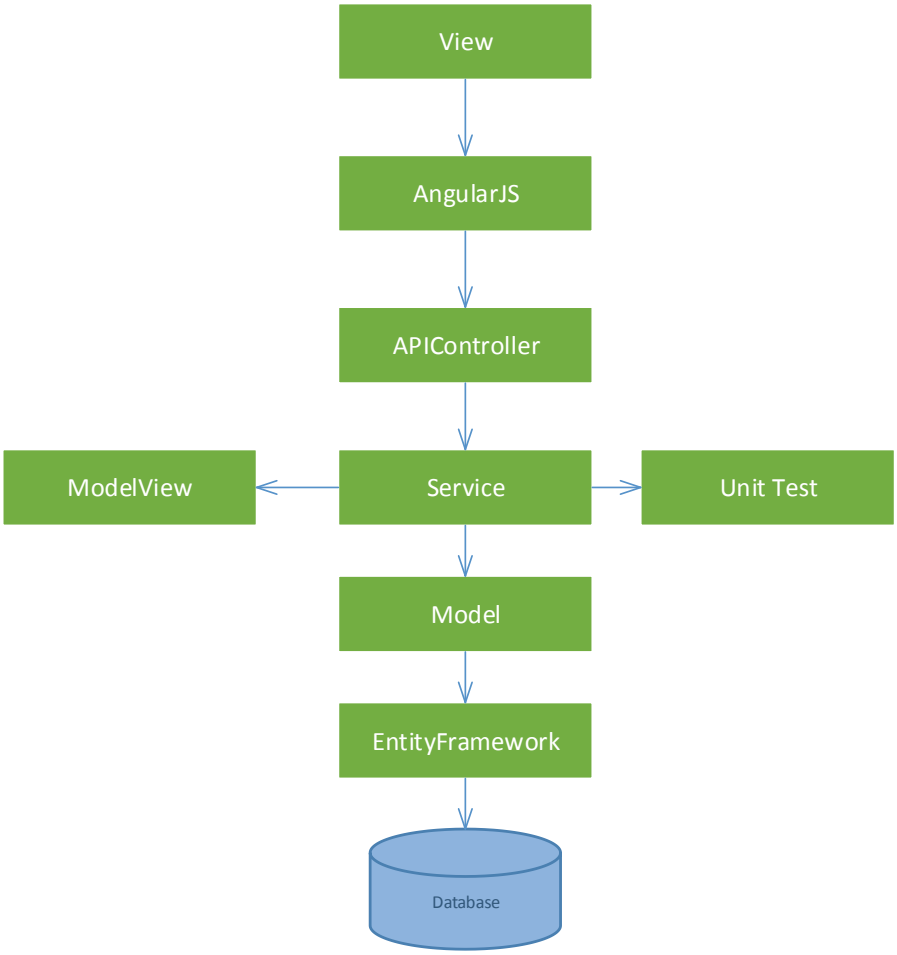


3 – Tier Architecture

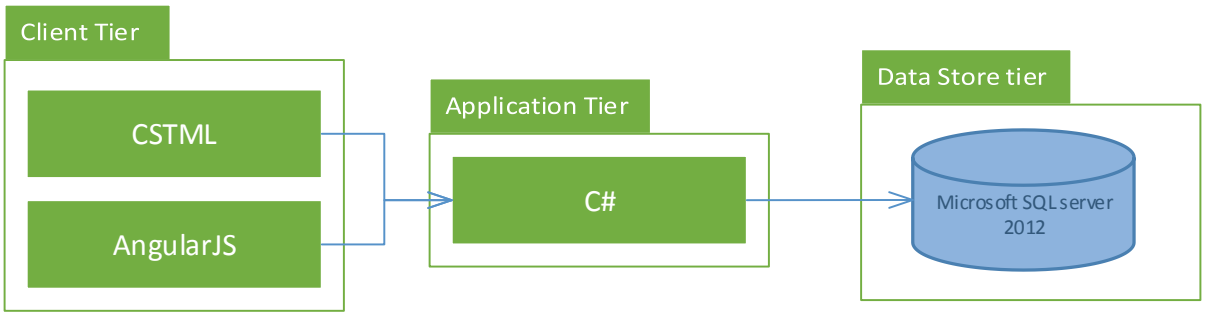
De C# MVC applicatie wordt in de volgende lagenstructuur verdeeld:

Laag	Inhoud
Web	3. Bestaat uit het eenvoudig tonen van de views 4. Client side interaction met AngularJS
Service	6. Hierin zit de controller en services van de Domain gespecifieerde classes 7. Verantwoordelijk voor het doorverwijzen van de routing van de gebruiker naar zijn gewenste view 8. Verantwoordelijk voor het juist afhandelen van een request tussen een controller en business domain 9. Op services zullen Unit tests worden gedefinieerd 10. Unit tests zullen op de service losgelaten worden waarbij de integriteit en de regels van het systeem waaraan hij zich moet houden worden gecheckt.
DataLayer	3. Hierin staan de business domeinklassen die benodigd zijn voor het realiseren van het project. 4. Implementatie van Entity Framework met zijn Context. Deze laag bestaat uit de communicatie met de database van Microsoft SQL en de business domain.

Fysieke Componenten

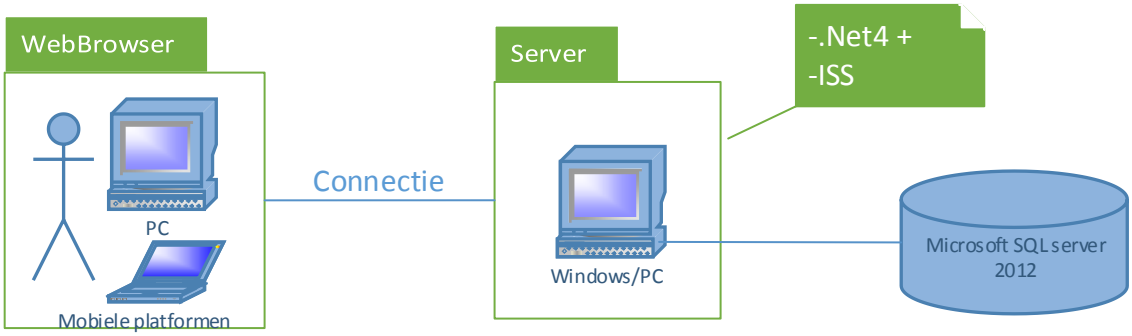


Tier Model



Deployment model

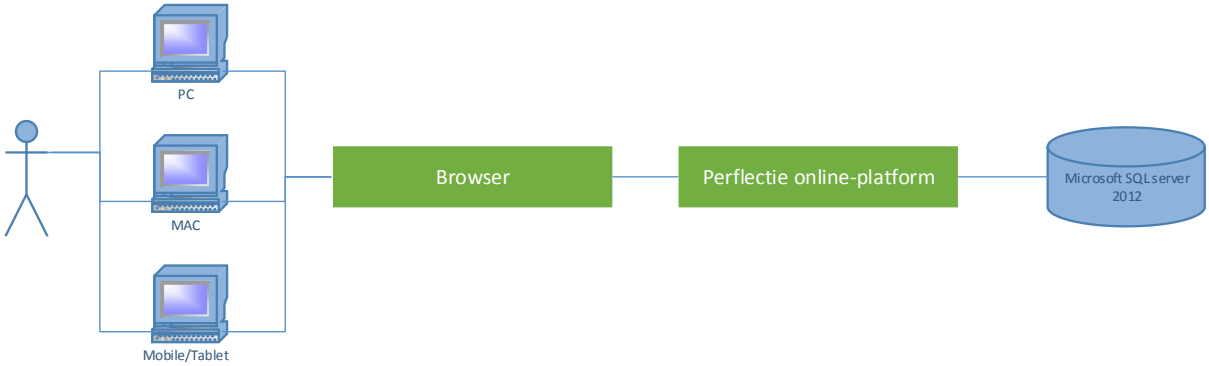
Dit is een voorbeeld hoe de omgeving eruit kan zien waarop de applicatie moet gaan draaien.



De Windows computer kan niet vervangen worden door elke ander computer, bijvoorbeeld Mac of Linux. Het is van belang dat op de computer Microsoft SQL server 2012+ draait omdat anders de webapplicatie geen gegevens kan opslaan.

Use Model

Dit is een voorbeeld hoe de omgeving eruit kan zien waarop de applicatie gebruikt gaat worden.



Er moet rekening mee gehouden worden dat een gebruiker met elk mogelijk elektronisch apparaat met een browser inlogt in de Perflectie online-platform. Het is van belang dat de gebruiker de platform geheel kan gebruiken en overzien.

G : Plan van aanpak

Perflectie heeft een methodiek voor gedragsverandering die wordt ondersteund door een online platform (web applicatie). Dit platform wordt momenteel gebruikt door klanten zoals Achmea, Independer, Reaal en Vodafone.

Om de groei van het aantal deelnemers in het platform te versnellen willen Jochem Aubel en Stefan Op de Woerd - de eigenaren van Perflectie - gaan werken met partners. Deze partners zijn bijvoorbeeld trainingsbureaus, coaches of consultants die medewerkers binnen bedrijven helpen met veranderprocessen. Het doel is dat dergelijke partners Perflectie kunnen inbedden in hun dienstverlening. Daarmee doen zij de feitelijke sales en implementatie van Perflectie, waardoor het business model schaalbaar wordt. Om dit interessant te maken voor partners moet de applicatie naadloos inpasbaar zijn in hun dienstverlening.

Uit het resultaat van het huidige online-platform blijkt dat de applicatie niet flexibel is. Feitelijk is er voor een partner niets instelbaar en is de partner dus niet in staat Perflectie zo in te richten dat het naadloos past binnen zijn dienstverlening. Om dit mogelijk te maken moet de applicatie flexibel gemaakt worden en moet er een beheeromgeving komen voor partners, waar zij de applicatie kunnen inrichten naar hun wensen.

Om verantwoordelijk te werk te gaan wordt als eerst een onderzoek gedaan naar de impact van een partner module op het huidige online-platform. Op basis van de resultaten van het onderzoek zal er een ontwerp gemaakt worden dat de nieuwe functionaliteiten van de Partner module dekt.

Probleemstelling

De initiële opdracht

Voor Perflectie moet er een Partner module ontwikkeld worden. Deze module zorgt ervoor dat een Partner de omgeving van zijn online-platform naar zijn eigen wens kan aanpassen. Mijn initiële opdracht was in het begin van mijn stage om de Partner module verder te ontwikkelen op het huidige online-platform van Perflectie.

De partner module houdt in dat voorgedefinieerde componenten toegevoegd kunnen worden door de Partner. Perflectie kan zelf ook de rol van Partner vervullen en moet ook in staat zijn gedefinieerde componenten te kiezen. Een Partner kan op zijn beurt weer een Perflectie component als basis gebruiken. Dit component kan de Partner personaliseren zonder het originele component aan te tasten. Wanneer Perflectie een verandering doorvoert naar dit component en de Partner heeft de attribuut niet gepersonaliseerd, dan krijgt hij de update doorgevoerd ook naar zijn omgeving. Zo heeft de partner altijd de beschikking over de laatste versie van het gekozen component.

Een partner kan gebruikers uitnodigen in zijn omgeving. Een gebruiker kan een programma volgen die de partner binnen zijn omgeving heeft gedefinieerd. De componenten binnen een programma moet door een gebruiker te volgen zijn en de gebruiker moet dit ook kunnen personaliseren.

Dit alles vraagt om een duidelijk inzichtelijk systeem dat gemakkelijk onderhoudbaar, maar ook uitbreidbaar is.

Het uitgangspunt van de opdracht, die ik in augustus 2014 heb gekregen van Jochem Aubel, luidde als volgt:

> *Ontwikkel voor Perflectie een Partner module in de omgeving van Perflectie zijn online-platform, met als doel dat een Partner de omgeving van zijn platform naar eigen wens kan aanpassen.*

Analyse van het huidige online-platform

Voordat ik aan de opdracht ben begonnen heb ik als eerst samen met Jochem Aubel afgesproken om vooraf een onderzoek te doen naar het huidige online-platform, waarin de volgende onderzoeksvraag centraal stond:

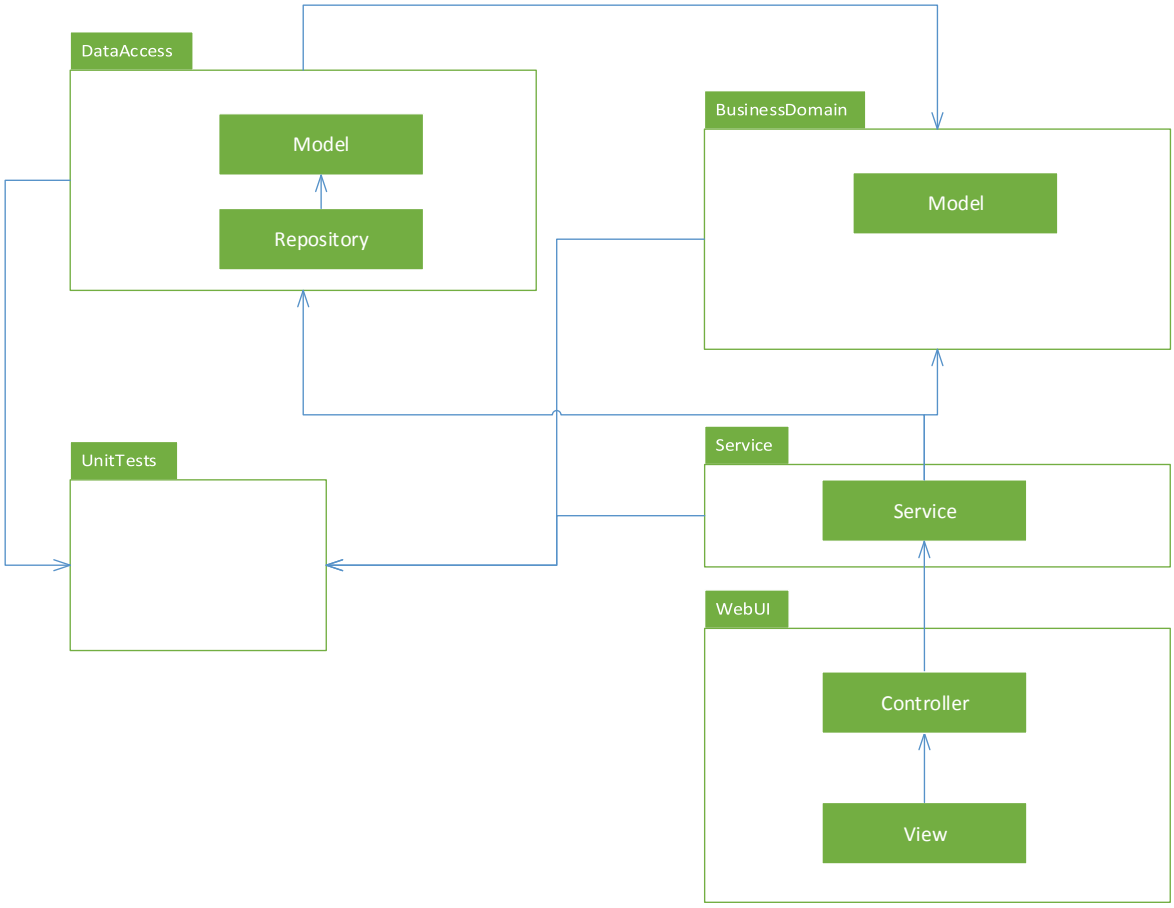
> *Wat is de impact van de partner module op de huidige online-platform en is dit verantwoordelijk?*

Deelonderzoeksvragen afgeleid van de hoofdonderzoeksvraag:

6. *Wat maakt een systeem onderhoudbaar en uitbreidbaar?*
7. *Aan welke voorwaarden moet een onderhoudbaar en uitbreidbaar systeem voldoen?*
8. *Wat is het probleem/wat zijn de problemen binnen het huidige online-platform van Perflectie?*
9. *Wat is het structuur binnen het huidige online-platform van Perflectie?*
10. *Voldoet het huidige online-platform aan de resultaat aan voorwaarden in deelvraag 2?*

Uit dit onderzoek (Zie bijlage Vooronderzoek) blijkt dat het huidige systeem van Perflectie, vanwege zijn grote omvang, op onderdelen niet te onderhouden is of er ongewenste afhankelijkheden voordoen. Dit heeft geresulteerd in een niet onderhoudbaar en niet uitbreidbaar systeem. Na een gesprek met Jochem Aubel zijn wij tot de conclusie gekomen om het systeem geheel opnieuw op te zetten, maar daarbij zoveel mogelijk functionaliteiten te gebruiken uit het oude systeem.

Om er echter voor te zorgen dat het nieuwe systeem niet in dezelfde baan zal lopen als het huidige online-platform zullen wij de bevindingen van het vooronderzoek (Zie bijlage Vooronderzoek) meenemen binnen het nieuwe project. Deze bevindingen bestaan uit een lijst van voorwaarden waar het systeem aan moet voldoen en vragen wat een programmeur aan zichzelf en aan andere kan stellen.



Figuur 3 Architectuur van het huidige online-platform

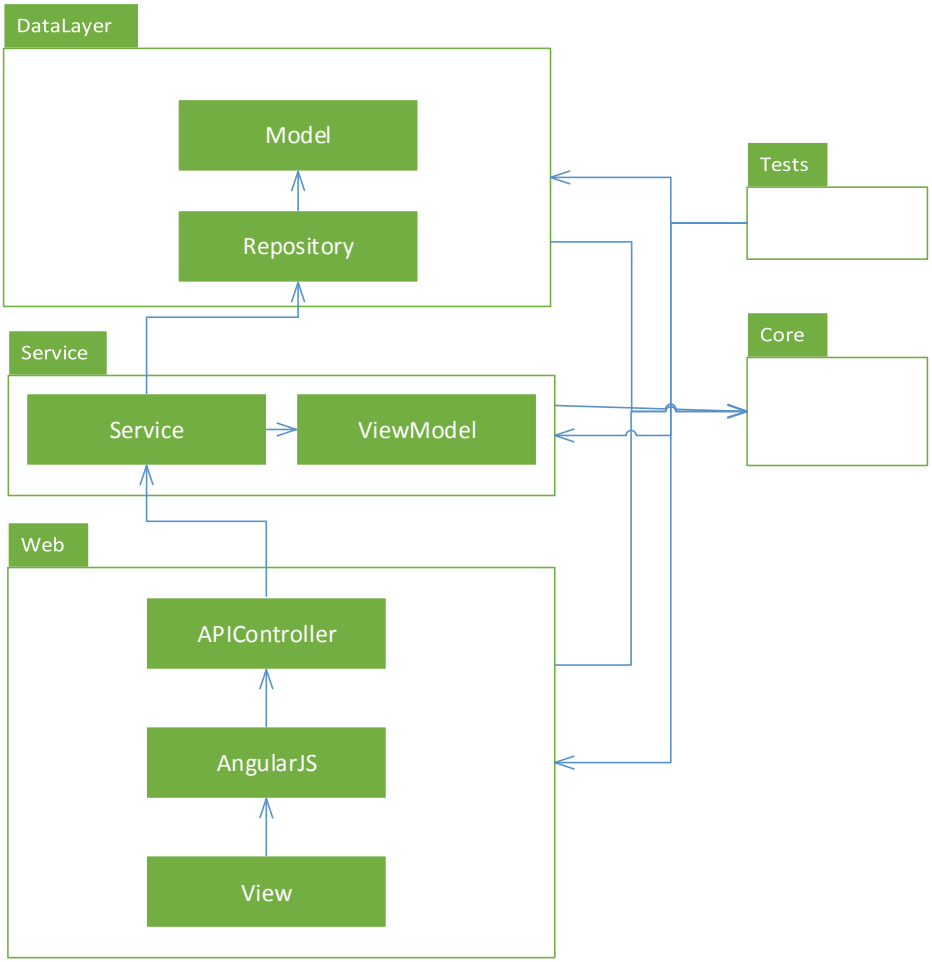
Conceptueel model

Aan het eind van dit project (2 december, 2014) zal er een nieuw ontwerp gemaakt en gerealiseerd zijn voor het nieuwe online-platform van Perflectie, waarbij het nieuwe systeem voldoet aan de eisen van de opdrachtgever, Perflectie en aan de voorwaarden beschreven in het Vooronderzoek. (Zie bijlage Vooronderzoek) De doelstellingen luiden als volgt:

Eisen

1. Het systeem is onderhoudbaar
2. Het systeem is uitbreidbaar
3. Het systeem houdt zich aan het ontwerp beschreven in de volgende documenten:
   a. Architecture notebook
   b. Technisch- en functioneel ontwerp
4. Functionaliteiten van het huidige online-platform van Perflectie
5. Functionaliteiten van de Partner module

Concept architectuur



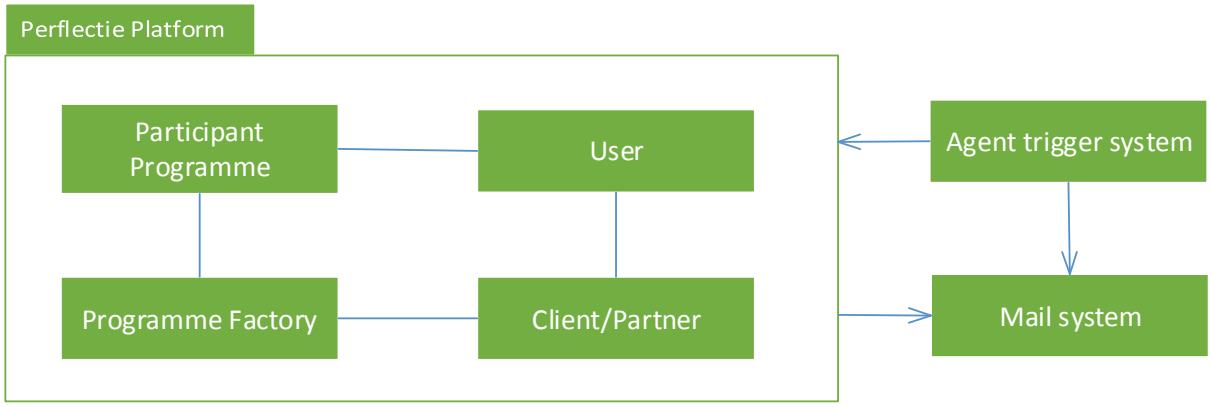
Figuur 4 Tier Architecture concept

Figuur 4 is een representatie van een nieuw architectuur voor het nieuwe online-platform, dit ter verbetering van de architectuur van het oude platform, zie figuur 3. Dit architectuur kan verandert worden gedurende het project, dit omdat er iteratief te werk gaat via de scrummethode.

Testen
Om de eisen en daarmee de kwaliteit te waarborgen zullen wij het systeem met behulp van TDD (Test Driven Development) testen. Hieruit kan afgeleid worden of het systeem aan de vastgestelde eisen voldoet.

Onderdelen van het nieuwe online-platform van Perflectie
Na een gesprek met Jochem Aubel zal het nieuwe online-platform voor 2 december moeten bestaan uit de volgende componenten:

- De programme factory
- Client/Partner
- Gebruikers
- Participants & feedbackmembers
- Mail systeem
- Agent trigger systeem



Figuur 5 Concept componenten voor het nieuwe platform

Deze onderdelen zullen zowel front-end, back-end als backoffice ontwikkeld worden en daarna in samenhang worden gebracht. Dit zal op een iteratieve wijze gebeuren en zo bij iedere iteratie verder gedetailleerd ontwikkeld worden.

Omdat na het vooronderzoek de opdracht enigszins is veranderd hebben wij hierop het uitgangspunt aangepast. Deze luidt nu als volgt:

> *Ontwerp en herbouw het Perflectie online-platform met de partner module inbegrepen, zodat het nieuwe systeem onderhoudbaar en uitbreidbaar is, voor 2 december.*

De voorwaarden die hieronder staan genoteerd komen overeen met de voorwaarden van het vooronderzoek (Zie bijlage Vooronderzoek). Door antwoord te bieden op iedere voorwaarden kan er zo op een correcte wijze een onderhoudbaar en uitbreidbaar systeem gebouwd worden.

Om het ontwerp van het nieuwe Perflectie online-platform zo onderhoudbaar mogelijk op te stellen, zullen wij dit vooraf het project doen. Om het ontwerp begrijpelijk te maken zullen wij de UML (Unified Model Language) standaard aanhouden. Dit komt overeen met de voorwaarde:

> Ontwerp het systeem voor onderhoudbaarheid van het begin af

Om de ontwikkeling van het nieuwe online-platform van Perflectie in goede banen te leiden zullen wij op een iteratieve methode het project managen en het product ontwikkelen. (Zie Aanpak) Dit bedekt de voorwaarde:

> Het systeem op een iteratieve methode managen/ontwikkelen.

Binnen Agile Scrum (Zie Aanpak & Bijlage Agile SCRUM) zijn er meerdere methodes om het project regulier te reviewen en de kwaliteit van het systeem te waarborgen. Dit bedekt de voorwaarde:

Reguliere reviews om de kwaliteit van het systeem te waarborgen

Binnen het scrumteam (Zie Aanpak & Bijlage Agile SCRUM) zal de werkwijze aangepast moeten worden om onderhoudbaar te werk te gaan. Deze werkwijze zal besproken worden binnen het scrumteam en zal meegenomen worden in de definition of done. (

Definition of done) Wanneer de werkwijze juist wordt gevolgd zal de volgende voorwaarden bedekt worden:

8. Leesbare code dat makkelijk te begrijpen is
9. Refactor de code om de begrijpbaarheid te verbeteren
10. Relevante documentatie dat programmeurs helpt om het systeem te begrijpen
11. Geautomatiseerde builds van het systeem zorgt voor gemakkelijk compileren van code
12. Continue integratie maakt de code gemakkelijk te builden en te testen
13. Version control helpt om de code, testen en documentatie up to date te houden binnen het projectteam
14. Verander de manier van werken, met als doel onderhoudbaarheid

Om de kwaliteit te waarborgen en om het systeem continu te kunnen verifiëren zal binnen het project Test Driven Development (TDD) gewerkt worden. Er zijn meerdere manieren om deze methode te volgen, maar uiteindelijk zorgt het voor een dezelfde doel. Een systeem dat onder test staat. Dit bedekt de volgende voorwaarde:

Geautomatiseerde tests zorgen voor gemakkelijke validaties bij veranderingen binnen het systeem

De hoofdvraag met zijn deelvragen

Hoofdvraag productsopdracht

> *"Hoe moet het nieuwe online-platform voor Perflectie zo ontworpen zijn dat, met de partner module inbegrepen, het systeem onderhoudbaar en uitbreidbaar is? "*

Deelvragen

1. *Wat maakt een systeem onderhoudbaar en uitbreidbaar en aan welke voorwaarden moet het systeem voldoen?*
2. *Wat zijn de eisen van Perflectie voor het nieuwe online-platform?*
3. *Welke structuur of structuren zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*
4. *Welke design oplossingen zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1 en eisen in deelvraag 2?*

De op te leveren producten

Product	Omschrijving
Literatuuronderzoek	Een onderzoek om de onderzoeksvragen te beantwoorden door middel van wetenschappelijke artikelen en het huidige online-platform.
Functioneel ontwerp	Een document waarin alle eisen en wensen staan genoteerd waar het online-platform aan moet voldoen.
Plan van aanpak	Een plan van aanpak ter verbetering van het online-platform
Technisch ontwerp	Een document van het technisch ontwerp van het project. Hierin is terug te lezen hoe het project is opgebouwd.
De programme factory	Voor het platform van perflectie zal er een factory gemaakt moeten worden. Deze factory zal de taak afhandelen zodat Partners en hun programma managers hun eigen programma's met daarbij modules kan beheren.

	Deelnemers kunnen, als zij een uitnodiging hebben ontvangen vanuit een programma, deelnemen aan een programma. Hier kunnen zij een module kiezen waar zij aan deel willen nemen en de volgorde in hoe zij deze programma willen volgen. Daarbij kunnen deelnemers feedbackmembers uitnodigen die feedback kunnen geven op de activiteiten van de deelnemer in zijn programma.
Client/Partner	Het platform van Perflectie zal ook Partners moeten kunnen beheren en op basis van de Partner zal er een omgeving binnen het online-platform gecreëerd worden. Deze omgeving kan door de Partner en zijn managers gepersonaliseerd worden.
Gebruikers(-beheer)	Binnen het platform moet er een gebruikerssysteem worden gebouwd waarbinnen rollen gedefinieerd kunnen worden. Deze rollen en daarbij zijn verantwoordelijkheden zal in het functioneel ontwerp verwerkt worden.
Participants & feedbackmembers	Als deelnemer heb je een eigen frontoffice waarbij je activiteiten uit kan voeren. Deelnemers kunnen, als zij een uitnodiging hebben ontvangen vanuit een programma, deelnemen aan een programma. Hier kunnen zij een module kiezen waar zij aan deel willen nemen en de volgorde in hoe zij deze programma willen volgen. Daarbij kunnen deelnemers feedbackmembers uitnodigen die feedback kunnen geven op de activiteiten van de deelnemer in zijn programma.
Mail systeem	Een generieke mail systeem waarbij gemakkelijk dynamische mailtemplates verstuurd kunnen worden op basis van activiteiten binnen het systeem.
Agent trigger systeem	Een generieke Agent waar triggers beheert kunnen worden. Deze triggers zorgen ervoor dat op basis van een activiteit een actie uitgevoerd kan worden wat makkelijk beheerbaar is door ook niet technische personeel van Perflectie.
Een nieuw online-platform voor Perflectie	-De verbetering van het online-platform. -Alle gedocumenteerde functionaliteiten. -Het structuur van het online-platform is zo opgebouwd dat het overeenkomt met wat er is vastgesteld in het technisch ontwerp en binnen de vooronderzoek. -Duidelijke ontwerpen dat gebruikt kunnen worden door programmeurs
Evaluatie online-platform	-User test document opstellen -Minimaal 20 kandidaten het online-platform laten gebruiken en het document in laten vullen -Feedback ontvangen
Aanpassing online-platform	-Feedback analyseren en toepassen op het prototype

Werkwijze

Aanpak

Voor het ontwikkelen van de partner module voor het online-platform van Perflectie zal in grote lijnen de volgende aanpak gebruikt worden om een taak of taken te voltooien:

Aanpak binnen het onderzoek

Het beantwoorden van de vragen binnen het onderzoek

Om het onderzoek succesvol te voltooien, in de gegeven tijd van zes weken beginnend met mijn stage, is de hoofdonderzoeksvraag onderverdeeld in deelvragen. De scope bestaande uit de deelvragen, zullen ieder met een aparte aanpak beantwoordt worden. Zo'n aanpak, dat varieert van het gebruik maken van voorgedefinieerde design- en architectuur patterns, wetenschappelijke artikelen etc., zullen ieder doelgericht de deelvraag juist beantwoorden.

Analyseer en beantwoord de volgende vragen op de volgende manier:

1. *Wat zijn de eisen voor het nieuwe online-platform?*
   a. *Dit zal beantwoordt worden door middel van vergaderen/interviews binnen iteraties*
2. *Welke structuur of structuren zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1?*
   a. *Dit zal beantwoordt worden door middel van een vergelijkingsonderzoek.*
3. *Welke design oplossingen zijn geschikt voor het nieuwe platform, wat voldoet aan de voorwaarden gedefinieerd in deelvraag 1?*
   a. *Dit zal beantwoordt worden door middel van een vergelijkingsonderzoek.*

Aanpak binnen het scrumteam

Om ervoor te zorgen dat het team het project binnen een juiste en geschikte methode binnen Perflectie afhandelt, is hiervoor een onderzoek verricht. Binnen dit onderzoek wordt er antwoordt gegeven welke projectmanagementmethode het beste bij de methodiek en principes van Perflectie past.

Binnen het meegeleverde onderzoeksresultaat (B : Management methode: Agile Scrum) staat beschreven welke managementmethode wij gaan gebruiken binnen het project. Alle voorstellen zijn goedgekeurd en zullen meegenomen worden tijdens het ontwikkelen van de partner-module.

Om Agile Scrum juist te kunnen uitvoeren binnen de organisatie hebben wij het volgende met elkaar afgesproken.

Project Constructie:

1. Selecteer of pak een taak wat in de sprint staat toegewezen; wanneer er geen taken meer zijn toegewezen zorg er dan voor dat je een taak oppakt die open staat en nog niet aan iemand is toegewezen.
2. Maak Unit tests van de business rules omschreven in de Iteration Plan van de Sprint waar de taak toe behoort.
3. Creëer indien nodig het model, repository, service, tests en controller voor de taak.
4. Creëer de Dependency injection voor de service voor zowel de test data als voor de real data.
5. Neem de bruikbare code over van het oude project naar het nieuwe project.

6. Pas de code aan of voeg code toe totdat elk unit test op groen staat. (Hou hier rekening mee met de regels die genoteerd zijn in hoofdstuk 8.4)
7. Wanneer de Unit Tests op groen staan en de taak is uitgevoerd kan de taak op Done worden gezet op de scrumboard.
8. Om de 2 weken de review bijwonen en nieuwe taken ontvangen voor de volgende sprint.
9. Mocht het nodig zijn, laat je code reviewen door een aanwezige programmeur.
10. Tijdens de review worden de afgeronde taken inclusief tests geanalyseerd door alle aanwezigen.

Project Documentatie:

1. Voor de start van een nieuwe sprint, zal de scope van de sprint gedocumenteerd worden wat beschreven staat in hoofdstuk 8.5 Als er vanuit de vorige sprint nog taken open staan zullen deze ook met de nieuwe sprint meegenomen worden.
2. Van de scope van de Sprint worden de Use cases en Use Stories genoteerd.
3. Aan het einde van een sprint zal in het document een evaluatie opgenomen worden met daarbij de backlog items en taken die volledig voltooid zijn.

Project Testing:

1. Tijdens het project zal er testdriven gewerkt worden. Dit houdt in dat continu Unit Tests worden gedraaid in de backend zodat de Business Rules worden gewaarborgt.
2. Ook zal er tijdens het ontwikkelen van het project, door Yvo Vogt en Jochem Aubel worden getest of de opgezette business rules en kwaliteitseisen worden overschreden.

Planning en Mijlpalen

Mijlpaal	Sprint	Datum
Kick off project werkwijze leren van Dependency Injection	1	08-09-2014 - 19-09-2014
Werkende "Fundering" voor het project	1	
Component factory af en werkende voorbeeld	1	
Partner Component Afgemaakt	2	22-09-2014 – 03-10-2014
User Component afgemaakt	2	
User Authorisatie en daarbij zijn kennen en kunnen afgemaakt	2	
Overige functionaliteiten toevoegen, als voorbeeld: Badges, score, test	3	06-10-2014 – 17-10-2014
	4	20-10-2014 – 31-10-2014
	5	03-11-2014 – 14-11-2014
	6	17-11-2014 – 28 -11 -2014
Werkend product dat live gezet is met daarbij de transactie van de oude database	7	01 -12 -2014 - 02 -12 -2014

Rollen en Personen

Tijdens dit project worden onderstaande rollen vervuld door de volgende personen.

Naam / Omschrijving	Toegekend aan:	Verantwoordelijkheden
Product eigenaar	Jochem Aubel Stefan Op de Woerd	Het maken van User Stories en het vaststellen van Business Rules. Door wie en wat moet het systeem kunnen?
Scrumteam	Bas Alblas, Dimitri Tholen, Benno van Dijk, Anne Boon, Jesse Buitenhuis, Serge Bekenkamp	Realiseert het project door het coördineren van en/of aansturen van collega's door middel van de SCRUM methode. Houdt zich bezig met het programmeren van software.
Specialist	Dimitri Tholen	Deelt kennis en installeert de structuur binnen het project.
Architect	Bas Alblas	Ontwerpt de structuur binnen het softwaresysteem.
Analist	Bas Alblas	Analyseert de flows binnen het project, en speurt naar verbeteringen.
Project Administrateur	Bas Alblas	Draagt zorg over de projectadministratie en houdt deze constant up to date.
Kwaliteitsbewaker	Yvo Vogt, Dimitri Tholen	Controleert of de systemen constant voldoen aan de vooraf gestelde normen.

Definition of done

MoScoW-methode

Must haves (zie 3.4 Conceptueel model)

- De programme factory
- Client/Partner
- Gebruikers
- Participants & feedbackmembers
- Mail systeem
- Agent trigger systeem
    - De mogelijkheid binnen het systeem om e-mails te versturen op basis van triggers dat afgehandeld wordt door een agent.
    - Een agent dat de database en code “schoon” houdt van ongewenste data en pas directe-input terugstuurt op basis van triggers binnen het online-platform van Perflectie
- Oude functionaliteiten van het huidige online-platform

Should haves

- Real-time feedback van het systeem naar de gebruiker

Could haves

- Het implementeren van de mogelijkheid om meerdere soorten persoonlijkheidstest voor een programma te kunnen kiezen.

Won’t haves

- De gebruiker kunnen op basis van een iteratieve wijze zijn competenties en levels ontwikkelen.

Werkwijze van een programmeur binnen het scrumteam

De werkwijze die wij onderling hebben afgesproken kan worden gediend als een checklist tijdens het programmeren. Er bestaat een checklist voor ieder vier fasen: Plan, red, green en refactor.

Plan

- Neem een taak vanuit de scrumboard op jezelf, als er geen eigenaar aan de taak is toegewezen.
- Bespreek binnen het scrumteam af of jou taak niet in de scope valt van een taak van iemand anders.
- Bekijk de iterationplan van het project om de user story en de requirements te verzamelen.

Red

- Bouw de unit test zo op dat de requirements die in de Plan fase is verzameld worden overgenomen in de code.
- Zet voor elke requirement de methode zo op dat de gewenste uitkomst verwacht wordt.
    - Na deze actie zal de unit tests op rood staan

Green

- Schrijf de code minimaal binnen de rode methodes om de gewenste resultaat te verkrijgen, zodat deze groen worden.

Refactor

- Verwerk de logica binnen het systeem en optimaliseer de code.
- Zorg hiervoor dat de benaming klopt, bij twijfel raadpleeg het scrumteam of de verantwoordelijke.
- Zorg dat de gemaakte methodes gemakkelijk leesbaar zijn en voeg hierbij comments toe.

Code smells

- De uitgewerkte component is na mening van het product eigenaar en project leider klaar.
- De Backend Applicatie maakt gebruik van Dependency Injection, MVC en Automapper en de programmeurs zullen deze technieken op een aantal correcte wijzen toepassen.
- Het systeem zal test driven geprogrammeerd worden, dit betekend dat eerst Unit Tests en daarna code wordt gemaakt binnen het project.
- NOT DONE (Code smells):

- Duplicated code: Identieke of erg op elkaar lijkende code die bestaat in meer dan een locatie
  - DO NOT REPEAT YOURSELF
- Long Methods: een methode, functie of procedure welke veel te groot is geworden
  - <10 lines of code = goed
  - 10> <50 lines of code = Methodes opsplitsen met zijn verantwoordelijkheden
  - 50> <75 Lines of code = “slecht”
  - 75> lines of code = reviewen en refactoren
- Large class
  - Een class dat te groot is geworden (een God object)
- Feature envy: een klasse die buitensporig veel gebruik maakt van methoden uit een andere klasse
- Inappropriate intimacy: een klasse met afhankelijkheden van uitvoeringsdetails over een andere klasse.
- Contrived complexity: gedwongen gebruik van overdreven ingewikkelde design patterns, waar simpelere designs ook zouden voldoen.
- Refused bequest: een klasse die de methode van een basisklasse op een dusdanige wijze overschrijft, dat het contract van de basisklasse niet onderhouden wordt door de afgeleide klasse.
- Lazy class / Freeloader: een klasse die bijna niets doet.
- Contrived complexity: gedwongen gebruik van overdreven ingewikkelde design patterns, waar simpelere designs ook zouden voldoen.
- Excessively long identifiers: in het bijzonder het gebruik van naamgeving voor doorverwijzing welke impliciet aanwezig zouden moeten zijn in de software architecture.
- Excessive use of literals: deze zouden moet worden geprogrammeerd als named constants, om de leesbaarheid te verbeteren en programmeerfouten te voorkomen.
- Realisatie in overeenstemming met ontwerp
- Code is comprehensible (te begrijpen)

Sprints

Voordat er een nieuwe of een opvolgende sprint van de vorige sprint wordt gestart, zal dit plan van aanpak uitgebreid worden om de planning van de aankomende sprint te bedekken. Zodoende zal het plan van aanpak helpen het project in goede banen te leiden op een iteratieve manier. Voor dit plan van aanpak is er een Architecture Notebook opgesteld (Zie bijlage Concept architecture notebook), waar het exacte ontwerp en architectuur van de applicatie in is omschreven. Dit zal leiden voor een werkend prototype na elke sprint.

De werkwijze en de daarbij horende documentatie om een sprint te bedekken zal bestaan uit:

- Als het aanwezig is, de voorgaande sprint evalueren en te testen op de Definition of Done. (Zie Hoofdstuk Definition of done)
- Overleg met het product eigenaar wat de Scope is van de Sprint en de daarbij horende classes.
- Overleg met het product eigenaar de attributen voor elk van die classes. (Zie hierboven genoemde punt)
- Hierbij zal ook de Business Rules van elk attribuut besproken en genoteerd worden.
- Analyseren van de User Story van het product eigenaar en hierbij de business rules noteren die bij het bijhorende backlog onderwerp en de daaronder taken horen.
- Ook zal er na de ontwikkeling van een sprint gedocumenteerd worden hoe de sprint is verlopen en een korte evaluatie of alle taken zijn voltooid. Anders worden deze meegebracht naar de volgende sprint.

Er zal een nieuwe sprint ontstaan om de twee weken totdat 2 december is bereikt.

De risico's

Dit gedeelte bevat een lijst van mogelijke risico's binnen het project. De kans, impact en prioriteit zijn gebaseerd op schattingen binnen het scrumteam.

Risico	Kans in %	Impact (0-5, 0=laag, 5=hoog)	Prioriteit (0-5, 0=laag, 5=hoog)	Risicobeperkende maatregel
Gebrek aan contact/input van een teamlid	40%	4	5	Contactgegevens uitwisselen, afspraken maken
Het niet houden aan het ontwerp dat binnen de documenten staan vastgesteld	50%	3	3	Afspraken maken aan de hand van welke documenten de applicatie gemaakt moet worden. Deze documenten gemakkelijk openbaar maken om er eenvoudig toegang tot te verkrijgen.
Focus op onderwerpen die buiten het onderzoek vallen.	15%	2	2	Goede richtlijnen voor het onderzoek vastleggen in het Plan van Aanpak
Onderzoeksvraag te abstract	20%	4	4	Kritisch kijken naar de onderzoeksvragen en de concreetheid
Ziekte van Dimitri Tholen. Dimitri bezit van ons allemaal de meeste kennis van het gebruik van de laatste technieken en software. Mocht Dimitri ziek worden dan kost het ons uitermate veel tijd om ons te verdiepen binnen deze stof.	60%	4	5	Goede afspraken maken hoe de nieuwe technieken geïmplementeerd moeten worden en in welke wijze.

H : Iteration plan

Sprints

Voordat er een nieuwe sprint of een opvolgende van de vorige sprint word gestart, zal de Iteration Plan uitgebreid worden om de planning van de aankomende sprint te bedekken. Zodoende zal de Iteration Plan helpen het project in goede banen te leiden op een iteratieve manier. Voor deze Iteration plan is er een Architecture Notebook opgesteld, waar het exacte ontwerp en architectuur van de applicatie in is omschreven. Dit zal leiden tot een werkende prototype na elke sprint.

De werkwijze en de daarbij behorende documentatie om een sprint te bedekken zal bestaan uit:

- Als het aanwezig is, de voorgaande sprint evalueren en te testen op de Definition of Done. (Zie Hoofdstuk Definition of done)
- Overleg met de product eigenaar wat de Scope is van de Sprint en de daarbij behorende classes.
- Overleg met de product eigenaar de attributen voor elk van die classes. (Zie hierboven genoemde punt)
    - Hierbij zal ook de Business Rules van elk attribuut besproken en genoteerd worden.

- Analyseer de User Stories dat door de product eigenaar is vastgesteld. Documenteer de business rules met daarbij de bijbehorende backlog items met de toebehorende taken.
- Ook zal er na het afloop van een sprint gedocumenteerd worden hoe de sprint is verlopen en een korte evaluatie of alle taken zijn voltooid. Anders worden deze taken meegebracht naar de volgende sprint.

Er zal een nieuwe sprint ontstaan om de twee weken totdat 12 december is bereikt.

Aanpak

Voor het ontwikkelen van de partner module voor het nieuwe online-platform van Perflectie zal in grote lijnen de volgende aanpak gebruikt worden om een taak of taken te voltooien:

Project Constructie:

1. Selecteer of pak een taak wat in de sprint staat toegewezen. Wanneer er geen taken meer zijn toegewezen zorg er dan voor dat je een taak oppakt wat open staat en nog niet aan iemand is toegewezen.
2. Maak Unit tests van de business rules omschreven in de Iteration Plan van de Sprint waar de taak toe behoort.
3. Creëer mocht het nog nodig zijn het model, repository, service, tests en controller voor de taak.
4. Creëer de Dependency injection voor de service voor zowel de test data als voor de real data.
5. Neem de bruikbare code over van het oude project naar het nieuwe project.
6. Pas de code aan of voeg code toe totdat elk unit test op groen staat. (Hou hier rekening mee met de regels die genoteerd zijn in hoofdstuk 7)
7. Wanneer de Unit Tests op groen staan en de taak is uitgevoerd kan de taak op done worden gezet.
8. Om de 2 weken de review bijwonen en nieuwe taken ontvangen voor de volgende sprint.
9. Mocht het nodig zijn, laat je code reviewen door een aanwezige programmeur.
10. Tijdens de review zal de gemaakte taken met zijn tests worden geanalyseerd door iedereen aanwezig.

Project Documentatie:

1. Voor de start van een nieuwe sprint, zal de scope gedocumenteerd worden wat beschreven staat in hoofdstuk 3. Als er in de vorige sprint nog taken zijn die open staan zal deze ook met de nieuwe sprint meegenomen worden.
2. Van de scope van de Sprint zullen de Use cases en Use Stories worden genoteerd.
3. Aan het einde van een sprint zal in het document een evaluatie gedocumenteerd worden met daarbij de backlog items en taken die volledig voltooid zijn.

Project Testing:

1. Tijdens de ontwikkeling van het project zal er testdriven gewerkt worden. Dit houdt in dat continu Unit Tests worden gedraaid in de back-end die de Business Rules waarborgen.
2. Ook zal tijdens het ontwikkelen van het project, door Yvo Vogt en Jochem Aubel (Zie hoofdstuk Rollen en Personen), worden getest of de gezette business rules en kwaliteit worden overschreven.

Definition of done

MoScoW-methode

Must haves (zie hoofdstuk Conceptueel model)

- De programme factory
- Client/Partner
- Gebruikers

- Participants & feedbackmembers
- Mail systeem
- Trigger systeem
- De mogelijkheid binnen het systeem om e-mails te versturen op basis van triggers dat afgehandeld wordt door een trigger systeem.
- De mogelijkheid binnen het systeem om functionaliteiten uit te voeren op basis van triggers dat afgehandeld wordt door een trigger systeem.
- Oude functionaliteiten van het huidige online-platform

Should haves

- Real-time feedback van het systeem naar de gebruiker
- De mogelijkheid om via CRUD functionaliteiten triggers te onderhouden binnen het systeem

Could haves

- Het implementeren van de mogelijkheid om meerdere soorten persoonlijkheidstest voor een programma te kunnen kiezen.

Won't haves

- De gebruiker kunnen op basis van een iteratieve wijze zijn competenties en levels ontwikkelen.

Werkwijze van een programmeur binnen het scrumteam

De werkwijze die wij onderling hebben afgesproken kan worden gediend als een checklist tijdens het programmeren. Er bestaat een checklist voor ieder vier fasen: Plan, red, green en refactor.

Plan

- Neem een taak vanuit de scrumboard op jezelf, als er geen eigenaar aan de taak is toegewezen.
- Bespreek binnen het scrumteam af of jou taak niet in de scope valt van een taak van iemand anders.
- Bekijk de iterationplan van het project om de user story en de requirements te verzamelen.

Red

- Bouw de unit test zo op dat de requirements die in de Plan fase is verzameld worden overgenomen in de code.
- Zet voor elke requirement de methode zo op dat de gewenste uitkomst verwacht wordt.
    - Na deze actie zal de unit tests op rood staan

Green

- Schrijf de code minimaal binnen de rode methodes om de gewenste resultaat te verkrijgen, zodat deze groen worden.

Refactor

- Verwerk de logica binnen het systeem en optimaliseer de code.
- Zorg hiervoor dat de benaming klopt, bij twijfel raadpleeg het scrumteam of de verantwoordelijke.
- Zorg dat de gemaakte methodes gemakkelijk leesbaar zijn en voeg hierbij comments toe.
    - Clean code
        - http://www.slideshare.net/mariosangiorgio/clean-code-and-code-smells?related=2
        - http://www.slideshare.net/arturoherrero/clean-code-8036914?related=3
        - http://not-at-school.blogspot.nl/2011/03/coding-rules-for-clean-and-robust-c.html

S.O.L.I.D

- Single responsibility principle
- Open/Closed principle principle
- Liskov subsitution principle
- Interface segregation principle
- Dependency inversion principle

K.I.S.S

- Keep It Simple Stupid

Code smells

- De uitgewerkte component is na mening van het product eigenaar en project leider klaar.

- De Backend Applicatie maakt gebruik van Dependency Injection, MVC en Automapper en de programmeurs zullen deze technieken op een aantal correcte wijzen toepassen.
- Het systeem zal test driven geprogrammeerd worden, dit betekend dat eerst Unit Tests en daarna code wordt gemaakt binnen het project.
- NOT DONE (Code smells):
    - Duplicated code: Identieke of erg op elkaar lijkende code die bestaat in meer dan een locatie
        - DO NOT REPEAT YOURSELF
    - Long Methods: een methode, functie of procedure welke veel te groot is geworden
        - <10 lines of code = goed
        - 10> <50 lines of code = Methodes opsplitsen met zijn verantwoordelijkheden
        - 50> <75 Lines of code = “slecht”
        - 75> lines of code = reviewen en refactoren
    - Large class
        - Een class dat te groot is geworden (een God object)
    - Feature envy: een klasse die buitensporig veel gebruik maakt van methoden uit een andere klasse
    - Inappropriate intimacy: een klasse met afhankelijkheden van uitvoeringsdetails over een andere klasse.
    - Contrived complexity: gedwongen gebruik van overdreven ingewikkelde design patterns, waar simpelere designs ook zouden voldoen.
    - Refused bequest: een klasse die de methode van een basisklasse op een dusdanige wijze overschrijft, dat het contract van de basisklasse niet onderhouden wordt door de afgeleide klasse.
    - Lazy class / Freeloader: een klasse die bijna niets doet.
    - Contrived complexity: gedwongen gebruik van overdreven ingewikkelde design patterns, waar simpelere designs ook zouden voldoen.
    - Excessively long identifiers: in het bijzonder het gebruik van naamgeving voor doorverwijzing welke impliciet aanwezig zouden moeten zijn in de software architecture.
    - Excessive use of literals: deze zouden moet worden geprogrammeerd als named constants, om de leesbaarheid te verbeteren en programmeerfouten te voorkomen.
    - Realisatie in overeenstemming met ontwerp
    - Code is comprehensible (te begrijpen)

Planning en Mijlpalen

Mijlpaal	Sprint	Datum
Kick off project werkwijze leren van Dependency Injection	1	08-09-2014 - 19-09-2014
Werkende “Fundering” voor het project	1	
Component factory af en werkende voorbeeld	1	
Partner Component Afgemaakt	2	22-09-2014 – 03-10-2014
User Component afgemaakt	2	
User Authorisatie en daarbij zijn kennen en kunnen afgemaakt	2	
Extra functionaliteiten toevoegen als voorbeeld: Badges, score, test	3	06-10-2014 – 17-10-2014

	4	20-10-2014 – 31-10-2014
	5	03-11-2014 – 14-11-2014
	6	17-11-2014 – 28 -11 -2014
Werkend product dat live gezet is met daarbij de transactie van de oude database	7	01 -12 -2014 - 12 -12 -2014

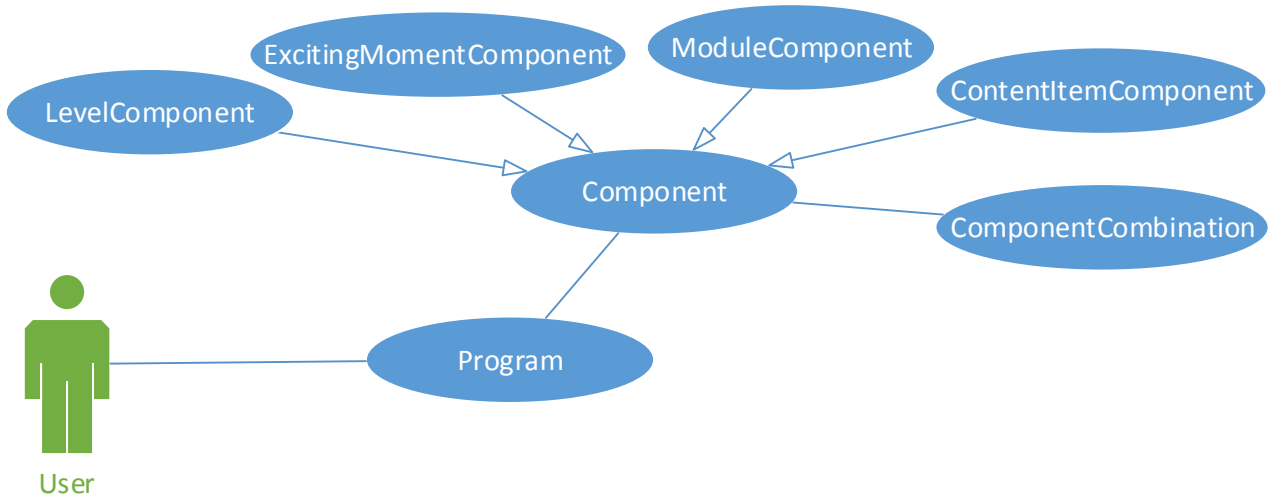
Rollen en Personen

Tijdens dit project worden onderstaande rollen vervuld door de volgende personen.

Naam / Omschrijving	Toegekend aan:	Verantwoordelijkheden
Product eigenaar	Jochem Aubel Stefan Op de Woerd	Het maken van User Stories en het vaststellen van Business Rules. Door wie en wat moet het systeem kunnen?
Project leider	Yvo Vogt Bas Alblas	Realiseert het project door het coördineren van en/of aansturen van projectmedewerkers door middel van de SCRUM methode.
Specialist	Dimitri Tolen	Deelt kennis en installeert de structuur binnen het project.
Architect	Bas Alblas	Ontwerpt de structuur binnen het softwaresysteem.
Analist	Bas Alblas	Analyseert de flows binnen het project, en speurt naar verbeteringen.
Project Administrateur	Bas Alblas	Draagt zorg over de projectadministratie en houdt deze constant up to date.
Software ontwikkelaar	Bas Alblas, Dimitri Tolen, Benno van Dijk, Anne Boon	Houdt zich bezig met het programmeren van software.
Kwaliteitsbewaker	Yvo Vogt, Dimitri Tolen	Controleert of de systemen constant voldoen aan de vooraf gestelde normen.

Sprint 1

Use Cases



User Stories

User Story	Story
Basisprogramma (Base Program, components and combinations)	Als superuser (en later als partner manager) wil ik … - een basisprogramma kunnen aanmaken, zien, wijzigen en verwijderen (lees: status op inactief zetten) - basisprogramma componenten kunnen aanmaken, zien, wijzigen en verwijderen. Mogelijke componenten zijn: module, level, spannend moment en content item - basisprogramma componentverbindingen kunnen aanmaken, zien, wijzigen en verwijderen en op volgorde zetten - instellen voor welke partners het basisprogramma toegankelijk is, opties nu: voor alle partners, alleen voor mijzelf Basisprogramma aanmaken, zien, wijzigen en verwijderen. Daarbinnen (dat kan nu ook al): - Modules aanmaken, zien, wijzigen en verwijderen - Levels aanmaken, zien, wijzigen en verwijderen - Spannende momenten aanmaken, zien, wijzigen en verwijderen - Content aanmaken, zien, wijzigen en verwijderen - Levels toevoegen aan en verwijderen uit modules - Spannende momenten toevoegen aan en verwijderen uit modules - Content items toevoegen aan en verwijderen uit modules - Content items toevoegen aan en verwijderen uit levels - Content items binnen een module op volgorde zetten - Content items binnen een level op volgorde zetten
OntwikkelProgramma	- een ontwikkelprogramma kunnen aanmaken, zien, wijzigen en verwijderen (lees: op status inactief zetten) - een basisprogramma selecteren en wijzigen dat als basis dient voor het ontwikkelprogramma - het ontwikkelprogramma kunnen ontkoppelen van het basisprogramma. Dit betekent dat alle componenten en -verbindingen uit het basisprogramma worden gekopieerd en dat het ontwikkelprogramma geen parent meer heeft - basisprogramma componenten kunnen zien, wijzigen en verwijderen (lees: status op inactief)

	- basisprogramma componentverbindingen kunnen zien en verwijderen - ontwikkelprogramma componenten kunnen aanmaken, zien, wijzigen en verwijderen - verbindingen tussen basisprogramma en/of ontwikkelprogramma componenten kunnen aanmaken, zien, wijzigen, verwijderen en op volgorde zetten - Als een basisprogramma component wijzigt dan wijzigt deze mee in het ontwikkelprogramma, mits deze niet is gewijzigd of verwijderd in het ontwikkelprogramma - Als een basisprogramma een component toevoegt, dan wordt deze toegevoegd in het ontwikkelprogramma - Voor een ontwikkelprogramma wordt pas een child van de base component toegevoegd als deze de parent component gaat wijzigen - Als een basisprogramma een component verwijdert, dan wordt deze verwijderd in het ontwikkelprogramma, tenzij deze al is gewijzigd in het ontwikkelprogramma Componentverbindingen - Een ontwikkelprogramma kan de volgorde van componentverbindingen uit het basisprogramma niet aanpassen - Een ontwikkelprogramma kan een componentverbinding uit het basisprogramma verwijderen (lees: op inactief zetten) - Een ontwikkelprogramma kan componentverbindingen tussen basis- en of child componenten aanmaken - Een ontwikkelprogramma kan eigen componentverbindingen op volgorde zetten, waarbij de volgorde wordt gedefinieerd als “na wie kom ik?” ofwel “wie is mijn voorganger in de volgorde?” - Als een basisprogramma een componentverbindinge verwijdert (lees: op inactief zet) dan is deze ook verwijderd voor het child program - Als een basisprogramma de volgorde van componentverbindingen wijzigt, wijzigt deze mee voor het child program, waarbij de zelf toegevoegde verbindingen van de child gekoppeld blijven aan dezelfde voorganger - Voor een ontwikkelprogramma wordt pas een child componentverbinding toegevoegd als deze de parent componentverbindinge gaat wijzigen. Het enige wat het ontwikkelprogramma kan wijzigen is de status: deze kan op inactief gezet worden
Programma Views	Als superuser wil ik een overzicht van programma's zien: - Het overzicht bevat de id, de naam, de status (bewerken, review, actief, inactief) en de toegankelijkheid van het programma en de naam van het programma waarvan dit programma is afgeleid - Het overzicht bevat een aantal andere attributen of afgeleiden van het programma: naam van de partner, aantal deelnemers in het programma, aantal modules, levels, spannende momenten, content items - Elke kolom heeft een filter, sorteer en zoek mogelijkheid Als superuser wil ik een detailscherm per programma: - id en naam van het programma - alle instelbare attributen van het programma (nader te bepalen = customization) - overzicht van programma’s die op dit programma zijn gebaseerd - doorklik mogelijkheid naar de detailpagina van die programma’s Als superuser wil ik voor elke component een overzichtsscherm: - Elk overzicht bevat de id, de naam, de status en de omschrijving van de component - Het module overzicht bevat aantal gekoppelde actieve levels, aantal gekoppelde actieve content items, aantal actieve+gepauzeerde deelnemers in deze module - Het level overzicht bevat aantal gekoppelde actieve modules, aantal gekoppelde actieve content items, aantal actieve+gepazeerde deelnemers die het level nu actief hebben, aantal actieve+gepauzeerde deelnemers die het level hebben geselecteerd maar niet actief hebben - Het spannende momenten bevat aantal gekoppelde actieve modules, aantal actieve+gepauzeerde deelnemers die dit spannend moment hebben geselecteerd - Het content item overzicht bevat type, aantal gekoppelde actieve levels, aantal gekoppelde actieve modules - Elke kolom heeft een filter, sorteer en zoek mogelijkheid

	Als superuser wil ik voor elke component een detailscherm: - het detailscherm bevat alle instelbare attributen van de component - bevat de aan die component gekoppelde componenten

Sprint backlog

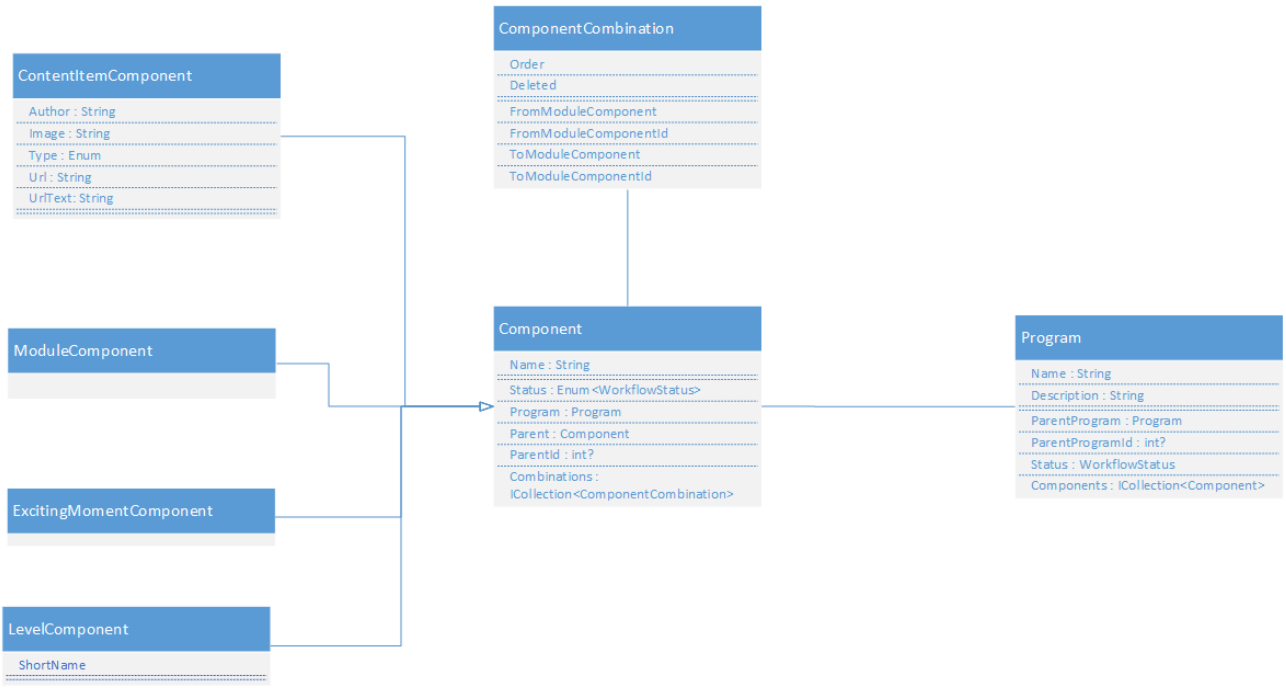
ID	Title 1	Title 2	Assigned To	Felocity
89	Nieuw project opzetten			0
135		Aanmaken MVC 5 project	Dimitri Tholen, Sebastiaan Alblas	3
136		Libraries installeren	Dimitri Tholen, Sebastiaan Alblas	3
138		Add Branch to source controller of TFS inside the Perflectie project	Dimitri Tholen, Sebastiaan Alblas	1
94	Unit test installaties			0
139		Unity installeren	Dimitri Tholen, Sebastiaan Alblas	2
140		Bootstrapper aanmaken	Dimitri Tholen, Sebastiaan Alblas	2
95	Layout opzetten			0
169		Layout opzetten	Dimitri Tholen, Sebastiaan Alblas	1
97	Basisprogramma (Base Program, components and combinations)			1
148		Afstemmen UML Diagram	Dimitri Tholen	10
103	Ontwikkelprogramma (Program, components and combinations)			1
102	Component			
164		Model defineren	Anne Boon	1
165		Repositories	Sebastiaan Alblas	4
166		Service en ViewModels aanmaken	Sebastiaan Alblas	5
167		Unit Test ontwerpen	Sebastiaan Alblas	3
219		Ik ben child en wil een parent component wijzigen of verwijderen	Dimitri Tholen	2
226		Ik wil een nieuwe component aanmaken binnen mijn programma	Sebastiaan Alblas	1
227		Ik wil componenten van een type binnen mijn programma zien	Sebastiaan Alblas	1
101	ModuleComponent			0
143		Model definieren	Sebastiaan Alblas	1
144		Repositories	Sebastiaan Alblas	4

145		Service en ViewModel aanmaken	Sebastiaan Alblas	5
146		Unit Test ontwerpen	Sebastiaan Alblas	3
147		Views backend aanmaken	Jesse Buitenhuis	2
232		ViewModel overzichtscherm Module	Dimitri Tholen	1
233		ViewModel detailscherm Module	Dimitri Tholen	3
100	LevelComponent			0
149		Model defineren	Sebastiaan Alblas	1
150		Repositories	Sebastiaan Alblas	3
151		Service en ViewModel aanmaken	Sebastiaan Alblas	5
152		Unit Test ontwerpen	Sebastiaan Alblas	3
153		Views backend aanmaken	Jesse Buitenhuis	2
156		Service en Viewmodel aanmaken	Sebastiaan Alblas	1
234		ViewModel overzichtsscherm Level	Dimitri Tholen	1
235		ViewModel detailscherm Level	Dimitri Tholen	1
99	ExcitingMomentComponent			0
154		Model defineren	Anne Boon	4
155		Repositories	Anne Boon	5
157		Unit tests ontwerpen	Anne Boon	6
158		Views backend aanmaken	Jesse Buitenhuis	3
236		ViewModel overzichtsscherm Spannende Momenten	Dimitri Tholen	1
237		ViewModel detailscherm Spannende Momenten	Dimitri Tholen	1
98	ContentItemComponent			0
159		Model defineren	Anne Boon	4
160		Repositories	Anne Boon	5
161		Service en ViewModels aanmaken	Anne Boon	6
162		Unit tests ontwerpen	Anne Boon	3
163		Views backend aanmaken	Jesse Buitenhuis	1
238		ViewModel overzichtsscherm ContentItems	Dimitri Tholen	1
239		ViewModel detailscherm ContentItem	Dimitri Tholen	1
203	ComponentCombination			0
204		Model Defineren	Sebastiaan Alblas	1
205		Repositories	Dimitri Tholen	1
206		Service en ViewModels aanmaken	Sebastiaan Alblas	5
207		Unit Test ontwerpen	Sebastiaan Alblas	3

224		Ik wil de volgorde van Content bij Level of Module wijzigen	Sebastiaan Alblas	8
225		Ik wil de volgorde van content items bij een module of level zien	Sebastiaan Alblas	1
228		Ik wil de levels en spannende momenten zien die aan een module zijn gekoppeld	Sebastiaan Alblas	8
229		Ik wil zien welke levels en modules aan contentitem zijn gekoppeld	Anne Boon	1
230		Ik wil een componentverbinding verbreken	Sebastiaan Alblas	1
231		Ik wil een componentverbinding aanleggen	Sebastiaan Alblas	1
242		Ik wil zien welke ContentItems aan een Module / Level zijn gekoppeld	Sebastiaan Alblas	1
171	Program			0
173		Model Defineren	Dimitri Tholen	1
174		Repositories	Dimitri Tholen	1
175		Service en Viewmodels aanmaken	Dimitri Tholen	5
176		Unit Tests ontwerpen	Dimitri Tholen	3
201		Views Backend Aanmaken	Jesse Buitenhuis	2
220		Ik wil mijn programma baseren op een ander programma	Sebastiaan Alblas	6
221		Ik wil mijn programma loskoppelen van mijn parent	Sebastiaan Alblas	1
222		Ik wil een programma dupliceren	Sebastiaan Alblas	2
240		ViewModel overzichtsscherm Programma's	Dimitri Tholen	1
241		ViewModel detailscherm Programma's	Dimitri Tholen	1
190	ProgramComponentCombination			148
197		Model Defineren	Sebastiaan Alblas	1
198		Repositories	Sebastiaan Alblas	1
199		Service en ViewModels	Sebastiaan Alblas	1
200		Unit Test ontwerpen	Sebastiaan Alblas	1
202		Create reference to BaseComponentCombination and ProgramComponent	Sebastiaan Alblas	2
210	Migratie Perflectie Content			
211		Database opzetten	Yvo Vogt	
212		Migratie Modules	Yvo Vogt	
213		Migratie levels	Yvo Vogt	

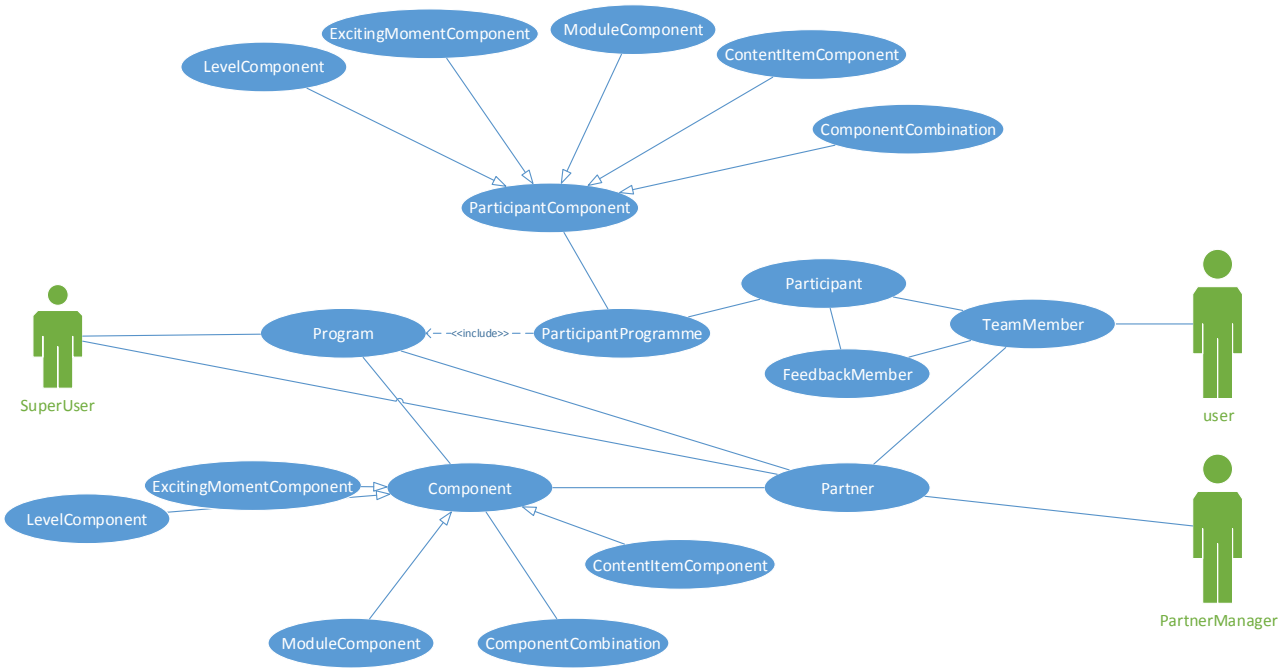
214	Migratie Spannende Momenten	Yvo Vogt
215	Migratie ContentItems	Yvo Vogt
216	Migratie ComponentCombinations	Yvo Vogt
315	Json omzetten naar ViewModels	Dimitri Tholen
316	API controllers	Dimitri Tholen
317	Afstemmen front end / back end	Dimitri Tholen
		0

Domain Diagram



Sprint 2

Use Cases



User Stories

User Story	Story
PartnerBeheer (B, 4)	Als superuser wil ik: Een partner aanmaken, zien, bewerken en verwijderen Status: actief of inactief
Partner Views (2)	Als superuser wil ik een overzichtsscherm van partners: Naam van de partner Id van de partner Status van de partner Aantal programma's binnen de partner Aantal deelnemers binnen de partner Als superuser wil een detailscherm per partner: Overzichtscherm van de programma's Managers van de partner
User Roles (5, D)	Een user kan de volgende rollen hebben: superuser manager program manager user De eerste drie rollen noemen we beheerrollen. Een superuser heeft toegang tot alle partners, programma's en deelnemers Een superuser kan alle users aanmaken, zien, bewerken en verwijderen Een superuser kan alle rollen toekennen en bewerken (dus ook andere superusers) Een superuser heeft alle rechten die een manager, program manager en user ook hebben Een manager is gekoppeld aan een of meerdere partners Een manager heeft toegang tot alle programma's en deelnemers die zijn gekoppeld aan de partner(s) waarvoor hij manager is

	Een manager kan de aan de partner(s) opengestelde programma's inzien en als basis gebruiken Een manager kan alle teamleden aanmaken, zien, bewerken en verwijderen die zijn gekoppeld aan de partner(s) waarvoor hij manager is Een manager kan de rollen manager, program manager en user toekennen en bewerken voor users gekoppeld aan de partner(s) waarvoor hij manager is Een manager heeft alle rechten die een programma manager en user ook hebben voor de parter(s) waaraan hij is gekoppeld Een program manager is gekoppeld aan een of meerdere programma's Een program manager heeft toegang tot alle programma's waarvoor hij program manager is Een program manager kan de aan de partner opengestelde programma's inzien en als basis gebruiken Een program manager kan alle teamleden aanmaken, zien, bewerken en verwijderen die zijn gekoppeld aan de programma('s) waarvoor hij program manager is Een program manager kan de rollen program manager en user toekennen en bewerken voor users gekoppeld aan de programma('s) waarvoor hij program manager is Een program manager heeft alle rechten die een user ook heeft voor alle programma's waaraan hij gekoppeld is Een user is gekoppeld aan een of meerdere programma's Een user kan per programma de modules, levels en spannende momenten inzien Een user kan 1 module, levels en spannende momenten selecteren, deselecteren en de attributen personaliseren [use case diagram maken]
User Authentication (8, J) (4 ViewModels, syntax afspraken)	Als superuser kan ik inloggen in de Back Office Als Manager kan ik inloggen in de Back Office Als Program Manager kan ik inloggen in de Back Office Als TeamMember kan ik inloggen in de Front Office Een user die TeamMember is, komt na inloggen in de Front Office, ongeacht zijn andere rollen Een user die TeamMember is in meerdere programma's kan kiezen in welk programma hij wil inloggen. Hij kan vanuit het ene programma switchen naar het andere programma Als hij ook andere rollen heeft, kan hij vanuit de Front Office naar de Back Office komen Een user die geen TeamMember is, komt na inloggen in de Back Office
Gebruikers Beheer (6, D)	Als superuser wil ik: andere superusers kunnen aanmaken, zien, bewerken en verwijderen (dat is: status stopped) bepalen of voor deze superuser al een User bestaat (op basis van voornaam, tussenvoegsel, achternaam) Zo ja, Superuser koppelen aan deze User en notificatie verzenden Zo nee, deze User aanmaken en benodigde gegevens uitvragen en uitnodiging verzenden Als superuser of manager wil ik binnen een partner: managers aanmaken, zien, bewerken en verwijderen (dat is: status stopped) bepalen of bij deze Manager al een User bestaat (op basis van voornaam, tussenvoegsel, achternaam) Zo ja, Manager koppelen aan deze User en notificatie verzenden Zo nee, deze User aanmaken en benodigde gegevens uitvragen en uitnodiging verzenden Als superuser, manager of program manager wil ik binnen een programma: program managers aanmaken, zien, bewerken en verwijderen (dat is: status stopped) bepalen of bij deze Program Manager al een User bestaat (op basis van voornaam, tussenvoegsel, achternaam)

	Zo ja, Program Manager koppelen aan deze User en notificatie verzenden Zo nee, deze User aanmaken en benodigde gegevens uitvragen en uitnodiging verzenden Als superuser, manager en program manager wil ik binnen een programma: TeamMembers aanmaken, zien, bewerken en verwijderen (dat is: status stopped) binnen een programma bepalen of bij deze TeamMember al een User bestaat (op basis van voornaam, tussenvoegsel, achternaam) Zo ja, TeamMember koppelen aan deze User Zo nee, deze User aanmaken en benodigde gegevens uitvragen Instellen of de TeamMember een deelnemer (Participant) is, of een Feedbackteamlid (FeedbackMember) De TeamMember status inclusief geschiedenis inzien Instellen wanneer de TeamMember een uitnodiging voor de intake ontvangt Instellen wanneer de TeamMember de intake mag afronden
Gebruikers Views (2)	Als superuser wil ik een overzichtspagina met voor alle users: user id, aanhef, volledige naam, bedrijf, e-mail adres, huidige status, rol op elke veld zit filter, sorteer en zoek mogelijkheid **Als superuser en manager wil ik binnen een partner een overzichtspagina met:** **overicht van users die manager zijn binnen deze partner** **overzicht van users die programma manager zijn van de programma's van deze partner** **overzicht van alle users die TeamMember zijn binnen de programma's van deze partner** Als superuser, manager en program manager wil ik binnen een programma een overzichtspagina met daarin per Participant: TeamMemberId, aanhef, volledige naam, bedrijf, e-mail adres, huidige ParticipantStatus, gekoppelde Module Als Module is gekoppeld, doorklik mogelijkheid naar detailscherm TeamMemberModule Op elk veld zit filter, sorteerangular en zoek mogelijkheid Als superuser, manager en program manager wil ik een detailpagina voor een user (voor zover ik toegang heb tot de user): kunnen inzien voor welke partners hij manager is, voor welke programma's hij program manager is, voor welke programma's hij TeamMember is (voor zover ik toegang heb tot die partners en programma's) kunnen doorklikken op TeamMember om in het detailscherm van TeamMemberModule te komen de user status incl geschiedenis kunnen inzien Een user met beheerrollen kan inloggen in de Front Office omgeving voor de TeamMembers waar hij toegang toe heeft
User Status (3 J, 2 BE) **in back end of front end?**	Statussen Elke user binnen Perflectie heeft op elk moment precies één status. De historie van de status wordt bijgehouden (begin en einddatum). De begindatum van de volgende status start 1 seconde na de einddatum van de vorige status. De laatste status heeft geen einddatum De laatste status is meestal de huidige status, maar dat hoeft niet. Als je hebt gepauzeerd en je hebt de einddatum van je pauze al ingesteld, is de laatste status actief, met als begindatum de einddatum van de pauze + 1 seconde De volgende statussen zijn beschikbaar: InvitedManager InvitedParticipant: je hebt een of meer uitnodigingen ontvangen om te starten met Perflectie, waarvan tenminste 1 als deelnemer, maar je hebt je nog niet aangemeld

	InvitedTeammember: je hebt een of meer uitnodigingen ontvangen om te starten met Perflectie als feedbackteamlid, maar je hebt je nog niet aangemeld UserDataSupplied: Je hebt je aangemeld, maar nog geen intake afgerond Active: Je hebt een intake afgerond Paused: Je hebt Perflectie gepauzeerd Stopped: Je bent gestopt met Perflectie Wat moet er gebeuren bij elke status: InvitedManager: komt in het aanmeldscherm en gaat na aanmelden door naar de Back Office omgeving InvitedParticipant: komt in het aanmeldscherm en gaat na aanmelden door naar de uitleg voor Participant InvitedFeedbackMember: komt in het aanmeldscherm en gaat na aanmelden door naar de uitleg voor FeedbackMemeber UserDataSupplied: gaat bij inloggen naar de intake stap waar hij is gebleven (zie status teammember). Als meerdere intakes openstaan, kom je in een keuzescherm waarin je de intake selecteert waarmee je verder wilt. Als hij geen TeamMember is (dus alleen beheerrol heeft) komt hij in de Back Office Active: je komt op de url waarop je klikte. Bij geen url kom je in je dashboard Paused: je komt in het pauzescherm, waar je de instellingen van je pauze kunt aanpassen: einddatum aanpassen of pauze direct beëindigen Stopped: kan niet inloggen. Melding om contact op te nemen met Perflectie
User aanmelden (2)	Ik ben nieuwe user, heb een uitnodiging ontvangen en wil: vanuit mijn uitnodigingsmail in een aanmeldscherm komen naam, tussevoegsel, achtenaam, email, telefoonnummer, profielfoto, bedrijf kunnen instellen
Participant Status (3)	Statussen Elke Participant binnen Perflectie heeft op elk moment precies één status. De historie van de status wordt bijgehouden (begin en einddatum). De begindatum van de volgende status start 1 seconde na de einddatum van de vorige status. De laatste status heeft geen einddatum De volgende statussen zijn beschikbaar: IntakeTest: komt in intakescherm van de test IntakeModule: komt in intakescherm voor selecteren Module IntakeMotivation: komt in intakescherm voor Motivation IntakeLevels: komt in intakescherm voor selecteren Levels IntakeExcitingMoments: komt in intakescherm voor selecteren Spannende Momenten IntakeTips: komt in intakescherm voor tips bij Spannende Momenten IntakeFeedbackMembers: komt in intakescherm voor uitnodigen FeedbackMembers IntakeMessages: komt in intakescherm voor selecteren berichteninstellingen Active: komt in Dashboard Completed: komt in Dashboard Stopped: komt in speciaal scherm, als de user geen andere (niet-gestopte) TeamMemers is binnen dit programma
FeedbackMember Status (2)	Statussen Elke FeedbackMember binnen Perflectie heeft op elk moment precies één status. De historie van de status wordt bijgehouden (begin en einddatum). De begindatum van de volgende status start 1 seconde na de einddatum van de vorige status. De laatste status heeft geen einddatum De volgende statussen zijn beschikbaar:

	FeedbackIntakeMotivation: komt in Feedback Intakescherm om Motivation te beoordelen FeedbackIntakeLevels: komt in Feedback Intakescherm om Levels te beoordelen FeedbackIntakeExcitingMoments: komt in Feedback Intakescherm om Spannende momenten te beoordelen FeedbackIntakeMessages: komt in Feedback Intakescherm voor berichteninstellingen Active: komt in Dashboard Completed: komt in Dashboard Stopped: komt in speciaal scherm, als de user geen andere (niet-gestopte) TeamMemer is binnen dit programma
ParticipantProgram (Intake) (12, J; 4 BE)	Ik ben deelnemer binnen een programma en ik wil: een Module die binnen het programma selecteren (als het programma maar 1 module heeft, is deze voor mij voorgeselecteerd) (alleen modules met status actief worden aangeboden) De naam en omschrijving van de Module personaliseren. Als niet wordt gepersonaliseerd pak attributen van component en zet deze er hard in een MotivationText, MotivationVideo (YouTube link), MotivationAudio, MotivationImage aan de Module toevoegen Levels selecteren die binnen het programma beschikbaar zijn en deze op volgorde zetten (alleen levels met status actief worden aangeboden) Naam en omschrijving van het Level personaliseren. Als niet wordt gepersonaliseerd pak attributen van component en zet deze er hard in Spannende momenten selecteren die binnen de Module beschikbaar zijn Een eigen Spannend Moment aanmaken en selecteren Bij elk Spannend Moment een tip toevoegen FeedbackMembers uitnodigen met een persoonlijke tekst en instellen wanneer zij de uitnodiging ontvangen Instellen op welke dagen en tijden je feedback wilt doorgeven Instellen welke typen coachberichten je wilt ontvangen Instellen met welke frequentie je coachberichten wilt ontvangen
ParticipantProgram (Changes) **(2)**	Ik ben deelnemer met een ParticipantProgram en ik wil iets aanpassen: Stoppen met het programma (mail naar FeedbackMembers; status Participant = stopped, status FeedbackMember = stopped) Andere Module selecteren: kan alleen als je een nieuwe uitnodiging vanuit het programma krijgt. Er wordt dan een nieuwe Participant aangemaakt. Naam of omschrijving van module wijzigen Motivatie wijzigen Levels of volgorde wijzigen of vervangen (kan alleen voor niet afgeronde levels) Naam of omschrijving van levels wijzigen Spannende momenten wijzigen, verwijderen of toevoegen Tips bij spannende momenten wijzigen, verwijderen of toevoegen FeedbackMembers verwijderen of toevoegen Wijzigen instellingen voor berichten (feedbackdagen en -tijden, typen en frequentie coachberichten)
ParticipantProgram (ManagementView) **1, BE** **2, J**	Ik ben superuser, manager of programma manager en wil van een deelnemer binnen het programma zien: De gekozen module De gekozen levels, de volgorde ervan en de status ervan de FeedbackMembers (incl. Participant), hun status (UserStatus en TeamMemberStatus) en de dagen dat ze feedback doorgeven
POP (2, J)	Ik ben Participant en wil een POP-module volgen: Een POP is een module zonder gekoppelde levels (die kan ik alleen kiezen als die in het programma aanwezig is)

	Ik kan dan kiezen uit alle beschikbaar levels in het programma
On boarding FeedbackMember **3 J**	Ik ben FeedbackMember, heb een uitnodiging ontvangen van een Participant en ik wil: via een button in de uitnodigingsmail in het platform komen als ik nog geen aangemelde user ben in het aanmeldscherm komen de motivatie van de Participant zien, daar Feedback op geven (1 tot 5 sterren) en commentaar op geven de levels met hun om schrijvingen zien die de Participant heeft gekozen de spannende momenten zien die de Participant heeft gekozen voor elk spannend moment een tip geven commentaar geven op de gekozen spannende momenten instellen op welke dagen en tijdstip ik feedback wil doorgeven
FeedbackMember (changes) **1 J**	Ik ben FeedbackMember en ik wil iets aanpassen: stoppen als FeedbackMember pauzeren als User wijzigen instellingen voor feedbackdagen en -tijdstip
User instellingen **1,5 J** **05 BE**	Ik ben user en wil iets wijzigen: mijn persoonlijke gegevens (attributen user) kunnen wijzigen mijn account kunnen pauzeren: begindatum en einddatum van de pauze instellen mijn account kunnen stoppen Een van mijn modules kunnen stoppen Een van mijn FeedbackMemberships kunnen stoppen

Sprint backlog

ID	Title 1	Title 2	Assigned To	Felocity
436	Sprint 2 viewmodels & automapper		Dimitri Tholen	0
435		foutmelding bij loginscherm	Jesse Buitenhuis	1
437		Knop nieuw programma aanmaken doet nog niets	Jesse Buitenhuis	1
438		Pagina nieuw programma	Jesse Buitenhuis	1
439		status programma veranderen	Jesse Buitenhuis	1
440		aantal deelnemers in programma overzicht	Jesse Buitenhuis	1
441		Navigatie management module fixen	Jesse Buitenhuis	1
442		deelnemerslijst binnen programma	Jesse Buitenhuis	1
443		laden lijst met relaties binnen programma	Jesse Buitenhuis	1
444		programma manager(s) binnen programma	Jesse Buitenhuis	1
445		module, level, spannend moment, contentitem detail scherm: status instellen, verwijderen en saven	Jesse Buitenhuis	1

446	level, module, spannend moment, contentitem aanmaken	Jesse Buitenhuis	1
447	overzicht module, level, spannend moment, contentitem: doorklikken naar detailscherm	Jesse Buitenhuis	1
448	(actieve!) componenten van de overerving worden niet meegeteld in programma overzicht	Dimitri Tholen	1
449	component editen van parent program moet nieuwe component aanmaken	Dimitri Tholen	1
450	type bij contentitem wordt niet goed meegegeven	Dimitri Tholen	1
451	contentitem: veld om afbeelding te uploaden	Jesse Buitenhuis	1
452	koppelen van componenten moet worden opgeslagen	Jesse Buitenhuis	1
453	aantallen (actieve + gepauzeerde) Psrticipant per module	Dimitri Tholen	1
454	Aantal (actieve+gepauzeerde) deelnemers die dit level als huidige level hebben & idem die dit level in hun persoonlijk programma hebben	Serge Bekenkamp	1
455	kolommen actief en inactief uit spannende momenten overzicht verwijderen	Jesse Buitenhuis	1
456	lijst (actieve + gepauzeerde) deelnemers in module + level detail overzicht	Jesse Buitenhuis	1
457	afvangen onjuiste url	Jesse Buitenhuis	1
458	gekoppelde (actieve) componenten tonen in overzicht en detailschermen	Dimitri Tholen	1
459	nieuwe partner toevoegen	Jesse Buitenhuis	2
460	data detailscherm partner laden niet	Jesse Buitenhuis	1
461	useroverzichtscherm fixen + gebruiker toevoegen	Jesse Buitenhuis	2
462	viewmodel user detail scherm		1
463	users/module routing fixen	Jesse Buitenhuis	2
465	api/account/login user terugsturen	Dimitri Tholen	6
466	routing na inloggen	Jesse Buitenhuis	4
468	intake: knoppen volgende en vorige activeren	Jesse Buitenhuis	4
471	intake: opslaan bij ga later verder, vorige en klikken op thermometer	Jesse Buitenhuis	4

472	naamgeving Viewmodel UsersDetail aanpassen	Dimitri Tholen	2
473	Laad user roles bij management/users/new	Dimitri Tholen	3
474	Angular User Service knopen	Dimitri Tholen	3
475	Viewmodel ProgrammeDetail toevoegen lijst met users	Dimitri Tholen	2
476	pfPagination cannot find 'length' fix	Jesse Buitenhuis	1
477	Programme Service Angular - knopen	Dimitri Tholen	2
478	Viewmodel Programme Overview Meesturen alle statussen	Dimitri Tholen	3
479	Account/Session Services en isAuthorized / isAuthenticated fixen	Jesse Buitenhuis	4
480	Dummydata partnermanagers in viewmodel Partner Detail vervangen voor echte data	Dimitri Tholen	1
481	Partner Edit naam + description in detailpagina mogelijk maken	Jesse Buitenhuis	2
482	Partner Service Angular knopen	Dimitri Tholen	6
483	programmeComponent Angular Service knopen	Dimitri Tholen	8
484	Ik ben een partner en ik wil elke unieke gebruiker binnen mijn omgeving verkrijgen	Dimitri Tholen	6
487	Fix/Refactor OverrideCombination extension method	Sebastiaan Alblas	6
489	Sendmail functionaliteiten	Dimitri Tholen	4
490	I am a component and i want to see all my linked components by type	Sebastiaan Alblas	4
491	I am a child component and i want to duplicate myself (als dit ergens wordt gebruikt...)	Sebastiaan Alblas	4
492	I am editing a programme and I override an attribute, creates a childcomponent in every layer	Sebastiaan Alblas	12
493	Verwijderen items programme, user etc	Jesse Buitenhuis	6
494	MergeUtilitiez OverrideCombination & OverrideComponent leesbaarder gemaakt en gerefactored	Sebastiaan Alblas	4
495	id ComponentCombinations wordt niet automatisch gevuld	Serge Bekenkamp	2

496	Status voor ComponentCombinations alleen active en inactive	Sebastiaan Alblas	3
499	ProgramLevelList : kolom inactive leest geen data in	Jesse Buitenhuis	2
501	Programme Overview	Dimitri Tholen	2
502	Viewmodel Modules Single (detail/edit): toevoegen gekoppelde contentitems, zelfde als levels en excitingmoments	Dimitri Tholen	2
543	Applicatie deployen	Dimitri Tholen	4
548	Auto login met cookie, en stuur User object naar Angular	Dimitri Tholen	1
549	Interceptor HTTP	Jesse Buitenhuis	1
560	Copy Programme / Create Child Programme	Dimitri Tholen	1
		Totaal felocity sprint 2 features afronden	150
96Server installeren			0
141	Transip.nl VPS huren	Dimitri Tholen	3
142	Inrichten Server	Dimitri Tholen	3
217 PartnerBeheer			0
250	Model defineren	Sebastiaan Alblas	1
251	Repositories	Sebastiaan Alblas	1
252	Service aanmaken	Sebastiaan Alblas	2
253	Unit Tests ontwerpen	Sebastiaan Alblas	3
258	Overzichtsscherm partners	Jesse Buitenhuis	2
259	Detailscherm partners	Jesse Buitenhuis	2
314	Beheerscherm Partner	Jesse Buitenhuis	2
319	ViewModels aanmaken	Jesse Buitenhuis	1

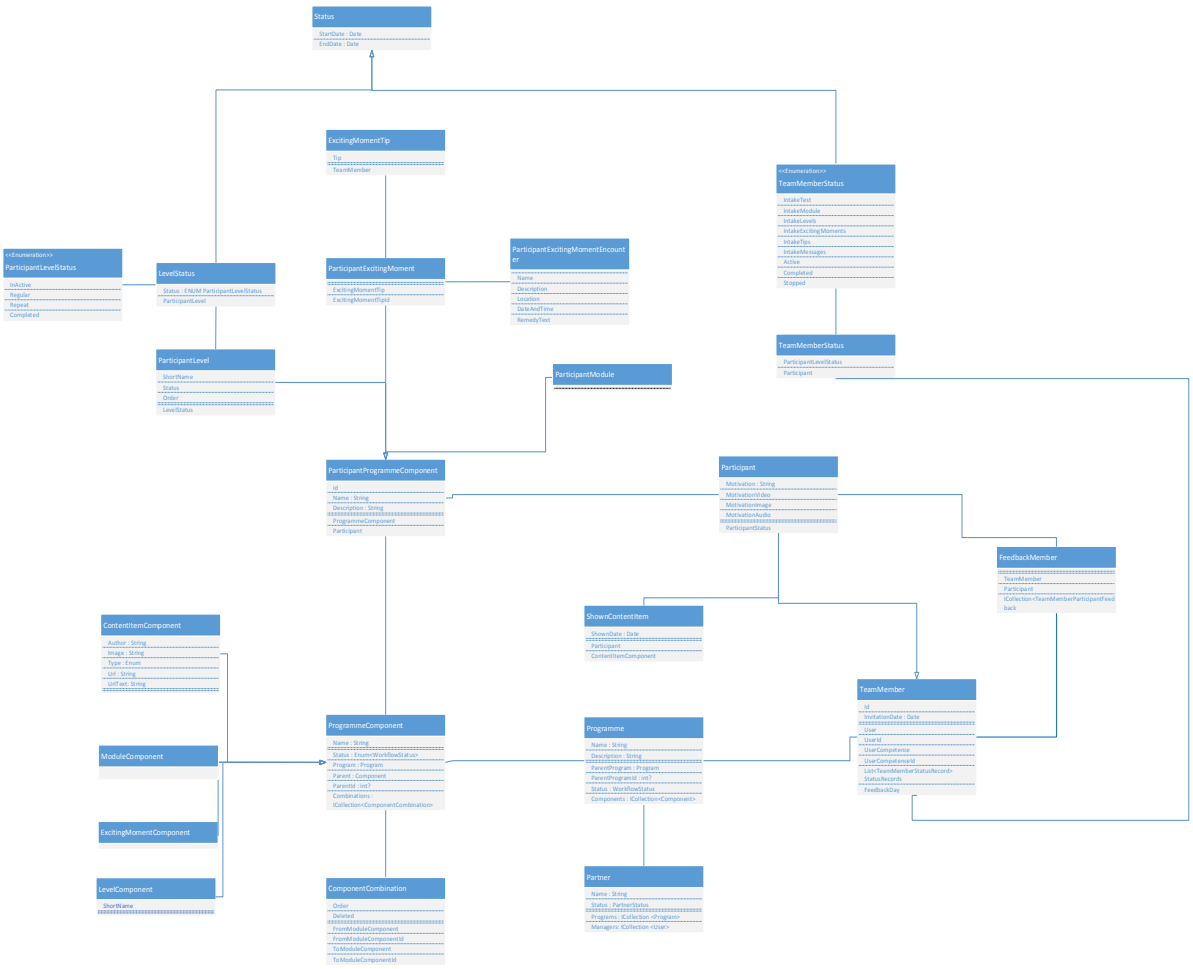
323	Een partner CRUD		2
324	Een status Actief of NonActief zetten		2
104 User Authentication			0
255	Repositories	Dimitri Tholen	4
254	Model Defineren	Dimitri Tholen	4
256	Service aanmaken	Dimitri Tholen	2
257	Unit tests ontwerpen	Dimitri Tholen	2
260	Inlogscherm	Jesse Buitenhuis	2
320	ViewModels aanmaken	Dimitri Tholen	2
321	Model Valideren	Dimitri Tholen	3
345	Als X user kan ik inloggen in de X	Jesse Buitenhuis	2
346	Een user die TeamMember is, komt na inloggen in de Front Office, ongeacht zijn andere rollen	Jesse Buitenhuis	2
348	Als hij ook andere rollen heeft, kan hij vanuit de Front Office naar de Back Office komen	Jesse Buitenhuis	2
349	Een user die geen TeamMember is, komt na inloggen in de Back Office	Jesse Buitenhuis	2
350	Syntaxes afspreken	iedereen	2
410	Iemand die geen teammembers is mag niet op front office pagina's komen	Jesse Buitenhuis	2
105 User Roles (user & superuser)			0
266	Model Defineren	Sebastiaan Alblas	1
269	Service aanmaken	Sebastiaan Alblas	1
270	Unit Test ontwerpen	Sebastiaan Alblas	3
271	Service en ViewModels aanmaken	Sebastiaan Alblas	2
325	User dat de volgende rollen kan bevatten	Dimitri Tholen	3
326	Een superuser heeft toegang tot alle partners, programma's en deelnemers	Dimitri Tholen	4

327	Een superuser kan alle users aanmaken, zien, bewerken en verwijderen	Dimitri Tholen	6
328	Een superuser kan alle rollen toekennen en bewerken (dus ook andere superusers)	Dimitri Tholen	5
329	Een superuser heeft alle rechten die een manager, program manager en user ook hebben	Dimitri Tholen	4
330	Een manager is gekoppeld aan een of meerdere partners	Dimitri Tholen	2
331	Een manager heeft toegang tot alle programma's en deelnemers die zijn gekoppeld aan de partner(s) waarvoor hij manager is	Dimitri Tholen	2
332	Een manager kan de aan de partner(s) opengestelde programma’s inzien en als basis gebruiken	Dimitri Tholen	6
333	Een manager kan alle teamleden aanmaken, zien, bewerken en verwijderen die zijn gekoppeld aan de partner(s) waarvoor hij manager is	Dimitri Tholen	2
334	Een manager kan de rollen manager, program manager en user toekennen en bewerken voor users gekoppeld aan de partner(s) waarvoor hij manager is	Dimitri Tholen	2
335	Een manager heeft alle rechten die een programma manager en user ook hebben voor de parter(s) waaraan hij is gekoppeld	Dimitri Tholen	2
336	Een program manager is gekoppeld aan een of meerdere programma's	Dimitri Tholen	2
337	Een program manager heeft toegang tot alle programma's waarvoor hij program manager is	Dimitri Tholen	2
339	Een program manager kan alle teamleden aanmaken, zien, bewerken en verwijderen die zijn gekoppeld aan de programma('s) waarvoor hij program manager is	Dimitri Tholen	2
340	Een program manager kan de rollen program manager en user toekennen en bewerken voor users gekoppeld aan de programma('s) waarvoor hij program manager is	Dimitri Tholen	2
341	Een program manager heeft alle rechten die een user ook heeft voor alle programma's waaraan hij gekoppeld is	Dimitri Tholen	2
342	Een user is gekoppeld aan een of meerdere programma's	Dimitri Tholen	2

343	Een user kan per programma de modules, levels en spannende momenten inzien	Dimitri Tholen	2
106 User status			0
279	Angular Services aanmaken	Jesse Buitenhuis	12
280	Angular Controllers aanmaken	Jesse Buitenhuis	12
360	Statussen X	Serge Bekenkamp	0
361	De volgende statussen zijn beschikbaar:	Sebastiaan Alblas	3
362	Wat moet er gebeuren bij elke status:	Jesse Buitenhuis	3
246User aanmelden		Jesse Buitenhuis	2
109Team member beheer (CRUD)		Jesse Buitenhuis	2
108Participant Status		Sebastiaan Alblas	2
364	Statussen X	Serge Bekenkamp	2
365	De volgende statussen zijn beschikbaar: X	Sebastiaan Alblas	3
208FeedbackMember			0
294	Statussen X	Sebastiaan Alblas	0
295	De volgende statussen zijn beschikbaar:	Sebastiaan Alblas	0
112Participant Program aanmaken			0
272	Intakeschermen participant	Jesse Buitenhuis	4
299	Service aanmaken	Sebastiaan Alblas	1
300	Model defineren	Sebastiaan Alblas	1
301	Repositories aanmaken	Sebastiaan Alblas	1
303	ViewModels aanmaken	Sebastiaan Alblas	5
244Participant Program inzien als beheerder			0

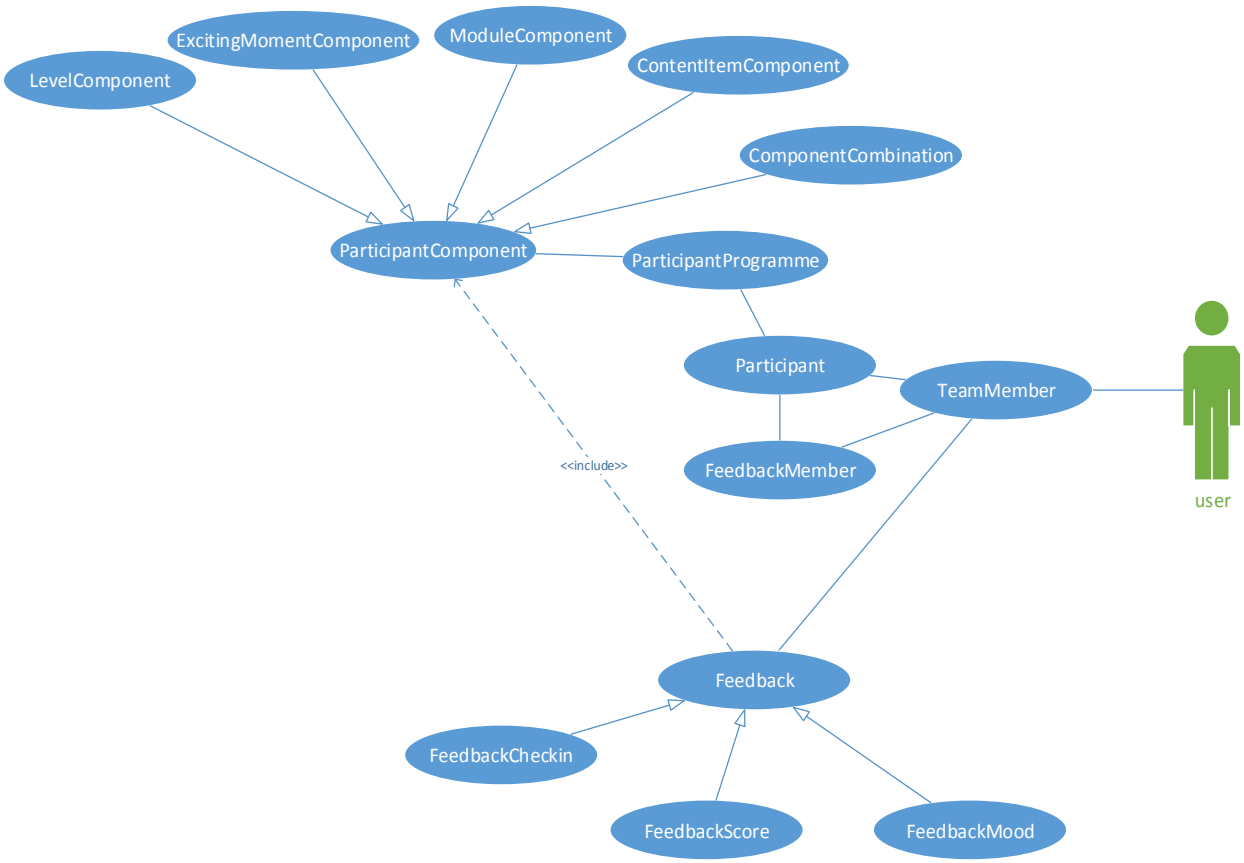
267	Detailscherm Participant	Jesse Buitenhuis	1
398	ViewModel Participant Program - Intake	Serge Bekenkamp	1
113	POP functionaliteit		0
368	Ik ben Participant en wil een POP-module volgen:	Sebastiaan Alblas	2
218	On Boarding FeebackMember		0
369	Ik ben FeedbackMember, heb een uitnodiging ontvangen van een Participant en ik wil:	Jesse Buitenhuis	2
			173

Domain Diagram



Sprint 3

Use Cases



User Stories

Sprint 3	
Feedback doorgeven	Als Participant en FeedbackMember kan ik dagelijks de volgende feedback doorgeven: score van 1 t/m 5 sterren op mijn level of NVT open feedback op deze score open feedback kan ik op prive zetten, zodat anderen deze niet zien op elk level dat is geselecteerd in het persoonlijke programma inchecken als dat lekker ging op totaalniveau een smiley (vrolijk, neutraal, verdrietig) Feedback kan worden doorgegeven met terugwerkende kracht Feedback in het verleden kan worden aangepast
Level halen	Als deelnemer behaal ik mijn level zodra ik op dat level 200 punten hebt behaald. Ik behaal op 3 manieren punten voor mijn level: Voor mijn eigen score (aantal punten = aantal sterren) Het gemiddelde van alle scores van mijn teamleden op een bepaalde dag. Deze wordt berekend zodra ik de scores cash. Het eerste moment dat dit mogelijk is, is om 9:00 de volgende dag. Als op het moment van cashen geen enkel teamlid feedback heeft doorgegeven, krijg ik om 9:00 nogmaals mijn eigen score als puntenaantal erbij Als ik na een puntenverlies feedback doorgeef, krijg ik de punten die ik heb verloren er weer bij

	Op sommige dagen kan een scoreverdubbelaar van kracht zijn. In dat geval krijg ik dubbele punten voor mijn eigen score Als ik een “topdag” heb gehad, ontvang ik bonuspunten. Een topdag is een dag dat ik van mezelf en al mijn teamleden (die een score hebben doorgegeven) minimaal 4 sterren heb gekregen, waarvan tenminste eenmaal 5 sterren. Dit wordt bepaald zodra alle scores binnen zijn of uiterlijk de dag erna omm 9:00 (als niet iedereen zijn score heeft doorgegeven). De eerste topdag wordt mijn puntentotaal aangevuld tot 40 punten, de 2e tot 80, 3e tot 120, 4e tot 160 en 5e tot 200 Ik kan ook punten verliezen als ik geen score doorgeef: Als ik na 3 reminders geen feedback heb doorgegeven, verlies ik bij het ontvangen van de 4e reminder 5 punten. Als ik na 4 reminders geen feedback heb doorgegeven, verlies ik bij het ontvangen van de 5e reminder 5 punten. Als ik na 5 reminders geen feedback heb doorgegeven, verlies ik bij het ontvangen van de 6e reminder 5 punten. Als ik na 6 reminders geen feedback heb doorgegeven, is bij de 7e reminder de scoreverdubbelaar van kracht NB: Feedback die is doorgegeven/gewijzigd nadat het puntentotaal is berekend (9:00 op de dag erna), hebben geen invloed meer op het puntentotaal Als ik mijn level/competentie haal (doordat ik mijn feedback cash) gebeurt het volgende: “feestelijke ervaring”: gefeliciteerd met het behalen van je level interactie om gevoel van succes te versterken volgende level: als default het geselecteerde level, met de keus om een ander level te kiezen de eerstvolgende feedback gaat over het nieuwe level
Dashboard	Ik ben user binnen een programma en wil in mijn dashboard zien: De laatste scores van mezelf en van mijn teamleden, waarbij er ook een teamscore wordt bijgehouden (het gemiddelde van de tot dat moment doorgegeven scores van teamleden op die dag). De scores moeten allemaal op dezelfde dag betrekking hebben, dus zodra de eerste score voor de nieuwe dag is doorgegeven, worden de andere scores leeggemaakt) → de “Jury” / Swipen door de verschillende teams waarin ik deelneem (incl. voortgang) / Houdt in design rekening met 0-3 teamleden + de deelnemer zelf De voortgang binnen mijn level en binnen mijn module (houd in design rekening met 1-10 levels) Mijn timeline (zie PBI ‘Timeline’) Mijn coach (als ik die heb) De ranglijst (als dit is ingeschakeld, *customization*) Een lijst met meldingen (achter een icoontje met aantal ongelezen meldingen, vgl. Facebook) Een lijst met vriendschapsverzoeken (achter een icoontje met aantal ongelezen meldingen, vgl. Facebook) (als dit is ingeschakeld, *customization*) Een lijst met berichten (achter een icoontje met aantal ongelezen berichten, vgl. Facebook) Ik ben deelnemer binnen een programma en wil in mijn dashboard kunnen doen: Feedback doorgeven (zie PBI ‘Feedback doorgeven’) voor alle teams waarin ik deelneem en voor alle dagen De scores van mijn teamleden ‘unlocken’ (dat kan pas als ik zelf feedback heb doorgegeven) - alleen als ik teamleden heb uiteraard Als ik mijn score doorgeef, zie ik het aantal punten in mijn progress bar oplopen Als ik een score van teamlid unlock, zie ik de teamscore veranderen. Als ik de laatste score van een teamlid unlock, zie ik het aantal punten van de de teamscore in mijn progress bar oplopen Levels kunnen wijzigen (volgorde wijzigen, level verwijderen, level toevoegen)

	Reageren op r: like / vraag / reactie Mijn timeline filteren (afeedback van een andelleen feedback op mijn eigen module zien, alleen coachberichten zien) Een bericht naar mijn coach sturen Spannende momenten agenderen (uitklapbalk?) Mijn motivatie oproepen (bijv. schilderij, overlay) en wijzigen. Motivatie komt in beeld op het moment dat ik feedback doorgeef Vrienden uitnodigen en uitnodigingen accepteren/weigeren (als dit is ingeschakeld, *customization*) Ik ben user binnen Perflectie en ik wil in mijn dashboard kunnen doen: Feedback doorgeven (zie PBI 'Feedback doorgeven') voor alle teams waarin ik deelneem en voor alle dagen Posten, reageren, delen, liken en disliken (alleen van coachberichten) van berichten op mijn timeline Een kudos, por of persoonlijk bericht sturen naar een van mijn vrienden
Call-to-actions **(meldingen)**	Ik ben user binnen Perflectie en bij bepaalde acties krijg ik een call-to-action: **Specifiek:** Openstaande feedback doorgeven (andere teams, gisteren) - voor 12:00 wordt feedback gisteren gevraagd, na 12:00 feedback van vandaag Eerste feedback vragen aan deelnemer of teamlid (als 5 werkdagen geen reactie) Feedback vragen aan deelnemer of teamlid (als 5 werkdagen geen feedback) Feliciteren met succes: Na behalen competentie (vrienden) Na afronden level (vrienden) Na doorgeven 50e, 100e (en dan elk 50-tal) feedback (teamleden) Fijne vakantie wensen 3,2,1 dagen voor start pauzeperiode van teamlid Openstaande intake afronden (geen optie nee bedankt) Level bekijken bij start nieuw level **Generiek** Spannende momenten doorgeven (als er geen spannende momenten zijn geagendeerd) Teamleden uitnodigen (bij 0 teamleden) - wat is de goede timing van dit bericht? Vrienden uitnodigen (indien nog niet op programma niveau bepaald) Profiel aanvullen (bijv. foto) Later uitwerken Spannende momenten evalueren Motivatie evalueren Extra requirements Vorm om call-to-action te presenteren kan zijn: uitlichten van onderdeel dat de aandacht moet krijgen of donkerder maken van de rest. Of "wolkje met pijl" [Na de CTA voor feedback doorgeven krijg je nog maximaal 1 andere CTA Als reactie op de CTA kun je doorgeven: "nee, bedankt", "herinner me later" Na "nee, bedankt" wordt de CTA (voor deze case) niet meer getoond Na "herinner me later" wordt de CTA de volgende keer bij het inloggen opnieuw getoond als die nog steeds van toepassing is Wegklikken wordt geinterpreteerd als "herinner me later" CTA's voor een specifieke case worden maximaal 3x getoond, tenzij er geen andere CTA is om te tonen Alle "actieve" CTA's worden als melding in het meldingen menu opgenomen Het tellertje van de meldingen telt de melding mee zolang deze openstaat (dat is: niet is opgevolgd of geweigerd)

Instant notifications Deelnemer	Ik ben user binnen Perflectie en bij bepaalde acties krijg ik een instant bericht binnen het platform: *Als ik [actie], krijg ik het bericht "[inhoud bericht]"* Wanneer bepaalde acties tot meerdere berichten leiden, wordt alleen de eerste getoond Meerdere berichten per actie mogelijk Ik ben deelnemer en ontvang de volgende berichten Als ik inlog: Als ik mijn 1e feedback doorgeef: als 4 of 5 sterren: meeste sterren tot nu toe op dit level Als ik mijn 1e feedback na een pauze doorgeef: Als ik mijn 1e feedback na een retentiebericht doorgeef: Als ik mijn 10e, 25e, 50e, 100e (en dan elk 50-tal) feedback doorgeef: Als ik voor de x-ste dag achter elkaar mijn feedback doorgeef: Als ik mijn x-ste feedback doorgeef: 1 ster: 2 sterren: 3 sterren: 4 sterren: 5 sterren: Als ik mijn eerste smiley doorgeef (blij): Als ik mijn eerste smiley doorgeef (neutraal): Als ik mijn eerste smiley doorgeef (droevig): Als ik mijn 3e (of meer) achtereenvolgende blije smiley doorgeef: Als ik mijn 3e (of meer) achtereenvolgende droevige smiley doorgeef: Als ik een blije smiley doorgeef: Als ik een neutrale smiley doorgeef: Als ik een droevige smiley doorgeef: Als ik voor het eerst een level incheck: Als ik alle levels van deze dag incheck: Als ik feedback van een teamlid unlock en daarmee mijn competentie behaal: special effect en daarmee mijn level behaal: special effect en daarmee staat vast dat ik een topdag heb: en als ik >1 ster meer heb doorgegeven dan een teamlid: en als ik >1 ster minder heb doorgegeven dan een teamlid: meeste sterren van teamlid op dit level tot nu toe: (gerelateerd aan een lagere score, dus nooit de eerste) 1 ster: 2 sterren: 3 sterren: 4 sterren: 5 sterren: en als dit de 10e, 25e, 50, 100e (en daarna elk 50-tal) keer is dat hij jou feedback heeft gegeven Als ik de feedback cash en ik kan bij de volgende feedback mijn level halen: en ik heb een topdag gehad: en ik heb gemiddeld >= 3 sterren behaald: en ik heb gemiddeld <3 sterren behaald: Als ik een teamlid por om te starten met feedback geven: Als ik een teamlid por om feedback door te geven: Als ik een spannend moment heb doorgeven: Als ik binnen 2 maanden 3x een coachberichten uit dezelfde categorie heb gedisliked: Als ik een direct ingaande pauze doorgeef: Als ik stop met Perflectie:

	badge behaald (later uitwerken) teskt in reactieveld aanpassen obv like/dislike contentitem Als ik een instant bericht heb gehad, wordt deze daarna aan mijn timeline toegevoegd (zie PBI 'Timeline' welke berichten daarop terecht komen)
Instant notifications Teamlid	Ik ben teamlid en ontvang de volgende berichten: (Teamlid kan ook feedback unlocken. Gelijk trekken aan deelnemer) Als ik inlog: Als ik mijn feedback doorgeef en de deelnemer daarmee zijn competentie behaalt: (special effect) Als ik mijn feedback doorgeef en de deelnemer daarmee zijn level behaalt: (special effect) Als ik mijn feedback doorgeef en daarmee staat vast dat de deelnemer een topdag heeft: Als ik mijn 1e feedback doorgeef: Als ik mijn 1e feedback na een pauze doorgeef: Als ik mijn 1e feedback na een retentiebericht doorgeef: Als ik mijn 10e, 25e, 50e, 100e feedback doorgeef: Als ik voor de x-ste dag achter elkaar mijn feedback doorgeef: Als ik mijn x-ste feedback doorgeef: meeste sterren tot nu toe op dit level: 1 ster: 2 sterren: 3 sterren: 4 sterren: 5 sterren: Als ik >1 ster meer doorgeef dan de deelnemer: Als ik >1 ster minder doorgeef dan de deelnemer: Als ik mijn eerste smiley doorgeef (blij): Als ik mijn eerste smiley doorgeef (neutraal): Als ik mijn eerste smiley doorgeef (droevig): Als ik mijn 3e (of meer) achtereenvolgende blije smiley doorgeef: Als ik mijn 3e (of meer) achtereenvolgende droevige smiley doorgeef: Als ik een blije smileys doorgeef: Als ik een neutrale smiley doorgeef: Als ik een droevige smiley doorgeef: Als ik voor het eerst een level incheck: Als ik het eerste level van deze dag incheck: Als ik het tweede tot een-na-laatste level van deze dag incheck: Als ik het laatste level van deze dag incheck: Als ik de deelnemer of een ander teamlid por om feedback door te geven: Als ik een direct ingaande pauze doorgeef: Als ik stop met Perflectie: badge behaald (later uitwerken)
Retentieberichten	Ik ben deelnemer en ik heb een aantal dagen geen feedback doorgegeven Fase 1: 5 werkdagen achtereen geen feedback bericht in FeedbackReminder deelnemer CTA voor teamleden bericht in morning message en alert teamleden Fase 2: 6-7 werkdagen achtereen geen feedback bericht in FeedbackReminder deelnemer CTA voor teamleden bericht in morning message en FeedbackReminder teamleden Fase 3: 7+ werkdagen achtereen geen feedback bericht in FeedbackReminder deelnemer CTA voor teamleden bericht in morning message en FeedbackReminder teamleden

	bericht in Timeline vrienden Ik ben teamlid en ik heb een aantal dagen geen feedback doorgegeven Fase 1: 5 werkdagen achtereen geen feedback bericht in FeedbackReminder teamlid CTA voor deelnemer Fase 2: 6-7 werkdagen achtereen geen feedback bericht in FeedbackReminder teamlid CTA voor deelnemer Fase 3: 7+ werkdagen achtereen geen feedback bericht in FeedbackReminder teamlid CTA voor deelnemer
Leerinterventies	Ik ben deelnemer en bij de volgende feedback ontvang ik een leerinterventie: nieuw maximum: een teamlid heeft mij voor het eerst 4 of 5 sterren gegeven (per teamlid) eerste keer 5 sterren op het level van een teamlid (eenmalig) verschil team: een teamlid heeft mij >1 meer of minder dan gegeven dan ik mijzelf heb gegeven level bijna gehaald: ik kan vandaag mijn level halen (eenmalig) Ik ben deelnemer en ontvang een leerinterventie ik krijg bericht hierover zodra ik de feedback unlock / eigen feedback doorgeef. Daarna komt dit bericht in de timeline (onder de feedback waarop de interventie betrekking heeft). Bericht komt direct in timeline als ik deze feedback nog niet heb gezien, maar er al feedback van een nieuwe dag is doorgegeven teamleden krijgen bericht hierover in de timeline
Coachberichten	Ik ben deelnemer en ik ontvang coachberichten om mijn persoonlijke ontwikkeling te versnellen: Ik kan instellen met welke frequentie en van welk type ik coachberichten wil ontvangen De berichten worden in mijn timeline geplaatst op de volgorde zoals vastgelegd in het ontwikkelprogramma. Er worden alleen berichten geplaatst die nog niet eerder aan mij getoond zijn De berichten worden in de timeline van mijn teamleden (die zelf niet deelnemen aan Perflectie) geplaatst voor zover ze nog niet eerder zijn getoond aan mijn teamlid
Berichtencentrum	Ik ben user in Perflectie en ik wil berichten versturen en ontvangen berichten hebben een afzender en ontvanger (altijd users binnen Perflectie) Perflectie of een (virtuele) coach kunnen afzender zijn ontvanger kan een groep zijn (Feedbackteam, Vrienden) berichten hebben een datum/tijdstip van verzending berichten hebben een status gelezen/ongelezen berichten worden verstuurd via een kanaal: melding, chat, mail, SMS berichten hebben een prioriteit (hoog/midden/laag) berichten zijn herkenbaar aan een berichtcode Ik ben superuser, manager of program manager en ik wil de Perflectie berichten van een user inzien op het detailscherm van de user: berichtcode, verzenddatum, kanaal, meegezonden attributen filteren op berichtcode
Timeline	Ik ben Participant binnen Perflectie en ik zie in mijn timeline: (tussen haakjes de prioriteit H/M/L) Feedback van teamleden (score op level, open feedback, emoticon en aantal ingecheckte levels), zodra ik de feedback heb gecashed (H) Leerinterventie van Perflectie obv de Feedback (H)

	Level behaald (H) Badge behaald (M) Coachberichten van Perflectie (M) Start en einde pauzeperiode teamlid (L) Stoppen van een teamlid (L) Ik ben FeedbackMember binnen Perflectie en ik zie in mijn timeline: Feedback van teamleden (score op level, open feedback, emoticon en aantal ingecheckte levels) (M) Leerinterventie van Perflectie obv de Feedback (M) Level behaald (H) Badge behaald (L) Coachberichten van Perflectie, als die nog niet eerder aan deze user zijn getoond (L) Start en einde pauzeperiode deelnemer (L) Stoppen van de deelnemer (L) Ik ben Vriend binnen Perflectie en ik zie in mijn timeline: Level behaald door een vriend (H) Retentiebericht voor een vriend (H) Badge behaald door een vriend (L) Posts van mezelf en van vrienden (M) Gedeelde berichten van vrienden (M) Reacties op posts in mijn timeline (M) Likes van posts in mijn timeline (M) De volgorde van de timeline wordt als volgt bepaald: Berichten worden gesorteerd op dag, de meest recente bovenaan Berichten van dezelfde dag worden getoond in volgorde van prioriteit, eerst hoog, dan midden, dan laag Berichten met dezelfde prioriteit worden getoond in volgorde van verzendtijdstip, meest recente bovenaan Ik ben user binnen Perflectie met een timeline en ik wil: Berichten posten, waarbij ik afbeelding of link (bijv naar video of artikel) kan toevoegen Een post kan prive, voor teamleden of voor vrienden (default) zijn Reageren op berichten van anderen Berichten van anderen delen met mijn teamleden of mijn vrienden (default) Berichten kunnen liken Coachberichten kunnen disliken Filteren op mijn eigen Feedback berichten Filteren op mijn eigen coachberichten

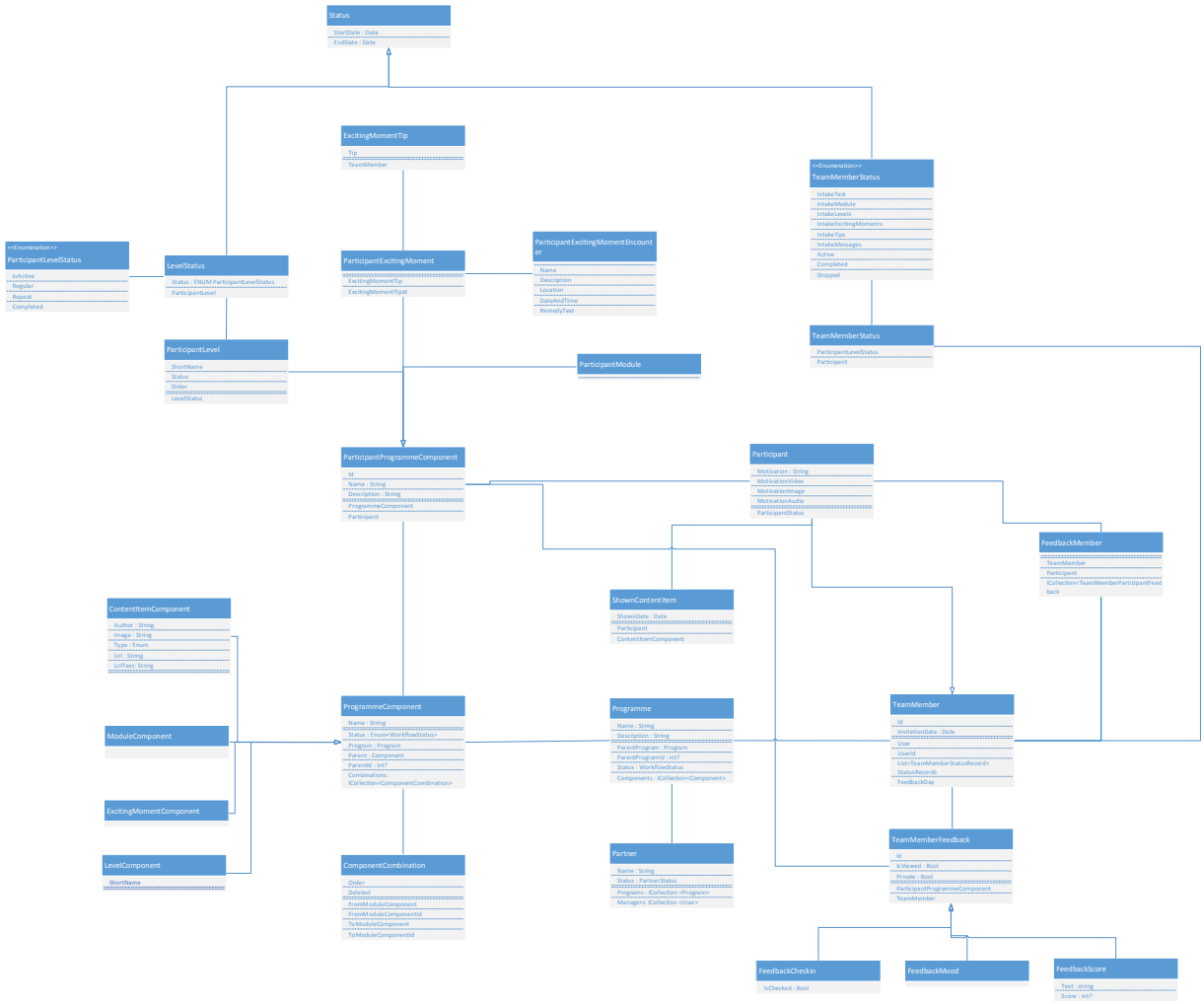
Sprint backlog

ID	Title 1	Title 2	Assigned To	Felocity
115	Feedback doorgeven		Sebastiaan Alblas	0
506		Ik ben deelnemer en geef mezelf feedback op een bepaalde datum op het actieve ParticipantLevel	Sebastiaan Alblas	3
507		Ik ben deelnemer en geef mezelf op een bepaalde dag een smiley op mijn ParticipantModule	Sebastiaan Alblas	3

508	Ik ben deelnemer en check op een bepaalde dag 1 of meer ParticipantLevels in	Sebastiaan Alblas	3
509	Ik ben feedbackmember en geef feedback op een bepaalde datum op het actieve ParticipantLevel	Sebastiaan Alblas	3
510	Ik ben feedbackmember en geef op een bepaalde dag een smiley op mijn ParticipantModule	Sebastiaan Alblas	3
511	Ik ben feedbackmember en check op een bepaalde dag 1 of meer ParticipantLevels in	Sebastiaan Alblas	3
512	Als ik als deelnemer of FeedbackMember feedback geef op het actieve ParticipantLevel wordt dit level automatisch ingecheckt	Sebastiaan Alblas	3
513	Als deelnemer kan ik mijn open feedback als prive markeren	Sebastiaan Alblas	3
514	Als feebackmember kan ik mijn open feedback als prive markeren	Sebastiaan Alblas	3
537	Model ontwerpen Feedbackgeven teammember binnen een Participant programme	Sebastiaan Alblas	3
538	Ontwerp Model omzetten naar Code	Sebastiaan Alblas	4
539	TeamMemberFeedback Unit tests Red	Sebastiaan Alblas	4
540	TeamMemberFeedback Unit Tests Green	Sebastiaan Alblas	4
541	TeamMemberFeedback Refactor	Sebastiaan Alblas	4
542	TeamMemberFeedback ViewModel + Controllers	Sebastiaan Alblas	2
118	Dashboard design en view		0
411	Inspiratie opzoeken design	Jesse Buitenhuis	2
412	Wireframe dashboard	Jesse Buitenhuis	4
413	Scorepaneel	Jesse Buitenhuis	3
414	Voortgang	Jesse Buitenhuis	5
415	Timeline	Jesse Buitenhuis	12
416	Coach + Coachberichten	Jesse Buitenhuis	3
417	Ranglijsten	Jesse Buitenhuis	2
418	Notifications, FriendRequests, Berichten	Jesse Buitenhuis	4

419	TeamMember: Feedback doorgeven	Jesse Buitenhuis	4
420	Ik ben user en zie pas mijn score als ik zelf feedback heb doorgegeven	Jesse Buitenhuis	2
421	Animaties progressbar en scores	Jesse Buitenhuis	3
422	Level wijzigingen	Jesse Buitenhuis	2
423	Interactie tussen users: Like, vraag, reactie	Jesse Buitenhuis	1
424	Timeline Groepen/Filters	Jesse Buitenhuis	2
425	Spannende Momenten Agenderen	Jesse Buitenhuis	1
426	Motivatie op kunnen roepen	Jesse Buitenhuis	1
427	Vrienden - Zoeken, Uitnodigen, Accepteren	Jesse Buitenhuis	1
428	Berichten sturen / Kudos geven etc	Jesse Buitenhuis	1
429	CallToActions uitdenken	Jesse Buitenhuis	2
430	Instant Notifications	Jesse Buitenhuis	1
431	Retentieberichten in timeline	Jesse Buitenhuis	2
432	Dashboard Ontwerpen	Jesse Buitenhuis	31
555	Datepicker installeren en aanpassen aan design	Jesse Buitenhuis	2
561	Creeeren algemene feedbackknop desktop site	Jesse Buitenhuis	1
			140

Domain Diagram



Sprint 4

Use Cases

Niet gedefinieerd voor sprint 4

User Stories

Sprint 4	
Feedback Reminder (SMS e/o e-mail)	Ik ben een user en ik wil een reminder om feedback door te geven Op de dagen en het tijdstip dat ik heb doorgegeven Via de kanalen (SMS en/of e-mail) die ik heb doorgegeven De tekst in de reminder hangt af van een aantal variabelen (de bovenste die van toepassing is, is bepalend voor het bericht) level gehaald retentie fase 1 retentie fase 2 retentie fase 3 nieuw maximum (als niet in ochtendmail) eerste keer 5 sterren op het level van een teamlid (als niet in ochtendmail) verschil team (als niet in ochtendmail) level bijna gehaald (als niet in ochtendmail) emoticon x in a row In de mail is het aantal ongelezen feedbacks te zien (als >0) In de mail is het aantal ongelezen chatberichten te zien (als >0 en bovenstaande =0) In de mail is het aantal ongelezen coachberichten te zien (als >0 en bovenstaande = 0) [met tekst en afbeelding] In de mail is het aantal likes op jouw posts te zien (als >0 en bovenstaande = 0) In de mail staat het als een teamlid vandaag is gepauzeerd of gestopt
Morning Message **E-mail**	Ik ben user en wil een ochtendbericht ontvangen ik wil instellen of ik dit bericht wel of niet ontvang ik ontvang dit bericht op de dagen dat ik een Feedback Reminder ontvang Het bericht wordt verstuurd om 9:00 De tekst in de reminder hangt af van een aantal variabelen nieuw maximum eerste keer 5 sterren op het level van een teamlid verschil team level bijna gehaald emoticon x in a row Referentie naar coachbericht: “er staat nog een [inspirerende video] voor je klaar [incl. de afbeelding] In de mail is het aantal ongelezen feedbacks te zien (als >0) In de mail is het aantal ongelezen chatberichten te zien (als >0 en bovenstaande =0) In de mail is het aantal ongelezen coachberichten te zien (als >0 en bovenstaande = 0) In de mail is het aantal likes op jouw posts te zien (als >0 en bovenstaande = 0) In de mail staat het als een teamlid vandaag is gepauzeerd of gestopt
Instant message (e-mail of SMS)	Ik ben user en in de volgende situaties ontvang ik een bericht per mail of SMS Iemand bij wie ik in het team zit heeft zijn level gehaald: Ik ontvang een e-mail. In dit geval wordt er geen morning message meer verstuurd Ik stop als deelnemer: mailbericht met mogelijkheid om toch nog verder te gaan Iemand anders vraagt mij om feedback door te geven: als mijn Feedback Reminder normaal via SMS wordt verstuurd ontvang ik een SMS bericht, anders een e-mail bericht

Chatfunctionaliteit	Ik ben user en wil een persoonlijk bericht sturen naar een andere user ik kan alleen berichten sturen naar vrienden en coach voor functionaliteit zie huidige platform en/of Facebook in mijn dashboard zie ik een melding dat ik ongelezen berichten heb
Vrienden	Ik ben user binnen Perflectie en ik wil vrienden maken: Ik kan andere users binnen Perflectie uitnodigen als vriend Ik kan uitnodigingen van andere users accepteren of weigeren ik kan vriendschappen verbreken ik ben automatisch vriend met mijn FeedbackMembers en met de Partcipants bij wie ik in het team zit Per programma is instelbaar of alle deelnemers van het programma automatisch vriend van elkaar zijn
Coach	Ik ben user binnen Perflectie en vervul de rol als coach van een of meer deelnemers: een coach kan zijn gekoppeld aan een deelnemer een coach kan zijn gekoppeld aan een programma. De coach is dan automatisch coach voor alle deelnemers in het programma, mits er geen coach aan de deelnemer zelf is gekoppeld de coach is zichtbaar in het dashboard van de deelnemer en de deelnemer kan via de chatfunctie contact opnemen met de coach de coach heeft een overzichtscherm in de BO van zijn coachees (de deelnemers die aan hem zijn gekoppeld), cf het deelnemersoverzicht in het programmascherm. Dit scherm is te filteren op programma de coach kan doorklikken op een deelnemer om in het persoonlijk moduleoverzicht te komen de coach heeft een eigen timeline met dezelfde informatie en mogelijkheden als ware hij vriend van al zijn coachees (hij ziet dus de berichten die de vrienden van de coachee krijgen, kan reageren op posts en kan zelf posts plaatsen die bij alle coachees op de timeline komen)
Programmaberichten	Ik ben superuser, manager of programma manager en wil een bericht plaatsen op de timeline van alle deelnemers in het programma
Spannende momenten agenderen	Ik ben Participant en ik wil spannende momenten kunnen agenderen: naam van het spannende moment, begintijd, eindtijd, soort spannend bericht (je kunt hierbij kiezen uit de spannende momenten die je tijdens de intake hebt geselecteerd) LATER: je kunt iemand uitnodigen om jouw na afloop van het spannend moment eenmalig feedback te geven je kunt instellen of je 15 minuten voor het moment een mail, een SMS of niets wilt ontvangen (algemene instelling, niet per moment) je kunt de spannende momenten die je in je intake hebt geselecteerd wijzigen, verwijderen of toevoegen
Ranglijst	Perflectie houdt een ranglijst bij van actieve deelnemers: voor elke dag dat je feedback doorgeeft op tenminste 1 module van jezelf of een ander ontvang je een “punt” de punten worden bijgehouden in de categorie “voor jezelf” of “voor een ander”, zodat we hier later desgewenst onderscheid in kunnen maken je ziet de ranglijst van je vrienden incl. jezelf je staat zelf altijd op de 3e positie (tenzij je op 1 of 2 staat) je ziet de ranglijst over de lopende kalendermaand je kunt je vrienden aanmoedigen of complimenteren inspiratie voor visuele weergave: iPhone App Runkeeper
Look & feel	Als partner kan ik mijn eigen look & feel in de applicatie gebruiken (nog niet door partner instelbaar):

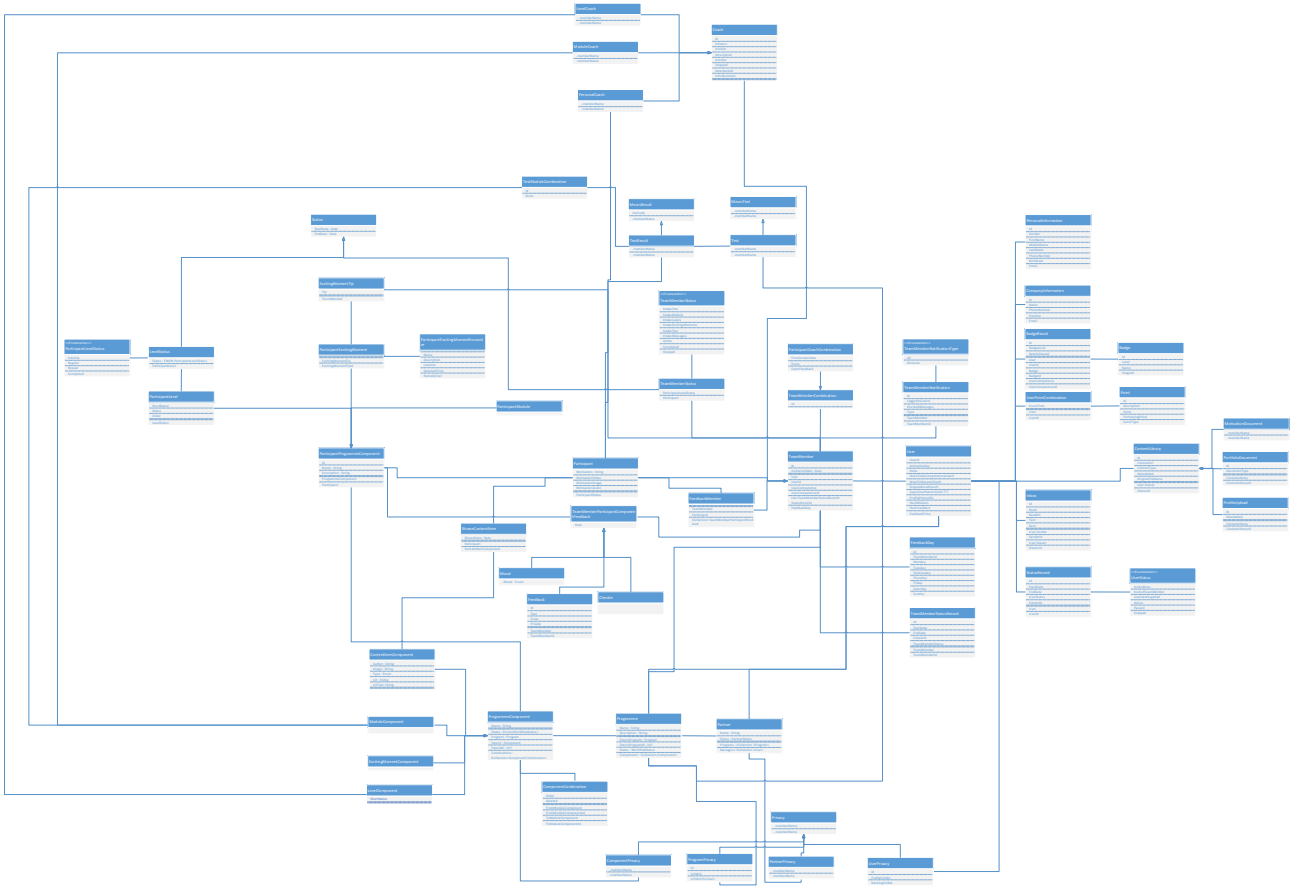
	eigen logo de 4 primaire kleuren De look & feel wordt doorgevoerd in de front office applicatie voor deelnemers binnen de programma's van de partner De look & feel wordt vooralsnog niet doorgevoerd in de back office, tenzij dit heel eenvoudig is.
Eigen taal	Als partner kan ik voor de de volgende termen mijn eigen termen gebruiken (nog niet door partner instelbaar): motivatie competentie/module level spannende moment feedbackteam feedbackteamlid vooruitgang feedback reflectie termen in de thermometer (talenten ontdekken, persoonlijk doel, motivatie & inspiratie, route naar succes, spannende momenten, tips & tricks, samen leren, meten = weten) NB: in de keuze van andere taal moten we rekening houden met aanverwante termen: meervoud competentie selecteren/ontwikkelen feedback doorgeven/vragen/geven spannende momenten doorgeven/hebben reflecteren De terminologie wordt gebruikt in de front office en alle communicatie middels het berichtencentrum De terminologie wordt vooralsnog niet doorgevoerd in de back office
Aantal levels variabel	Als beheerder binnen een programma kan ik instellen hoeveel levels de deelnemers in het programma kunnen selecteren. Als deelnemer kan moet ik in de intake exact dit aantal levels selecteren Als ik later levels wijzig moet ik exact dit aantal levels behouden
Intake flexibel instelbaar	Als programma kan ik bepalen welke stappen van de intake ik beschikbaar wil stellen en in welke volgorde (nog niet noodzakelijk door programma instelbaar). De volgende stappen zijn beschikbaar om te selecteren: test module (verplicht - voor levels) motivatie levels (verplicht - na module) spannende momenten tips (alleen beschikbaar als spannende momenten is geselecteerd - na spannende momenten) feedbackteam berichteninstellingen (verplicht) De deelnemer krijgt deze stappen te zien in de intake.

	De elementen die niet zijn geselecteerd komen niet terug in de front office en alle communicatie middels het berichtencentrum

Sprint backlog

Niet gedefinieerd voor sprint 4.

Domain Diagram



I : Interviews : Het nieuwe online-platform & gemaakte functionaliteiten

Sebastiaan Alblas

Naam : Sebastiaan Alblas
Functie: Backend developer

• Kan ik code vinden dat gerelateerd is aan een specifiek probleem of verandering?
Problemen kunnen gemakkelijk gevolgd worden naar de kern. Hierdoor is het gemakkelijk om een probleem op te lossen. Als bij een verandering het systeem problemen krijgt dan is dit gemakkelijk af te lezen bij de real-time tests.

• Kan ik code begrijpen? Kan ik uitleggen wat de bedoeling is achter de functionaliteiten aan iemand anders?

Sommige code is erg complex maar dan is het na ongeveer 10 minuten te begrijpen. Hierdoor is de merendeel van de code gemakkelijk uit te leggen. Dit is vooral door de vastgelegde principes, methodes en technieken.

• Is het gemakkelijk om de code te veranderen? Is het gemakkelijk voor mij om vast te stellen wat er veranderd moet worden als consequentie van de vorige verandering? Zijn de hoeveelheden en omvang van deze consequentie klein?

De code is gemakkelijk te veranderen. Het is helaas niet goed vastgelegd in documentatie. Dit is jammer want dit vereist uitzoekwerk, wat tijd kost. Maar als er veranderingen plaatsvinden binnen het systeem is dit gemakkelijk te herleiden tot de rest van het systeem met behulp van real-time tests.

• Kan ik snel een verandering binnen het systeem verifiëren?

Door middel van real-time tests is er een overzicht te zien of het systeem aan alle tests voldoet. Hiermee wordt het systeem continu geverifieerd.

• Kan ik een verandering doorvoeren met een laag risico dat er bestaande functionaliteiten omvallen?

Door middel van real-time tests is het gemakkelijk te zien of je een functionaliteit binnen het systeem aantast.

• Als er een bug ontstaat, is het probleem snel en gemakkelijk te detecteren en te diagnoseren?

Door middel van de debug mode van Visual studio is het gemakkelijk om het probleem te diagnoseren. Doormiddel van breakpoints kun je stap voor stap door het systeem lopen en zo het punt detecteren waar het fout gaat. Dit kan soms wel sloom gaan omdat elke verandering gekeken moet worden of het werkt. Hiervoor moet het project gebuild worden en dit kost veel tijd voor kleine veranderingen.

Serge bekenkamp

Naam : Serge Bekenkamp
Functie: Backend developer

• Kan ik code vinden dat gerelateerd is aan een specifiek probleem of verandering?
Over het algemeen is het weinig tot geen moeite om de code te vinden voor een probleem doordat het opgesplitst is in 3 delen namelijk data, services, en api's

• Kan ik code begrijpen? Kan ik uitleggen wat de bedoeling is achter de functionaliteiten aan iemand anders?
Ik denk dat voor 95% van de code makkelijk is uit te leggen wat het doet. Dit doordat we goede standaarden hebben afgesproken over de namen van functies en variabelen waardoor het meteen duidelijk is wat ze doen.

• Is het gemakkelijk om de code te veranderen? Is het gemakkelijk voor mij om vast te stellen wat er verandert moet worden als consequentie van de vorige verandering? Zijn de hoeveelheden en omvang van deze consequentie klein?

Op wat uitzonderingen na in code die heel veel gebruikt wordt is het vrij makkelijk om code te veranderen

• Kan ik snel een verandering binnen het systeem verifiëren?
ja door de unit tests die zijn geschreven is het makkelijk om te zien of er businessrules geschonden worden.

• Kan ik een verandering doorvoeren met een laag risico dat er bestaande functionaliteiten omvallen?
ja door de unit tests van de service en datamodel is het makkelijk om te zien wanneer bestaande functionaliteit niet meer werkt door een aanpassing

• Als er een bug ontstaat, is het probleem snel en gemakkelijk te detecteren en te diagnoseren?
Meeste problemen die we tot nu toe gehad hebben waren makkelijk te detecteren doordat deze vooral in de ui / api laag zaten. Dit was dan meestal snel gediagnostiseerd door een of 2 breakpoints op de plekken waar die specifieke functionaliteit werd aangeroepen.

Dimitri tholen

Naam : Dimitri Tholen
Functie: Backend developer / Specialist

• Kan ik code vinden dat gerelateerd is aan een specifiek probleem of verandering?
Het is gemakkelijk om code die door andere mensen binnen het team is gemaakt te begrijpen en bij het optreden van problemen te veranderen.

• Kan ik code begrijpen? Kan ik uitleggen wat de bedoeling is achter de functionaliteiten aan iemand anders?
Omdat wij onderling dezelfde methodes hebben vastgelegd is het gemakkelijk om functionaliteiten uit te leggen en te begrijpen. Alleen de code die gemaakt is in het begin van dit project voldoet nog niet volledig aan de vastgelegde methodes.

• Is het gemakkelijk om de code te veranderen? Is het gemakkelijk voor mij om vast te stellen wat er verandert moet worden als consequentie van de vorige verandering? Zijn de hoeveelheden en omvang van deze consequentie klein?
De impact van verandering is gemakkelijk te zien. Door middel van de real-time tests kan je zo een overzicht van het systeem zien of er geen business rules gebroken worden. Alleen aan het eind van dit project is er te gehaast gewerkt waardoor de unit tests niet geheel betrouwbaar meer zijn.

• Kan ik snel een verandering binnen het systeem verifiëren?
Door middel van de real-time tests hoor je het systeem gemakkelijk te kunnen verifieren

• Kan ik een verandering doorvoeren met een laag risico dat er bestaande functionaliteiten omvallen?
Door middel van de unit tests moet het wijzigen of breken van functionaliteiten op tijd gediagnoseerd kunnen worden.

• Als er een bug ontstaat, is het probleem snel en gemakkelijk te detecteren en te diagnoseren?
Tot nu heb ik geen tot nauwelijks moeite gehad om code te begrijpen en het probleem te detecteren.

Jesse Buitenhuis

Naam : Jesse Buitenhuis
Functie: Front-end developer / Specialist

• Kan ik code vinden dat gerelateerd is aan een specifiek probleem of verandering?
Door het implementeren van AngularJs en de MVVM pattern in de front-end gaf mij de vrijheid om het zo op te bouwen dat het gemakkelijk is om specifieke problemen of veranderingen te vinden. Het is alleen wel onduidelijk wanneer AngularJs een foutmelding geeft. Zo'n foutmelding verwijst niet naar het specifieke probleem.

• Kan ik code begrijpen? Kan ik uitleggen wat de bedoeling is achter de functionaliteiten aan iemand anders?

Het is voor mij gemakkelijk om code te begrijpen en uit te leggen. Dit omdat door middel van de architectuur en de opbouw van de front-end gemakkelijk is specifieke functionaliteiten te vinden.

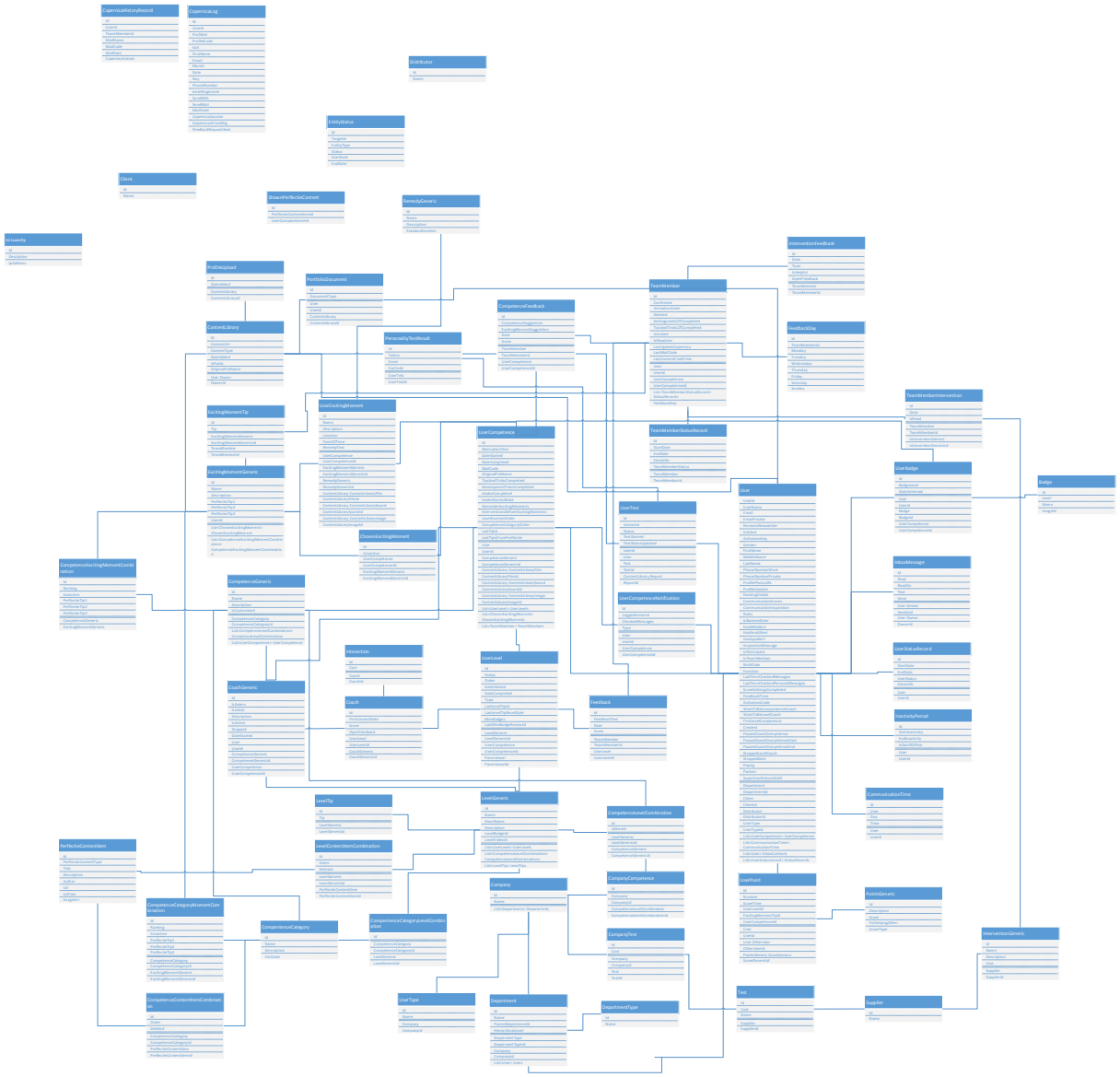
• Is het gemakkelijk om de code te veranderen? Is het gemakkelijk voor mij om vast te stellen wat er verandert moet worden als consequentie van de vorige verandering? Zijn de hoeveelheden en omvang van deze consequentie klein?
Het is gemakkelijk om code te veranderen, maar het is voor mij niet zichtbaar of ik hiermee iets anders van de front-end aantast.

• Kan ik snel een verandering binnen het systeem verifiëren?
Op dit moment is dit voor de front-end niet mogelijk om te doen, maar alle logica wordt door de backend afgehandeld. Dus de verantwoordelijkheid voor het verifiëren ligt niet bij de front-end.

• Kan ik een verandering doorvoeren met een laag risico dat er bestaande functionaliteiten breken?
De front-end wordt niet geverifieerd door functionaliteiten. Hierdoor kan het zijn dat andere functionaliteiten in de front-end breek bij veranderingen. Maar omdat de front-end in gesloten delen is opgebouwd zal dit geen massaal impact hebben in het systeem.

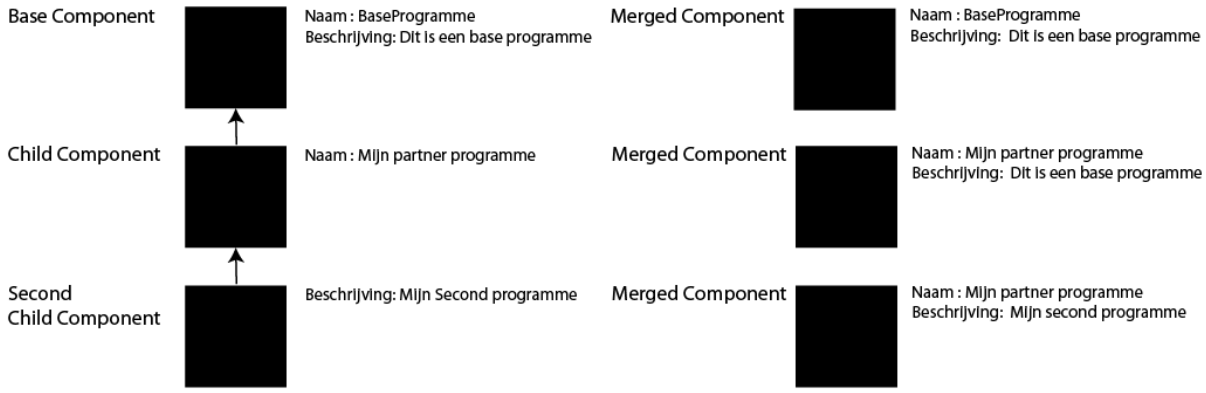
• Als er een bug ontstaat, is het probleem snel en gemakkelijk te detecteren en te diagnoseren?
AngularJs geeft geen duidelijke aanwijzing waarde fout in de code zich voordoet.

J : DOMAIN UML DIAGRAM PERFLECTIE, 11-08-2014



K : Uitgewerkt concept Merge functionaliteit

Het concept is voorafgaande aan sprint 1 door Sebastiaan Alblas uitgewerkt en gerealiseerd door middel van het gebruik van “gewone” taal. Aan het concept is te zien hoe de communicatie verloopt tussen de erving van data vanaf de child tot aan de base. Daarbij mag een onderdeel alleen zijn Parent aanroepen. Om het concept te realiseren en te testen zijn er altijd drie lagen structuren gebruikt voor de casussen.



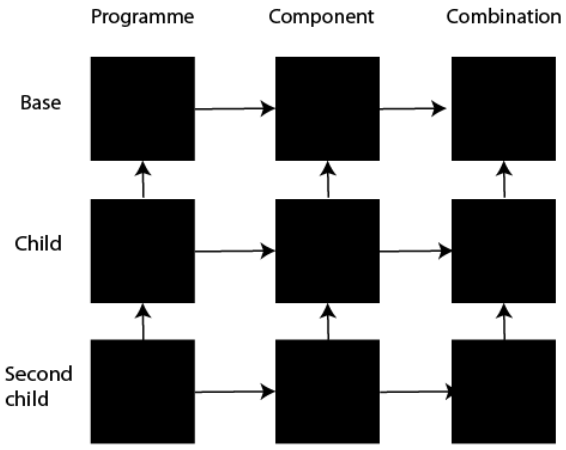
Figuur 15 Merge concept

Merging

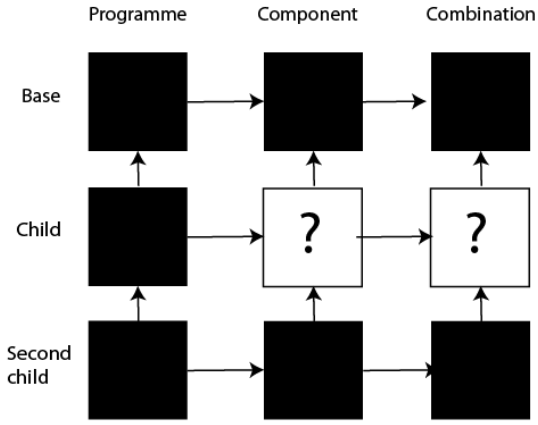
Zowel het programma als al zijn onderdelen en combinaties kunnen gebruik maken van de Merge functionaliteit. Alle velden die de geselecteerde component nog niet heeft overschreven krijgt hij door de merge functionaliteit ingevuld van zijn parents. (zie figuur 15) Hier is als voorbeeld genomen een base, een child en een second child component.

De base heeft geen parent. Als daar de merge functionaliteit op wordt uitgevoerd krijg je een representatie wat overeenkomt van de base object.

De child heeft alleen de naam overschreven van zijn parent. Als daar de merge functionaliteit op wordt uitgevoerd dan krijg je een representatie van de overschreven naam maar ook de beschrijving van zijn parent.



Figuur 16 Programme structuur



Figuur 17 Concept structuur overridden

Als laatst is er de second child component. De second child component heeft alleen de beschrijving overschreven. Als hier de merge functionaliteit op wordt uitgevoerd dan krijg je als representatie een object met de naam van zijn parent, de child component, en de beschrijving van zichzelf.

Concept structuur programme

In figuur 16 is de structuur te zien van een programme. Een programme heeft een één op meer relatie naar een component en een component heeft een één op meer relatie naar combinaties. Dit houdt in dat één onderdeel meerdere onderdelen van de andere kan bevatten.

Als een child gemerged wordt dan krijg hij ook de componenten en daarbij de combinaties van zijn parents. Als de child hiervan een component of

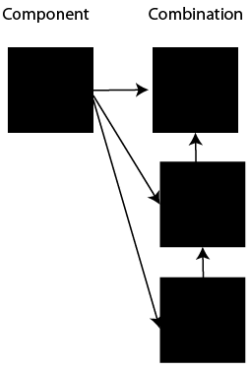
combinatie wil wijzigen dan wordt daarbij het onderdeel overschreven en wordt dus een child aangemaakt binnen het programme met de overgeschreven data.

Daarom kan een child nooit een component of combinatie die niet binnen zijn programma zit aanpassen, alleen wat binnen Perflectie genoemd wordt, overridden.

Concept overriding

Overridden is de term die wordt gehanteerd binnen de programme factory, voor het wijzigen van een parent door een child. Maar wat als de second child iets wilt overridden van de base maar de child heeft dat nog niet gedaan?

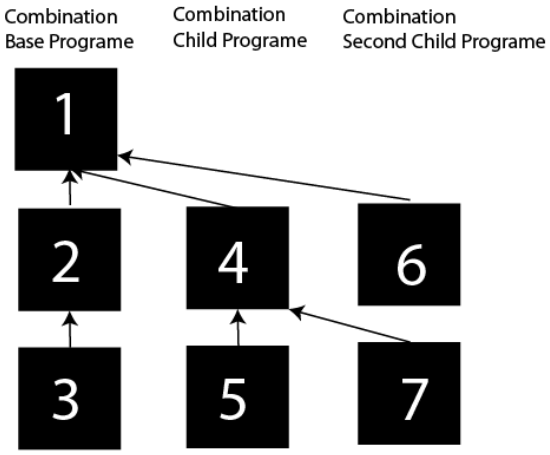
Wanneer er een gat ontstaat binnen het grid omdat een laag nog geen component of combination heeft overridden, maar een laag onder hem wil dat wel doen, dan moet het grid gesloten worden in elke laag. (zie situatie schets figuur 17) Stel in figuur 17 wil de second child de base combination overridden. Een directe verwijzing is niet mogelijk omdat dat de grid structuur verstoord. Daarom moet de child ook het component en combination overridden zodat het grid zijn structuur behoud.



Figuur 18 Component combinations

Concept ordenen

Het moet mogelijk zijn om componenten te kunnen ordenen binnen een programma. Zo bestaat een combination uit een FromComponent en een ToComponent. Zo kan een FromComponent bijvoorbeeld een module, meerdere levels hebben door middel van combinations. (zie figuur 18)



Figuur 19 Combinations met lagen

Dit kan niet gemakkelijk opgelost worden door middel van een cijfer die de positie bijhoudt. Dit komt omdat wanneer een parent de order wijzigt van de combinaties dit ook meegenomen moet worden bij de children. Ook al heeft een child zijn eigen componenten in de order toegevoegd.

Om dit op te lossen is er gebruik gemaakt van de lagen van een programma waarmee de combinatie wordt beheerd. In figuur 18 zijn er gecijferde combinaties te zien. Elke combinatie heeft dezelfde FromComponent maar een andere ToComponent. Toch wil je de ToComponents ordenen. Dit kan gedaan worden als je een combinatie linkt met een andere combinatie als te zien in figuur 19. 3 linkt naar 2 en 2 linkt naar 1. Als je de combinaties opvraagt van de base programme dan krijg je de volgorde terug:

- 1
- 2
- 3

Het child programme heeft twee eigen combinaties met daarbij zijn eigen ToComponenten genaamd nummer 4 en 5. Nummer 4 linkt naar 1 en hierdoor vraagt het child programma dat zijn combinatie eerder moet dan de base component. Nummer 5 linkt naar nummer 4 en hoort erna te komen. Als je de volgorde van de child programme opvraagt dan krijg je de volgende lijst terug:

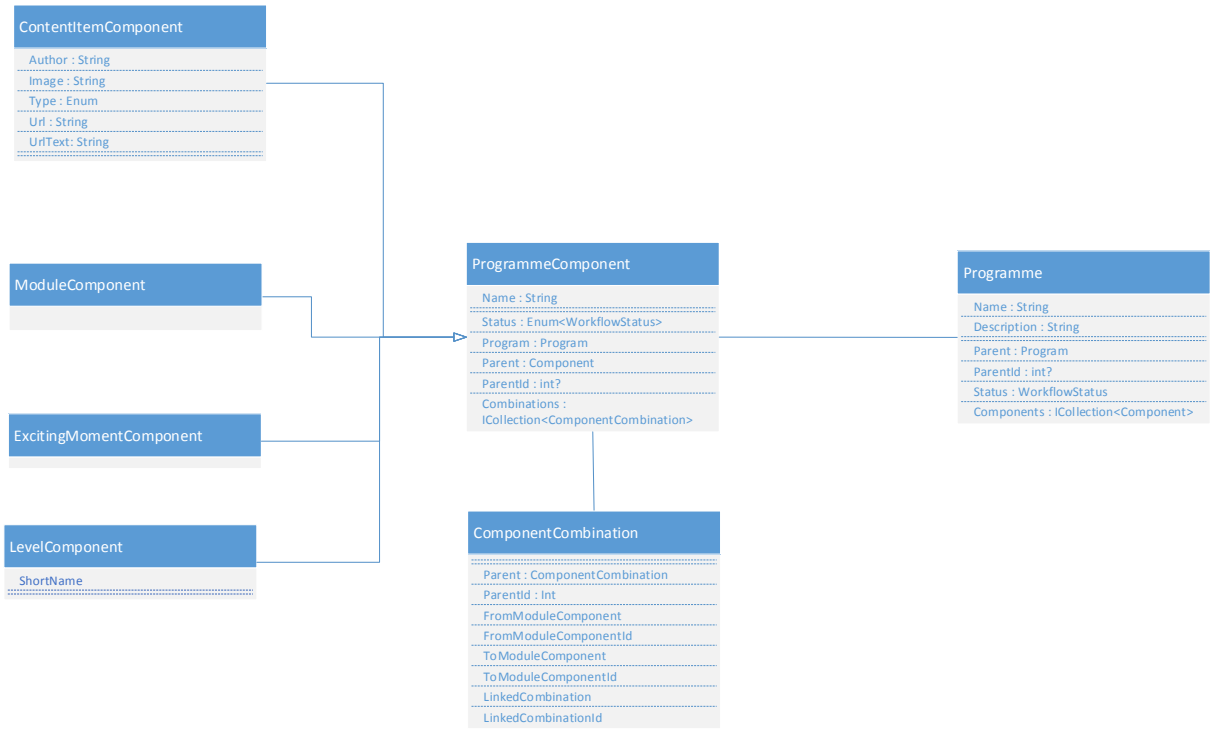
- 1
- 4
- 5
- 2
- 3

De second child programme heeft twee eigen combinaties met daarbij zijn eigen ToComponenten genaamd nummer 6 en 7. Nummer 6 linkt naar 1 en hierdoor vraagt de second child programma dat zijn combinatie eerder moet dan de child en base component. Nummer 7 linkt naar nummer 4 en hoort daardoor eerder te

komen dan de child component 5. Als je de volgorde van de second child programme opvraagt dan krijg je de volgende lijst terug:

- 1
- 6
- 4
- 7
- 5
- 2
- 3

Domain-driven design
Na het gezamenlijk uitwerken van het concept is er met de Product Owner (zie hoofdstuk 5.1 De gebruikte methodes), Jochem Aubel, overgegaan tot het ontwerpen van de domain model. Vanuit het concept en de domain model zal de functionaliteiten gerealiseerd worden binnen de applicatie.



Figuur 20 Domain model Programme factory

Om alle klassen, die kunnen Mergen, een gezamenlijke property te geven is er voor de Parent property gekozen. De Parent is een verwijzing van een klasse naar zichzelf. Zo kunnen er oneindig veel lagen gemaakt worden van een klasse.

ComponentCombination – Klasse
ComponentCombinatie heeft een extra verwijzing naar zichzelf genaamd LinkedCombination om zo de orde vast te kunnen stellen van combinations.

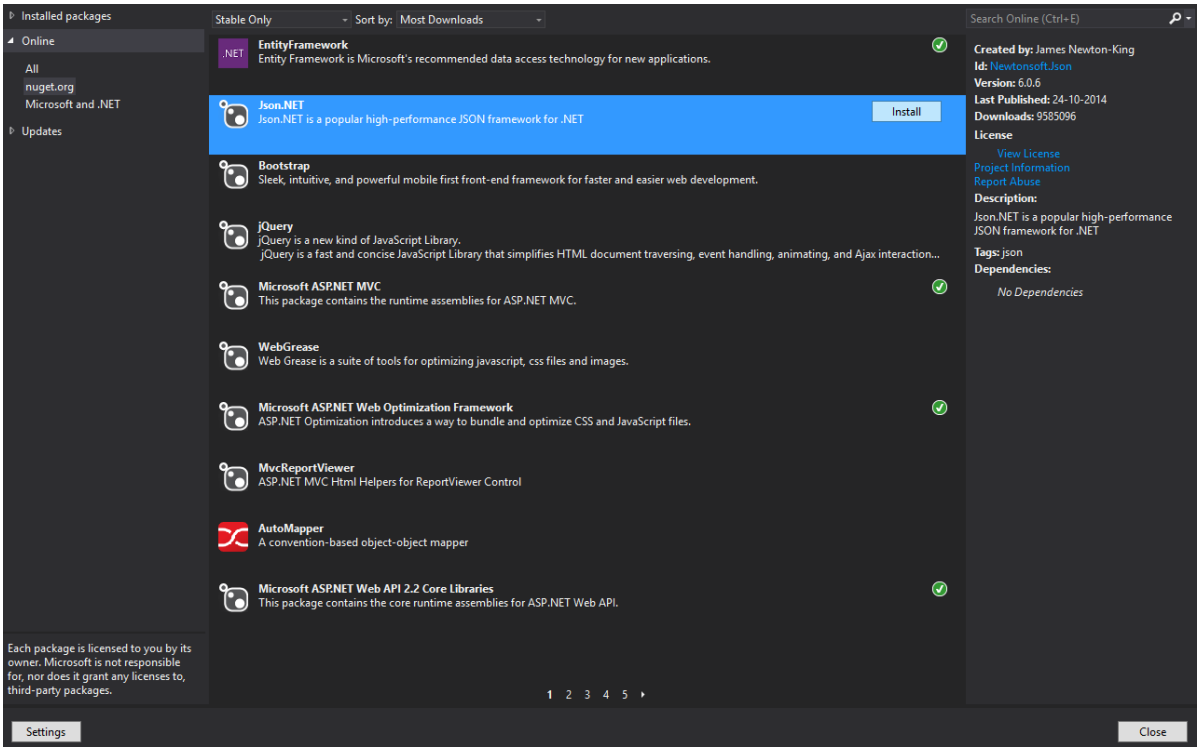
L : De gebruikte software & extensies

Om het nieuwe online-platform te realiseren is er de volgende software en extensies gebruikt.

Microsoft developer tools
Het project zal met behulp van Microsoft developer tools gerealiseerd worden. Deze keuze komt vanuit het feit dat Perflectie voor de C# programmeertaal (zie hoofdstuk 4.1.6 Conclusie te adviseren programmeertaal) en het .Net Framework heeft gekozen. Om de applicatie te kunnen uitvoeren moet op de server een Windows besturing systeem geïnstalleerd zijn met de benodigde .Net versie. Dit geeft als voordeel dat er gebruik gemaakt kan worden van de Microsoft developer tools. Dit houdt in dat elke tool dat gebruikt wordt voor productie, compatible is met elkaar en zo gemaakt is om met elkaar te kunnen communiceren. Dit geeft voor een ontwikkelaar toegang tot een sneller productie omgeving. Deze compatibiliteit wordt verder beschreven per software tool.

Visual studio

Visual studio is de tool dat gebruikt gaat worden door de ontwikkelaars van Perflectie. Visual studio creëert een programmeerontwikkelomgeving waarbinnen er diverse complete sets aan tools gebruikt kunnen worden in diverse programmeertalen om met name Windows applicaties te ontwikkelen. Binnen Visual studio wordt er gebruik gemaakt van meerdere library's en het .NET Framework. Door het gebruik hiervan is het gemakkelijker voor de ontwikkelaar om snel Windows applicaties te realiseren.



Figuur 7 NuGet package manager

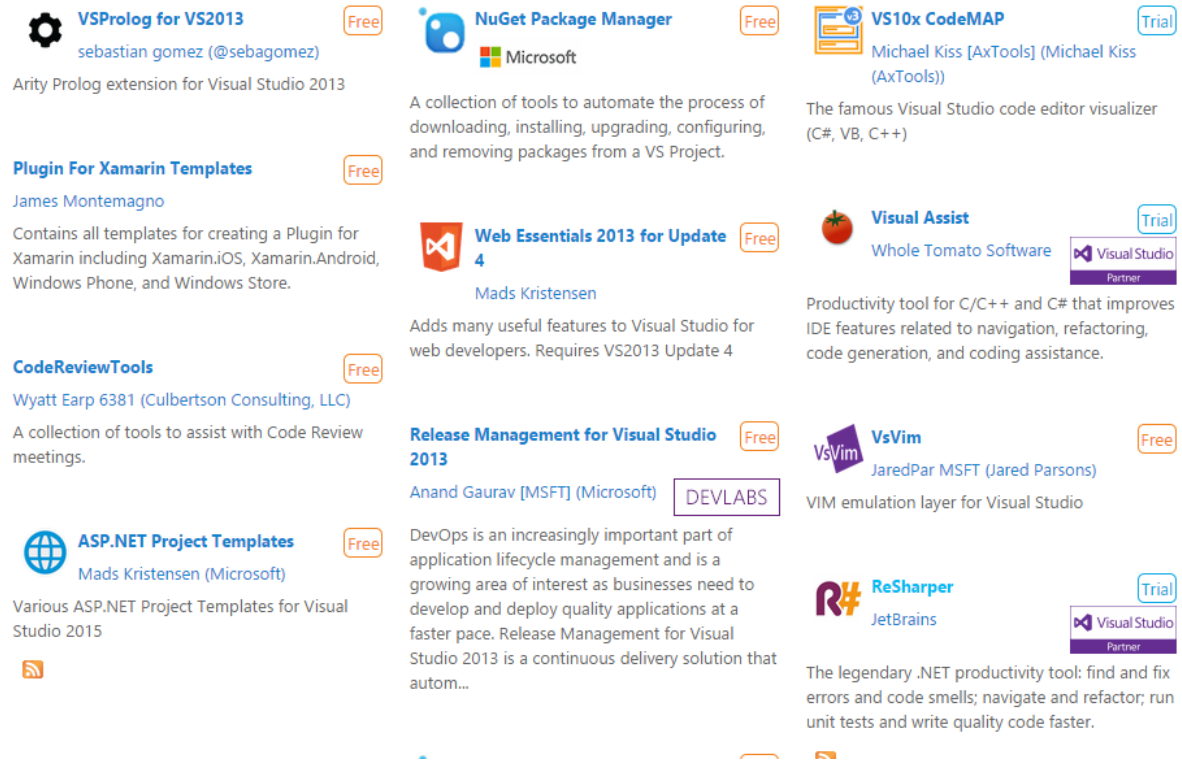
In het verleden bood Microsoft Visual studio zowel commerciële als gratis versies aan. Tegenwoordig bieden zij versies aan specifiek voor een platform die gratis beschikbaar zijn. Dit bezorgt voor een onderneming geen extra licentie kosten.

NuGetPackageManager
Externe losse componenten worden ook wel packages genoemd. Een package is een losstaand werkend subsysteem dat gebruikt kan worden binnen een systeem.

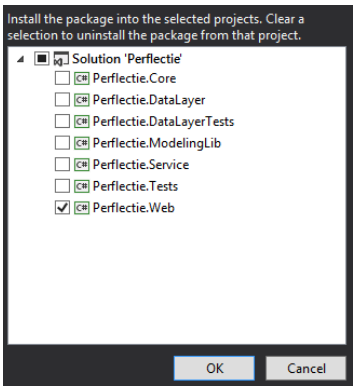
Om packages te implementeren en te beheren binnen visual studio is de NuGet package manager geschikt. Binnen NuGet is het mogelijk om externe packages gemakkelijk online te vinden en daarna in je project te installeren.

Deze packages kunnen daarna beheerd worden om beschikbaar gesteld te worden binnen een laag van de applicatie of door de gehele applicatie. (zie figuur 7)

Visual studio extensies
Naast packages is het ook mogelijk om Visual studio uit te breiden met extra software, templates etc. dit wordt ook wel een extensie genoemd. Net als de NuGet manager kan je gemakkelijk met visual studio zelf of via het internet extensies downloaden en installeren. Extensies zijn ontwikkeld door verschillende bedrijven en daarom kan het zijn dat bij sommige een licentie betaald moet worden voor gebruik.
Niet



Figuur 8 Visual studio gallery extensions



Figuur 9 Package het instellen van een tier

Microsoft SQL server management studio

Mircosoft SQL server management studio is een developing tool om servers en databases te onderhouden. Binnen de tool kan via een userinterface of via SQL query's, datastructuren beheerd worden. Ook bestaan er meerdere mogelijkheden om data te exporteren naar verschillende bestandsformaten zoals Excel of Sql. De tool is gratis beschikbaar en te downloaden.

Team foundation server

Team foundation server (TFS) is een online web-applicatie welke zowel als source control voor het project als een projectmanagement platform gebruikt kan worden. Er kan gratis een account worden aangemaakt waardoor er een eigen omgeving ontstaat waarin een project gestart kan worden. Als er een nieuwe project gestart wordt kunnen er maximaal vijf actieve gebruikers gratis deelnemen, daarnaast kunnen oneindig veel gebruikers toegevoegd worden die het project kunnen inzien. Er kan een gewenste projectmanagementmethode gekozen worden en de gewenste source control methode.

Wanneer een projectmanagementmethode is gekozen dan kan deze beheerd worden door de actieve gebruikers. Als er een source control is ingesteld dan kan deze ook beheerd worden door de actieve gebruikers.

Windows server

Om de applicatie te kunnen hosten is de Windows server of een Windows besturingssysteem nodig. Windows server is specifiek opgezet om Windows applicaties te draaien in een server omgeving. Hierbij krijg je een userinterface met een set aan tools om zo gemakkelijk applicaties te draaien en zo beschikbaar te stellen voor externe gebruikers. Ook kan een Windows server zo ingesteld worden dat het gemakkelijk is om software updates uit te voeren op live applicatie.

M : Het opzetten van de ontwikkelomgeving

Definiëren van tieren/lagen binnen een solution

Om de architectuur vast te stellen, welke in de architecture notebook (zie bijlage F : Nieuwe architecture notebook) is vastgesteld kunnen binnen het nieuwe online-platform met Visual studio lagen worden gedefinieerd. Deze lagen en de communicatie onderling kan niet gebroken worden, de details van deze lagen zijn niet zichtbaar voor de lagen die hier geen toegang toe hebben.

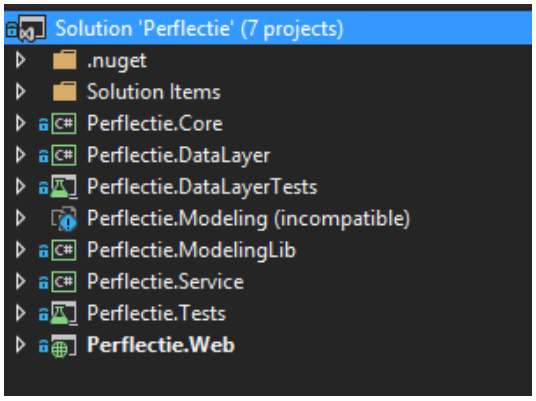
Om een component binnen het project toe te voegen in een solution van Visual studio, kun je door rechter muisklik op de solution, add -> new project en nieuw component toevoegen.

Binnen Perflectie zijn er vijf lagen aangemaakt, deze zijn:

- Web – Web applicatie
- Service – Class library
- DataLayer – Class library
- Core– Class library
- Tests– Class library

Al deze lagen maken gebruik van .Net versie 4.5.

De web laag mag alleen de service aanroepen, waarbij de service alleen de DataLayer mag aanroepen. De core kan gebruikt worden door iedere laag en de Tests kan de web, service en DataLayer gebruiken. (zie hoofdstuk 5.2.2 Gekozen programmeertalen & architectuur



Figuur 60 Solution lagen

Om dit binnen de applicatie te definiëren moet je een referentie toevoegen binnen een project. Dit kan door rechter muisklik op het project, add -> reference -> solution -> projects. Hierbinnen kan je de lagen aanvinken waar de laag gebruik van mag maken.

Test driven development

Er is een populaire manier van testen binnen softwareprojecten, genaamd Test driven development (TDD). Als TDD is ingericht kan er effectief getest worden binnen het gehele systeem. De Unit Tests zullen blijven bestaan ook wanneer het systeem in gebruik is genomen. Wanneer het systeem wordt uitgebreid of aangepast en een requirement wordt verbroken tijdens een Unit test dan zie je dit direct binnen het systeem. Zo kunnen bugs vroegtijdig opgespoord en opgelost worden.

Binnen het huidige online-platform zijn er geen tests aanwezig. Dit blijkt ook na elke nieuw release van het online-platform van Perflectie. Na elke release geeft Perflectie aan dat er bugs aanwezig zijn binnen het systeem. Deze bugs worden ook weer ontdekt door de eindgebruikers. Om deze bugs zo vroeg mogelijk en liefst niet in de productieomgeving tegen te komen moeten er unit tests geïmplementeerd worden. Met behulp van Dependency Injection (Developer's Guide to Dependency Injection Using Unity, 2014) kan een system lousely coupled opgezet worden. Dit in combinatie met Unit testing kan je zowel test data opzetten voor de unit tests als ook real-time database koppelen.

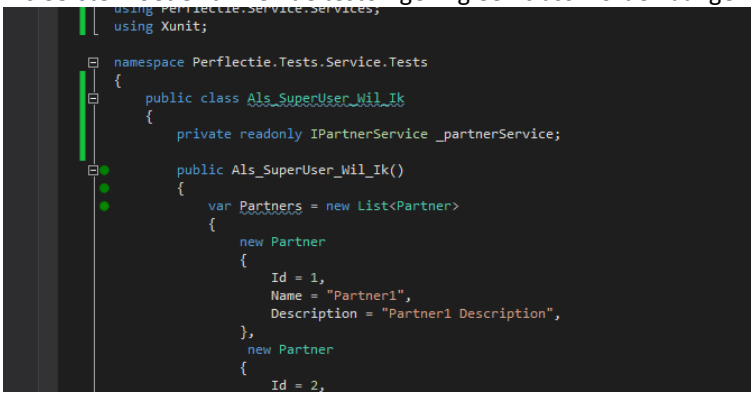
Om een test driven systeem te ontwikkelen is het nodig om vanaf het begin binnen deze methode te werken. Binnen deze methode zijn er enkele plug-ins en stappen nodig die moeten worden voltooid. De plug-in xUnit zorgt ervoor dat binnen het project unit tests aangewezen kunnen worden. Met nCrunch kunnen er real-time

tests gedraaid worden, wat binnen xUnit is gedefinieerd, om zo, wanneer je aan het programmeren bent, te zien of je geen andere tests aantast.

Wanneer de plug-ins en extensies zijn geïnstalleerd kunnen er tests gemaakt worden. Een programmeur binnen een scrumteam weet wanneer hij een test moet aanmaken door te kijken naar zijn taken op het scrumboard. Als de programmeur bijvoorbeeld de volgende taak toegewezen krijgt:
Als SuperUser wil ik:

- Een partner aanmaken, zien, bewerken en verwijderen
- Status: actief of inactief

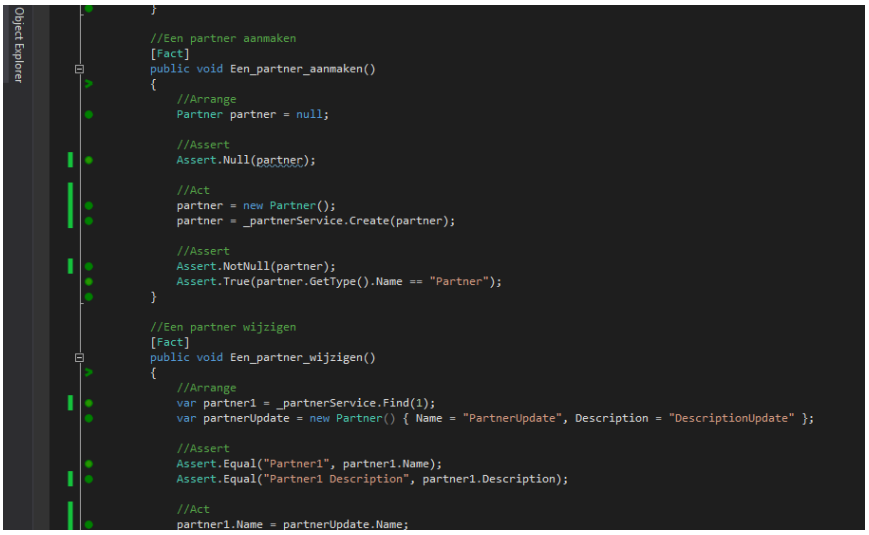
Als eerste moet er binnen de testomgeving een class worden aangemaakt met de naam: Als_superuser_wil_ik.



Figuur 11 Unit test Als_SuperUser_Wil_ik

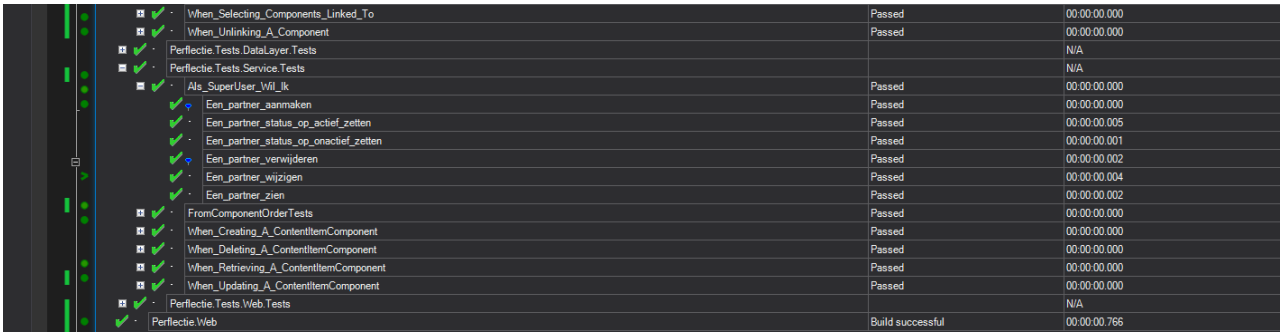
Om de unit tests volledig te maken op de eisen en wensen van het product eigenaar zullen de volgende methodes worden toegevoegd:

- Een partner
  - Aanmaken
  - Zien
  - Bewerken
  - Verwijderen
- Een partner zijn status
  - Actief zetten
  - Inactief zetten



Figuur 12 SuperUser met zijn methodes toegevoegd met daarbij een werkende xUnit test dat gedraaid wordt met nCrunch

Wanneer de methodes zijn toegevoegd zal deze uitgebreid worden tot deze methodes aan alle eisen van de unit tests voldoen. Hierna worden de functionaliteiten van een methode omgezet en gerefactored binnen het software systeem, wat later door middel van unit tests wordt doorverwezen. Wanneer het systeem volledig is uitgebreid met Unit tests krijgen de belanghebbende het volgende overzicht van het project.f



Figuur 13 Overzicht lijst unit tests

Wanneer later een aanpassing aan het systeem een error geeft binnen het systeem kun je het op deze manier direct opzoeken en oplossen.