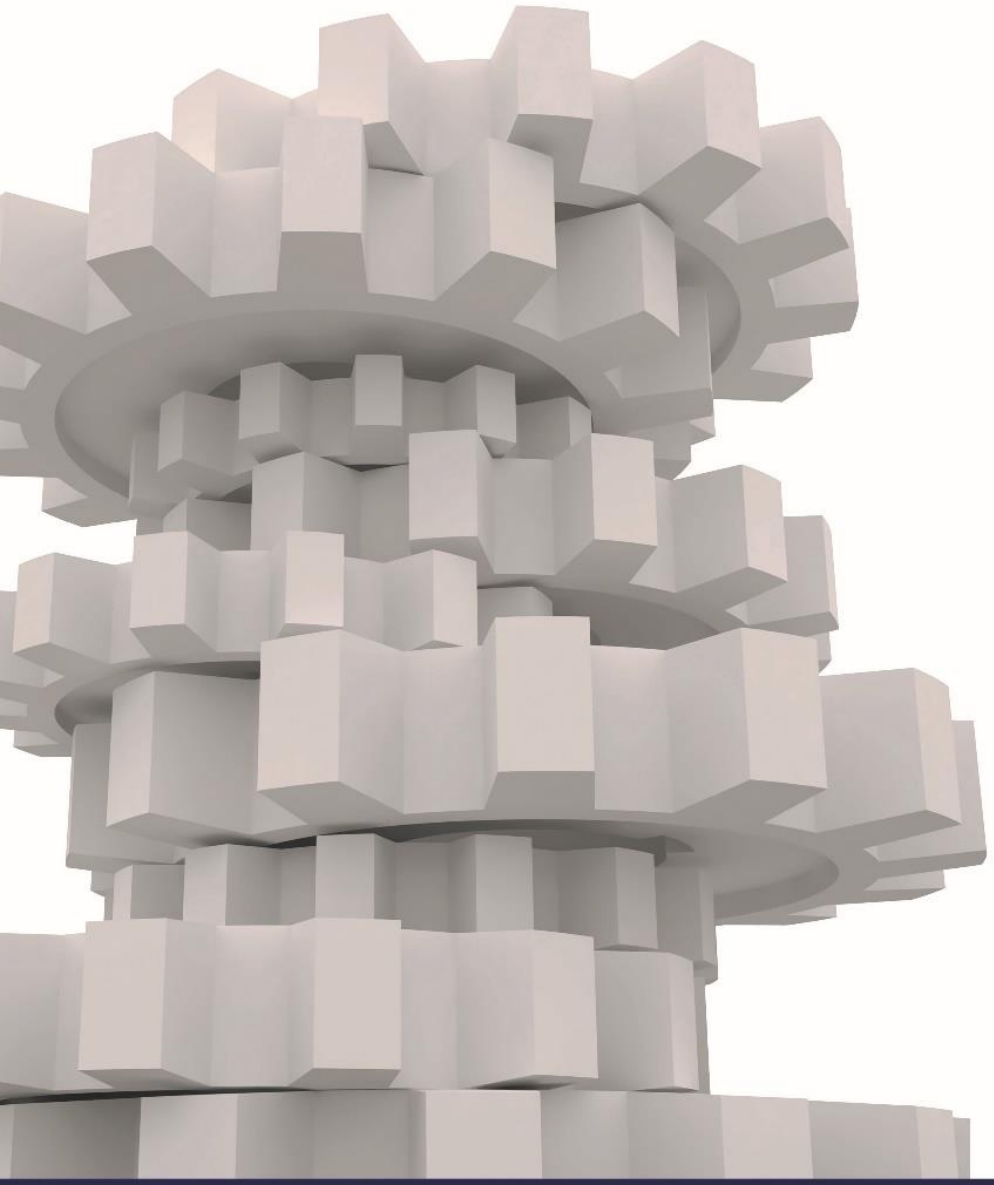


Afstudeerverslag
HelloID Directory Agent

Versie: 1.3.0

Auteur: Robbert Brussaard

TOOLS4EVER
IDENTITY GOVERNANCE & ADMINISTRATION

REFERAAT

In dit verslag worden het proces en de evaluatie van mijn afstudeerproject ‘Ontwikkelen van een (modulaire) authenticatie- en synchronisatieservice voor HelloID bij Tools4ever B.V.’ beschreven.

Afstudeerder: Robbert Brussaard

Onderwijsinstelling: De Haage Hogeschool

Locatie: Den Haag

Variant: Deeltijd c.q. avond

Opleiding: Informatica

Periode: 09-09-2016 tot en met 09-01-2017

Eerste begeleider: Arie Toet

Tweede begeleider: Paul Smit

Bedrijf: Tools4ever B.V.

Bedrijfsmentor: George Bakker

Opdrachtgever: George Bakker

Descriptoren: C#, Single Sign On, Windows Service, Multi-threading, Scrum, Test Driven Development (TDD), Unified Modeling Language UML, Requirements, Active Directory, Design Patterns, Win32 API, Large Object Heap, Synchronisatie

VOORWOORD

Dit eindverslag heb ik geschreven als informaticastudent aan De Haagse Hogeschool. Eind augustus 2016 ben ik gestart met mijn afstudeerproject bij het bedrijf Tools4ever B.V, waar ik op het moment van schrijven al twee jaar fulltime met plezier werk.

Dit verslag is bedoeld voor alle betrokkenen bij mijn afstudeerproject en mijn studieloopbaan. Ik wil deze gelegenheid gebruiken om een aantal personen te bedanken.

Ten eerste wil ik mijn collega's van de HelloID-afdeling bedanken voor hun steun, adviezen en feedback tijdens mijn studie en afstuderen. In het bijzonder wil ik mijn manager en begeleider George Bakker bedanken. Sinds ik bij Tools4ever B.V. werk, heb ik enorm veel van hem geleerd, zowel op theoretische en praktijkaspecten als op persoonlijk vlak.

Ten tweede wil ik mijn afstudeerbegeleiders Arie Toet en Paul Smit bedanken voor de tijd die zij in het begeleiden en beoordelen van mijn afstudeerproject hebben gestoken. Ook wil ik hen en de andere docenten van De Haagse Hogeschool bedanken voor de kennis die zij aan mij hebben overgedragen.

Ten derde wil ik mijn vrienden en medestudenten Dominique Nihot en Roald Koning hartelijk danken. Wij hebben tijdens onze studie veel gelachen, maar vooral samen hard gewerkt aan alle projecten en vraagstukken. Tijdens onze afstudeerprojecten zijn wij trouw aan elkaar gebleven door minimaal een zondag in de maand af te spreken en kritisch naar elkaars werk te kijken, feedback te geven en het werk naar een hoger niveau te brengen. Daarvoor dank.

Ten vierde wil ik Tools4ever B.V. bedanken voor de mogelijkheid om hier fulltime te werken en gelijktijdig af te studeren. Tijdens het project zijn alle faciliteiten die ik nodig had tot mijn beschikking gesteld en dat heeft mij tijdens het afstudeerproject veel geholpen. Daarnaast wil ik ook alle collega's bedanken die altijd klaar hebben gestaan bij vragen en de tijd namen om deze te beantwoorden.

Ten slotte wil ik mijn vriendin Mandy Snoek in het bijzonder bedanken. Zij heeft veel moeten doorstaan tijdens mijn studie. In de avonden en weekenden dat ik moest studeren was zij alleen en konden wij niet altijd naar familie of vrienden. Desondanks stond zij altijd voor mij klaar en zorgde zij ervoor dat ik gezond at en niet te veel hooi op mijn vork nam. Daarvoor dank.

Robbert Brussaard

Baarn, 25 december 2016

INHOUDSOPGAVE

Referaat 1

Voorwoord 2

Documentinformatie 10

Versiebeheer 10

Verzendlijst 10

1 Inleiding 11

2 Bedrijf / organisatie 12

2.1 Over Tools4ever 12

2.2 Over het HelloID-team 12

2.3 Over HelloID 13

3 Opdrachtomschrijving 14

3.1 Opdrachtomschrijving 14

3.1.1 Huidige situatie 14

3.1.2 Gewenste situatie 15

4 Aanloopfase 17

4.1 Plan van aanpak 17

4.1.1 Fasering 18

4.1.2 Methode en technieken 18

4.2 Requirements verzamelen 23

4.2.1 Eliciteren 23

4.2.2 Verwerken requirements en werken met MoSCoW 27

4.3 Testplan 28

4.3.1 Aanpak 28

5 Sprint 1: Ontwerpen en implementeren van het domeinmodel 30

5.1 Ontwerpen van het nieuwe domeinmodel 30

5.1.1 Profiel- en attributenstructuur 31

5.1.2 Opnemen van identity providers 32
5.1.3 Authenticatie-templates 33
5.2 Opstellen van EER, RRM en RIM. 33
5.2.1 Overerving in de database 34
6 Sprint 2: Communicatie tussen agent en HelloID 38
6.1 Opzetten van Windows Service 38
6.2 Ontwikkelen van modulaire structuur 39
6.2.1 ServiceLocator 40
6.2.2 Inversion of Control 42
6.2.3 Thread-safe maken van Resolve methode 43
6.3 De verbinding naar HelloID 44
6.3.1 Websocket in combinatie met polling 45
6.3.2 Een veilige verbinding opzetten en uitwisseling van berichten 50
6.4 Testbaar maken van code 52
7 Sprint 3: Inloggen via inloggegevens uit Active Directory 54
7.1 Domein bepalen door middel van inloggegevens 54
7.1.1 Gebruik van Win32 API 56
7.2 Gegevens uit Active Directory halen 57
7.3 Integratietests 58
8 Sprint 4: Synchronisatiemechanisme 59
9 Sprint 5: Demo-omgeving 62
9.1 Aanpassing van synchronisatiemechanisme 62
9.1.1 De problemen 62
9.1.2 Oplossing 64
9.2 Demo-omgeving 65
10 Evaluatie 67
10.1 Procesevaluatie 67
10.1.1 Werken met Scrum 69
10.1.2 Werken met Test-Driven Development 71

10.2 Productevaluatie 73
10.3 Competentie-evaluatie 74
10.3.1 Uitvoeren analyse door definitie van requirements 74
10.3.2 Opstellen gegevensmodel voor database en Ontwerpen, bouwen en bevragen van een database 74
10.3.3 Ontwerpen systeemdeel 75
10.3.4 Bouwen applicatie 75
10.3.5 Uitvoeren van en rapporten over het testproces 76
11 Bronvermeldingen 77
12 Definities en afkortingen 80
12.1 Definities 80
12.2 Afkortingen 83
13 Figuren 84
14 Bijlage A: Afstudeerplan 85
Bijlage B: Bedrijfsbezoek 93
Bijlage C: Voortgangsverslag 94
Voortgang 94
Aanloopfase 94
Sprint 1 94
Sprint 2 95
Sprint 3 96
Conclusie 96
Bijlage D: Formulier tussentijds assessment 97
Bijlage E: Plan van aanpak 101
Samenvatting 101
Versiebeheer 101
Verzendlijst 101
Uitgangspunten 101
Projectdefinitie 102

Bedrijf / organisatie 102
Opdrachtomschrijving 102
Doelstelling 103
Afbakening 103
Projectorganisatie 104
Product owner 104
Stakeholders 104
Projectteam 104
Interne afspraken 104
Werkwijze 105
Fasering 105
Aanloopfase 105
Cyclische scrum-fase (sprints) 105
Methodieken 106
Scrum 106
Test Driven Development (TDD) 106
MoSCoW 107
Planning 108
Activiteiten 108
Requirements verzamelen 108
Ontwerpen 108
Ontwikkelen 108
Testen 108
Inrichten demonstratie omgeving 109
Schrijven van handleiding 109
Globale planning 109
Mijlpaal producten 109
Bijlage F: Requirementsrapport 110
Versiebeheer 110

Verzendlijst 110
Uitgangspunten 110
Inleiding 111
Bedrijfsvisie 111
Huidige situatie 111
Gewenste situatie 111
Businessrequirements 112
Belanghebbende/Actoren 112
Eindgebruikers 112
Gebruikers 112
Beheerders/Consultants 112
Gebruikersomgeving 112
Systeem 113
Context 113
Requirements 114
Gebruikersrequirements 114
Gebruikers 114
Beheerders/consultants 115
Softwarerequirements 122
Functionaliteit 122
Betrouwbaarheid 123
Bruikbaarheid 124
Efficiency 125
Onderhoudbaarheid 126
Overdraagbaarheid 126
Prioriteiten van requirements 127
Bijlage G: Ontwerpdocumentatie 129
Versiebeheer 129
Verzendlijst 129

Samenvatting 130
Architecturale structuur 130
Klassendiagram Directory agent 131
Core pacakage 131
Shared package 132
Klassendiagram HelloID 133
Communicatie tussen Agent en HelloID 134
Context diagram 134
Elementen 135
Activity diagram: flow van een authenticatie aanvraag 136
Sequentie diagram: verbinding opzetten 137
Module API 138
Module resolvement 139
Sequenctie diagram: resolven van IUserManager 140
Domeinmodel HelloID 141
Analyse klassendiagram 141
Beschrijving van klassen 142
Enhanced Entity-Relationship model (EERM) 144
Relationeel representatiemodel (RRM) 145
Relationeel implementatie model (RIM) 146
Grafische weergave in HelloID 149
Agent overview 149
Geen providers aanwezig 149
Provider aanwezig 150
Agent wizard 151
Bijlage H: Testplan 153
Versiebeheer 153
Verzendlijst 153
Introductie 154

Domein 154
Test basis 154
Documenten 154
Applicaties 154
Verantwoordelijkheden 155
aanpak 156
White-box tests 156
Module tests 156
Integratie tests 156
Black-box tests 157
Error guessing 157
Bijlage I: Voorbeeld testrapportage 158

DOCUMENTINFORMATIE

VERSIEBEHEER

Versie	Datum	Omschrijving
1.0	01-09-2016 t/m 5-11-2016	Eerste conceptversie met initiële structuur en inhoud.
1.1	11-12-2016	Tweede conceptversie, waarbij de hoofdstukken voor 80% zijn geschreven. Deze versie is gebruikt voor de TTA.
1.2	25-11-2016	Afgeronde versie van het afstudeerverslag. Alle hoofdstukken zijn geschreven en de feedback die uit de TTA is gekomen is verwerkt.
1.3	02-01-2017	Feedback op inhoud en Nederlands van begeleider verwerkt.

VERZENDLIJST

Wie	Contactgegevens
George Bakker	G.Bakker@tools4ever.com
Arie Toet	A.Toet@hhs.nl
Paul Smit	P.Smit@hhs.nl

1 INLEIDING

De studie informatica sluit ik af met een afstudeertraject waaruit moet blijken dat ik kan functioneren binnen een bedrijf en dat ik wat ik geleerd heb tijdens mijn studie kan toepassen in de praktijk. Tijdens dit afstudeertraject heb ik gewerkt aan een Windows Service die gegevens kan synchroniseren tussen het bronsysteem Active Directory (en vergelijkbare systemen) en het cloud-product HelloID.

Voor u ligt mijn afstudeerverslag van dit traject. Dit document heeft als doel om de werkzaamheden die ik tijdens mijn afstuderen heb verricht te beschrijven en keuzes die ik heb gemaakt toe te lichten.

In hoofdstuk 2 worden het team en de organisatie waar ik dit afstudeertraject heb uitgevoerd geïntroduceerd. In hoofdstuk 3 ga ik in op de opdracht en beschrijf ik de huidige en de gewenste situatie. In hoofdstuk 4 bespreek ik de aanloopfase, waarin ik het project heb opgestart. In deze fase heb ik mijn plan en werkwijze opgesteld en de wensen en eisen van de opdrachtgever in kaart gebracht.

Onderdeel van mijn werkwijze was het gebruik van de ontwikkelmethode Scrum, waarmee in korte perioden aan het product gewerkt wordt. Deze korte perioden worden sprints genoemd. Om de HelloID Directory Agent te ontwikkelen, heb ik vijf sprints gehouden. Aan elke sprint is in dit verslag een hoofdstuk gewijd.

In hoofdstuk 5 licht ik de eerste sprint toe, waarin ik ben gestart met het ontwerpen van een nieuw domeinmodel. In de tweede sprint, die wordt beschreven in hoofdstuk 6, heb ik de communicatie tussen de HelloID Directory Agent en HelloID ontwikkeld. In dit hoofdstuk wordt ook besproken hoe ik de code testbaar heb gemaakt. De communicatietechnologie had ik nodig om het inloggen via gegevens uit een gegevensbron te kunnen ontwikkelen in de derde sprint, toegelicht in hoofdstuk 7. Hoofdstuk 8 gaat in op de vierde sprint, de ontwikkeling van het synchronisatiemechanisme. Het verloop van de vijfde en laatste sprint wordt toegelicht in hoofdstuk 9. Deze sprint stond in het teken van het opzetten van de acceptatieomgeving.

Tot slot kijk ik in hoofdstuk 10 terug op mijn afstudeertraject. Ik geef hier mijn eigen oordeel over de totstandkoming van de producten (het proces), de opgeleverde producten en resultaten (het product) en in welke mate ik de competenties en beroepstaken heb behaald.

2 BEDRIJF / ORGANISATIE

2.1 OVER TOOLS4EVER

Tools4ever is een internationaal bedrijf dat ontstaan is in Nederland. Sinds 1999 levert en ontwikkelt het gestandaardiseerde en betaalbare oplossingen op het gebied van Identity Governance & Administration (IGA), die in tegenstellig tot andere IGA-oplossingen eenvoudig te implementeren en te beheren zijn. Tools4ever is in Nederland de absolute marktleider op het gebied van IGA. Het bedrijf bedient een breed scala van organisaties variërend in grootte, van driehonderd tot meer dan tweehonderdduizend gebruikeraccounts. Zowel kleine ondernemingen als multinationals, onderwijsinstellingen, zorginstellingen en (lokale) overheden zetten de oplossingen van Tools4ever in. Het gaat hierbij zowel om non-profit als profit organisaties.

Tools4ever is een jonge, dynamische en groeiende organisatie. Medewerkers krijgen veel vrijheid en kansen om mee te helpen aan de groei van het bedrijf; eigen initiatief van de medewerkers wordt sterk gewaardeerd. Medewerkers krijgen bij Tools4ever de ruimte om zichzelf verder te ontwikkelen aan de hand van opleidingen en doorgroeimogelijkheden. In totaal heeft Tools4ever zeven vestigingen waar veel mee wordt samengewerkt. Op het kantoor in Nederland werken ongeveer zestig medewerkers. Het Nederlandse kantoor bestaat uit acht afdelingen: Secretariaat, Administratie, Recruitment, Marketing, Planning & Support, Ontwikkeling, Sales en Consultancy. De afdeling Ontwikkeling is onderverdeeld in drie teams: IAM, ERAM en het team waar ik in werk, HelloID.

2.2 OVER HET HELLOID-TEAM

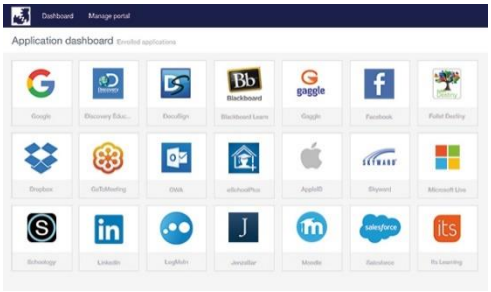
HelloID is een nieuw Single Sign On-product dat is ontwikkeld door George Bakker (Team lead developer en opdrachtgever van deze afstudeeropdracht), Aad Lutgert (Tester) en mijzelf (Developer). Met de start van de ontwikkeling van HelloID is ook het HelloID-team ontstaan. Na ongeveer een jaar is een nieuwe developer het team komen versterken en begin januari 2017 start een front-end engineer om de vormgeving naar een hoger niveau te brengen.

Wij hebben HelloID ontwikkeld voor de cloud, waardoor het product dag en nacht voor alle klanten beschikbaar is. Op dit moment heeft HelloID wereldwijd meer dan honderdduizend actieve gebruikers. Dit zorgt ervoor dat wij als team veel direct contact hebben met onze buitenlandse kantoren; dat maakt ons team uniek binnen Tools4ever.

2.3 OVER HELLOID

HelloID is een cloud-gebaseerde Single Sign On-oplossing waarmee gebruikers door slechts één keer in te loggen snel, eenvoudig en veilig hun webapplicaties kunnen openen op computer, tablet of smartphone (Bring your own device, BYOD).

Het product wordt aangeboden in vorm van een internetportaal. Het doel is dat de eindgebruiker op eenvoudige wijze het portaal kan vinden en vervolgens laagdrempelig toegang krijgt tot de bedrijfswebapplicaties.



Eenmaal ingelogd op het portaal vindt de eindgebruiker een overzicht van de beschikbare cloudapplicaties, bijvoorbeeld Google Apps, Salesforce, Office365, LinkedIn en Twitter. De eindgebruiker hoeft bij het openen van deze applicaties niet nogmaals in te loggen, maar kan direct met de applicatie aan de slag.

3 OPDRACHTOMSCHRIJVING

3.1 OPDRACHTOMSCHRIJVING

Halverwege 2015 is Tools4ever begonnen met de ontwikkeling van een nieuw SSO-product, HelloID. HelloID is een IDaas (Identity as a service) cloud-oplossing waarbij men kan inloggen op een online portaal. Eenmaal ingelogd op het portaal ziet de eindgebruiker een overzicht van beschikbare cloudapplicaties (bijvoorbeeld, Google Apps, Office 365, LinkedIn en Github). De eindgebruiker hoeft bij het openen van deze applicaties niet nogmaals in te loggen.

Bij HelloID kan op verschillende manieren worden ingelogd op het portaal. Dit kan onder andere via een Active Directory (AD) connectie, waarbij met een gebruikersnaam- en wachtwoordcombinatie kan worden ingelogd. De AD-connectie wordt gemaakt door een service die geïnstalleerd is in de omgeving van de klant. De service maakt een verbinding aan naar HelloID en kan vervolgens inlogacties op HelloID controleren om de identiteit van een gebruiker te verifiëren.

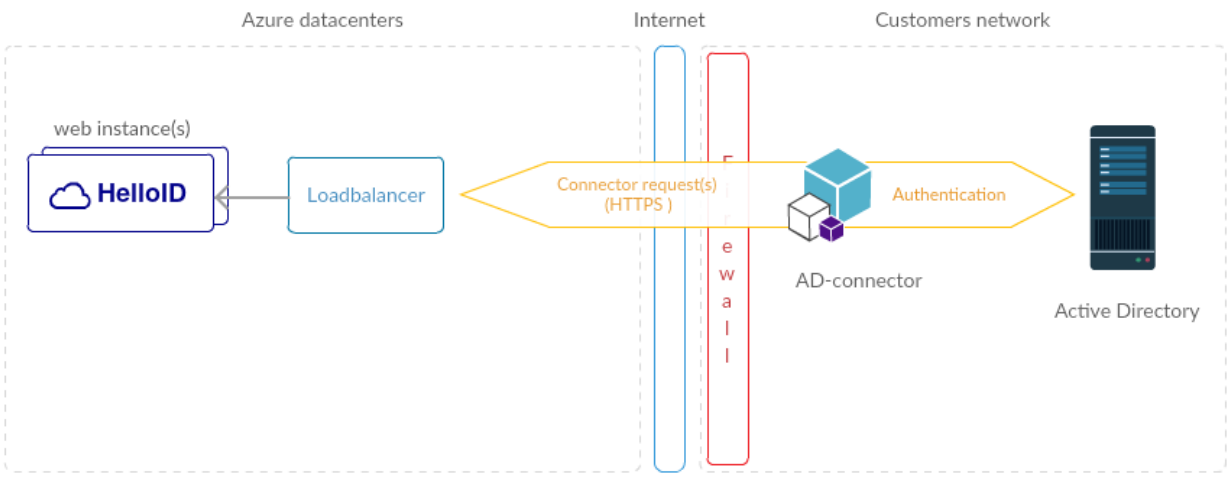
3.1.1 HUIDIGE SITUATIE

Tools4ever levert nu geen generieke connector. De huidige connector ondersteunt alleen AD als bron voor identiteiten. Het bedrijf levert wel koppelingen via het SAML-protocol (John Hughes, 2004). De meeste systemen ondersteunen echter geen SAML. Doordat verschillende bronnen niet worden ondersteund, is het niet mogelijk om alle klantomgevingen te ondersteunen. Het is voor een klant niet mogelijk om een ander bronsysteem dan AD of een systeem dat geen SAML ondersteunt te gebruiken om inlogacties te controleren. Andere bronnen die klanten zouden willen gebruiken zijn bijvoorbeeld SQL-gebaseerd DBMS en andere LDAP-gebaseerde systemen dan AD of Tools4ever's Single Sign On (SSO) product E-SSOM.

Een ander probleem is dat de huidige AD-connector niet beschikt over onderstaande functionaliteiten, waardoor de dienst (de combinatie van de AD-connector en HelloID) niet zo effectief is als deze zou kunnen zijn:

- De status van een AD-Connector tonen in HelloID;
- Gegevens van gebruikers synchroniseren met HelloID;
- Offline ondersteuning voor gegevens in HelloID;

Doordat bovenstaande functionaliteiten ontbreken, loopt Tools4ever potentiële klanten mis of moeten klanten zelf synchronisatiemechanismes schrijven, wat niet ten goede komt aan het gebruiksgemak en de efficiëntie. Hierdoor staat het product HelloID minder sterk in de markt dan Cloud SSO-producten van concurrenten.



FIGUUR 1 HUIDIGE SITUATIE MET AD-CONNECTOR

In Figuur 1 is een diagram te zien van de huidige situatie, waarbij alleen authenticatie met AD mogelijk is.

3.1.2 GEWENSTE SITUATIE

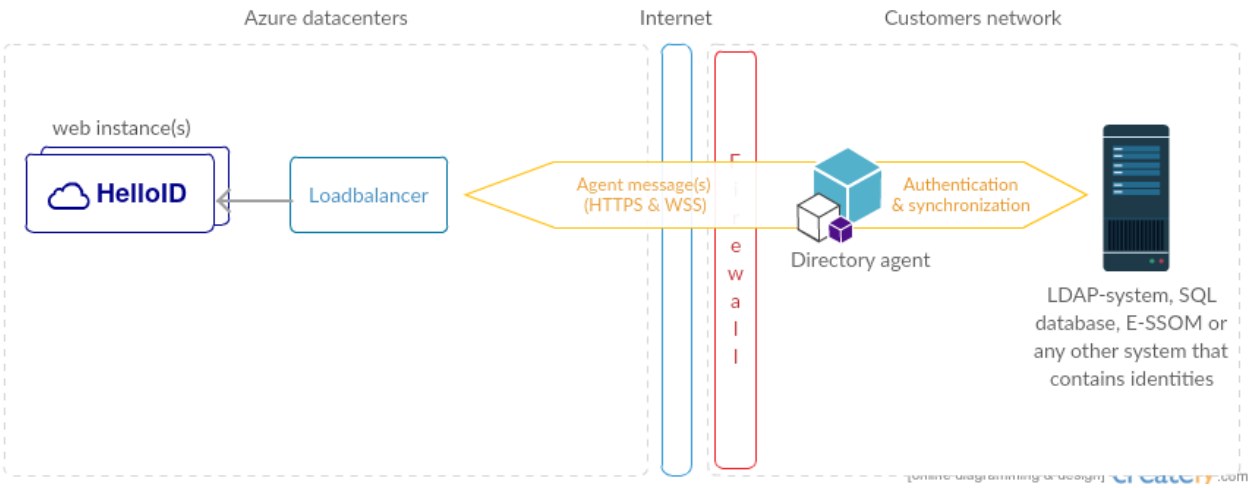
In de gewenste situatie zijn klanten niet meer genoodzaakt om zelf een synchronisatiemechanisme te schrijven om gebruikersgegevens te synchroniseren. Dit mechanisme is opgenomen in de nieuwe connector: de HelloID directory agent.

De nieuwe connector ondersteunt naast AD ook andere gegevensbronnen en is modulair opgezet, zodat, indien nodig, eenvoudig een gegevensbron kan worden toegevoegd. Out-of-the-box worden in de gewenste situatie de volgende gegevensbronnen ondersteund: AD (LDAP), SQL en E-SSOM[1]. Hierdoor kunnen meer klanten gebruikmaken van HelloID. Indien wij hun bron moeten ondersteunen, kunnen wij deze eenvoudig toevoegen aan de connector.

Verder bevat de directory agent ook de functionaliteit om gebruikers tegen gegevensbronnen te authentiseren en kan deze de status van de connectie weergeven in HelloID.

Met de realisatie van de HelloID Directory Agent is HelloID gebruiksvriendelijker en bruikbaar voor meer klanten. Dit heeft als gevolg dat HelloID beter in de markt geplaatst kan worden en kan concurreren met soortgelijke producten.

[1] Tijdens het verloop van het traject is gebleken dat andere requirements belangrijker waren dan out-of-the-box ook SQL en E-SSOM te ondersteunen. Zie hiervoor paragraaf 4.2.1.1 (Interview met opdrachtgever).



FIGUUR 2 GEWENSTE SITUATIE MET DIRECTORY AGENT

Het diagram in Figuur 2 laat zien dat de agent in de nieuwe situatie zowel authenticatie als synchronisatie mogelijk maakt en meerdere bronsystemen ondersteunt.

4 AANLOOPFASE

De eerste twee weken van mijn afstudeertraject heb ik gebruikt voor de aanloopfase. In deze fase heb ik mijn project 'opgestart': ik heb beschreven wat de aanpak zou zijn, met wie ik zou samenwerken en wat ik uiteindelijk zou gaan maken. De planning en aanpak heb ik beschreven in een plan van aanpak. Wat ik zou maken is beschreven in vorm van requirements in een requirementsrapport. Om te ontdekken wat de requirements waren, heb ik in de aanloopfase onder andere interviews gehouden met medewerkers van Tools4ever.

4.1 PLAN VAN AANPAK

Aan het begin van het traject heb ik een plan van aanpak opgesteld. Hierin heb ik beschreven hoe de organisatie en projectorganisatie in elkaar steken, wat de opdracht inhoudt, wat mijn werkwijze is en wanneer activiteiten worden uitgevoerd. Dit bood aan het begin van het traject enige houvast voor het verloop van het project. Daarnaast dient het plan van aanpak ook als overeenkomst met Tools4ever; wij moeten het eens zijn over de op te leveren producten en over hoe en wanneer ik deze realiseer. Deze overeenkomst heb ik gesloten met George Bakker, mijn afstudeerbegeleider en expert op het gebied van Single Sign On en Active Directory.

4.1.1 FASERING

Het project bestond uit twee fases. De eerste fase was de aanloopfase, waarin ik, zoals gezegd, het plan van aanpak heb geschreven en in de vorm van requirements beschreven heb wat ik ga maken. De volgende fase bestond uit vijf sprints. Tijdens een sprint worden onderdelen ontworpen, ontwikkeld en getest. Na elke sprint kan een nieuwe sprint worden gestart om het product uit te breiden of te verbeteren. Wat in de sprint wordt uitgevoerd, wordt bepaald in de sprintplanning. Zo is iteratief aan dit project gewerkt. Tools4ever hanteert een time-box (doorlooptijd) van twee weken.

Een sprint is een time-box (Network, TimeBox, 2008) die periodiek terugkeert. Dit heb ik in mijn plan van aanpak beschreven als een cyclische Scrum-fase. Tijdens en na elke sprint worden de sprint en software geëvalueerd. Op basis van deze evaluatie kunnen nieuwe requirements ontstaan. Deze worden in kaart gebracht en toegevoegd aan het requirementsrapport en de backlog van HelloID. In paragraaf 4.1.2.1 bespreek ik Scrum en hoe Tools4ever dit hanteert.

4.1.2 METHODE EN TECHNIEKEN

4.1.2.1 SCRUM

Tijdens dit project heb ik gewerkt volgens de software-ontwikkelmethode Scrum, omdat dit de standaard is bij Tools4ever B.V. en het afstudeertraject zo synchroon liep met de algemene sprint van HelloID. Zo kon het werk op dezelfde manier worden ingepland en geëvalueerd worden.

4.1.2.1.1 SCRUM

Scrum is een iteratieve ontwikkelmethode, waarbij nauw met de opdrachtgever en andere belanghebbenden wordt samengewerkt om elke sprint werkende software of valide documentatie op te leveren (Jeff Sutherland, 2014, p. 3). Zo kunnen nieuwe ideeën, kansen en inzichten naar voren komen tijdens het proces. Per sprint stellen we met elkaar vast wat binnen die sprint wordt ontwikkeld. Aan het einde van de sprint kijken we kritisch met elkaar naar het resultaat.

De HelloID Directory Agent wordt ontwikkeld op basis van eisen en wensen (requirements) vanuit Tools4ever. Met Scrum kan tijdens het ontwikkeltraject op een goede en flexibele manier met deze requirements worden omgegaan. Tools4ever hanteert Scrum als standaard bij het softwareontwikkelproces.

Scrum kent een aantal time-boxen voor voorgeschreven gebeurtenissen, zoals de sprint en bijeenkomsten tussen teamleden, bijvoorbeeld de sprintplanning (Jeff Sutherland, 2014, p. 8). Hieronder heb ik beschreven welke gebeurtenissen Tools4ever hanteert en hoe deze worden ingevuld.

4.1.2.1.2 Sprint (uitvoerende fase)

Tijdens een sprint worden onderdelen voor projecten (indien nodig) ontworpen, en vervolgens ontwikkeld en getest. De doorlooptijd van een sprint bij Tools4ever is twee weken. Na elke sprint kan een nieuwe sprint worden gestart om het project uit te breiden of te verbeteren. Wat in een sprint wordt uitgevoerd, wordt bepaald in de sprintplanning (zie paragraaf 4.1.2.1.4). Zo wordt iteratief aan een project gewerkt.

4.1.2.1.3 Daily stand-up

Elke werkdag van een sprint houdt het HelloID-team een stand-up met een time-box van vijftien minuten. Tijdens deze stand-up is het Scrum-team aanwezig en worden de volgende drie vragen beantwoord:

- Wat heb je gisteren gedaan?
- Loop je ergens tegenaan?
- Wat ga je vandaag doen?

Door het beantwoorden van deze vragen houden de teamleden elkaar op de hoogte en is duidelijk of iemand ergens tegenaan loopt. In dat geval kan een collega diegene helpen zoeken naar een oplossing.

4.1.2.1.4 Sprintplanning

De sprintplanning wordt altijd voor aanvang van een sprint gehouden en wordt bijgewoond door de producteigenaar/opdrachtgever (George Bakker), Scrum-Master (ik) en de rest van het Scrum-team (Aad Lutgert en Peter Vos). Tijdens de sprintplanning beschrijft de producteigenaar de requirements met de hoogste prioriteit aan het team. Op basis hiervan wordt door het Scrum-team en de producteigenaar het doel van de sprint bepaald en wordt de sprint backlog opgesteld. De time-box van onze sprintplanning is vier uur.

4.1.2.1.5 Sprintdoel

Een sprintdoel is een korte zin of samenvatting over wat het team van plan is te bereiken tijdens de sprint. Het sprintdoel kan worden gebruikt voor snelle rapportage buiten de sprint, bijvoorbeeld voor partijen die willen weten waar het team aan werkt, maar geen behoefte hebben aan de details van alle backlog items (zoals consultants en sales- of accountmanagers).

4.1.2.1.6 Sprint backlog

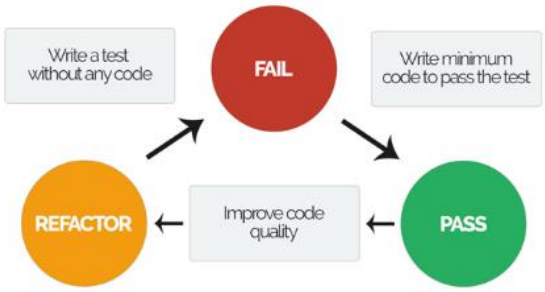
Naast het sprintdoel wordt ook de backlog opgesteld. De sprint-backlog is een lijst van requirements (user stories) die het team verwacht binnen de time-box van de sprint te kunnen realiseren. Zoals aangegeven wordt de sprint-backlog opgesteld door de producteigenaar en het Scrum-team. Deze backlog wordt bijgehouden en beheerd in de Scrum-tool Jira.

4.1.2.1.7 Retrospective

Na elke sprint houden wij een bijeenkomst (de 'retrospective') van ongeveer twee uur. In deze bijeenkomst evalueren wij onze eigen functioneren in de sprint, waarbij zowel gekeken wordt naar zaken die goed zijn verlopen als naar zaken die verkeerd zijn verlopen. Met het hele team overleggen wij hoe we zaken kunnen verbeteren. Een goed plan wordt als een user story opgenomen in de volgende sprint.

4.1.2.2 TEST DRIVEN DEVELOPMENT

In dit project heb ik nog een ontwikkelmethode toegepast, namelijk Test Driven Development (hierna te noemen: TDD). Bij TDD is het testen van software een integraal onderdeel van de softwareontwikkeling.



Bij deze ontwikkelmethode worden voor een nieuw deel code eerst de tests geschreven. De testresultaten zijn eerst negatief, omdat er geen implementatiecode is. Vervolgens wordt code geschreven om te zorgen voor positieve testresultaten, waarna de code wordt 'gerefactored' om de kwaliteit en beheersbaarheid ervan te verbeteren (Fowler, Refactoring, 2002, p. xvi). Deze activiteiten worden ook wel Red, Green en Refactor genoemd (Dan Pilone, 2008)

Ik heb TDD tijdens dit project toegepast, omdat wij voor de HelloID te weinig onze code testen. Door ervaringen op te doen met TDD en deze te delen met collega's, zouden wij deze methode in de toekomst kunnen toepassen bij de ontwikkeling van HelloID. HelloID wordt elke twee weken verder ontwikkeld, waarna een release-versie wordt opgeleverd. Soms kunnen aanpassingen aan delen van de code echter ongewenste fouten veroorzaken op andere locaties in de code. Met unit-tests zouden wij deze fouten eerder kunnen ontdekken en oplossen. TDD zorgt ervoor dat de ontwikkelaar unit-tests schrijft voor de code die wordt ontwikkeld.

Een andere reden om TDD toe te passen, is dat ik tijdens dit afstudeerproject ook een techniek wilde toepassen waar ik weinig ervaring mee had. Ik wil mijzelf continu verbeteren en ik vind dat ik dat doe door ook ervaring op te doen met het schrijven van unit-tests, waardoor ik anders moet nadenken over de code die ik schrijf. Hiermee volg ik een belangrijke tip uit het boek The *p*ragmatic programmer, namelijk: 'care about your craft' (Andrew Hunt, 2000). Met deze tip wordt bedoeld dat een ontwikkelaar om zijn vakgebied moet geven en altijd de kans moet grijpen om zichzelf te verbeteren.

4.1.2.3 MODELLERINGSTALEN EN -TECHNIEKEN

4.1.2.3.1 UNIFIED MODELING LANGUAGE

Voor het beschrijven van de architectuur en structuur van de software heb ik gebruikgemaakt van de modelleringstaal Unified Modeling Language (UML). UML is een modelmatige taal om objectgeoriënteerde[2] analyses en ontwerpen te maken voor informatiesystemen (Uml.org, 2005). Deze taal is de standaard in de software-enginering-industrie. Zowel op De Haagse Hogeschool als bij Tools4ever B.V. wordt deze taal gebruikt om onder andere klassendiagrammen, volgordediagrammen (sequence diagrams) en activiteitendiagrammen te ontwerpen.

4.1.2.3.2 ENHANCED ENTITY-RELATIONSHIP MODEL

Ook heb ik gebruikgemaakt van het Enhanced Entity-Relationship-model (EER-model of EERM). Dit model is bedoeld om een conceptueel datamodel inzichtelijk te maken en wordt vooral gebruikt om de structuur van entiteiten die in een database wordt opgeslagen te beschrijven (Cohen, 2002). Tijdens dit afstudeertraject heb ik het EER-model gebruikt om conceptuele diagrammen te maken en mijn databasestructuur te beschrijven.

[2] Dit is een paradigma dat gebruikt wordt bij programmeren, dataopslag of modelleren. Bij deze benadering wordt het model of informatiesysteem opgebouwd uit objecten.

4.1.2.4 PROGRAMMEERTAAL C#

De directory agent heb ik ontwikkeld in de programmeertaal C#. Deze taal is beschikbaar gesteld door Microsoft. C# is in mijn optiek een elegante en eenvoudige taal om objectgeoriënteerde software te schrijven. C# kan gebruikt worden om Windowsapplicaties te schrijven, waaronder Windows Services (Microsoft, Introduction to the C# Language and the .NET Framework, 2015).

Doordat ik C# heb gebruikt, kan de directory agent alleen in Windows-omgevingen worden geïnstalleerd. De keuze om C# te gebruiken is samen met de opdrachtgever gemaakt. Die keuze is er vooral op gebaseerd dat mijn expertise ligt in het schrijven van Windows-software en het gebruik van C#, waardoor ik de Directory Agent sneller en efficiënter kon ontwikkelen.

4.1.2.5 TOOLS

Voor verschillende processen, zoals het ontwikkelen van software, heb ik gebruikgemaakt van een aantal tools. Deze tools licht ik hieronder kort toe:

- **Jira** is een webapplicatie die het ontwikkelproces ondersteunt, door de requirements te organiseren en de voortgang hierop inzichtelijk te maken. Bij Tools4ever is de Jira zo geconfigureerd dat deze werkt op basis van het Scrum-proces. Wij registreren de requirements in Jira als user stories (zie paragraaf 4.1.2.1.6, Sprint backlog) en zetten deze in de juiste sprint, waarna de voortgang hierop gevolgd kan worden in het systeem.

- **Visual studio 2015** is een softwarepakket dat programmeurs een ontwikkelomgeving biedt. Het pakket bevat een grote set aan tools die de programmeur ondersteunen bij schrijven van de software. Het bevat bijvoorbeeld ondersteuning om de syntax van de programmeertaal C# duidelijk te maken, een debugger waardoor stap voor stap, tijdens het uitvoeren van de code, door de code heen kan worden gelopen om problemen op te sporen.

- **Team Foundation Server (TFS)** is een softwarepakket van Microsoft dat versiebeheer van onder andere de programmatuur regelt. Met dit pakket heb ik de code en testresultaten bijgehouden. Visual studio 2015 werkt samen met dit product, waardoor ik het gewijzigde werk eenvoudig kon opslaan in het versiebeheer.

4.2 REQUIREMENTS VERZAMELEN

In de aanloopfase heb ik de eerste versie van het requirementsrapport gemaakt om aan alle betrokkenen duidelijk te maken wat voor systeem moest worden ontwikkeld en welke redenen daaraan ten grondslag lagen. Daarnaast diende dit om aan te geven uit welke requirements de Directory Agent bestaat en in welke volgorde de requirements moesten worden gerealiseerd.

Globaal was de afdeling al bekend met de eisen en wensen voor de agent, maar deze waren nooit officieel vastgelegd. Ik heb eerst de meest globale requirements vastgelegd in een document. Dit was gelijk een goede voorbereiding op het bepalen van andere requirements.

Bij het opstellen en bepalen van de requirements heb ik veel gehad aan het Handboek Requirements (Swart, 2010). Dit boek bood mij een handvat bij het opstellen van het document en gaf informatie over de beste manieren om requirements te ontlokken. Zo heb ik bijvoorbeeld een prototype van de installatie-wizard opgesteld, waardoor ik meer details van requirements heb kunnen ontdekken (in paragraaf 4.2.1.3 ga ik hier nader op in).

4.2.1 ELICITEREN

Om requirements te eliciteren, heb ik gebruikgemaakt van twee elicitatietechnieken. Ik heb hierbij de processtappen uit het handboek gevolgd.

TABEL 1: ELICITATIETECHNIEKEN VOLGENS SWART EN DE DOOR MIJN GEBRUIKTE TECHNIEKEN

Processtap	Elicitatietechnieken	Door mij gebruikt
1. Positioneer het systeem binnen het businessdomein	Interviewen, workshops houden	Interviewen
2. Definieer de gewenste oplossing	Interviewen, prototype, workshops houden	Interviewen
3. Detailleer de requirements	Interviewen, prototype, workshops houden, observeren	Prototype

Eerst heb ik een interview gehouden met de opdrachtgever, George Bakker, om het systeem in het businessdomein te plaatsen. In dit interview hebben we het ook over de gewenste oplossing gehad. Daaarnaast heb ik, beheerder, Aad Lutgert geïnterviewd om te achterhalen wat volgens hem de

gewenste oplossing is. Tot slot heb ik een sessie met beiden gehouden, waarvoor ik een prototype van de installatie en het gebruik van de agent had gemaakt. Aan de hand van het prototype kon ik details in kaart brengen die ik anders niet achterhaald had.

4.2.1.1 INTERVIEW MET OPDRACHTGEVER

In het interview met de opdrachtgever heb ik het systeem in het businessdomein geplaatst. Zo zijn de beoogde procesverbeteringen, die met de ontwikkeling van de Directory Agent behaald moesten worden, duidelijker geworden. Een van de procesverbeteringen was al duidelijk:

- De opdrachtgever wil dat een klant niet afhaakt bij een intake omdat hun identiteitsbron niet wordt ondersteund.

Andere belangrijke procesverbeteringen die uit het gesprek naar voren kwamen zijn:

- De opdrachtgever wil dat klanten niet zelf een synchronisatiemechanisme hoeven te schrijven.
- De beheerder wil eenvoudig zelf kunnen bepalen hoe en waarnaartoe gegevens worden gesynchroniseerd tussen het bronsysteem en HelloID.

Deze laatste twee business-requirements hadden een grote impact op het project, omdat deze nog belangrijker zijn dan de ondersteuning van meerdere identiteitsbronnen (Active Directory is de belangrijkste bron, omdat deze door de meeste klanten wordt gevraagd). Dit betekende dat andere zaken minder belangrijk werden om ontwikkeld te worden.

In het interview is vervolgens de gewenste oplossing voor de Directory Agent bepaald. We hebben in kaart gebracht aan welke functionele en niet-functionele eisen het systeem moet voldoen om de business requirements te behalen. Hieruit werd vooral duidelijk welke instellingen ook moeten worden opgenomen bij het beheren van de Directory Agent. Het ging hierbij bijvoorbeeld om:

- Offline modus voor authenticatie: de beheerder wil kunnen aangeven dat gebruikers mogen inloggen op het portaal wanneer de agent offline is.

- Optie om aan te geven of gebruikers mogen worden aangemaakt in HelloID als zij geen account hebben in HelloID, maar wel in het bronsysteem (JIT-provisioning (Staaij, 2014, p. 86)).

4.2.1.2 INTERVIEW MET BEHEERDER

De beheerder, Aad Lutgert, vertegenwoordigt de belangrijkste gebruikersgroep die te maken heeft met de Directory Agent: de consultant/beheerder. In het interview met hem stond vooral de vraag centraal wat voor die groep de gewenste oplossing is.

Aad gaf aan dat hij blij was met de voorbereiding die was gedaan met de requirements die ik eerder had opgesteld. Hierdoor kon hij beter focussen en aangeven wat wel of niet gewenst was.

Hij noemde als belangrijkste factor dat het voor beheerders eenvoudig moet zijn om de agent te downloaden, installeren en configureren in HelloID. In zijn ogen zou het doorlopen van een configuratie-wizard helpen, omdat een beheerder dan minder fouten kan maken.

Belangrijke requirements die uit dit gesprek naar voren kwamen, zijn:

- De beheerder moet de agent eenvoudig kunnen configureren in HelloID aan de hand van een configuratie-wizard;

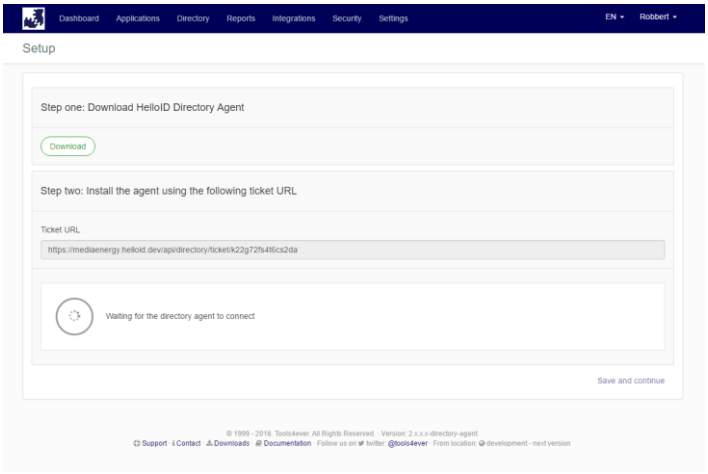
- De agent moet in de omgeving van de klant te installeren zijn door middel van een portaal-URL en een geheime sleutel;

- De agent moet na installatie in de wizard aangegeven dat hij succesvol een verbinding kan maken met het HelloID-systeem.

4.2.1.3 GEBRUIK VAN EEN PROTOTYPE

Uit de interviews was duidelijk dat de configuratie-wizard het belangrijkste functionele deel zou worden dat voor de gebruiker zichtbaar is. Ik wilde op dit gebied meer requirements ontlokken, wat ik heb gedaan door een prototype te maken.

Door het prototype te maken, kwam ik zelf al tot nieuwe inzichten. Zo heb ik ervoor gekozen om een Ticket URL op te nemen in plaats van de voorgestelde portaal URL en een geheime sleutel. De Ticket URL is een combinatie van die twee, waardoor de gebruiker een waarde minder hoeft te configureren.



FIGUUR 3 PROTOTYPE VAN DE WIZARD

In Figuur 3 is een prototype van de wizard weergegeven. Samen met de opdrachtgever en de beheerder heb ik gekeken naar deze visualisatie van de requirements. De opdrachtgever en de beheerder moesten een voorstelling maken van de toekomstige situatie, wat leidde tot een aantal nieuwe inzichten:

- In de configuratie-wizard moet het mogelijk zijn om bronspecifieke instellingen op te nemen, zoals de organisatie-units die mogen synchroniseren (Active Directory-specifiek);
- In de configuratie-wizard moet het mogelijk zijn om een deel van de mapping in te zien en eventueel aan te passen.
- Het moet mogelijk zijn om het proces te annuleren.

4.2.2 VERWERKEN REQUIREMENTS EN WERKEN MET MOSCOW

Ik heb de requirements verwerkt in het requirementsrapport. Hierin heb ik de context, business-, gebruikers- en softwarerequirements opgenomen. Voor het beschrijven van requirements heb ik het format van een user story gehanteerd:

<Identificatiecode>	**<Titel>**	**<Prioriteit>**
Als <actor> wil ik <eis/wens>.		

De gebruikersrequirements heb ik gegroepeerd op belanghebbende. De software-requirements heb ik ingedeeld op de ISO 9126:2001 standaard. Deze standaard heb ik ook in mijn interviews gebruikt, door te vragen wat de geïnterviewden dachten bij bepaalde kwaliteitseigenschappen. Daarnaast heb ik ook zelf requirements voor de kwaliteitseigenschappen geformuleerd. Om de requirements die ik zelf had opgesteld te verifiëren, heb ik deze doorgenomen met de opdrachtgever.

Bij het verzamelen en verwerken van de requirements werd duidelijk dat het project veel groter zou worden en langer zou duren dan mijn afstudeerproject. Ik moest daarom prioriteiten stellen aan de hand van de vragen wat het afstudeerproject tot een succes maakt en welke onderdelen na het afstuderen kunnen worden opgepakt. Om deze prioriteiten te stellen hebben ik in overleg met de opdrachtgever gebruikgemaakt van MoSCoW-methode. MoSCoW staat voor: **M**ust have, **S**hould have, **C**ould have en **W**on't have (MoSCoW-methode, 2015). Met behulp van de MoSCow-methode werd inzichtelijk welke requirements belangrijk zijn voor het succes van deze afstudeeropdracht en welke requirements in een volgende sprint (buiten het afstuderen) kunnen worden opgepakt.

4.3 TESTPLAN

Nadat ik klaar was met het opstellen en verwerken van de requirements, heb ik een testplan opgesteld. Dit plan had als doel om de betrokkenen bij het project een overzicht te bieden van de soorten testen en activiteiten.

4.3.1 AANPAK

Zoals gezegd heb ik gebruikgemaakt van de ontwikkelmethode TDD (zie paragraaf 4.1.2.2), waarbij het testen een integraal onderdeel is van de softwareontwikkeling. Bij TDD worden 'white-box tests' opgesteld. In mijn aanpak heb ik beschreven wat voor soort tests dit zijn, wat deze tests inhouden, hoe deze moeten worden toegevoegd aan het project en welke ondersteunende software hierbij wordt gebruikt. Naast de white-box tests is ook een black-box test opgenomen.

4.3.1.1 WHITE-BOX TESTS

White-box tests zijn tests die worden geïmplementeerd met kennis van het gehele systeem (Grood, 2008, p. 218). Het doel van deze tests is het testen van de interne structuren en de werking van het systeem. Een bekende vorm van een white-box test in een moduletest (of unit-test).

4.3.1.1.1 MODULETEST

De eerste testsoort die ik heb beschreven is de moduletest. Moduletests zijn kleine stukken code of scripts die onafhankelijk softwarecomponenten testen (Grood, 2008, pp. 76,218). Deze tests mogen geen andere componenten of externe bronnen aanspreken. Ze worden geschreven om de correcte werking en structuur van een individueel component te testen.

Met TDD worden voor tests opgesteld bij nieuw stuk code. Bij elke wijziging kunnen deze tests opnieuw worden uitgevoerd. Hierdoor kunnen eventuele problemen (bugs) vroeg worden gedetecteerd.

Om de tests op te stellen heb ik gebruikgemaakt van Microsoft Visual Studio en de standaard softwarebibliotheek die hierbij geleverd wordt om unit-tests op te stellen. Naast Visual Studio heb ik gebruikgemaakt van de extensie DotCover, geleverd door JetBrains. Door middel van deze extensie kunnen rapportages van de moduletests in pdf-formaat opgeslagen en gearchiveerd worden.

4.3.1.1.2 INTEGRATIETEST

De tweede testsoort dit ik in het testplan heb beschreven is de integratietest. Deze tests zijn net als moduletests kleine stukken code. In tegenstelling tot de moduletests, testen de integratietests de samenwerking tussen componenten of het aanspreken van externe bronnen (Grood, 2008, pp. 48,218). De integratietests hebben als doel om de betrouwbaarheid van de samenwerking tussen componenten te verifiëren. Het opstellen, rapporteren en archiveren gebeurt op dezelfde manier als bij de moduletests.

4.3.1.2 BLACK-BOX TESTS

Black-box tests testen de functionaliteit van een applicatie zonder kennis van de interne werking en structuur van het systeem (Grood, 2008, p. 219). Bij Tools4ever wordt ook gebruikgemaakt van error guessing. In elke sprint werden ontwikkelde requirements getest door onze tester (Aad Lutgert) door middel van error guessing.

4.3.1.2.1 ERROR GUESSING

Zoals de naam impliceert, is error guessing het gissen naar fouten (Koomen, 2014, p. 641). Tools4ever focust hierbij vooral op arbitraire input in de userinterface. De waarde van deze tests ligt in het onverwachte. Op basis van ervaring gaan wij op zoek naar foutgevoelige plekken in het systeem en bedenken wij tests tijdens de testuitvoering.

Voor deze tests zijn geen scripts of testscenario's uitgeschreven. Gevonden fouten zijn vermeld in requirements die zijn geregistreerd in Jira, waarin wij de voortgang van de requirements en de sprint bijhouden (paragraaf 4.1.2.5 gaat dieper in op Jira). Wanneer een fout werd gevonden, werd deze door mij hersteld en kon opnieuw worden getest.

5 SPRINT 1: ONTWERPEN EN IMPLEMENTEREN VAN HET DOMEINMODEL

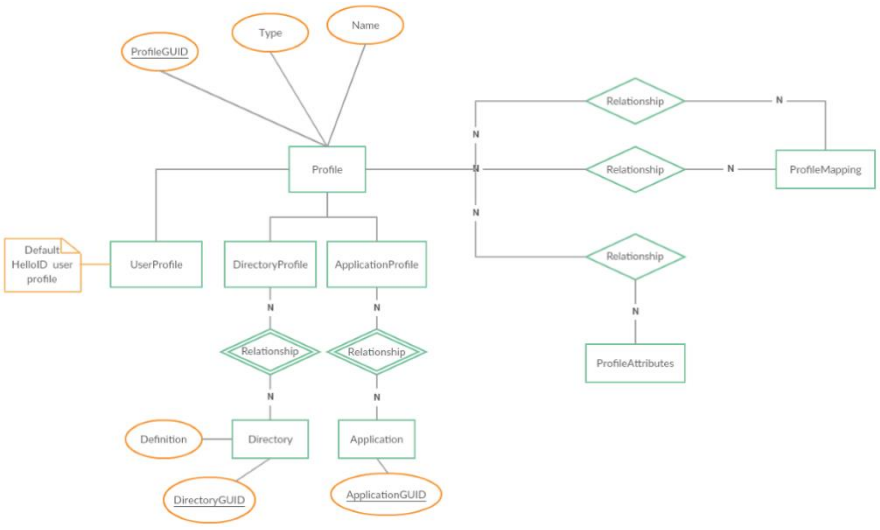
In de eerste sprint stond een nieuw domeinmodel centraal. Het nieuwe domeinmodel moest nieuwe onderdelen en wijzigingen die de Directory Agent in het model van HelloID teweegbrengt in kaart brengen.

5.1 ONTWERPEN VAN HET NIEUWE DOMEINMODEL

De sprint ben ik gestart met het ontwerpen van het nieuwe domeinmodel. Met de begeleider was besloten dat ik in deze sprint het model iteratief zou maken en vervolgens zou presenteren aan de rest van het team.

Ik ben begonnen met een EER-model (zie Figuur 4), waarin ik vooral duidelijk wilde maken dat wij rekening moesten houden met verschillende soorten profielen. Uit het bronsysteem worden onder meer gebruikersgegevens en groepsgegevens opgehaald. Ergens moet worden vastgelegd hoe deze gegevenssets eruitzien. Dit wilde ik bereiken met de entiteit Profile. Naast het bronsysteem zijn er gegevenssets in HelloID waarnaartoe gegevens moeten worden overgezet. Ook hiervan moest de definitie worden vastgelegd, die in een Profile-entiteit zou kunnen worden opgeslagen.

Tussen de profielentiteiten zou vervolgens een mapping (ProfileMapping) tot stand kunnen komen om daarin te bepalen hoe en waarnaartoe velden worden overgezet tijdens synchronisatie.



FIGUUR 4 EERSTE EER-MODEL IN DE CONCEPTUELE FASE

De presentatie van dit model zorgde voor veel opmerkingen, discussies en inzichten.

5.1.1 PROFIEL- EN ATTRIBUTENSTRUCTUUR

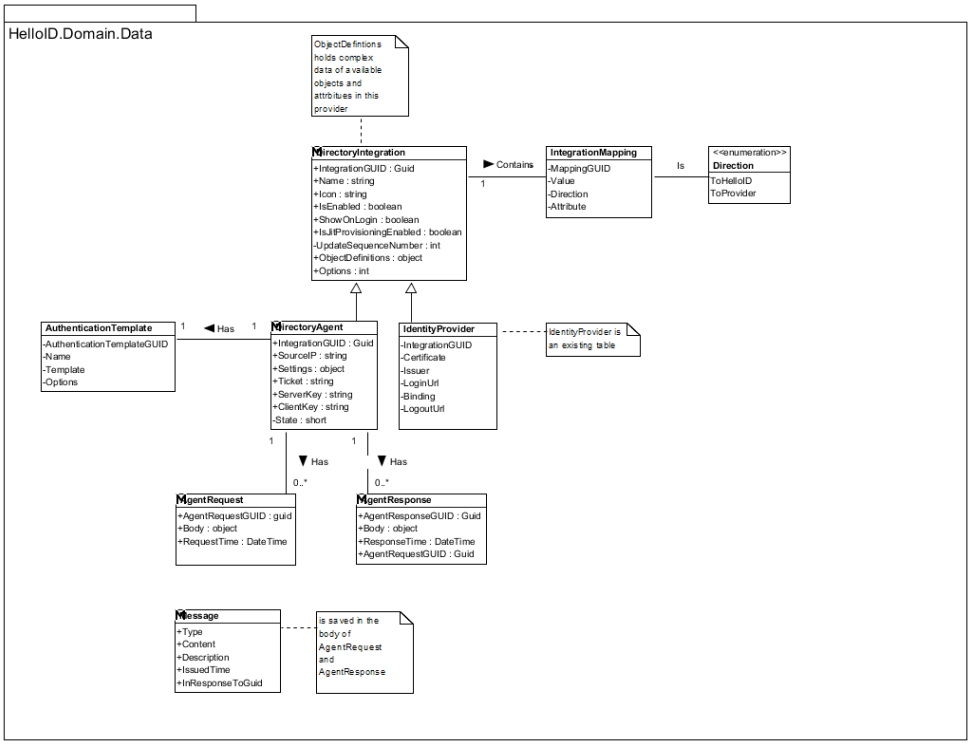
Het belangrijkste discussiepunt ging over de structuur van profielen en de attributen die de profielen bevatten. Mijn begeleider stelde dat uit de mapping (entiteit met attribuut voor bronveld en attribuut voor doelveld) het profiel en de attributen kunnen worden afgeleid en dat het redundant is deze in aparte entiteiten op te slaan. Ik was het hier niet mee eens; ik vond dat de mapping en de definitie van gegevens twee verschillende zaken waren.

Ik stelde dat wij met de definitie van gegevens ook andere acties willen uitvoeren die losstaan van de mapping. Zo zouden bijvoorbeeld regels kunnen worden uitgevoerd op een bepaald attribuut. Een voorbeeld hierbij is: als het attribuut ‘Status’ van een gebruiker ‘Uit dienst is’, dan mag de gebruiker niet gesynchroniseerd worden naar HelloID. Het attribuut hoeft niet te zijn opgenomen in de mapping. Mijn begeleider was door mijn stelling echter nog niet overtuigd.

In een volgend model heb ik de structuur van profielen en attributen anders uitgewerkt (zieFiguur 5). In dit model zijn geen extra profiel- en attributenentiteiten aanwezig, maar worden profieldefinities bijgehouden in een complexe datastructuur door middel van een JSON-bestand. Bij de presentatie van dit model heb ik meer argumenten genoemd waarom het opnemen van een definitie nodig is:

> Als een definitie van de datastructuur beschikbaar is, kan in de userinterface kenbaar worden gemaakt welke attributen aanwezig zijn in het bronsysteem, los van de vraag of deze wel of niet in de mapping aanwezig zijn. Het kan namelijk zo zijn dat klanten sommige attributen niet willen synchroniseren en daarom niet in de mapping opnemen. Wanneer deze attributen niet meer in de mapping staan, is nog wel duidelijk dat deze kunnen bestaan in het bronsysteem;

Op basis van deze argumentatie en het model was mijn begeleider wel overtuigd. Wij hoefden geen omslachtige structuur op te nemen, doordat wij de definitie in een JSON-bestand opnamen. Een bijkomstigheid bij het opslaan van de structuur in JSON is dat wij beter rekening kunnen houden met complexe typen data, zoals attributen die uit een objectstructuur bestaan. Een andere bijkomstigheid is dat het uitbreiden van deze structuur flexibel is, waardoor wij verwachten minder databasewijzigingen in het model hoeven uit te voeren.



FIGUUR 5 ANALYSE KLASSENDIAGRAM

5.1.2 OPNEMEN VAN IDENTITY PROVIDERS

Een inzicht dat ik kreeg bij het maken van dit domeinmodel is dat de structuur van onze identity providers erin past. Een identity provider is een entiteit binnen HelloID en staat voor een identiteitsbron die identiteiten levert door middel van een federatieprotocol als SAML. Als wij de identity providers in dit model opnemen, dan kunnen wij de synchronisatie en mappingfunctionaliteiten ook bij de identity providers gebruiken. Net als de Directory Agent is een identity provider een bron van identiteiten. Deze gegevens willen wij overzetten naar HelloID. In bovenstaand model (Figuur 5) heb ik dit daarom opgenomen als een afgeleide van een basisentiteit, genaamd DirectoryIntegration.

Het team was het eens met deze wijzigingen. Om de huidige structuur van identity providers over te zetten, moeten wel veel data worden gemigreerd en moeten veel grafische weergaven worden aangepast. Wij hebben besloten om wel de identity providers op te nemen in dit model, maar dat het migratiewerk en andere aanpassingen buiten de scope van het afstudeerproject vallen en daarom in een later traject zullen worden opgepakt.

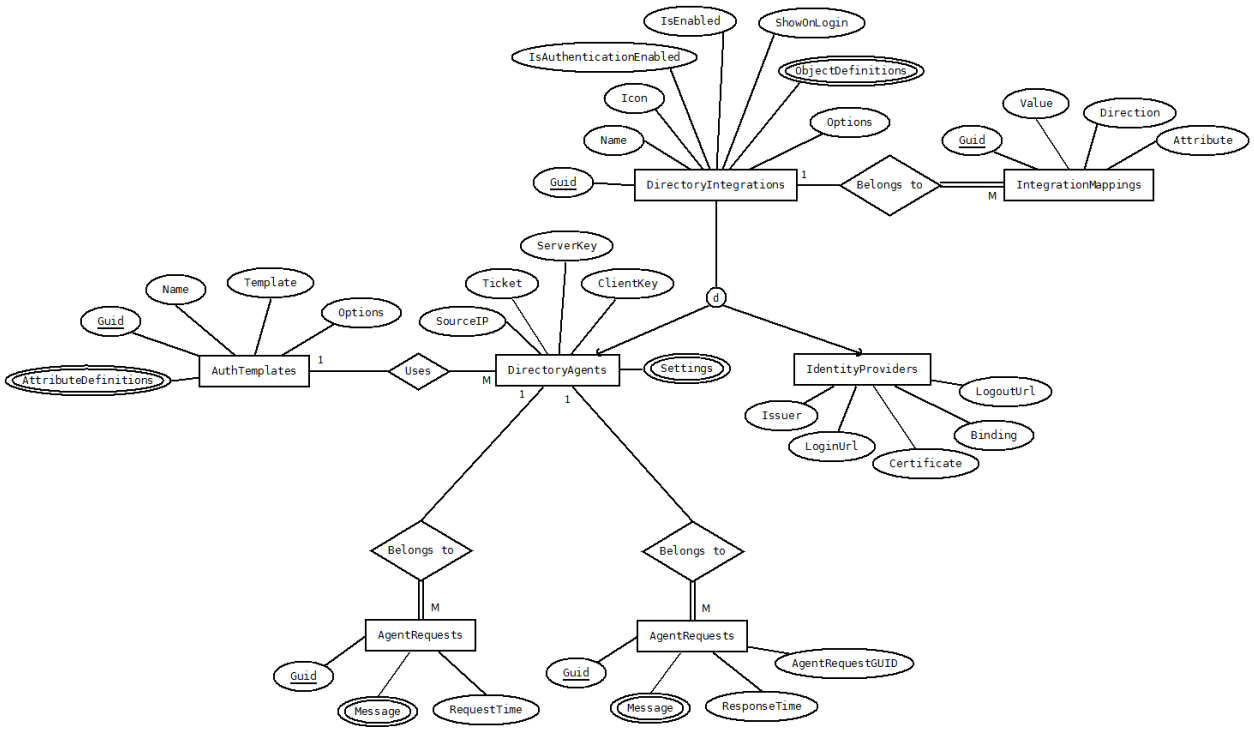
5.1.3 AUTHENTICATIE-TEMPLATES

Een ander inzicht dat tijdens het ontwerpen van het model aan het licht kwam, is dat elke bron waarschijnlijk met andere velden wil authentiseren. De meeste systemen zullen een gebruikersnaam- en wachtwoordcombinatie gebruiken, maar andere systemen, zoals Active Directory, hebben bijvoorbeeld ook een domeinnaam nodig of willen gebruikmaken van een schoolnaam of gebruikerstype. Om dit te faciliteren moet ook het inlogscherm uit te breiden zijn met deze velden. Hiervoor heb ik een entiteit AuthenticationTemplate opgenomen.

Ook hiervan vonden mijn collega's dat dit in model moet worden opgenomen, maar dat het schrijven van de software voor de loginschermen buiten de scope van het afstudeerproject valt.

5.2 OPSTELLEN VAN EER, RRM EN RIM.

De goedgekeurde versie van het model heb ik vertaald naar een Enhanced Entity-Relationship (EER) model, Relational Representation Model (RRM) en een Relational Implementation Model (RIM). Deze modellen heb ik opgesteld met behulp van eigen kennis en het boek Principes van Database, van Guy de Tré. Het RRM en RIM zijn terug te vinden in de ontwerpdocumentatie.



FIGUUR 6 EER-MODEL VAN DE DATABASE

5.2.1 Overerving in de database

Tijdens het opstellen van het EER, RRM en RIM moest ik rekening houden met de overerving in het domeinmodel van de basisentiteit DirectoryIntegration en de afgeleide entiteiten DirectoryAgent en IdentityProvider. Voor overerving in de database kan een aantal strategieën gekozen worden (Fowler, Chapter 12. Object-Relational Structural Patterns, 2002). Deze worden op de volgende pagina's toegelicht.

Single table inheritence (ook wel: Table per hierarchy inheritence)

Dit is de eenvoudigste aanpak; alle attributen van subtypes worden in dezelfde tabel opgenomen. Het subtype wordt aangeduid door middel van een extra attribuut. Het type bepaalt welke kolommen wel of niet van toepassing zijn. Als ik mijn model als uitgangspunt neem, zal de tabel er ongeveer zo uitzien:

Tabel: DirectoryIntegrations

Guid	Type	Name	Issuer	Ticket
Fef23d-sd29…	IDP	Alpha	Salesforce	NULL
86f3B-d300-e…	AGENT	Beta	NULL	4f3Hhsdd23d==
Kolommen die alle subtypen moeten bevatten			Subtype specifieke kolommen	

Deze aanpak heeft een aantal nadelen:

- Het is niet meteen duidelijk welke attributen wel of niet van toepassing zijn op een bepaald subtype;
- Invullen van bepaalde gegevens kan niet afgedwongen worden door NOT NULL in de tabel op te nemen; dit moet bijvoorbeeld geregeld worden door middel van CHECK constraints of in de programmatuur van de applicatie. Dit kan problematisch zijn als meer subtypen worden toegevoegd.

Concrete table inheritence (ook wel: Table per concrete class)

In een andere aanpak wordt één tabel per subtype gemaakt, waarbij alle velden die gedeeld worden door de subtypen worden herhaald. De tabellen zouden er dan op de volgende manier uitzien:

Tabel: IdentityProvider

Guid	Name	Issuer
Fef23d-sd29…	Alpha	Salesforce

Table: DirectoryAgent

Guid	Name	Ticket
86f3B-d300-e…	Beta	4f3Hhsdd23d==

Deze aanpak lost een aantal problemen uit de vorige aanpak op:

- Invullen van verplichte velden kan worden afgedwongen door een NOT NULL-beperking op te nemen;
- Als nieuwe subtypen toegevoegd worden, kan een nieuwe tabel worden opgenomen en hoeven geen kolommen te worden toegevoegd aan een bestaande tabel, waarvoor constraints en eventuele applicatielogica aangepast moeten worden;
- Het risico bestaat niet dat velden worden gevuld van een ander subtype dan de bedoeling was;
- Het is niet nodig om een extra kolomtype op te nemen. Het wordt namelijk al bepaald door de metadata van de tabel zelf: de tabelnaam.

Deze aanpak heeft echter ook een aantal nadelen:

- De gedeelde kolommen worden alsnog gemixt met subtype-specifieke kolommen;
- Wanneer een nieuw subtype wordt toegevoegd, moeten de gedeelde kolommen worden herhaald in de tabel van het subtype;
- Het bijhouden van de identifier wordt een uitdaging, zeker als dit een auto increment zou zijn (dat is in dit geval niet van toepassing).
- Het ophalen van gedeelde kolommen van alle subtypen van DirectoryIntegrations zou moeten worden gefaciliteerd door middel van UNION-query's. Indien een subtype moet worden gemaakt, moeten de UNION-query's worden aangepast.

Voorbeeld van UNION-query:

```
SELECT Guid, Name FROM IdentityProvider

UNION

SELECT Guid, Name FROM DirectoryAgent
```

Class table inheritence (ook wel: Table per type)

In een andere aanpak wordt voor het basistype en voor elk subtype een tabel aangemaakt. In de basistabel worden alle gedeelde kolommen opgenomen en in elke subtypetabel de subtype-specifieke kolommen. In het subtype geldt de identifier, de primaire sleutel, ook als vreemde sleutel naar de basistabel.

Dit ziet er zo uit:

Tabel: DirectoryIntegrations

Guid	Name
Fef23d-sd29…	Alpha
86f3B-d300-e…	Beta

Tabel: IdentityProvider

Guid	Issuer
Fef23d-sd29…	Salesforce

Tabel: DirectoryAgent

Guid	Ticket
86f3B-d300-e…	4f3Hhsdd23d==

Deze aanpak lost de volgende problemen op:

- Het invullen van verplichte velden kan worden afgedwongen door een NOT NULL-constraint op te nemen;
- Als nieuwe subtypen worden toegevoegd, kan een nieuwe tabel worden opgenomen en hoeven geen kolommen te worden toegevoegd aan een bestaande tabel, waarvoor constraints en eventuele applicatielogica aangepast moeten worden;
- Er is geen risico dat velden worden gevuld van een ander subtype dan de bedoeling was;
- Het is niet nodig om een extra kolomtype op te nemen. Het wordt namelijk al bepaald door de metadata van de tabel zelf: de tabelnaam.
- Gedeelde kolommen kunnen op één plek worden toegevoegd of verwijderd;
- De identifier wordt in de basistabel bijgehouden.
- Het wordt eenvoudiger om gedeelde kolommen op te halen, omdat hiervoor geen UNION-query geschreven hoeft te worden.

Deze aanpak heeft wel een nadeel:

- Om een tuple (rij) te verwijderen, moet zowel de basistabel als de subtype-specifieke tabel worden bijgewerkt.

Gekozen aanpak

Ik heb gekozen voor de laatste aanpak, class table inheritence, omdat ik het belangrijk vind dat het invullen van verplichte velden kan worden afgedwongen met een NOT NULL-constraint en dat kolommen zo min mogelijk herhaald hoeven worden. Hiernaast hoeft bij het ophalen van basiskolommen geen query worden geschreven met een UNION, waardoor het beheer van deze query's ook wegvalt.

Het nadeel dat zowel de basistabel als de subtypespecifieke tabel moet worden bijgewerkt om een tuple te verwijderen, is opgelost door in het RIM een cascading delete op te nemen: wanneer een tuple uit de basistabel wordt verwijderd, wordt ook de bijbehorende tuple uit de subtype-tabel verwijderd.

6 SPRINT 2: COMMUNICATIE TUSSEN AGENT EN HELLOID

Bij de tweede sprint was het doel om een goede verbinding en daarmee communicatielijn tussen de Directory Agent en HelloID te ontwikkelen. Hiervoor heb ik een Windows Service geschreven die een verbinding opzet en die controleert of nieuwe berichten in HelloID aanwezig zijn die geconsumeerd moeten worden door deze service. Daarnaast ben ik begonnen met het ontwerp en de ontwikkeling van de modulaire architectuur voor de identiteitsbronnen.

6.1 OPZETTEN VAN WINDOWS SERVICE

Voordat het afstudeertraject begon, hadden wij al besloten dat de agent in een Windows Service zou worden ontwikkeld en in C# zou worden geschreven. Dit betekent wel dat de service platformspecifiek is en alleen op Microsoft Windows kan worden geïnstalleerd. Deze beslissing hebben wij genomen ten behoeve van ontwikkelsnelheid. 99 procent van de klanten van Tools4ever heeft een Windows-omgeving en mijn expertise ligt in het schrijven van Windows-software. Op dit platform kan ik daardoor sneller en efficiënter code schrijven.

Het project heb ik aangemaakt in de nieuwste Visual Studio. Deze versie bevat een van de nieuwste C# compilers (versie 6.0), waardoor ik gebruik kon maken van handige nieuwe taalfeatures die code vereenvoudigen en verduidelijken. Zo heb ik tegelijkertijd deze features leren gebruiken. Een voorbeeld van een nieuwe feature is NULL propagation; om een NullReferenceException te voorkomen, zou je onderstaande oplossing kunnen schrijven.

```
if (result != null)
{
       result.DoSomething();
       return;
}
```

Door NULL propagation te gebruiken, wordt de code eenvoudiger en beter leesbaar. Hieronder is een stukje code weergegeven waarbij NULL propagation wordt gebruikt.

```
result?.DoSomething();
return;
```

Nadat ik het project had opgezet, heb ik een klasse aangemaakt die nodig is om het project als een Windows Service te kunnen draaien. Deze klasse moet overerven van de klasse ServiceBase. Vervolgens moet deze klasse worden geïnstantieerd en mee worden gegeven aan de statische methode Run van ServiceBase. Dit moet gebeuren in de main entry point van de applicatie.

```
public static class Program
{
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        public static void Main(params string[] args)
        {
              ServiceBase.Run(new WindowsService());
        }
}
```

6.2 Ontwikkelen van modulaire structuur

Toen ik een werkende service had, ben ik gestart met het ontwerpen en ontwikkelen van de architectuur van de agent. Hierbij heb ik rekening gehouden met het feit dat de service modulair opgezet moet zijn en door ons en derde partijen uit te breiden moet zijn. Het moet eenvoudig zijn om in de toekomst nieuwe identiteitsbronnen toe te voegen.

Mijn oplossing hiervoor is het runtime inladen van een identiteitsbron, dynamic-link libraries (DLL). Voor elke identiteitsbron wordt een aparte DLL geschreven en in de root folder van de Directory Agent geplaatst. Door middel van de configuratie wordt bepaald welke DLL wordt ingeladen.

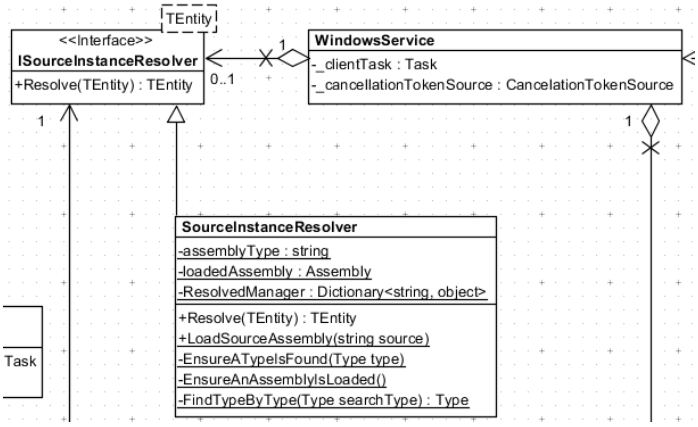
Een voordeel hiervan is dat derde partijen eenvoudig hun eigen implementatie schrijven en compileren naar een DLL; zij kunnen hun eigen implementatie en dus identiteitsbron gebruiken in combinatie met ons product. Dit voordeel geldt ook voor ons: als wij bijvoorbeeld CSV als bron willen ondersteunen, kunnen wij deze implementatie schrijven in een aparte DLL en meeleveren met de agent.

Een ander voordeel is dat het op deze manier mogelijk is om op afstand de implementatie van identiteitsbronnen te updaten. Als er een bug zit in een van de identiteitsbronnen, dan zouden wij de DLL bij alle klanten op afstand kunnen bijwerken.

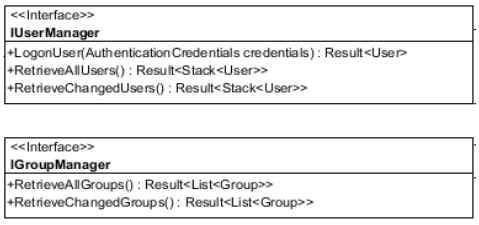
Een nadeel van runtime inladen van een DLL is dat de performance afneemt als componenten vaak moeten worden ingeladen. Om dit probleem te voorkomen heb ik gebruikgemaakt van een ServiceLocator klasse die die verantwoordelijkheid op zich neemt en die door een cachingmechanisme (Dictionaries, Cache, n.d.) de performanceproblemen oplost, omdat hierdoor componenten maar eenmalig worden aangemaakt.

6.2.1 SERVICELOCATOR

Binnen mijn architectuur is de SourceInstanceResolver-klasse (Figuur 8 SourceInstanceResolver) een ServiceLocator-klasse (zie de paragraaf 6.2.2). Deze klasse is verantwoordelijk voor het aanmaken van concrete objecten waarvan andere objecten afhankelijk zijn en die een gemeenschappelijk thema hebben, ook wel een abstract factory genoemd (Shvets, 2015). Verder bevat deze klasse een cachingmechanisme om eerder aangemaakte objecten op te zoeken, zodat objecten alleen indien nodig opnieuw hoeven worden aangemaakt.



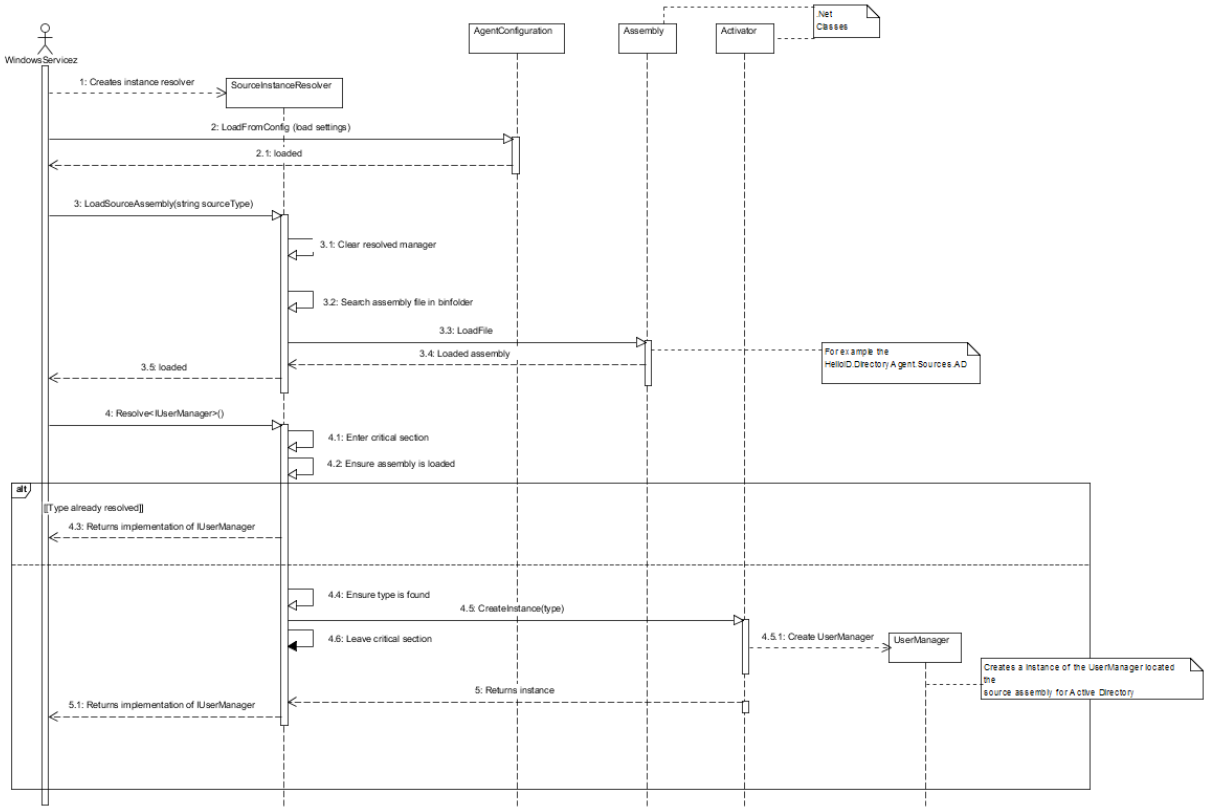
FIGUUR 8 SOURCEINSTANCERESOLVER



FIGUUR 9 PUBLIEKE API

Deze SourceInstanceResolver leest uit de bin folder in de root folder van de agent alle assemblies die voldoen aan de naamgeving: HelloID.DirectoryAgent.Sources.<SourceType>. Als de Resolve-methode wordt uitgevoerd, dan zoekt hij alleen klassen die overerven van interfaces van de publieke API. De publieke API bevat alle publieke elementen om een identiteitsbron te maken die door de agent kan worden gebruikt (Figuur 9).

Om te verduidelijken hoe een DLL wordt ingeladen en hoe een klasse wordt opgezocht (en indien nodig aangemaakt), heb ik in mijn ontwerpdocumentatie onderstaand sequentiediagram (Figuur 10 Resolving IUserManager) opgenomen.



FIGUUR 10 RESOLVING IUSERMANAGER

In bovenstaand diagram is te zien hoe een implementatie voor de IUserManager van een module wordt ingeladen. De Windows Service instantieert de SourceInstanceResolver. Dan laadt de service de configuratie in. Deze configuratie bevat een type dat aangeeft welke source assembly (DLL) moet worden gebruikt als een klasse wordt opgezocht en ingeladen. De service roept de methode LoadSource Assembly aan, waarna de assembly wordt opgezocht en ingeladen. De service roept de methode Resolve aan met het aangegeven type interface. Deze methode stapt een critical section in (zie de paragraaf 6.2.3)). Dan zorgt de methode ervoor dat alles correct is ingeladen en maakt een instantie aan, waarna de methode deze teruggeeft aan de service.

6.2.2 Inversion of Control

Door gebruik te maken van de SourceInstanceResolver klasse heb ik onder andere de principes van Inversion of Control (IoC) en Dependency Injection (DI) toegepast (Inversion of Control Containers and the Dependency Injection pattern, 2004).

Inversion of Control
IoC in een principe waarbij, onafhankelijk van welke taal wordt gebruikt, geïnstantieerde objecten niet zelf andere objecten aanmaken. In plaats daarvan krijgen zij de benodigde objecten van een externe dienst. Voorbeelden hierbij zijn het gebruik van Dependency Injection door middel van constructor injection of gebruikmaken van een ServiceLocator-klasse.

Dependency Injection
Dependency Injection betekent dat afhankelijke objecten worden verkregen door middel van IoC, maar dat hierbij geen gebruik wordt gemaakt van concrete objecten maar van abstracties of interfaces. Een bijkomstigheid hiervan is dat de code testbaar is.

ServiceLocator-klasse
In principe is een ServiceLocator-klasse een factory-klasse (een klasse die verantwoordelijk is voor het aanmaken van objecten) die geïnstantieerde objecten opzoekt of, indien geen object gevonden wordt, er één instantieert op het moment dat het nodig is.

Door deze principes toe te passen, is de code beter uit te breiden. Daarnaast ontstaat er een losse koppeling, omdat niet meer naar de concrete implementaties wordt verwezen, maar naar een abstracties.

Behalve dat een losse koppeling bereikt wordt en de performanceproblemen opgelost worden, is er het voordeel dat de implementatie bij de unit-tests kan worden uitgewisseld door ‘mockobjects’, objecten om de eigenschappen en werking van andere objecten te simuleren (Mockobject, 2013). Hierdoor is de testbaarheid van de code verbeterd.

De volgende pagina toont een voorbeeld van hoe ik Dependency Injection heb toegepast met behulp van de SourceInstanceResolver-klasse (mijn ServiceLocator). De MessageBus verwacht een implementatie van UserManager. In dit voorbeeld heb ik een extra constructor gemaakt waaraan de ServiceLocator kan worden meegegeven; normaal gesproken zou dit buiten de klasse om gebeuren.

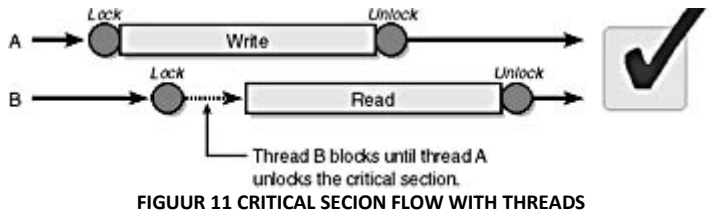
```
using HelloID.DirectoryAgent.Internal;
using HelloID.DirectoryAgent.Shared.Interfaces;

internal class MessageBus : IMessageBus
{
    public readonly IUserManager _manager;

    public MessageBus(ISourceInstanceResolver r) : this(r.Resolve<IUserManager>())
    {

    }

    public MessageBus(IUserManager manager)
    {
        _manager = manager;
    }
}
```

6.2.3 THREAD-SAFE MAKEN VAN RESOLVE METHODE

Voor de implementatie van de SourceInstanceResolver moest de klasse thread-safe zijn. Om te voorkomen dat threads elkaar in de weg zitten bij het aanmaken of het opvragen van een klasse, heb ik onder andere gebruikgemaakt van het lock keyword (Microsoft, lock Statement (C# Reference), 2015). Dit markeert het statement block als een 'critical section'.

Een critical section verzekert dat Thread A niet de sectie in kan als thread B in die sectie is. Thread A wacht tot het object is vrijgegeven door Thread B (zie Figuur 11). Voorbeeld van het lock statement:

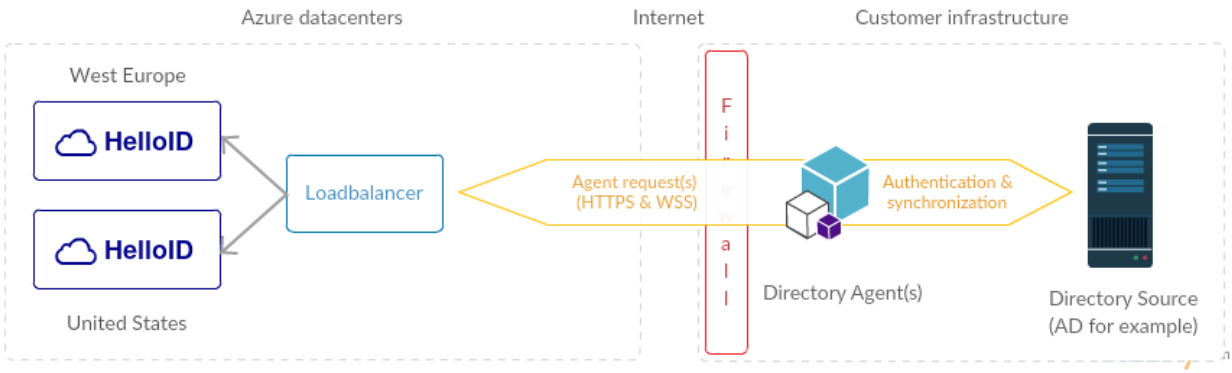
```
public TEntity Resolve<TEntity>()
{
       lock (ResolvedManagers)
       {
              //statement
       }
}
```



FIGUUR 11 CRITICAL SECION FLOW WITH THREADS

6.3 De verbinding naar HelloID

Nadat ik de algemene architectuur had ontworpen, ben ik gestart met het mechanisme om een verbinding op te zetten naar HelloID. Over deze verbinding moeten berichten tussen de twee systemen worden uitgewisseld. Om de context waarin de systemen zich bevinden te verduidelijken, heb ik een contextdiagram gemaakt en opgenomen in de ontwerpdocumentatie.



FIGUUR 12 CONTEXTDIAGRAM

Deze Directory Agent is ontwikkeld om bijvoorbeeld te faciliteren dat gebruikers op HelloID kunnen inloggen met gegevens uit een ander systeem of gegevens kunnen synchroniseren tussen een bronsysteem en HelloID. De agent moet in de omgeving van de klant worden geïnstalleerd. De agent zet een verbinding op via HTTPS (Dictionaries, HyperText Transfer Protocol Secure, n.d.) naar een HelloID-instantie die wordt bepaald door de loadbalancer (Load balancing, 2015). Als de verbinding is goedgekeurd, dan wordt deze omgezet naar een WSS-verbinding (WebSocket, 2016). Over deze verbinding kunnen vervolgens berichten worden uitgewisseld.

De directory agent vraagt door middel van een pollingmechanisme (Polling (techniek), 2016) of er berichten zijn die verwerkt moeten worden. De berichten geven aan of er een authenticatie-aanvraag heeft plaatsgevonden of dat gegevens moeten worden gesynchroniseerd. Deze berichten worden verwerkt en indien nodig wordt het achterliggende bronsysteem geraadpleegd om de gegevens te controleren of op te halen. Berichten die verwerkt zijn worden teruggestuurd naar HelloID.

6.3.1 WEBSOCKET IN COMBINATIE MET POLLING

Het is redelijk uniek dat een pollingmechanisme wordt gebruikt in combinatie met een websocket; dit was een interessante ontwerpbeslissing van de opdrachtgever en mij. In eerste instantie wilden we enkel een websocket gebruiken om berichten uit te wisselen.

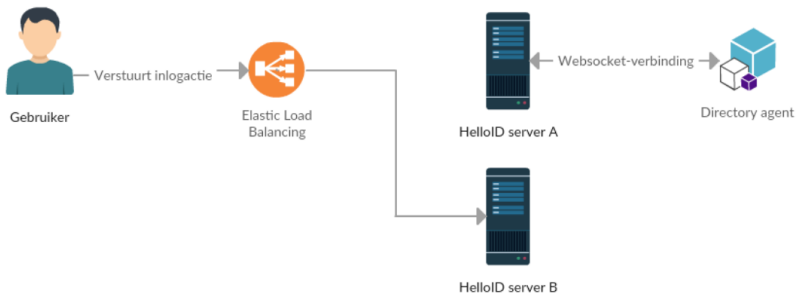
Een websocket-verbinding opzetten gebeurt door vanaf de client een handshake-request te versturen over een http(s)-verbinding. De webserver interpreteert de handshake en zet een bi-directionele TCP-verbinding (TCP) op (Transmission Control Protocol, 2016). De server laat de client weten dat van protocol moet worden gewisseld via de response op het handshake-request. Over de TCP-verbinding kunnen zowel naar de client als naar de server berichten worden gestreamd.

Theoretisch kan een inlogactie via deze verbinding op de volgende manier werken:

FIGUUR 13 AGENT BEREIKBAAR

1. De gebruiker vult zijn gegevens in op het inlogscherm;
2. De server wordt genotificeerd;
3. De server streamt de inlogactie naar de Directory Agent;
4. De agent verifieert de gegevens bij het bronsysteem;
5. De agent streamt de goedkeuring naar de server;
6. De gebruiker is ingelogd.

Bij HelloID had ik er echter mee te maken dat Tools4ever meerdere webservers draait in verschillende regio's (Amerika en Europa). Doordat de websocket-verbinding altijd alleen tussen één server en client een verbinding heeft, geeft dat problemen in de volgende situatie:

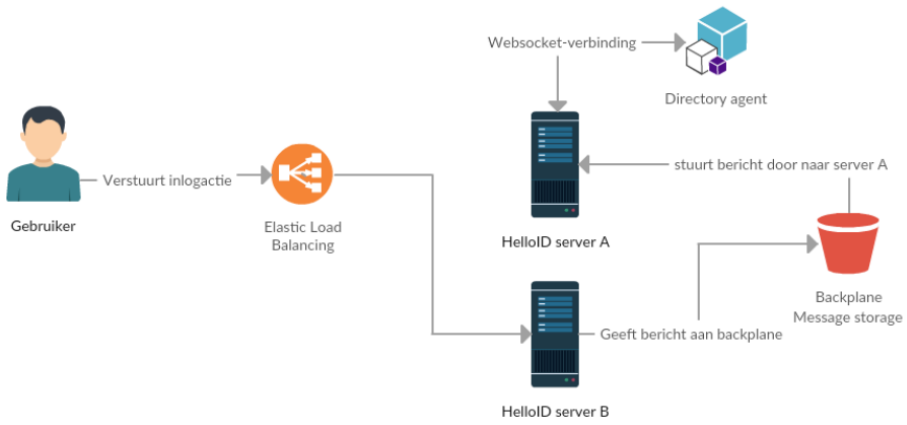


FIGUUR 14 AGENT NIET BEREIKBAAR

1. De agent heeft een verbinding met server A;
2. De gebruiker vult zijn gegevens in op het inlogscherm;
3. De loadbalancer notificeert server B;
4. Het bericht zal niet worden verwerkt en de gebruiker kan niet inloggen met de gegevens uit het bronsysteem;

In een loadbalanced-omgeving kan dit probleem worden opgelost met een backplane, een extra server die berichten van een server doorstuurt naar andere servers.

Als ik dit had toegepast, dan zou dit op de volgende manier werken:



FIGUUR 15 MET BACKPLANTE

1. De agent heeft een verbinding met server A;
2. De gebruiker vult zijn gegevens in op het inlogscherm;
3. Loadbalancer notificeert server B;
4. Server B geeft het bericht door aan de backplane;
5. Server A ontvangt het bericht van de backplane;
6. Server A streamt de inlogactie naar de Directory Agent;
7. De agent verifieert de gegevens bij het bronsysteem;
8. De agent streamt de goedkeuring naar server A;
9. Server A geeft het bericht door aan de backplane;
10. Server B ontvangt het bericht van de backplane;
11. De gebruiker is ingelogd.

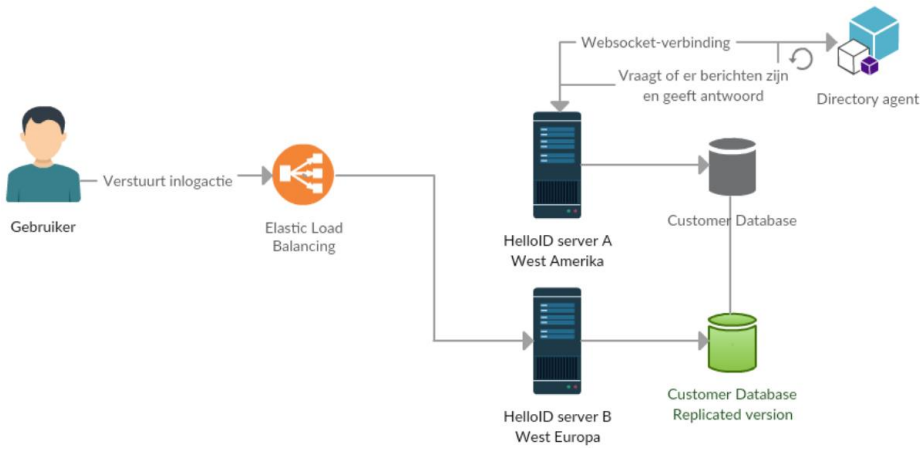
Hoewel dit een goede oplossing is, werkt deze niet voor ons product. Onze servers staan verspreid over Amerika en Europa. Een backplane is locatiegebonden, wat vertragingen kan veroorzaken, waardoor de gebruiker te lang moet wachten op antwoord van de server. Om ervoor te zorgen dat de berichten door alle servers in de loadbalanced-omgeving kunnen worden gebruikt en er zo min mogelijk vertraging is, hebben wij ervoor gekozen om een pollingmechanisme te gebruiken over een websocket-verbinding.

Bij het gebruik van een pollingmechanisme wordt standaard om een bepaald aantal seconden (vaak dertig seconden) in een dergelijke context een http(s)-request verstuurt naar de server. Als deze aanvraag bij de server wordt verwerkt, geeft de server niet direct antwoord. De server wacht tot er een bericht in de database van een klant is gezet. Als een bericht aanwezig is, verstuurt hij het bericht naar de client.

Het gebruik van een dergelijk mechanisme kent veel overhead bij het opzetten van een http(s)-request en er worden middelen (thread) op de server bezet gehouden. Dit kan voor vertragingen zorgen bij andere http(s)-requests die door de server worden behandeld.

Dit was voor ons de aanleiding om de technieken met elkaar te combineren. De agent zet een websocket-verbinding op met de server en streamt over die verbinding steeds de vraag of er berichten aanwezig zijn op de server. Door het gebruik van de websocket-verbinding is er geen overhead en door de korte berichten worden minder middelen op de server bezet gehouden.

De berichten kunnen door elke server zijn aangemaakt in de database. De database wordt gerepliceerd naar locaties die dichterbij de andere servers staan. Indien er berichten zijn, worden deze behandeld door de agent en wordt het antwoord teruggestuurd. Het antwoord wordt in de database gezet en kan door elke server worden opgepakt om afgehandeld te worden. Hoe de verbinding werkt is te zien op de volgende pagina.



FIGUUR 16 SOCKET MET POLLINGMECHANISME

1. De agent heeft een verbinding met server A;
    a. De agent vraagt met een interval of berichten aanwezig zijn;
    b. Als er berichten zijn, zal de agent vaker om berichten vragen.
2. De gebruiker vult zijn gegevens in op het inlogscherm;
3. Loadbalancer notificeert server B;
4. Server B zet het bericht in de database;
5. Server B wacht tot een antwoord aanwezig is in de database;
6. Server A raadpleegt de database;
7. De agent vraagt aan Server A of er berichten zijn;
8. Server A antwoordt en streamt de vraag naar de agent;
9. De agent verifieert de gegevens bij het bronsysteem;
10. De agent streamt de goedkeuring naar server A;
11. Server A zet het bericht in de database;
12. Server B leest het bericht uit de database;
13. De gebruiker is ingelogd.

Het is interessant om te melden dat wij eerder een connector hebben geschreven die enkel het pollingmechanisme gebruikt. In een van onze testomgevingen heb ik handmatig zowel het mechanisme van de connector getest als dat van de HelloID Directory Agent. Bij de connector varieerde de duur van ongeveer 1 tot 4 seconden, bij de nieuwe agent ligt de duur tussen de 500 milliseconden en 2 seconden.

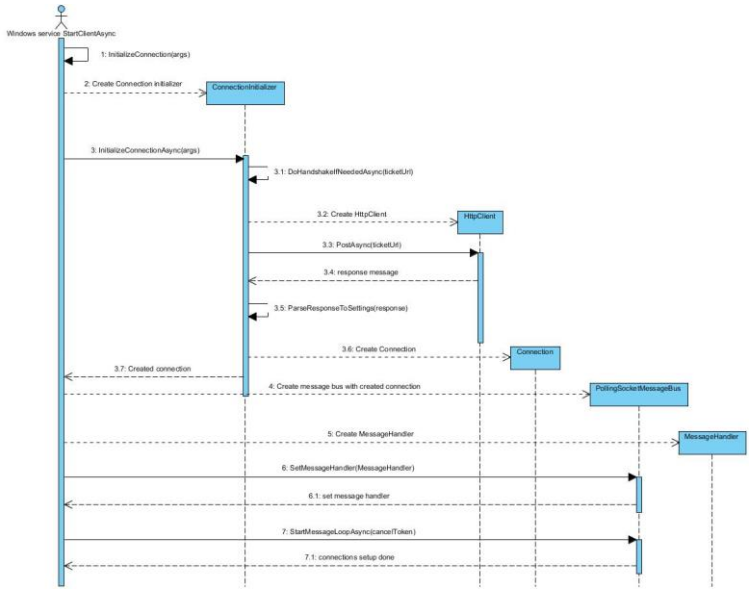
Daarnaast heb ik een stresstest gedraaid waarbij gelijktijdig duizend inlogacties werden verstuurd. Al deze aanvragen werden binnen een seconde verwerkt en teruggestuurd naar HelloID. Ik ben trots op dit resultaat en de structuur van de verbinding die ik heb opgezet.

6.3.2 EEN VEILIGE VERBINDING OPZETTEN EN UITWISSELING VAN BERICHTEN

Om ervoor te zorgen dat een verbinding veilig kan worden opgezet, moet eerst vanuit de agent een aanvraag worden gedaan, zodat geverifieerd kan worden dat de versie gedownload is uit ons portaal. Zo kan een kwaadwillende niet zomaar een agent maken en deze koppelen aan portalen van onze klanten.

Hiervoor heb ik een mechanisme ontworpen waarbij de klant de agent moet installeren met een ticket URL. Bij de installatie van de agent wordt de ticket URL opgegeven. Wanneer de agent voor het eerst een verbinding opzet, zal de agent het publieke deel van zijn certificaat delen met HelloID. HelloID deelt een publiek deel van een ander certificaat dat alleen bij die klant beschikbaar is.

In een sequentiediagram heb ik uitgewerkt hoe een verbinding wordt opgezet vanuit de agent. De klassen van de agent zijn opgenomen in het klassendiagram Directory Agent in de bijlage Ontwerpdocumentatie.

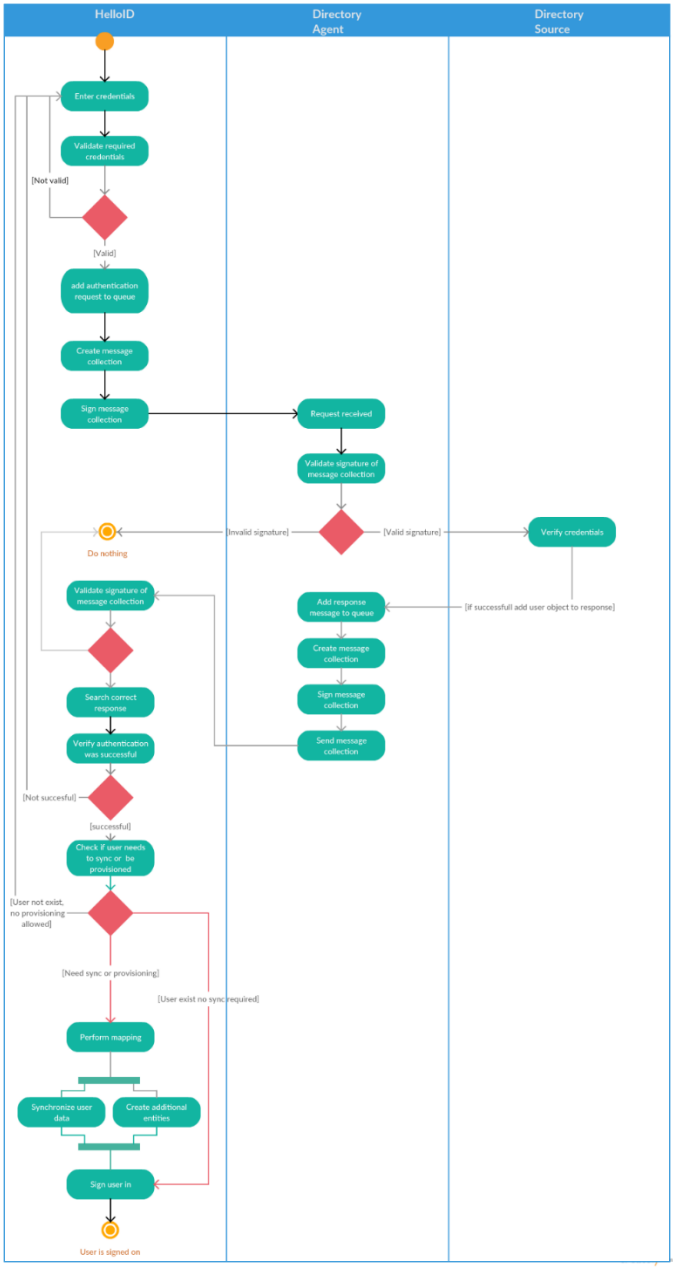


FIGUUR 17 SEQUENCE DIAGRAM VERBRINDING OPZETTEN

Nadat de verbinding opgezet is, kunnen de systemen berichten met elkaar uitwisselen. Hierbij zal ieder systeem de berichten bij het versturen ondertekenen met het publieke deel van het andere systeem. Vervolgens kan het systeem met een privédeel van zijn certificaat bevestigen dat het bericht van een systeem komt dat hij vertrouwt. Als HelloID dus een bericht stuurt naar de agent, wordt dit ondertekend met het publieke deel van het certificaat van de agent. De agent kan met zijn privédeel van het certificaat bevestigen dat het bericht van een systeem komt dat hij vertrouwt.

Om een beter beeld te geven van hoe de uitwisseling van berichten werkt, heb ik de flow van een authenticatieaanvraag verwerkt in een activity-diagram. Het diagram in Figuur 18 toont ook aan

wanneer de mapping en synchronisatie moet plaatsvinden als een gebruiker een authenticatieaanvraag doet.



FIGUUR 18 ACTIVITY DIAGRAM

6.4 TESTBAAR MAKEN VAN CODE

Met behulp van de methodiek Test-Driven Development heb ik terwijl ik mijn code schreef ook bijbehorende moduletests ontwikkeld. Deze methode had ik nooit eerder gehanteerd en ik vond het moeilijker werken dan ik had verwacht. Ik betrapte mijzelf er af en toe op dat ik al een stuk code had geschreven, maar nog geen bijbehorende tests, waarna ik deze alsnog schreef. In de volgende sprint heb ik hier meer op gelet.

Het vereist een andere manier van werken en denken om code goed testbaar te maken. Ik heb een Windows Service geschreven die veel gebruik moet maken van low-level API's om bijvoorbeeld Threads op de Thread pool in te plannen, of de Win32 DLL aanspreekt om een domain controller op te zoeken.

Om de code goed te kunnen testen, zonder bijvoorbeeld onnodig Threads te starten, is het belangrijk om de code zo te schrijven dat deze afhankelijk is van abstracties, niet van concrete implementaties. Dit gebeurt veel door Dependency Injection toe te passen. Ik heb nu adaptief geprogrammeerd, waardoor de code gemakkelijk is aan te passen. Dit heb ik bijvoorbeeld gedaan om de klasse MessageHandler te testen. Hiervoor heb ik een abstractie gemaakt voor de ThreadPool-klasse. Zo kon ik in mijn tests een mockobject meegeven (Mockobject, 2013) dat niet de ThreadPool aanspreekt. Op de volgende pagina is een code-voorbeeld weergegeven.

```
//Standaard API call
return ThreadPool.QueueUserWorkItem(callback, state);

//ThreadPool API in een abstractie, afgeleid van IWinThreadPoolWrapper interface
public class WinThreadPoolWrapper : IWinThreadPoolWrapper
{
       public bool QueueUserWorkItem(WaitCallback callback, object state = null)
       {
            return ThreadPool.QueueUserWorkItem(callback, state);
        }
}

//Een mockedobject van de interface IWinThreadPoolWrapper
public class FakeWinThreadPoolWrapper : IWinThreadPoolWrapper
{
        public bool QueueUserWorkItem(WaitCallback callback, object state = null)
        {
             //Actie wordt niet ingepland op de thread pool, maar direct uitgevoerd
            callback?.Invoke(state);
            return true;
        }
}

//Test waarbij de mockedobject wordt gebruikt
[TestMethod]
public async Task HandleSingleMessageAsync_IsConnectedResponseIsSentBack()
{
       FakeMessageBus messagebus = new FakeMessageBus();
       IMessageHandler messageHandler = new MessageHandler(messagebus,
                                        new FakeSourceInstanceResolver(),
                                        new FakeWinThreadPoolWrapper());
       int nrofMessagesSent = 1;

       await messageHandler.HandleSingleMessageAsync(Message.Create("is connected"));

       Assert.AreEqual(nrofMessagesSent, messagebus?.SendMessages?.Count);
}
```

7 SPRINT 3: INLOGGEN VIA INLOGGEGEVENS UIT ACTIVE DIRECTORY

Omdat het raamwerk (de basis) van de agent was opgezet, lag de focus in de derde sprint op succesvol inloggen op HelloID met gegevens uit een externe bron. In deze sprint de eerste bron, Active Directory (AD).

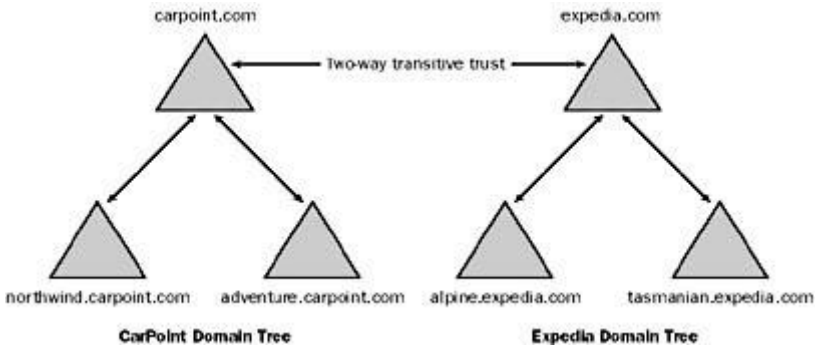
Om AD te ondersteunen, heb ik volgens mijn modulaire architectuur (zie paragraaf 6.2) een aparte DLL geschreven die runtime kan worden ingeladen. Deze DLL bevat de implementatie om de AD te raadplegen.

7.1 DOMEIN BEPALEN DOOR MIDDEL VAN INLOGGEGEVENS

Ik heb ook een klasse gemaakt die de interface IUserManager overerft. Deze interface beschrijft onder andere hoe het publieke deel van de LogonUser eruit moet komen te zien. Hierbij moet als parameter de AuthenticationCredentials-klasse worden meegeven en de output is een generieke Result-klasse met een User-object.

De AuthenticationCredentials-klasse kan een lijst van variabele namen en objecten bevatten. In het geval van Active Directory bevat een instantie van deze klasse een gebruikersnaam, een wachtwoord en eventueel een standaard domeinnaam.

In bijna elk bedrijfsnetwerk is een zogeheten Active Directory forest beschikbaar (Network, Active Directory forest (AD forest), 2012). Dit is een overkoepeling voor meerdere domeinen binnen een bedrijfsnetwerk. Hierdoor kunnen gebruikersnamen hetzelfde zijn, maar binnen verschillende domeinen zijn opgeslagen. Bij elk domein hoort weer een eigen domain controller.



FIGUUR 19 DOMAIN FOREST

AD ondersteunt dat gebruikers op verschillende domeinen inloggen, en doet dit op verschillende manieren, bijvoorbeeld door het standaarddomein mee te sturen bij het inloggen of de domeinnaam af te leiden uit de gebruikersnaam. Dit laatste kan ook weer op een aantal manieren gebeuren, bijvoorbeeld via de SAM-Account-Name (Microsoft, SAM-Account-Name attribute, n.d.), User Principal Name (UPN) (Microsoft, User-Principal-Name attribute, n.d.) of een combinatie hiervan met een alias voor de domeinnaam.

1. **Gebruikersnaam o.b.v. SAM-Account-Name:** `Rbd-develop\test.user.0001`
2. **Gebruikersnaam o.b.v. UPN:** `test.user.0001@Rbd-develop.local`
3. **Gebruikersnaam o.b.v. SAM-Account-Name (t4e alias):** `t4e\test.user.0001`

   Deze drie voorbeelden leiden naar het domein Rbd-develop.local

Dit heeft als gevolg dat in de agent de input van de gebruikersnamen gevalideerd en geverifieerd moet worden om de juiste domain controller te achterhalen. Dit wordt onder andere gedaan door eerst de domeinnaam uit de gebruikersnaam te halen, bijvoorbeeld met onderstaande code waarin de domainnaam wordt afgeleid uit een UPN.

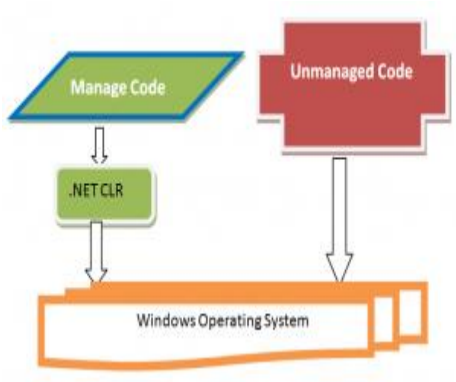
```
//controleer voor UPN
int index;
if ((index = userName.IndexOf("@", StringComparison.Ordinal)) > 0)
{
        _domainToUse = userName.Substring(index + 1, userName.Length - index - 1);
        _recommendedSearchType = AdUserSearchType.ByUpn; //user principal
}
```

FIGUUR 20 VOORBEELDCODE VOOR UPN

7.1.1 GEBRUIK VAN WIN32 API

Om vervolgens de domain controller te raadplegen wordt via de Win32 API het netwerk aangesproken om op basis van de domeinnaam informatie op te vragen van de controller. In deze informatie is de officiële naam van de controller te vinden.

Er was geen standaard abstractie beschikbaar voor de actie die ik wilde aanroepen bij de Win32 API. Hiervoor moest ik 'unmanaged code' aanspreken.



Managed code is bijvoorbeeld wat de c# compiler compileert. Deze code wordt uitgevoerd via de Common Language Runtime (CLR). De CLR biedt diensten aan als het opruimen van geheugen waar geen verwijzingen meer naartoe gaan, de garbage collection, (Microsoft, Garbage Collection, 2016). De code wordt 'gemanaged' door de CLR (Gregory, 2003).

Unmanaged code is code die direct wordt vertaald naar machine code. Deze code wordt bijvoorbeeld gecompileerd door traditionele C/C++ compilers (Gregory, 2003).

De unmanaged code die ik moest aanspreken bevindt zich in de netapi32.dll. Hiervoor heb ik gebruikgemaakt van DLLImport-attribuut (Microsoft, DllImportAttribute Class, n.d.). Dit maakt een statisch toegangspunt voor unmanaged code om de code te benaderen in managed code. Zie een voorbeeld hiervan hieronder.

```
[DllImport("Netapi32.dll")]
private static extern int NetApiBufferFree([In] IntPtr buffer);
```

FIGUUR 21 DLLIMPORT

7.2 Gegevens uit Active Directory halen

Op het moment dat de gebruikersnaam correct is en bijbehorende domain controller bekend is, kan met deze informatie de Active Directory worden aangesproken om bijvoorbeeld gegevens op te halen. Na kort onderzoek stuitte ik op de mogelijkheden die het .Net Framework biedt om AD te raadplegen. Het framework biedt de namespace System.DirectoryServices aan, waarmee het LDAP-protocol kan worden gebruikt om de AD te raadplegen.

> **Lightweight Directory Access Protocol** is een netwerkprotocol dat beschrijft hoe directory services zoals AD moeten worden benaderd over TCP/IP (Lightweight Directory Access Protocol, 2013).

Gebruik van dit protocol heeft als voordeel dat andere bronnen die LDAP gebruiken ook kunnen worden ondersteund of dat hier al de nodige code voor is geschreven die ik kan hergebruiken. Bovendien heeft Tools4ever intern veel ervaring met dit protocol.

In Figuur 22 is weergegeven hoe ik een klasse uit de System.DirectoryServices namespace heb gebruikt om gebruikersgegevens op te halen.

```
using (DirectorySearcher adsSearcher = new DirectorySearcher(entry))
{
        adsSearcher.Filter = searchSet.GetUserSearchFilter();
        SearchResult adsSearchResult = adsSearcher.FindOne();
        user = userFactory.CreateFromSearchResult(adsSearchResult);
}
```

FIGUUR 22 DIRECTORY SEARCHER KLASSE

Aan het bovenstaande voorbeeld wordt een DirectoryEntry meegegeven. Deze entry bevat een LDAP-string waarmee ik aangeef welke directory (domain controller) ik wil aanspreken. De string ziet er ongeveer zo uit: `LDAP://dc=Rbd-develop,dc=local`.

Vervolgens geef ik aan de searcher ook een filter op. In het geval van AD wil ik filteren op gebruikers (zodat niet op groepen wordt gezocht) en op UPN of SAM Account Name. De search string ziet er bijvoorbeeld zo uit: `(&(sAMAccountType=805306368)(samAccountName=R.Brussaard)).`

7.3 INTEGRATIETESTS

Tijdens de ontwikkeling van de agent heb ik gebruikgemaakt van TDD. Wat ik naast het moduletesten belangrijk vond, is testen of de integratie tussen de code en het bronsysteem goed werkte. Mijn collega Aad Lutgert heeft een testomgeving opgezet met daarin een domain controller en Active Directory. Voor deze testomgeving heb ik Powershell-scripts[3] gemaakt die bijvoorbeeld twintigduizend gebruikers kunnen aanmaken. (Zie onderaan de pagina de Powershell create user script.)

Met deze omgeving heb ik integratietests opgezet, om zo de manier waarop ik in mijn code de AD raadpleeg te waarborgen. Hiernaast kan ik deze omgeving gebruiken voor testdoeleinden. De gehele omgeving is virtueel en kan op elk moment worden teruggezet naar een status met gebruikers of een status zonder gebruikers.

```
Import-Module ActiveDirectory

 $userIndex = 0

 1..20000 | Foreach-Object {

 $userID = "{0:0000}" -f ($userIndex++)
 $userName = "usr$userID"

 New-ADUser -Name "$userName" `
   -AccountPassword (ConvertTo-SecureString "alpha123qwe,./" -AsPlainText -Force) `
   -City "City" `
   -Company "Company" `
   -Country "US" `
   -Department "Department" `
   -Description ("TEST ACCOUNT " + $userID + ": This user account does not repre-
sent a real user and is meant for test purposes only") `
   -DisplayName "Test User ($userID)" `
   -Division "Division" `
   -EmailAddress "$userName@Rbd-develop.local" `
   -EmployeeNumber "$userID" `
   -EmployeeID "ISED$userID" `
   -Enabled $true `
   -Fax "703-555-$userID" `
   -GivenName "Test" `
   -HomePhone "703-556-$userID" `
   -Initials "TU$userID" `
   -MobilePhone "703-557-$userID" `
   -Office "Office: $userID" `
   -OfficePhone "703-558-$userID" `
   -Organization "Organization" `
   -POBox "PO Box $userID" `
   -PostalCode $userID `
   -SamAccountName $userName `
   -State "Virginia" `
   -StreetAddress "$userID Any Street" `
   -Surname "User ($userID)" `
   -Title "Title" `
   -UserPrincipalName "$userName@Rbd-develop.local"
}
```

FIGUUR 23 POWERSHELL CREATE USER SCRIPT

[3] Powershell is een taakgebaseerde scripttaal van Microsoft die speciaal gebouwd is om administratieve operaties te automatiseren (Microsoft, Scripting with Windows PowerShell, 2014).

8 SPRINT 4: SYNCHRONISATIEMECHANISME

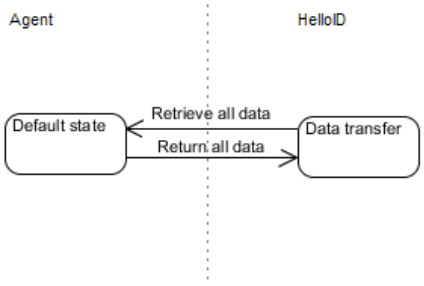
De agent zorgt er niet alleen voor dat gebruikers kunnen authentiseren tegen een identiteitsbron, maar moet ook zorgen voor gebruikers- en groepsgegevenssynchronisatie tussen bronsysteem en HelloID. Bij het synchroniseren van deze gegevens is het van groot belang[4] dat de klant kan bepalen hoe en naar welke velden de gegevens worden gesynchroniseerd in HelloID. Het doel van deze sprint was dan ook een synchronisatie- en mappingmechanisme te ontwikkelen.

Mijn begeleider heeft al vaker synchronisatiemechanismes gebouwd. Voordat ik aan de ontwikkeling van het mechanisme begon, hebben mijn begeleider en ik daarom overlegd over de manier waarop gegevens moet worden verplaatst.

Eerst stelden we vast dat het bronsysteem het leidende systeem moet zijn in de synchronisatie met HelloID. Klanten willen immers de gegevens uit het bronsysteemsysteem overzetten naar HelloID en niet andersom. Het is belangrijk dat exact dezelfde gegevens beschikbaar zijn in HelloID. HelloID zal in deze versie nog niet het bronsysteem bijwerken.

Om de gegevens te verplaatsen heb ik twee strategieën voorgesteld: een ‘Full Transfer’ of een ‘Timestamp Transfer’.

Full Transfer
Bij een Full Transfer wordt, zoals de naam al suggereert, de volledige verzameling van de gegevens (dataset) uit het bronsysteem verstuurd naar HelloID. HelloID vervangt de bestaande dataset (als deze bestaat), met de dataset die is ontvangen, of voegt de nieuwe dataset aan HelloID toe.



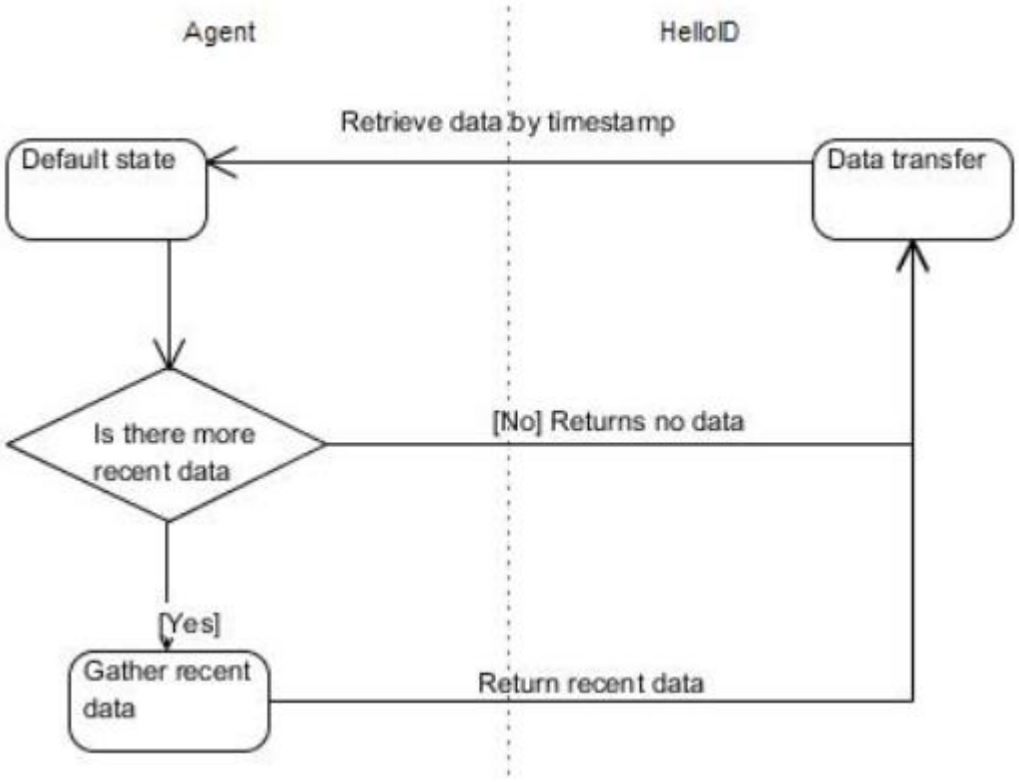
FIGUUR 24 FULL DATA TRANSFER

Het voordeel van deze strategie is dat deze simpel te implementeren is. Het nadeel is dat deze strategie veel tijd en middelen kost. Bij 20.000 gebruikers moet de directory agent elke keer

[4] Dit belang kwam naar voren uit de requirements-analyse (zie paragraaf 4.2.1).

20.000 gebruikers ophalen (wat een dure operatie kan zijn) en versturen over de verbinding met HelloID. HelloID moet deze vervolgens allemaal verwerken, wat veel geheugen kan kosten.

Timestamp Transfer
Bij een Timestamp Transfer moet HelloID de directory agent raadplegen met een tijdstip dat is verkregen uit de vorige synchronisatie. Dit tijdstip is gelijk aan het tijdstip van de ontvangst van het laatst aangepaste object door HelloID. De agent geeft alleen een dataset terug met daarin gegevens die later zijn aangepast dan het meegegeven tijdstip.



FIGUUR 25 TIMESTAMP TRANSFER

Een voordeel van deze strategie is dat deze minder tijd en middelen kost dan een Full Transfer. Slechts initieel, wanneer geen tijdstip bekend is, moeten alle gegevens in het systeem worden opgehaald. Een nadeel van deze strategie is dat niet duidelijk is of bepaalde gegevens verwijderd zijn uit het bronsysteem.

Wij waren het met elkaar eens om niet de Full transfer-strategie te gebruiken, omdat dit zonde is van de middelen. Hierbij moet ook gedacht worden aan de kosten van Azure: bij veel transacties op de webserver en database worden meer kosten gerekend voor het gebruik.

Mijn begeleider heeft vaker met de strategie Timestamp Transfer gewerkt en stelde dat deze nog een nadeel heeft: de onderliggende bronsystemen kunnen problemen hebben met het consistent houden van het tijdstip waarop objecten zijn aangepast. Het tijdstip wordt vaak bepaald door de instelling van de server waar het bronsysteem is geïnstalleerd.

Hij stelde voor dat ik, in plaats van met een tijdstip (timestamp), zou werken met een Update Sequence Number (USN). In HelloID wordt het hoogste USN bijgehouden en gebruikt om de meest recente wijzigingen op te halen.

In eerste instantie raadpleegden wij Active Directory, dat net als veel andere bronsystemen voor elk object een USN bijhoudt. Bij systemen die geen USN ondersteunen kunnen wij de USN en identifiers bijhouden in de agent.[5]

Bovenstaande lost het nadeel dat onduidelijk is of gegevens uit het bronsysteem zijn verwijderd niet op. In deze versie van het synchronisatiemechanisme hebben wij dit opgelost door in een apart proces periodiek (bijvoorbeeld elke dag) een lijst te sturen van alle object-identifiers [6] die HelloID in het systeem heeft staan. Object-identifiers die niet meer bestaan in het bronsysteem worden teruggestuurd naar HelloID, die deze vervolgens verwijdert.

Op basis van het hierboven beschreven overleg heb ik in deze sprint het synchronisatiemechanisme geschreven.

[5] Deze functionaliteit wordt gebouwd in een volgende versie van de HelloID Directory Agent.

[6] Dit is vergelijkbaar met een Burgerservicenummer dat een uniek persoon identificeert, alleen identificeert een object-identifier hier een uniek object dat bestaat in de database van HelloID.

9 SPRINT 5: DEMO-OMGEVING

Bij de vorige sprints lag de focus op het ontwerp en de ontwikkel van de agent. In deze sprint lag focus op het gereedmaken van een stabiele versie en deze werkend opleveren in een demo-omgeving. De eerste week van deze sprint heb ik besteed aan het finetunen van de agent.

9.1 AANPASSING VAN SYNCHRONISATIEMECHANISME

De belangrijkste aanpassing die ik in deze sprint heb gedaan is de manier waarop de agent gebruikersgegevens uitwisselt met HelloID. Voorheen werden de gegevens van alle gebruikers verzameld en als één berichtobject naar de server gestuurd. Door middel van integratietests, waarbij ik meerdere grote berichten tegelijk naar HelloID stuurde, heb ik onderstaande problemen ontdekt.

9.1.1 DE PROBLEMEN

Een groot bericht (met meer dan 1000 gebruikersgegevens in een bericht) zorgde ervoor dat de server te lang (langer dan 4 seconden) een thread bezet hield op de server. Dit is een probleem als meerdere threads tegelijkertijd worden vastgehouden; de server kan dan niet meer tijdig reageren wanneer een gebruiker HelloID benadert.

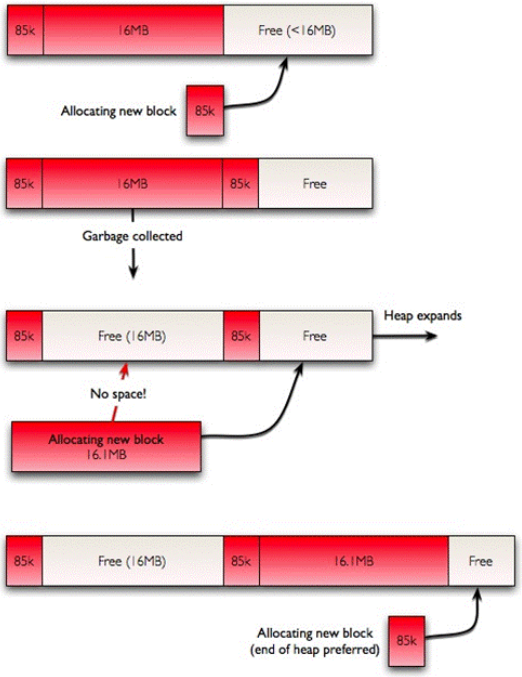
Daarnaast heeft de server bij een groot bericht niet genoeg geheugen om meerdere acties tegelijk uit te voeren, omdat objecten van andere aanvragen niet meer in het geheugen kunnen worden gealloceerd. Dit probleem is tweeledig. Ten eerste duurt het verwerken van de gegevens langer, waardoor er langer een referentie is naar de gegevens. De objecten die de gegevens bevatten kunnen zo niet tijdig worden opgeruimd door de garbage collector. Als gevolg hiervan komt er voor een langere tijd geen geheugen vrij.

Ten tweede worden objecten waarvan het geheugen groter is dan 85.000 bytes (Stephens, 2016) gealloceerd op de Large Object Heap (LOH). Het alloceren van objecten op de LOH heeft de volgende twee gevolgen:

1. Bij het alloceren wordt er rekening mee gehouden dat het object groter kan zijn dan 16MB en moet worden opgedeeld in verschillende segmenten (Netkachov, 2015). Dit gebeurt in .Net door het op te slaan in segmenten van 16MB. Wanneer het object kleiner is dan 16MB, wordt maar één segment van 16MB gereserveerd[7] voor het object. Wanneer het object groter is dan 16MB, wordt nog een blok van 16MB gereserveerd (zie Figuur 26 Large object heap reseveren geheugen).

[7] Geheugen wordt gereserveerd op het RAM-geheugen.

2. De garbage collector gaat anders om met het opruimen van objecten die in de LOH staan. In plaats van de objecten te verwijderen en segmenten vrij te geven, worden de segmenten gereserveerd als 'vrij' en mogen deze opnieuw gebruikt worden om grote objecten te alloceren. Hierdoor zou een situatie kunnen ontstaan waarin er niet genoeg geheugen is om nieuwe objecten te alloceren.



FIGUUR 26 LARGE OBJECT HEAP RESEVEREN GEHEUGEN

Garbage collector

Binnen .NET framework wordt het alloceren en vrijgeven van geheugen beheerd door de garbage collector. De collector bepaalt automatisch welke data-objecten in een programma niet meer benaderd kunnen worden en geeft het geheugen vrij dat door deze objecten gebruikt wordt (Microsoft, Garbage Collection, 2016).

Large Object Heap

De garbage collector van .Net deelt het geheugen bij het beheren ervan in kleine en grote objecten in. Grote objecten worden in de Large Object Heap gezet, een apart deel van het geheugen waar dynamisch geheugen kan worden gealloceerd.

Deze problemen komen in principe niet voor wanneer slechts één klant verbinding maakt met een server; het geheugen kan dan snel genoeg vrijgegeven worden. Maar HelloID heeft een loadbalanced-omgeving waarbij meerdere klanten (lees agents) tegelijk verbinding maken met dezelfde server (deze verbinding is besproken in paragraaf 6.3). Hierdoor zullen meerdere synchronisaties tegelijk plaatsvinden en kan het geheugen van een server vollopen.

9.1.2 OPLOSSING

De oplossing voor dit probleem is de LOH te vermijden. Dit heb ik gedaan door in de agent de gegevens op te delen in kleinere berichtobjecten (die 85kb niet overschrijden) en deze direct te versturen naar de server. Dit had wel als gevolg dat de agent een groot aantal referenties naar gebruikerobjecten moest bijhouden, waardoor dit proces voor een langere tijd veel geheugen in beslag kon nemen.

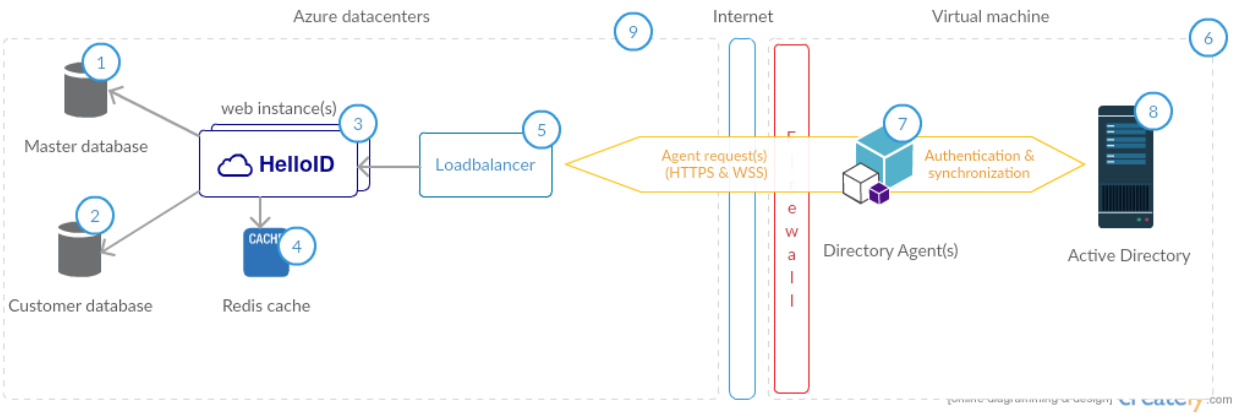
Om ervoor te zorgen dat het geheugen in de agent werd vrijgegeven, heb ik gebruikgemaakt van een stack, waarbij de referentie verwijderd wordt zodra het gebruikerobject gelezen is (Microsoft, Stack.Pop Method (), n.d.). Terwijl de berichten verstuurd werden, werd zo ook het geheugen vrijgegeven.

```
private static void ProcessUsers(Result<Stack<User>> result)
{
	if (result == null || result.ResultObject == null) return;

	while (result.ResultObject.Count > 0)
	{
		List<User> userToSend = new List<User>();

		for (int i = 0; i < 50; i++)
		{
			if (result.ResultObject.Count == 0) break;
				userToSend.Add(result.ResultObject.Pop());
		}

		client.AddMessageToQueue(MessageType.RetrieveUsersSuccess, userToSend);
	}
 }
```

FIGUUR 27 PROCESS USERS VOORBEELD

9.2 DEMO-OMGEVING

De focus lag naast op het finetunen van de agent, ook op het inrichten van een werkende demo-omgeving. Het is van belang dat een demo-omgeving zo gelijk mogelijk is aan de productieomgeving. In Figuur 28 Ingerichte demo-omgeving is te zien hoe de omgeving in elkaar zit.



FIGUUR 28 INGERICHTE DEMO-OMGEVING

Elementnummer	Naam	Omschrijving
1	Master database	Deze database wordt beheerd door Tools4ever en wordt op een databaseserver in Azure gehost. In deze database worden klantgegevens bewaard, waaronder de gegevens van de customer database.
2	Customer database	De customer database bewaart informatie van de klant, waaronder diens gebruikers en de applicaties die zij gebruiken. Ook deze database wordt op een databaseserver in Azure gehost.
3	Web instance(s)	De web instance is een webserver waarop de applicatie HelloID draait. Hiervan kunnen meerdere instanties actief zijn. Deze webserver(s) worden gehost in Azure.
4	Redis cache	De Redis cache bewaart sessies die door gebruikers worden gestart via HelloID. Zo kunnen gebruikers die van instantie wisselen

Elementnummer	Naam	Omschrijving
		gebruikmaken van dezelfde sessie. De Redis cache wordt in Azure gehost.
5	Loadbalancer	De loadbalancer zorgt ervoor dat bij veel verkeer een nieuw web instance wordt gestart en zal bij weinig verkeer web instances op inactief zetten.
6	Virtual machine	Deze virtuele machine (VM) representeert het netwerk van een klant. De VM draait in het netwerk van Tools4ever en verbindt via dit netwerk ook met de servers die door Microsoft Azure worden gehost.
7	HelloID Directory Agent	Dit is het component dat ik heb ontwikkeld. Dit component zorgt ervoor dat de gegevens uit Active Directory kunnen synchroniseren met HelloID.
8	Active Directory	Dit is de identeiteitsbron die ik in de demo-omgeving heb ingericht om mee te testen.
9	Azure datacenters	Azure datacenters hosten de servers die nodig zijn om een HelloID-omgeving te draaien. Azure wordt aangeboden door Microsoft. Alle servers die ik heb opgetuigd draaien in West-Europa.

10 EVALUATIE

10.1 PROCESEVALUATIE

Het project ben ik gestart met een aanloopfase. In deze eerste fase heb ik een plan van aanpak opgesteld, met daarin de interne afspraken, mijn werkwijze en de planning. Als ik hierop terugkijk, kan ik stellen dat ik de planning goed heb gevolgd. Ik ben begonnen met het verzamelen van requirements en ben daarna gelijk gestart met de eerste sprint. Doordat ik de ontwikkelmethode Scrum heb gehanteerd, was het mogelijk om de requirements op basis van nieuwe inzichten tijdens het traject te verrijken, te veranderen of te laten vervallen.

Het heeft mij veel geholpen tijdens dit traject om prioriteiten goed af te bakenen. Zo kwamen de opdrachtgever en ik tijdens het verzamelen van requirements tot de conclusie dat sommige requirements belangrijker waren dan requirements die wij voorafgaand aan het afstudeerproject hadden opgesteld. Zo was het belangrijker om te bepalen hoe en waarnaartoe gegevens worden gesynchroniseerd dan om meerdere identiteitsbronnen te ondersteunen. (Zie voor meer informatie hierover paragraaf Interview met opdrachtgever4.2.1.1.)

Om te bepalen welke requirements belangrijk genoeg waren voor het succes van deze afstudeeropdracht, heb ik (in samenspraak met de opdrachtgever) de MoSCoW-methode toegepast. Requirements die niet de prioriteit ‘must have’ hebben, worden in een volgende sprint (buiten de scope van de afstudeeropdracht) opgepakt. (Zie voor meer informatie hierover paragraaf 4.2.2.)

In projecten naast dit afstudeerproject analyseert Tools4ever geen requirements, maar worden de eisen en wensen door belanghebbenden geformuleerd. Door dit project heb ik veel geleerd over het nut van het analyseren van requirements. Mijn nieuwe inzichten zijn de kwaliteit van het product ten goede gekomen en ik ben van plan om deze ervaring en techniek toe te passen bij toekomstige ontwikkeltrajecten. Wel had ik graag meer tijd gehad om specifiekere requirements te formuleren. Dit had waarschijnlijk tijd gescheeld tijdens het ontwikkelen en het opstellen van tests.

Na de aanloopfase zijn wij gestart met het Scrum-proces en dus ook met de eerste sprint. Tijdens elke sprint heb ik onderdelen van het product ontworpen, ontwikkeld en getest. Het testen stond volledig in het teken van TDD. Mijn terugblik op Scrum is te lezen in paragraaf 10.1.1, de terugblik op de tests met TDD is te lezen in paragraaf 10.1.2.

De eerste sprint stond volledig in het teken van het ontwerp en de implementatie van een nieuw domeinmodel binnen het huidige model van HelloID. Om het nieuwe model te modelleren en te presenteren heb ik de modelleringstaal UML gehanteerd.

Eerst heb ik het nieuwe model gepresenteerd in een EER-Model. Dit model was niet duidelijk genoeg en gaf volgens mijn collega's onvoldoende inzicht. In het vervolg van het traject heb ik dit soort modellen in de vorm van een klassendiagram gepresenteerd.

Het ontwerpen van dit model bracht nieuwe vragen met zich mee, bijvoorbeeld met betrekking tot de vraag hoe identity providers in dit model passen of hoe moet worden gewerkt met authenticatietemplates. Om dit afstudeertraject te beschermen heb ik samen met mijn opdrachtgever besloten dat deze vragen buiten de scope van dit traject vallen en in de toekomst in een ander traject worden opgepakt.

Na de eerste sprint kon het 'echte' ontwikkelen beginnen. Ik startte in de tweede sprint met het ontwikkelen van de Windows Service en de communicatie tussen de agent en HelloID. In de derde sprint lag de focus op inloggen via gegevens uit de AD en in vierde sprint op het synchronisatiemechanisme. Bij de ontwikkeling van deze service heb ik rekening gehouden met een modulaire structuur, het managen van meerdere Threads en, omdat ik met TDD werkte, met het testbaar maken van de code.

Ik werk nu ongeveer twee jaar bij Tools4ever en ongeveer een decennium in dit vakgebied. Mijn opdrachtgever heeft veel vertrouwen in mijn keuzes en beslissingen. Ik was vrij om te bepalen hoe ik het ontwerpen, ontwikkelen en testen aanpakte. Ik heb dit als erg prettig ervaren; het haalde in mij het beste naar boven.

Indien ik voor problemen sta die een grote impact hebben op het product of als ik ergens mee worstel, dan zorg ik ervoor dat ik deze problemen eerst onderzoek (een voorbeeld hiervan is te vinden in paragraaf 0) en daarna bespreek met een expert of met de opdrachtgever. Vervolgens bepaal ik op basis hiervan een oplossing en pas ik deze toe. Het kan ook zijn dat ik de problemen kort toelicht tijdens de stand-up en een teamlid een idee heeft om mij op weg te helpen.

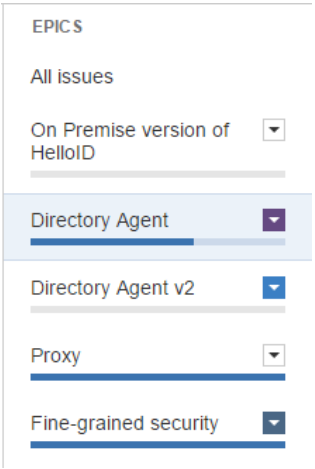
10.1.1 Werken met Scrum

In de uitvoerende fases heb ik de ontwikkelmethode Scrum gehanteerd. Scrum is een flexibele manier om (software)producten te maken, waarbij korte lijnen zijn tussen belanghebbende en teamleden, in korte fasen werkende producten worden opgeleverd en snel op nieuwe kansen en inzichten kan worden ingespeeld.

Tools4ever hanteert de Scrum-methodiek nu ongeveer een jaar. Welke Scrum-gebeurtenissen het bedrijf hanteert en hoe hier invulling aan is gegeven heb ik beschreven in paragraaf 4.1.2.1.

Het resultaat van dit project is uiteindelijk de nieuwe functionaliteit voor het product HelloID. Mede omdat Tools4ever Scrum al hanteert, besloot ik om dit project op te nemen in het bestaande Scrum-proces van HelloID. Daardoor kon op dezelfde manier werk worden ingepland en geëvalueerd. Bovendien had ik zo goed zicht op doorlopende ontwikkelingen van het HelloID-product.

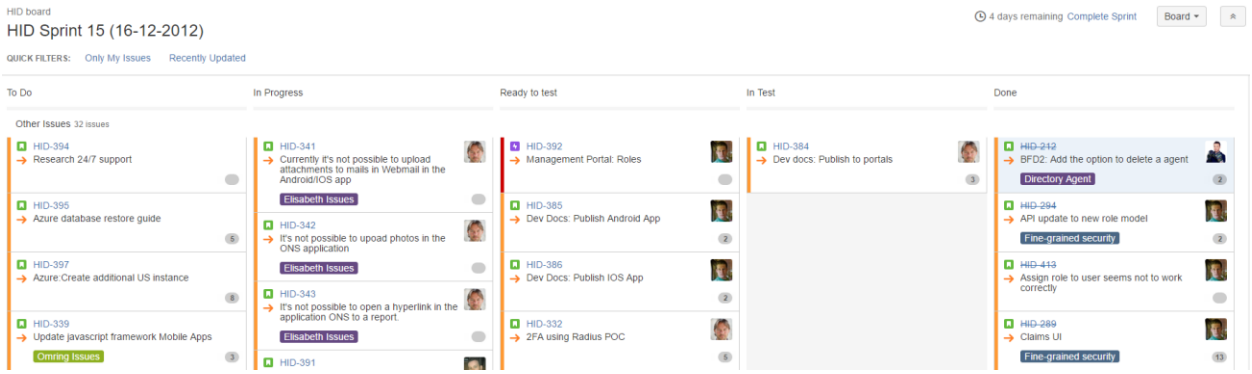
> Uniek aan dit project was dat de functionaliteit, die meerdere sprints zou kosten om op te nemen in het product, veel groter was dat wij normaal in ons proces opnemen. Dit was voor Tools4ever en mij een nieuwe ervaring.
>
> Wij hebben dit opgelost door een ‘Epic’ op te nemen in onze Scrum-tool Jira. Deze ‘Epic’ is een manier om user stories (requirements) te categoriseren en in ons geval om aan te geven dat de stories die zich hierin bevinden onderdeel zijn van een grotere functionaliteit.



FIGUUR 29 EPICS IN JIRA

Nadat wij meer requirements hadden verzameld en (met behulp van de MoSCoW-methode) prioriteiten hadden bepaald, hebben we deze ‘Epic’ gesplitst in ‘HelloID Directory Agent’ en ‘HelloID Directory Agent v2’; de stories onder ‘v2’ kunnen na het afstuderen worden opgepakt.

Voorafgaand aan elke sprint hebben wij een sprintplanning gehouden om te bepalen wat het sprintdoel was en welke requirements (stories) zouden worden opgepakt tijdens de sprint. Tijdens de sprints heb ik aan de software gewerkt, die werd getest met behulp van moduletests (zie paragraaf 10.1.2). Aan het einde van elke sprint hebben wij een ‘retrospective’ gehouden, waarin we een evaluatie van het gemaakte werk en feedback op ons proces combineerden.



FIGUUR 30 VOORBEELD SPRINTBOARD

Ik vind dat wij nog kunnen werken aan de manier waarop feedback wordt gegeven op het gemaakte werk. Nu wordt de feedback vaak gegeven tijdens het ontwikkelen, met als gevolg dat de kwaliteit van het visuele deel van ons product minder is. Dat vind ik jammer.

Ik heb al voorgesteld dat wij ook gebruikmaken van de Scrum-review, waarbij personen een korte presentatie krijgen van het werk, waarna feedback kan worden gegeven. Ik heb voorgesteld dat wij een kleine gebruikersgroep uitnodigen die uiteindelijk met ons product aan de slag gaat. Dit is goed ontvangen door mijn opdrachtgever en door onze salesmanager, Tjeerd Seinen.

Persoonlijk vind ik het prettig om met Scrum te werken, vooral omdat wij elke dag een korte vergadering houden (stand-up), waarin we bekijken wat iedereen heeft gedaan, gaat doen en waar iemand tegenaan loopt. Dit werkt voor mij motiverend. Als ik ergens tegenaan loop, is de kans groot dat een ander teamlid een oplossing heeft en mij op weg kan helpen.

Ons werk wordt visueel gemaakt met behulp van een groot touchscreen (voorheen met post-its) met daarop alle user stories waaraan gewerkt wordt. Hierdoor is in één opslag duidelijk wat de status is van de sprint.

Tijdens dit traject heb ik weer de kracht van Scrum kunnen ervaren. Wij hebben alles uit de aanloopfase kunnen inplannen via de sprintplanningen en grote items kunnen waarborgen met behulp van 'Epics'. De betrokkenheid van elk teamlid was te voelen bij elke stand-up, waarbij we elkaar op weg hielpen bij problemen. Ook hebben we kunnen reflecteren op ons proces tijdens de retrospective. Wel vind ik dat de evaluatie van ons product beter kan.

10.1.2 Werken met Test-Driven Development

In dit traject heb ik mij ook gericht op de methodiek Test-Driven Development (TDD), waarmee gelijktijdig de code en de bijhorende moduletests worden geschreven.

Een van de redenen dat ik in dit traject ben gaan werken met TDD is dat ik hier ervaring mee wilde opdoen, zodat ik de nieuw opgedane kennis kon delen met Tools4ever. Ik zie nu goed in wat de voordelen, nadelen en valkuilen zijn van deze methodiek.

Persoonlijk vind ik het grootste voordeel dat bij het opstellen van de programmatuur regressietests worden opgesteld. Op elk moment, zowel bij het ontwikkelen als bij het opleveren van de software, kunnen de tests worden uitgevoerd. Zo kan worden geverifieerd of de functionaliteit nog werkt zoals deze zou moeten werken. Dit helpt bij het verzekeren van codekwaliteit en het elimineren of verminderen van bugs.

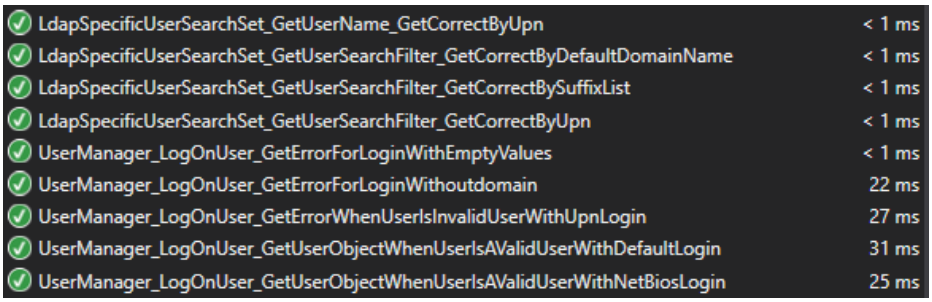
Een ander voordeel vind ik dat TDD mijn moraal als programmeur bij het schrijven van software ondersteunt. Het werkt motiverend om te kunnen zien of de code naar behoren blijft werken en dat fouten direct terug te zien zijn in de testresultaten.

Nog een voordeel vind ik dat TDD mij stimuleert om anders na te denken over mijn code. Voorheen zou ik low-level API's direct in de code aanroepen, maar om de code testbaar te maken moet dit abstracter worden gezien en pas ik encapsulation toe (Dictionaries, Encapsulation (computing), n.d.). Zo gebeurt dit binnen een andere klasse, die ik tijdens het testen kan vervangen door een gesimuleerd object. Dit heeft als gevolg dat er een losse koppeling ontstaat en dat de uitbreidbaarheid van de code verbetert (dit is aan de orde gekomen in paragraaf 6.4). Ik vind dat ik mijzelf heb verbeterd op het gebied van software schrijven. Mijzelf verbeteren was de tweede reden waarom ik TDD wilde toepassen.

Een nadeel aan TDD vind ik dat wanneer de programmeur (in dit geval ikzelf) iets over het hoofd ziet, dit niet in code en ook niet in de bijhorende tests is terug te zien. In sprint 3 heb ik bijvoorbeeld veel gewerkt aan de mogelijkheid om in te loggen via inloggegevens uit AD. Er mogen veel variaties of aliassen van de gebruikersnaam die gebruikt wordt bij deze inlogactie worden ingevoerd. Dat betekent dat deze scenario's ook allemaal getest moeten worden. Als ik een scenario over het hoofd heb gezien, dan zal dit scenario nooit getest worden.

Ik had gewild dat alle scenario's van tevoren bekend waren geweest en dat de requirement-analyse meer specifieke details zou hebben bevat. In het vervolg zal ik hier meer op letten wanneer ik requirements opstel.

Bovengenoemd nadeel kan ik in de toekomst voorkomen, maar ik kan mij voorstellen dat programmeurs desondanks zaken over het hoofd kunnen zien. Dit is wellicht op te lossen door het werk te verifiëren met behulp van onderlinge toetsing.



FIGUUR 31 SCREENSHOT VAN RESULTATEN

Toen ik de programmatuur voor dit afstudeertraject schreef, ben ik een aantal keer in de valkuil getrapt dat ik veel code had geschreven zonder bijbehorende tests te schrijven (dit is aan de orde gekomen in paragraaf 6.4). Dit moest ik mijzelf aanleren, wat niet altijd gemakkelijk ging. Het gaat nu echter al steeds beter. In mijn volgende project zal ik er nog meer op focussen de discipline te verbeteren.

Een andere valkuil vind ik het schrijven van simulatiecode (stubs en mocks). De software die ik heb gemaakt is vaak afhankelijk van andere bronnen, zoals AD of een verbinding met HelloID. Hoewel ik testcode heb geschreven die de werking van deze bronnen simuleert, wil dat niet zeggen dat de simulatie altijd de echte werking weergeeft. Het is de vraag wat je moet doen als de bron niet meer bereikbaar is of als een bron een onverwachtse melding geeft. Om deze reden heb ik ervoor gekozen om ook integratietests toe te voegen die echte bronnen aanspreekt, zodat ik de werking met een echte bron kan garanderen (dit is besproken in paragraaf 7.3).

Ik kan na dit traject zeggen dat ik veel ervaring op heb gedaan en dat ik vind dat ik mijzelf door het gebruik van TDD op een aantal vlakken heb verbeterd. Deze ervaringen en kennis kan ik goed delen met mijn collega's, waardoor wij TDD wellicht kunnen gaan toepassen in het ontwikkeltraject van HelloID.

10.2 PRODUCTEVALUATIE

Ik ben trots op het eindresultaat, een modulaire service met bijbehorende documentatie en moduletests, waarbij binnen HelloID kan worden bepaald hoe en waarnaartoe gegevens worden gesynchroniseerd. Dit laatste heeft de complexiteit van dit product verhoogd.

Het eindresultaat bevat veel meer dan wat met de vorige AD-connector mogelijk was, bijvoorbeeld inloggen met gegevens uit een bronsysteem. Voorheen was het alleen mogelijk om de AD te raadplegen, maar met de modulaire structuur is het mogelijk om ook veel andere systemen te raadplegen (bijvoorbeeld SQL of andere LDAP-gebaseerde systemen).

De agent bevat een uniek soort verbinding met HelloID en een complex synchronisatiemechanisme dat gegevens synchroniseert aan de hand van een interval of pas op het moment dat dit nodig is (Just in time).

In het afstudeerplan lag de focus meer op het ondersteunen van meerdere bronnen. Dit is echter tijdens het verzamelen van requirements en ontwikkelen van de service veranderd. Het werd belangrijker om te bepalen hoe en waarnaartoe gegevens worden gesynchroniseerd binnen HelloID. Doordat ik meer tijd heb besteed aan de ontwikkeling, ben ik niet aan het maken van de handleiding toegekomen. De handleiding wordt nog voor de oplevering van de agent gemaakt door mijn collega Aad Lutgert.

Mijn opdrachtgever en ik zijn tevreden over de door mij geschreven code; deze is flexibel, uit te breiden en begrijpelijk voor andere ontwikkelaars. Doordat ik met TDD heb gewerkt, kunnen regressietests uitgevoerd worden. De code sluit aan op de gemaakte documentatie (klassendiagrammen, een EER-model, etcetera). Volgens mijn collega's George Bakker en Peter Vos wordt efficiënt omgegaan met middelen, zoals de database en het geheugen van de server en de cliënt.

Een van mijn demo's van het eindresultaat, waarin ik aantoon dat binnen vijf minuten twintigduizend gebruikers worden gesynchroniseerd en gelijktijdig kan worden ingelogd, leverde de volgende reacties op van belanghebbenden:

> "Wat werkt dit snel" – Tjeerd Seinen
>
> "Beheerders kunnen nu bepalen welk veld voor de gebruikersnaam wordt gebruikt? Reden voor een feestje!" – Aad Lutgert

Eind februari 2017 zullen wij dit onderdeel officieel opleveren bij een release van ons product HelloID. Hoewel de service alle 'must-haves' bevat, hebben wij besloten om nog een aantal 'should haves' en 'could haves' (die onder de 'Epic' v2 vallen) te ontwikkelen (en te testen), voordat het product opgeleverd wordt.

10.3 Competentie-evaluatie

Als ik terugkijk op de uitvoering van de afstudeeropdracht, kan ik concluderen dat ik alle competenties die ik in mijn afstudeerplan heb opgenomen heb uitgevoerd op het niveau dat ik had aangegeven. Daarnaast heb ik werkzaamheden uitgevoerd die aantonen dat ik ook andere competenties heb uitgevoerd op niveau 3 of hoger. Hieronder beschrijf ik per competentie welke werkzaamheden ik heb uitgevoerd.

10.3.1 Uitvoeren analyse door definitie van requirements

Tijdens de aanloopfase heb ik zelfsturend de analyse uitgevoerd om business requirements en gebruikers- en softwarerequirements in kaart te brengen (dit is aan de orde gekomen in paragraaf 4.2). Ik ben gestart met het ontlokken van eisen en wensen aan de hand van interviews met belanghebbenden. Om meer details te ontlokken heb ik vervolgens een sessie gehouden waarbij ik een prototype van het installatieproces heb gebruikt.

Tijdens het verzamelen van de requirements bleken sommige wensen van groter belang dan andere, wat de scope van het project aanpaste. Om de scope van het project te bewaken heb ik aan de hand van de MoSCoW-methode de requirements geprioriteerd. De requirements en prioriteiten zijn door de opdrachtgever gevalideerd.

De verzamelde requirements heb ik verwerkt in een requirementsrapport. Daarnaast heb ik de requirements vertaald naar user stories in onze backlog, die is bijgehouden en beheerd in de Scrum-tool Jira.

10.3.2 Opstellen gegevensmodel voor database en Ontwerpen, bouwen en bevragen van een database

De eerste sprint die ik heb uitgevoerd stond volledig in het teken van het ontwerpen van het nieuwe domeinmodel waarmee HelloID wordt uitgebreid (zie de hoofdstuk 5). Met behulp van een EER-model en een klassendiagram heb ik zelfstandig een gegevensmodel ontworpen voor een grote en bedrijfkritische applicatie, waarin sprake is van meerdere relaties tussen twee objecten, geserialiseerde gegevens en overerving.

Ik heb de volledigheid en duidelijkheid van het model steeds getoetst door het model iteratief te presenteren. Aan de hand daarvan konden wij als team beslissen of we tevreden waren over het nieuwe domeinmodel waarmee HelloID werd uitgebreid.

Zelfsturend heb ik het gegevensmodel vertaald naar een relationeel representatiemodel (RRM) en bijbehorend relationeel implementatiemodel (RIM; zie paragraaf 5.2). Het RIM is in SQL geschreven, zodat hij direct kon worden geïmplementeerd in de toen huidige situatie. In het RIM is rekening gehouden met geldende beperkingen op de data en met databaseovererving.

10.3.3 Ontwerpen systeemdeel

Tijdens elke sprint heb ik zelfsturend delen voor de Directory Agent en HelloID ontworpen. In sprint 2 heb ik bijvoorbeeld veel aandacht besteed aan het ontwerp van de modulaire structuur en de verbinding tussen de agent en HelloID.

Alle diagrammen en ontwerpen heb ik opgenomen in het document zijn te vinden in de ontwerpdocumentatie. Hierin staat voor zowel de agent als voor HelloID een klassendiagram. In deze klassendiagrammen is onder andere rekening gehouden met abstractie, koppeling en cohesie (zie paragraaf6.2). Daarnaast heb ik ervoor gezorgd dat de code uit te breiden, herbruikbaar en testbaar is. Al het bovenstaande is onder andere bereikt door design patterns toe te passen.

Naast de klassendiagrammen heb ik sequentiediagrammen gebruikt om complex gedrag uit te beelden (zie paragraaf 6.3.2). Om aan de lezer duidelijk te maken in welke context de systemen staan, heb ik een contextdiagram gemaakt.

Het ontwerp had betrekking op een objectgeoriënteerde applicatie waarbij rekening moest worden gehouden met een multi-tenant architectuur in een loadbalanced omgeving.

10.3.4 Bouwen applicatie

De HelloID Directory Agent heb ik zelfsturend opgezet en gebouwd. De agent is een objectgeoriënteerde multi-threaded Windows Service, geschreven in C#, die aansluit op bestaande software (zie hoofdstuk 6,7 en 8). Binnen de service heb ik een framework gebouwd waarin modulair identiteitsbronnen kunnen worden ondersteund (zie paragraaf 6.2). Ik heb gebruikgemaakt van het .NET framework en low level APIs van bijvoorbeeld Windows (zie paragraaf 7.1.1).

Bij het schrijven van de software heb ik de ontwikkelmethode Test Driven Development gehanteerd, wat zorgde voor een grote focus op testbaarheid. Daarnaast heb ik gefocust op hergebruik en heb ik de software zo geschreven dat deze eenvoudig kan worden uitgebreid met nieuwe functionaliteiten.

De ontwikkeling heeft plaatsgevonden binnen de geavanceerde ontwikkelomgeving van Tools4ever, waar gebruik wordt gemaakt van de tool Visual Studio en versiebeheer met Team Foundation Services. Tijdens de ontwikkeling is gebruikgemaakt van een demo-omgeving voor consultants en is er een testomgeving ingericht met domain controllers om meerdere integratietests mee op te zetten (zie paragraaf 7.3).

10.3.5 UITVOEREN VAN EN RAPPORTEN OVER HET TESTPROCES

Tijdens de afstudeeropdracht heb ik Test Driven Development gehanteerd, waarbij ik gelijktijdig code en testscripts heb geschreven, zowel voor moduletests als voor integratietests. Uit deze tests volgt automatisch een rapportage.

Deze methodiek had ik nog niet eerder gehanteerd. Om ervoor te zorgen dat ik deze methodiek goed kon oppakken, heb ik een algemeen testplan geschreven. Hierin heb ik beschreven wat de aanpak voor het testen tijdens dit traject was, wat de testbasis was en welke tooling gebruikt zou worden.

Deze methodiek is beschreven in de paragrafen 4.1.2.2, 4.3, 6.4 en 7.3.

11 BRONVERMELDINGEN

Andrew Hunt, D. T. (2000). *The Pragmatic Programmer.* Addison-Wesley.

Brussaard, R. (2016, January 27). *SAML, wat is het en wat doet het*. Opgehaald van Tools4ever B.V.: https://www.tools4ever.nl/blog/2016/saml-wat-is-het-en-wat-doet-het/

Cohen, J. (2002, december 4). *Enhanced Entity Relationship (EER) Model.* Opgehaald van userwww.sfsu.edu: http://userwww.sfsu.edu/dchao/EnhancedERDRelationalDBF06.ppt

Dan Pilone, R. M. (2008). Test Driven Development. In R. M. Dan Pilone, *Head First Software Development* (p. 281). United States of America: O'Reilly Media, Inc. .

Dictionaries, O. (sd). *Cache*. Opgehaald van Oxford Dictionaries: https://en.oxforddictionaries.com/definition/cache

Dictionaries, O. (sd). *Encapsulation (computing)*. Opgehaald van Oxford Dictionaries: https://en.oxforddictionaries.com/definition/encapsulation

Dictionaries, O. (sd). *HyperText Transfer Protocol Secure*. Opgehaald van Oxford Dictionaries: https://en.oxforddictionaries.com/definition/http

Fowler, M. (2002). Chapter 12. Object-Relational Structural Patterns. In M. Fowler, *Patterns of Enterprise Application Architecture.* Addison Wesley.

Fowler, M. (2002). *Refactoring.* Addison-Wesley.

Gregory, K. (2003, April 28). *Managed, Unmanaged, Native: What Kind of Code Is This?* Opgehaald van Developer.com: http://www.developer.com/net/cplus/article.php/2197621/Managed-Unmanaged-Native-What-Kind-of-Code-Is-This.htm

Grood, D.-J. d. (2008). *TestGoal.* Den Haag: Sdu Uitgevers.

Inversion of Control Containers and the Dependency Injection pattern. (2004, January 23). Opgehaald van martinfowler.com: http://martinfowler.com/articles/injection.html#InversionOfControl

Jeff Sutherland, K. S. (2014). The Scrum Guide. In J. Sutherland, *The Scrum Guide.* Scrum.Org.

John Hughes, E. M. (2004). *Security Assertion Markup Language.* OASIS. Opgehaald van http://xml.coverpages.org/saml.html

Koomen, T. (2014). *TMap Next.* Vianen.

Kurige. (2010, Augustus 25). *Difference between "managed" and "unmanaged"*. Opgehaald van Stackoverflow: http://stackoverflow.com/questions/3563870/difference-between-managed-and-unmanaged

Lightweight Directory Access Protocol. (2013, March 22). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol

Load balancing. (2015, Augustus 24). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Load_balancing

Microsoft. (2014, August 4). *Scripting with Windows PowerShell*. Opgehaald van MSDN: https://technet.microsoft.com/en-us/library/bb978526.aspx

Microsoft. (2015, July 20). *Introduction to the C# Language and the .NET Framework*. Opgehaald van msdn.microsoft.com: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

Microsoft. (2015, July 20). *lock Statement (C# Reference)*. Opgehaald van MDSN: https://msdn.microsoft.com/nl-nl/library/c5kehkcz.aspx

Microsoft. (2016). *Garbage Collection*. Opgehaald van msdn.microsoft.com: https://msdn.microsoft.com/en-us/library/0xy59wtx(v=vs.110).aspx

Microsoft. (sd). *DllImportAttribute Class*. Opgehaald van MSDN: https://msdn.microsoft.com/en-us/library/system.runtime.interopservices.dllimportattribute(v=vs.110).aspx

Microsoft. (sd). *SAM-Account-Name attribute*. Opgehaald van MSDN: https://msdn.microsoft.com/en-us/library/ms679635(v=vs.85).aspx

Microsoft. (sd). *Stack.Pop Method ()*. Opgehaald van MSDN: https://msdn.microsoft.com/en-us/library/system.collections.stack.pop(v=vs.110).aspx

Microsoft. (sd). *User-Principal-Name attribute*. Opgehaald van MSDN: https://msdn.microsoft.com/en-us/library/ms680857(v=vs.85).aspx

Mockobject. (2013, March 20). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Mockobject

MoSCoW-methode. (2015 , October 6). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/MoSCoW-methode

Netkachov, A. (2015). *LOH Fragmentation*. Opgehaald van alexatnet.com: https://alexatnet.com/articles/memory-leaks-in-net-applications/large-object-heap-fragmentation

Network, T. (2008). *TimeBox*. Opgehaald van searchsoftwarequality.techtarget.com: http://searchsoftwarequality.techtarget.com/definition/timebox

Network, T. (2012, June). *Active Directory forest (AD forest)*. Opgehaald van http://searchwindowsserver.techtarget.com/: http://searchwindowsserver.techtarget.com/definition/Active-Directory-forest-AD-forest

Polling (techniek). (2016, Augustus 20). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Polling_(techniek)

Refactoren. (2014 , December 3). Opgehaald van Wikipedia: https://nl.wikipedia.org/wiki/Refactoren

Shvets, A. (2015). In A. Shvets, *Design Patters Explained Simply* (p. 10). SourceMaking.com.

Staaij, R. v. (2014). *Handboek Identity & Access management.* Academic Service.

Stephens, M. (2016, May 31). *Large Object Heap Uncovered*. Opgehaald van blogs.msdn.microsoft.com: https://blogs.msdn.microsoft.com/maoni/2016/05/31/large-object-heap-uncovered-from-an-old-msdn-article/

Swart, N. D. (2010). *Handboek Requirements.* Eburon business.

Tim Howes, M. S. (1997). *LDAP: Programming Directory-Enabled Applications with Lightweight Directory Access Protocol.* Indianapolis: Macmillan Technical Publishing.

Transmission Control Protocol. (2016, December 20). Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/Transmission_Control_Protocol

Tré, G. d. (2007). Principes van Databases. In G. d. Tré, *Principes van Databases.*

Uml.org. (2005 , July). *WHAT IS UML*. Opgehaald van Uml.org: http://www.uml.org/what-is-uml.htm

WebSocket. (2016, October 7). Opgehaald van Wikipedia: https://en.wikipedia.org/wiki/WebSocket#Overview

12 DEFINITIES EN AFKORTINGEN

12.1 DEFINITIES

Begrip	Definitie
.Net Framework	Dit is een software raamwerk dat door Microsoft is ontwikkeld.
Azure	Dit is een dienst van Microsft waarmee een aantal internetdiensten wordt aangeboden, zoals: het hosten van servers in hun datacenters.
Black-box test	Categorie waarin tests vallen die geen kennis hebben van de interne werking en structuur van het systeem. Het hoofddoel is het testen van functionaliteiten.
BYOD **Bring your own device**	Dit is een concept waarbij medewerkers hun eigen privé apparaten gebruiken (zoals smartphone of laptop) voor zakelijk gebruik. Daarnaast betekent het concept ook dat het voor onze dienst niet uitmaakt of het via een smartphone, tablet of wat voor apparaat dan ook gebruikt wordt. Het werk via elk apparaat.
Cloud **Cloud computing**	Cloud of cloud computing is een concept waarbij via het internet hardware, software of gegevens beschikbaar worden gesteld.
Compiler	Dit is een stuk software dat een hogere programmeertaal (zoals c# broncode) vertaald naar een lagere computertaal zoals bijvoorbeeld machine code. Een compiler wordt meestal gebruikt om broncode om te zetten naar een computerprogramma.
EER(m) **Enhanced entity relationship (model)**	Diagram voor het weergeven van een gegevensmodel voor een database.
Error guessing	Testtechniek waarbij door de tester wordt gegist naar fouten.
IDaas, **Identity as a service**	Identity as a service is een authenticatie infrastructuur dat is gebouwd en beheerd in de Cloud. Het wordt ook vaak omschreven als SSO voor in de Cloud.

Begrip	Definitie
Identity provider	Een entiteit binnen HelloID wat staat voor een identiteitsbron dat identiteiten levert door middel van een federatieprotocol.
Integratie tests	Methode om de samenwerking tussen software componenten te testen.
JIT-provisioning	Het aanmaken van resources op het moment dat ze nodig zijn. Just in time (JIT).
JSON	JSON of JavaScript Object Notation is een gegevensformaat wat in vorm van javascript objecten is genoteerd.
Refactoring	Refactoren is het herstructureren van de broncode, met als doel de leesbaarheid of onderhoudbaarheid te verbeteren.
RIM **Relational Implementation model**	Een model voor de implementatie van een gegevensmodel voor een database.
RRM **Relational Representation model**	Een model voor de relaties tussen entiteiten binnen een gegevensmodel van een database.
Scrum	Scrum is een softwareontwikkelingsmethode om software te maken. Er wordt gewerkt in kleine teams die in korte sprints (time-box binnen Scrum), met een vaste lengte van een tot vier weken, werkende software opleveren. Binnen een sprint wordt de software ontwikkeld en getest.
SSO **Single Sign On**	Single Sign On software ondersteund het volgende principe: Gebruikers hoeven slechts eenmalig in te loggen om daarna toegang te krijgen tot meerdere applicaties (zonder opnieuw in te loggen).
Synchronisatie	Een proces dat zorgt dat twee of meer systemen exact dezelfde gegevens-set heeft op een bepaald moment.

Begrip	Definitie
TTD **Test Driven Development**	Een softwareontwikkelingsmethode waarbij (module/integratie) tests eerst worden geschreven en daarna pas de implementatie van de programmatuur.
Unit tests/Module tests	Methode om software modules of stukken programmatuur (units) afzonderlijk te testen.
User story/ requirement	Een korte beschrijving van een wens of eis wat een belanghebbende van het te ontwikkelen/testen systeem verwacht.
White-box tests	Categorie waarin tests vallen die kennis hebben van het gehele systeem. Het hoofddoel is het testen van de interne structuren en werking van het systeem.

12.2 AFKORTINGEN

Afkoring	Van
AD	Active Directory
SQL	Structured Query Language
SSO	Single Sign On
TTD	Test Driven Development
IDaas	Identity as a service
JIT	Just in time
IoC	Inversion of Control
BYOD	Bring your own device
EER(M)	Enhanced Entitiy Relationship-model
HDA	Hello Directory Agent

13 FIGUREN

Figuur 1 Huidige situatie met AD-connector 15
Figuur 2 Gewenste situatie met Directory agent 16
Figuur 3 prototype van de wizard 26
Figuur 4 Eerste EER-model in de conceptuele fase 30
Figuur 5 Analyse klassendiagram 32
Figuur 6 EER model van de database 33
Figuur 7 Main entry point 39
Figuur 8 SourceInstanceResolver 40
Figuur 9 Publieke api 40
Figuur 10 Resolving IUserManager 41
Figuur 11 Critical secion flow with threads 43
Figuur 12 Contextdiagram 44
Figuur 13 Agent bereikbaar 45
Figuur 14 Agent niet bereikbaar 46
Figuur 15 Met backplante 47
Figuur 16 socket met pollingmechanisme 49
Figuur 17 Sequence diagram verbrinding opzetten 50
Figuur 18 Activity Diagram 51
Figuur 19 Domain forest 54
Figuur 20 Voorbeeldcode voor UPN 55
Figuur 21 DllImport 56
Figuur 22 Directory Searcher klasse 57
Figuur 23 powershell Create user script 58
Figuur 24 Full data transfer 59
Figuur 25 Timestamp transfer 60
Figuur 26 Large object heap reseveren geheugen 63
Figuur 27 Process users voorbeeld 64
Figuur 28 Ingerichte demo-omgeving 65
Figuur 29 Epics in Jira 69
Figuur 30 Voorbeeld sprintboard 70
Figuur 31 Screenshot van resultaten 72

14 Bijlage A: Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen*)

Afstudeerblok: 2016-2.1 (start uiterlijk 29 augustus 2016)
Startdatum uitvoering afstudeeropdracht: 1 augustus 2016
Inleverdatum afstudeerdossier volgens jaarrooster:

Studentnummer: 13070290
Achternaam: dhr. Brussaard (*) *weghalen niet van toepassing*
Voorletters: R
Roepnaam: Robbert
Adres: C.P.M. Rommeplantsoen 52
Postcode: 1067WT
Woonplaats: Amsterdam
Telefoonnummer: +31 64 56 00 428
Mobiel nummer: +31 64 56 00 428
Privé emailadres: R.Brussaard@tools4ever.com

Opleiding: Informatica
Locatie: Den Haag
Variant: deeltijd c.q. avond

Naam studieloopbaanbegeleider: Merie Heijne
Naam begeleidend examinator: Arie Toet
Naam tweede examinator: Paul Smit

Naam bedrijf: Tools4ever B.V.
Afdeling bedrijf: Development
Bezoekadres bedrijf: Amalialaan 126C
Postcode bezoekadres: 3743 KJ
Postbusnummer: -
Postcode postbusnummer: -
Plaats: Baarn
Telefoon bedrijf: +31 35 54 83 255
Telefax bedrijf: +31 35 54 32 736
Internetsite bedrijf: https://www.tools4ever.nl

Achternaam opdrachtgever: dhr. Bakker (*) *weghalen niet van toepassing*
Voorletters opdrachtgever: G
Titulatuur opdrachtgever: Ingenieur, Bachelor of Science in Information Technology
Functie opdrachtgever: Senior and lead software engineer
Doorkiesnummer opdrachtgever: +31 65 37 83 007
Email opdrachtgever: g.bakker@tools4ever.com

Achternaam bedrijfsmentor: dhr. Bakker (*) *weghalen niet van toepassing*
Voorletters bedrijfsmentor: G
Titulatuur bedrijfsmentor: Ingenieur, Bachelor of Science in Information Technology

Functie bedrijfsmentor: Senior and lead software engineer
Doorkiesnummer bedrijfsmentor: +31 65 37 83 007
Email bedrijfsmentor: g.bakker@tools4ever.com
NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder: +31 64 56 00 428
Functie afstudeerder (deeltijd/duaal): Senior software developer

Titel afstudeeropdracht:

Ontwikkelen van een (modulaire) authenticatie en synchronisatie service voor HelloID bij Tools4ever B.V.

Opdrachtomschrijving

1. Bedrijf

Tools4ever is een internationaal bedrijf dat sinds 1999 gestandaardiseerde en betaalbare Identity Governance & Administration (IGA) oplossingen ontwikkelt en levert, die in tegenstellig tot andere IGA-oplossingen eenvoudig te implementeren en te beheren zijn. Tools4ever is in Nederland dé absolute marktleider op het gebied van IGA.

Met deze oplossingen bedient Tools4ever een breed scala van organisaties variërend in grootte, van driehonderd tot meer dan tweehonderdduizend gebruiker accounts. Zowel kleine ondernemingen, multinationals, onderwijsinstellingen, zorginstellingen als (lokale) overheid zetten de oplossingen van Tools4ever in. Het betref zowel non-profit als profit organisaties.

Tools4ever is een jonge, dynamische en groeiende organisatie. Medewerkers krijgen veel vrijheid en kansen om mee te helpen aan de groei van dit bedrijf. Eigen initiatief van de medewerkers wordt enorm gewaardeerd. Daarom krijgen de mensen die bij Tools4ever de ruimte om zichzelf verder te ontplooien en te ontwikkelen, door middel van opleidingen en doorgroeimogelijkheden. Samen met ongeveer honderdtwintig medewerkers wereldwijd staan wij sterk en bouwen we aan ons merk.

Het afgelopen jaar is Tools4ever begonnen met de ontwikkeling van een nieuw SSO-product HelloID. HelloID is een IDaas (Identity as a service) Cloud-oplossing waarbij een online portaal wordt aangeboden waarop men kan inloggen, eenmaal ingelogd op het portaal vindt de eindgebruiker een overzicht van beschikbare cloudapplicaties (bijvoorbeeld, Google Apps, Office 365, LinkedIn, Github) de eindgebruiker hoeft bij het openen van deze applicaties niet nogmaals in te loggen (SSO, Single Sign On). Zie https://www.tools4ever.nl/software/helloid-idaas-cloud-single-sign-on/ voor meer informatie.

Bij dit bedrijf ben ik werkzaam als software developer en werk ik aan IAM (Identity and access management) oplossingen. Op dit moment werk ik op de afdeling waar SSO-oplossingen worden ontwikkeld. Deze afdeling bestaat op dit moment uit vier personen; De verwachting is dat dit jaar het aantal personen groet naar vijf.

2. Probleemstelling

Bij het product HelloID [1] kan op verschillende manieren worden ingelogd. Dit kan onder andere via een Active Directory (AD) connectie, waarbij met een gebruikersnaam en wachtwoord combinatie moet worden ingelogd. De AD-connectie (hierna te noemen AD-Connector) is een service die wordt geïnstalleerd in de omgeving van de klant. De service maakt een verbinding naar HelloID aan en kan vervolgens inlogacties op HelloID controleren (om de identiteit van een gebruiker te verifiëren) tegen de AD binnen de omgeving van een klant.

Op dit moment wordt alleen AD als bron van identiteiten ondersteund. Het is hierdoor voor een klant niet mogelijk om een ander bronsysteem te gebruiken om inlog acties te controleren. Doordat verschillende bronnen niet worden ondersteund is het niet mogelijk om alle klantomgevingen te ondersteunen. Andere bronnen die klanten zouden willen gebruiken zijn bijvoorbeeld:

- SQL-gebaseerd DBMS [3];
- LDAP-gebaseerd systeem dan AD [4];
- Tools4ever's Single Sign On (SSO) product E-SSOM [2];
- CSV-bestanden [5];

Naast dat de AD-connector niet andere bronnen ondersteunt, controleert de huidige AD Connector alleen inlogacties en synchroniseert het niet de status van de Connector of gegevens van gebruikers met HelloID. Hierdoor is de service niet zo effectief als dat de service zou kunnen zijn.

Het is nu niet mogelijk om:

- De status van een AD-Connector uit te lezen;
- Gegevens van gebruikers te synchroniseren met HelloID;
- Instellingen te maken zodat wanneer de AD-connector offline is er gebruik gemaakt mag worden van gesynchroniseerde gegevens;
- Andere gegevensbronnen dan Active Directory te gebruiken om gebruikers in te loggen op HelloID.

Doordat de AD-connector bovenstaande onderdelen niet bevat, loopt Tools4ever potentiële klanten mis en staat het product HelloID minder sterk in de markt dan Cloud SSO-producten van concurrenten.

3. Doelstelling van de afstudeeropdracht

Het ontwikkelen van een nieuwe service* staat centraal tijdens de afstudeeropdracht. Tools4ever wil dat de nieuwe service de huidige AD-connector vervangt. Bij het ontwikkelen van deze service zijn de volgende doelstellingen opgesteld:

1. Het ondersteunen van verschillende gegevensbronnen

   De nieuwe service moet niet alleen Active Directory als bron van identiteiten en gegevens ondersteunen, maar ook bronnen als andere LDAP-systemen, SQL-databases, Tools4ever's E-SSOM of zelfs bestanden zoals CSV-bestanden.

   Out-of-the-box, zullen LDAP-systemen (eerst AD), SQL-databases en E-SSOM worden ondersteund.

2. Nieuwe service moet uit te breiden zijn

   De service moet modulair zijn gebouwd, zodat in de toekomst eenvoudig een nieuwe bron kan worden toegevoegd door Tools4ever of door klanten zelf.

3. Synchroniseren van gegevens

   Niet alleen moeten inlogacties gecontroleerd worden tegen het bronsysteem, maar ook moeten gebruikergegevens uit het bronsysteem met HelloID gesynchroniseerd kunnen worden.

**De nieuwe service kan alleen gebruikt worden op een Windows-besturingssysteem.*

4. Resultaat

Als de afstudeeropdracht succesvol is uitgevoerd: is een effectieve modulaire service gerealiseerd, die de huidige AD-connector kan vervangen. Deze service is in staat om verschillende gegevensbronnen te raadplegen. De service kan inlogacties controleren en kan gebruiker gegevens met HelloID synchroniseren.

De service zal de volgende modules 'out-of-the-box' bevatten:

- LDAP-module;
- SQL-module;
- E-SSOM-module;

Door deze oplossing is het mogelijk dat HelloID bruikbaar is voor meer klanten. Hierdoor kan HelloID zich beter in de markt kunnen plaatsen en concurreren met soortgelijke producten.

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

Activiteit	Aantal dagen	Opmerking
Gesprekken met experts, stakeholders e.d.	3	
Plan van aanpak schrijven	5	
Requirementsrapport opstellen	3	
Algemeen testplan	3	Een algemeen plan met daarin beschreven hoe wordt getest tijdens de ontwikkeling.
Ontwerpdocumentatie	8	Bevat de ontwerp documentatie van de architectuur en beschrijvingen van eventuele database aanpassingen
Bouw applicatie	42	
Finetuning van het product	2	
Handleiding schrijven	3	
Demonstratie omgeving inrichten	1	
Opbouwen afstudeerdossier	15	Dit zal gedurende de gehele afstudeer periode worden bijgewerkt.

Vooraf aan de ontwikkeling zullen requirements worden vergaard van verschillende opdrachtgevers. Samen met de opdrachtgevers wordt de omvang van het project beheerst. Daarvoor wordt een geprioriteerde lijst met requirements (zowel functionele als niet-functionele eisen en wensen) bijgehouden. Om de lijst te prioriteren wordt de MoSCoW-principe [6] gehanteerd.

Om op wijzigingen in de requirements in te spelen wordt tijdens de ontwikkeling gebruik gemaakt van de agile software-ontwikkelmethodiek SCRUM. Dit is de standaard die wordt gehanteerd bij Tools4ever. Dit project zal in het lopende SCRUM-proces van HelloID binnen Tools4ever worden opgenomen en uitgevoerd.

SCRUM is een iteratieve ontwikkelmethode, waarbij nauw met de opdrachtgever en andere stakeholders wordt samengewerkt om elke sprint (time-box waarin software wordt ontwikkeld en getest) werkende software op te leveren. Dat is prettig, want het biedt ruimte voor nieuwe ideeën, kansen en inzichten die naar voren komen tijdens het ontwikkelen.

Het beschrijven van de architectuur en structuur zal gebeuren door middel van de modelleringstaal UML. Dit wordt opgenomen in de ontwerpdocumentatie.

De nieuwe service zal worden ontwikkeld in C#. Aanpassingen aan HelloID zullen worden verricht met C#, T-SQL, HTML en CSS.

De projectorganisatie bestaat:

Rol	Persoon	Opmerking
Afstudeerder, Software developer, ontwerper, tester en projectleider	Robbert Brussaard	
Expert	George Bakker	Voor eventuele technische vragen kan George worden benaderd.
Functioneel tester	Aad Lutgert	Door Aad wordt het product functioneel getest (het beschrijven van de functionele testen valt buiten de scope van het afstudeerproject).

Op te leveren (tussen)producten

- Plan van aanpak;
- Requirementsrapport;
- Algemeen testplan;
- Ontwerpdocumentatie;
    - Systeemarchitectuur;
    - Indien nodig: database design;
- Gerealiseerde applicatie (De nieuwe service);
- Handleiding;
- Ingerichte demonstratie omgeving;
- Rapportage van unit tests;

6. Te demonstreren competenties en wijze waarop

1.4 Uitvoeren analyse door definitie van requirements Niveau 3

De requirements van meerdere stakeholders voor de modulaire service zullen worden opgesteld. Deze requirements beschrijven zowel functionele als niet- functionele requirements. Het is goed mogelijk dat er tegenstrijdigheden zijn in de eisen en wensen van verschillende stakeholders, mocht dit het geval zijn dan wordt hier rekening mee gehouden. De requirements worden vertaald naar de product backlog. Het interviewen en verwerken van de requirements zal zelfstandig worden uitgevoerd.

3.2 Ontwerpen systeemdeel Niveau 3

Het betreft het ontwerpen van een objectgeoriënteerde applicatie. In de ontwerpdocumentatie zal de structuur van de applicatie worden beschreven met behulp van de modelleringstaal UML. Het ontwerpen en schrijven van deze documentatie zal zelfstandig worden uitgevoerd.

3.3 Bouwen applicatie Niveau 4

De applicatie die ontwikkelt moet worden, zal object georiënteerd worden opgebouwd, waarbij geavanceerde concepten van de programmeertaal C# aan de orde komt. De applicatie moet aansluiten op bestaande software (HelloID, AD, en meer systemen). Het ontwikkelen van deze applicatie zal gebeuren in een geavanceerde ontwikkelomgeving en staat in verbinding met een versiebeheertool (TFS van Microsoft). De ontwikkeling van de applicatie en eventuele aanpassingen aan HelloID zal zelfstandig plaatsvinden.

3.5 Uitvoeren van en rapporteren over het testproces Niveau 3

Tijdens en na de bouw van de applicatie moet deze worden getest. Dit zal met behulp van unit testing worden gedaan. Het doel van unittesten is om softwaremodulen of stukken programmatuur onafhankelijk van elkaar te kunnen testen op correcte werking. In het algemene testplan zal worden beschreven hoe wordt getest en hoe de resultaten van de unit tests worden gerapporteerd. Het schrijven van unit testen en het uitvoeren hiervan zal zelfstandig en sturend worden uitgevoerd.

Verwijzingen

[1] https://www.tools4ever.nl/software/helloid-idaas-cloud-single-sign-on/
[2] https://www.tools4ever.nl/software/enterprise-sso-single-sign-on-authentication-management/
[3] https://nl.wikipedia.org/wiki/Databasemanagementsysteem
[4] https://nl.wikipedia.org/wiki/Lightweight_Directory_Access_Protocol
[5] https://en.wikipedia.org/wiki/Comma-separated_values
[6] https://nl.wikipedia.org/wiki/MoSCoW-methode

BIJLAGE B: BEDRIJFSBEZOEK

De heer Arie toet, mijn begeleidend examinator, heeft op 24 oktober 2016 een bezoek gebracht aan het bedrijf Tools4ever B.V. Tijdens dit bezoek hebben de opdrachtgever, George Bakker, en ik Arie kennis laten maken met mijn werkomgeving. Wij hebben Arie een rondleiding gegeven door het kantoor en de HelloID-afdeling.

Vervolgens hebben George, Arie en ik plaatsgenomen in de kleine vergaderruimte waar Arie de formele gang van zaken omtrent het afstuderen heeft uitgelegd. Met nadruk op de terugkoppelmomenten en waaraan het afstudeerdossier moet voldoen. Hiernaast heeft Arie ook aangestipt dat wij erover moeten nadenken of wij het afstudeerdossier willen laten opnemen in de HBO-kennisbank.

Na de formele gang van zaken hebben wij besproken hoe afstuderen tot nu gaat. Hierbij heb ik kenbaar gemaakt dat het goed verloopt en dat ik veel uren maak om de reis die ik november ga maken (naar Peru) te compenseren. Ook heb ik aangegeven dat andere eisen en wensen naar voren zijn gekomen en dat dit de scope van het project heeft aangepast. De opdrachtgever en ik hebben de eisen en wensen zo geprioriteerd zodat een succesvol product kan worden afgeleverd aan het einde van het afstudeertraject en dit na het traject verder kan worden opgepakt (ik heb een vast contract bij Tools4ever B.V.) in een nieuw ontwikkeltraject.

Wij hebben de volgende afspraken gemaakt m.b.t. de terugkoppelmomenten:

- Voor 15 november 2016 bespreken wij het concept afstudeerdossier;
    - Over de mail hebben wij afgesproken dat dit donderdag 10 november 2016 zal plaatsvinden op de Kromhoutkazerne in Utrecht.
- Begin december 2016 houden wij het tussentijds assessment;
- In de week van 31 oktober tot 6 november zal ik het voortgangsverslag inleveren en op Blackboard een elektronisch portfolio beschikbaar maken.

Wij hebben op dit moment nog niet besloten of wij wel of niet het afstudeerdossier laten opnemen in de HBO-kennisbank. Hier komen wij (Tools4ever B.V.) later op terug.

BIJLAGE C: VOORTGANGSVERSLAG

VOORTGANG

AANLOOPFASE

In de eerste twee weken, de aanloopfase, van mijn afstudeertraject ben ik begonnen met het opstarten van mijn project. Het opstarten van het project houd in dat ik een plan van aanpak heb opgesteld en in kaart heb gebracht wat ontwikkeld moet worden. Het geen wat ik in kaart heb gebracht heb ik beschreven in vorm van functionele en niet-functionele requirements in een requirementsrapport (gebruik gemaakt van Handboek Requirements van Nicole de Swart[8]).

Voor het verzamelen van de requirements heb ik interviews gehouden en discussies gevoerd met behulp van een prototype. Uit deze analyse kwamen interessante procesverbeteringen die wij bij voorhand bij het bepalen van de afstudeeropdracht niet hadden bedacht.

Bijvoorbeeld de procesverbetering dat klanten zelf kunnen bepalen hoe en waarnaartoe gegevens worden gesynchroniseerd van het bronsysteem naar HelloID is belangrijker dan het ondersteunen van meerdere gegevensbronnen. Dit omdat de huidige bron (Active Directory) de meest gebruikte bron is, en deze bron voor het grootste gedeelte ook door toekomstige/nieuwe klanten gebruikt zal worden.

Het ontwikkelen van een dergelijk proces (mapping mechanisme) is complex en zal daarom een grote impact hebben op het afstudeertraject. Om ervoor te zorgen dat het project niet uitloop en dat een succesvol product is gerealiseerd na het afstudeertraject hebben de opdrachtgever en ik de MoSCoW[9]-methode toegepast op de requirements te prioriteren. Zo zijn de requirements die nodig zijn voor de bovenstaande procesverbetering als ‘Must-have’ gemarkeerd. De requirements voor het initiële proces om meerdere identiteitsbronnen te ondersteunen hebben wij gemarkeerd als ‘should have’.

SPRINT 1

Na de aanloopfase heeft bij de start van de sprint een sprintplanning plaatsgevonden (zoals bij elke sprint). In deze planning hebben wij als team bepaald welke user story’s (lees requirements) worden opgepakt. Deze sprint had voor mij als doel: het ontwerpen van een nieuw domeinmodel. Bij het ontwerpen van dit model heb ik veel discussie gehad met mijn collega’s, vooral met George Bakker de opdrachtgever. Deze discussie ging vooral over de structuur van de entiteiten (profielen en attributen) die de structuur van een bronsysteem moeten weergeven in ons systeem.

De opdrachtgever stelde dat uit de mapping (tabel met kolom voor bronveld en kolom voor doelveld) het profiel kon worden afgeleid. Dit is op zekere hoogte waar, maar ik heb hem duidelijk gemaakt dat

[8] Swart, N. D. (2010). Handboek Requirements. Eburon business.
[9] https://nl.wikipedia.org/wiki/MoSCoW-methode

wij bronvelden willen gebruiken om regels af te dwingen en dat het kan dat een bronveld hiervoor niet in de mapping hoeft voor te komen. Op basis hiervan en door het mogelijk te maken van complexe datastructuren door middel van het opslaan van het profiel in een JSON-document heb ik de opdrachtgever kunnen overtuigen.

Vervolgens heb ik voor het ontwerp een EER, RRM en RIM opgesteld en database wijzigen doorgevoerd op de HelloID-database.

Bij nader inzien had ik er beter voor kunnen kiezen om de competenties: 2.1 opstellen gegevens model voor database en 2.2 ontwerpen, bouwen en bevragen van een database op te nemen in mijn afstudeerplan. Het is mijn plan om deze als competentie-evaluatie op te nemen in mijn afstudeerverslag.

SPRINT 2

In de volgende sprint heb ik mij gericht op het opzetten van de communicatie tussen de agent en HelloID. De agent is een multi-threaded Windows service dat via een websocket een verbinding maakt met HelloID en vervolgens berichten uitwisselt. Hoe het opzetten van de verbinding moet werken en hoe de uitwisseling van berichten werkt heb ik beschreven in de ontwerpdocumentatie.

Ik ben trots op hoe de agent in elkaar zit en hoe hij werkt. Bij de ontwikkeling heb ik veel rekening gehouden met gelijktijdige authenticatieaanvragen. Ik heb bijvoorbeeld een test gedraaid waarbij duizend aanvragen tegelijk binnenkwamen. Deze aanvragen werden binnen een seconde verwerkt en teruggestuurd naar HelloID.

Tijdens de ontwikkeling heb ik mij ook gericht op het hanteren van de methodiek Test Driven Development, waarbij ik zorg dat tijdens het ontwikkelen ook bijbehoren module tests worden geschreven. Dit vind ik toch wel moeilijk; Om de discipline te blijven houden om eerst de tests te schrijven en dan daarna de implementatie. Ik betrap mijzelf vaak erop dat ik al een stuk code heb geschreven, maar geen bijbehorende tests heb geschreven. Dan schrijf ik ze achteraf. Hier ga ik in het vervolg nog beter opletten.

Wel vind ik het schrijven van de module tests leuk en een voordellen hebben op wanneer je ze niet zal schrijven. Het is motiverend om te zien dat code blijft werken zoals het moet en wanneer je een fout hebt gemaakt dit direct terug te zien is in de testresultaten als de module tests zijn uitgevoerd. Een ander voordeel is dat je anders nadenkt over de architectuur, een voorbeeld: voorheen zou ik low-level API’s direct in de code aanroepen, maar om dit testbaar te maken moet dit abstracter worden gemaakt en pas ik encapsulatie toe, zodat dit binnen een andere klasse gebeurt, die ik tijdens het testen kan vervangen door een gesimuleerd object. Dit heeft als gevolg dat er een losse koppeling ontstaat en dat de uitbreidbaarheid van de code verbeterd.

SPRINT 3

Het doel van de derde sprint was het inloggen via inloggegevens uit Active Directory (AD). In het begin van het ontwikkelen van dit stuk ging het erg gemakkelijk. Het was 'gewoon' gegevens uit het bronsysteem halen, en de melding verwerken bij het loginscherm in HelloID. Maar naarmate dat ik met de ontwikkeling hiervan bezig was kwamen steeds meer niet genoemde eisen aan het inloggen via AD naar boven wat de oude AD-connector ondersteund en wat de nieuwe ook moet ondersteunen.

Hierdoor werd de ontwikkeling hiervan iets taaier. Ik moest bijvoorbeeld rekening houden met alle manieren waarop een gebruiker kan inloggen op AD (dit zijn er nogal wat) en moest ervoor zorgen dat op basis van de gegevens de juiste domain controller wordt opgezocht. Hiervoor moest ik een low-level API van Win32 aanspreken via C#. Via C++ kan dit rechtstreekst worden aangesproken, maar in C# moet er een DLL Import plaatsvinden. Hierbij moest ik goed rekeninghouden dat referenties naar bepaalde blokken in het geheugen weer worden vrijgegeven, wanneer de code klaar met de actie is.

CONCLUSIE

Over het algemeen vind ik dat de ontwikkeling, documentatie en het bewaken van de scope goed gaat. Ik vind het leuk om te doen en ik heb een trots gevoel over het geen wat ik aan het maken ben. Wel vind ik dat beter om moet gaan met het test driven development.

BIJLAGE D: FORMULIER TUSSENTIJDS ASSESSMENT

Bespreking concept	**Tussentijds assessment**	Eerste beoordeling

Formulier tussentijds assessment

Student: Robert Brussaard **Studentnummer:** 13070290

Datum: 14 december 2016 **eerste / tweede TTA:** eerste

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

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

Beroepstaken op afgesproken niveau uitgevoerd?		O	T	V	G
1	1.4 Uitvoeren analyse door definitie van requirements (niveau 3)			x	
2	3.2 Ontwerpen systeemdeel (niveau 3)			x	
3	3.3 Bouwen applicatie (niveau 4)			x	
4	3.5 Uitvoeren van en rapporteren over het testproces (niveau 3)			x	

Producten	O	T	V	G
Tussenproducten			x	
Eindproducten			x	

Effectief communiceren	O	T	V	G
Binnen afstudeerbedrijf			x	
Afstudeerdossier				x

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

<u>Toelichting per beoordelingscriterium</u>

Aanpak
Aanpak is weldoordacht en er is goed gebruik gemaakt van standaard methoden en technieken.

Beroepstaken op afgesproken niveau uitgevoerd?
Alle beroepstaken zijn adequaat uitgevoerd. Rapportage over het testproces is nog niet volledig.

Producten
Hoewel nog niet helemaal af (sprint 4 en 5 moeten nog beschreven worden) waren alle producten van goede kwaliteit. Punt van zorg is wel het taalgebruik, dit is zondermeer slecht.

Effectief communiceren
Communicatie met examinatoren is goed, Uit verslag blijkt ook een goede communicatie binnen afstudeerbedrijf

Reflectie
Student weet goed wat hij doet en wat hij wil. Dit blijkt met name uit een uitgebreide evaluatie.

Advies

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

Besluit student

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

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

Naam begeleidend examinator: Arie Toet

Naam tweede examinator: Paul Smit

Datum: 16 december 2016

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

BIJLAGE E: PLAN VAN AANPAK

SAMENVATTING

Dit document bevat het plan van aanpak voor het ontwikkeltraject van de HelloID Directory Agent. Alle onderdelen, conclusies en aanbevelingen zijn verwerkt in dit document, met eventuele verwijzingen naar andere documenten die betrekking hebben tot dit document.

VERSIEBEHEER

Versie	Auteur	Datum	Omschrijving
1.0	Robbert Brussaard	08-09-2016	Inleiding, Doelstellingen, Afspraken, Werkwijze en planning bijgewerkt.
1.1	Robbert Brussaard	17-09-2016	MoSCoW toegevoegd aan werkwijze.

VERZENDLIJST

Wie	Contactgegevens
George Bakker	G.Bakker@tools4ever.com
Tjeerd Seinen	T.Seinen@tools4ever.com
Aad Lutgert	A.Lutgert@tool4ever.com
Peter Vos	P.Vos@tools4ever.com
Robbert Brussaard	R.Brussaard@tools4ever.co
Arie Toet	A.J.Toet@hhs.nl
Paul Smit	P.R.Smit@hhs.nl

UITGANGSPUNTEN

- Gesprekken met George Bakker en Tjeerd Seinen;

PROJECTDEFINITIE

BEDRIJF / ORGANISATIE

Tools4ever is een internationaal bedrijf dat sinds 1999 gestandaardiseerde en betaalbare Identity Governance & Administration (IGA) oplossingen ontwikkelt en levert, die in tegenstellig tot andere IGA-oplossingen eenvoudig te implementeren en te beheren zijn. Tools4ever is in Nederland dé absolute marktleider op het gebied van IGA. Met deze oplossingen bedient Tools4ever een breed scala van organisaties variërend in grootte, van driehonderd tot meer dan tweehonderdduizend gebruiker accounts. Zowel kleine ondernemingen, multinationals, onderwijsinstellingen, zorginstellingen als (lokale) overheid zetten de oplossingen van Tools4ever in. Het betref zowel non-profit als profit organisaties.

OPDRACHTOMSCHRIJVING

Halverwege vorig jaar (2015) is Tools4ever begonnen met de ontwikkeling van een nieuw Single Sign On-product HelloID. HelloID is een IDaas (Identity as a service) Cloud-oplossing waarbij een online portaal wordt aangeboden waarop men kan inloggen, eenmaal ingelogd op het portaal vindt de eindgebruiker een overzicht van beschikbare cloudapplicaties (bijvoorbeeld Google Apps of Office 365) de eindgebruiker hoeft bij het openen van deze applicaties niet nogmaals in te loggen (Single Sign On), dit handelt HelloID af.

Bij HelloID [1] kan op verschillende manieren worden ingelogd. Dit kan onder andere via een Active Directory (AD) connectie, waarbij met een gebruikersnaam en wachtwoord combinatie kan worden ingelogd. De AD-connectie wordt gemaakt door een service die is geïnstalleerd in de omgeving van de klant. De service maakt een verbinding naar HelloID aan en kan vervolgens inlogacties op HelloID controleren (om de identiteit van een gebruiker te verifiëren) tegen de AD binnen de omgeving van een klant.

Op dit moment wordt alleen AD als bron van identiteiten ondersteund. Het is hierdoor voor een klant niet mogelijk om een ander bronsysteem te gebruiken om inlog acties te controleren. Doordat verschillende bronnen niet worden ondersteund is het niet mogelijk om alle klantomgevingen te ondersteunen. Andere bronnen die klanten zouden willen gebruiken zijn bijvoorbeeld:

- SQL-gebaseerd DBMS;
- LDAP-gebaseerd systeem;
    - AD is ook een LDAP-gebaseerd systeem, het is de bedoeling dat ook andere LDAP gebaseerde systemen worden ondersteund;
- Tools4ever's Single Sign On (SSO) product E-SSOM;
- CSV-bestanden.

Naast dat de huidige AD-connector geen andere bronnen ondersteunt dan AD bevat het ook de onderstaande functionaliteiten niet.

- De status van een AD-Connector tonen in HelloID;
- Gegevens van gebruikers te synchroniseren met HelloID;
- Offline ondersteuning voor gegevens in HelloID.

Doordat de AD-connector bovenstaande functionaliteiten mist is de dienst (combinatie van AD-connector en HelloID) niet zo effectief als dat de het zou kunnen zijn.

Doordat de AD-connector bovenstaande onderdelen niet bevat is de dienst (combinatie van AD-connector en HelloID) niet zo effectief als dat het zou kunnen zijn. Als gevolg staat het product HelloID minder sterk in de markt dan Cloud SSO-producten van concurrenten en lopen wij (Tools4ever) potentiële klanten mis.

DOELSTELLING

Om de dienst te verbeteren en het product HelloID beter in de markt te plaatsen zal een nieuwe Windows-service worden ontwikkeld dat de huidige AD-connector zal vervangen. Deze nieuwe service moet naast Active Directory ook andere gegevens bronnen ondersteunen. Out-of-the-box zal de service AD en SQL en ESSOM als gegevensbron moeten ondersteunen.

Om in de toekomst ervoor te zorgen dat de service eenvoudig kan worden uitgebreid met gegevensbronnen: moet de service modulair worden opgezet en een goede interface bieden. Via deze interface is het mogelijk dat Tools4ever zelf de uitbreidingen schrijft, maar biedt het ook de mogelijkheid dat een derde partij of klant deze uitbreidingen kan schrijven.

Met deze gegevensbronnen moet de bestaande functionaliteit, het authentiseren, worden ondersteund. Maar de nieuwe service moet ook gebruiker gegevens kunnen synchroniseren naar HelloID.

AFBAKENING

Onderstaand wordt een afbakening gedefinieerd waarin wordt aangegeven welke aspecten en activiteiten binnen de opdracht vallen.

- Alleen de producten (zie mijlpaal producten) en activiteiten die in dit document staan beschreven vallen binnen de opdracht;
- De service zal alleen worden ontwikkeld voor op Windows;
- Wijzigingen die een maand voor de afronding van het traject worden bekend gemaakt, worden niet meegenomen in dit traject;
- Implementeren van de nieuwe service bij klanten valt niet binnen de opdracht.

PROJECTORGANISATIE

PRODUCT OWNER

Wie	Rol
George Bakker	Expert, Manager HelloID, Senior softwareontwikkelaar

STAKEHOLDERS

Wie	Rol
Tjeerd Seinen	Salesmanager
Sebastiaan Otten	Development manager
Peter Vos	HelloID softwareontwikkelaar, Gebruiker

PROJECTTEAM

Wie	Rol
Aad Lutgert	Tester, implementatie consultant
Robbert Brussaard	Projectuitvoerder, Scrum-master, Senior softwareontwikkelaar, Tester

INTERNE AFSPRAKEN

- Documenten die tijdens deze opdracht worden geschreven worden beheerd en gearchiveerd op de Dropbox van Robbert Brussaard. Een kopie van deze documenten worden bewaard op het interne netwerk van Tools4ever;
- Programmatuur wordt in de Team Foundation Server, het versie beheersysteem, van Tools4ever opgeslagen;
- Documenten gebruiken het Tools4ever sjabloon.

WERKWIJZE

Hieronder is beschreven welke fases het project kent en welke methodieken worden gehanteerd tijdens dit ontwikkeltraject.

FASERING

AANLOOPFASE

Tijdens deze fase wordt dit document, het plan van aanpak en de planning, opgesteld. Naast dit document zal in de beginperiode de requirements voor de HelloID Directory Agent verzamelt en in kaart gebracht. Op basis van deze requirements wordt de eerste sprintplanning uitgevoerd en bepaald wat in de eerste sprint wordt ontwikkeld.

CYCLISCHE SCRUM-FASE (SPRINTS)

Na de aanloopfase wordt direct gestart met de eerste sprint (zie hoofdstuk Scrum). Tijdens een sprint worden onderdelen voor dit project ontworpen, ontwikkeld en getest. Na elke sprint kan een nieuwe sprint worden gestart om het product uit te breiden of te verbeteren. Wat in die sprint wordt uitgevoerd wordt bepaald in de sprintplanning. Zo wordt iteratief met een terugkerende time-box aan dit project gewerkt. De time-box (doorlooptijd) die wij bij Tools4ever hanteren is twee weken.

Tijdens en na elke sprint wordt de sprint en software geëvalueerd. Op basis van deze evaluatie kunnen nieuwe requirements ontstaan. Deze worden in kaart gebracht en toegevoegd aan de requirements analyse en backlog van HelloID.

Methodieken

Scrum

Scrum is een iteratieve ontwikkelmethode, waarbij nauw met de opdrachtgever en andere belanghebbende wordt samengewerkt om elke sprint werkende software of valide documentatie op te leveren. Dat is prettig, want het biedt ruimte voor nieuwe ideeën, kansen en inzichten die naar voren gaan komen tijdens het proces. Per sprint stellen we met elkaar vast wat binnen die sprint wordt ontwikkeld. Aan het einde van de sprint bekijken we het resultaat kritisch met elkaar.

De HelloID Directory Agent wordt ontwikkeld op basis van eisen en wensen (requirements) vanuit Tools4ever. Door middel van Scrum kan tijdens het ontwikkeltraject op een goede en flexibele manier met deze requirements worden omgegaan. Bij Tools4ever wordt Scrum als standaard gehanteerd bij het softwareontwikkelproces.

Zie meer informatie over Scrum op: http://www.scrumguides.org/scrum-guide.html

Test Driven Development (TDD)

Tijdens het ontwikkeltraject wordt ook gebruik gemaakt van Test Driven Development (hierna te noemen TDD). TDD is een software-ontwikkelmethode waarbij het testen van software een integraal onderdeel is van softwareontwikkeling. De ontwikkelmethode hanteert dat bij het schrijven van een nieuwe unit (deel van de code) eerst de bijbehorende tests worden geschreven daarna pas de implementatie van dat deel code.

Deze methode draagt bij aan de kwaliteit van de code en levert metaforische parachutes waardoor bij wijzigen tijdens het traject wordt gecontroleerd of eerder geschreven units nog steeds correct werken.

TDD complementeert de Scrum ontwikkelmethode goed. Het zorgt ervoor dat tijdens het ontwikkelen direct wordt getest en uiteindelijk dat eerder geschreven software blijft werken zoals dat was bedoeld. Waardoor een betere garantie kan worden afgegeven voor werkende software aan het einde van een sprint.

MoSCoW

Bij het in de kaart brengen van de requirements moeten ook prioriteiten moeten worden gesteld. Dit zal worden gedaan aan de hand van de methode: MoSCoW[10]. Met deze methode worden de requirements ingedeeld in verschillende categorieën.

MoSCoW staat voor een ezelbruggetje, waarbij de medeklinkers in het woord de categorieën vertegenwoordigen. Hieronder staan de categorieën met een verklaring opgesomd in volgorde van het meest belangrijk tot minder belangrijk:

- **Must have**
  Requirements met deze prioriteit moeten in het eindproduct zijn verwerkt om het tot een succes te brengen.

- **Should have**
  De requirements die deze prioriteit hebben gekregen zijn zeer welkom in het eindproduct. Deze zouden het product naar een hoger niveau tillen.

- **Could have**
  Indien er genoeg tijd is zullen de requirements met deze prioriteit worden opgenomen in het eindproduct.

- **Won't have**
  Requirements met deze prioriteit zullen in dit traject niet worden opgenomen in het eindproduct. In een vervolgtraject zal dit interessant zijn om wel op te nemen in het eindproduct.

[10] https://nl.wikipedia.org/wiki/MoSCoW-methode

PLANNING

ACTIVITEITEN

REQUIREMENTS VERZAMELEN

Om een goed beeld te krijgen van de functionaliteiten en eigenschappen die de Directory Agent moet bezitten worden de requirements hiervoor in kaart gebracht. Om de requirements te verzamelen worden verschillende elicitatietechnieken gebruikt:

- Interviewen van belanghebbende;
- Het maken van prototype (dit kan als item tijdens een sprint);

Ontdekte requirements zullen in een requirementsrapport worden opgenomen. Op basis van deze requirements zal bij een sprintplanning worden bepaald wat op dat moment moet worden ontwikkeld of uitgewerkt. Bij elke sprint wordt dus opnieuw de relevantie en prioriteit van een requirement bekeken.

Tijdens een sprint kan het voorkomen dat nieuwe ideeën of inzichten zich voordoen. Deze worden als requirement opgenomen. Tijdens het afstudeertraject is het goed mogelijk dat de initiële lijst met requirements is gegroeid of is veranderd. Bij het opstellen van de initiële lijst van requirements worden de prioriteiten van de lijst gesteld aan de hand van de MoSCoW-methode.

ONTWERPEN

Tijdens elke sprint worden software requirements beschreven en toegevoegd aan de ontwerpdocumentatie. Dit kan per requirement verschillen wat nodig is. Bijvoorbeeld: als de requirement database wijzigingen vereist dan moet voor dat gedeelte een database-ontwerp worden opgenomen, hierbij kan gebruik worden gemaakt van EER-modellen. Voor het weergeven van andere ontwerpen zoals klassendiagrammen wordt UML gebruikt.

ONTWIKKELEN

Het ontwikkelen kan betrekking hebben op zowel de Directory Agent als wijzigingen op het HelloID-portaal. In elke de sprint zullen de ingeplande requirements worden ontwikkeld. Tijdens het ontwikkelen wordt gebruikt gemaakt van de software-ontwikkelmethode TDD. De Directory Agent zal worden ontwikkeld in C#. Aanpassingen aan het portaal kunnen betrekking hebben met de volgende technieken: C#, SQL, HTML, CSS of Javascript.

TESTEN

Zoals eerder aangegeven wordt tijdens het ontwikkelen gebruik gemaakt van TDD en volgen hieruit unit tests. Om duidelijk te maken hoe deze tests worden opgezet en hoe deze worden gerapporteerd, zal een algemeen testplan worden opgesteld.

Inrichten demonstratie omgeving

Net voor het einde van het afstudeertraject wordt voor de werkende software op dat moment een demonstratie omgeving opgetuigd. De demonstratie omgeving bestaat uit een HelloID-portaal, een virtuele Windows server met daarop een Active Directory en de nieuwe Directory Agent.

Schrijven van handleiding

Voor het gebruik en installatie van de HelloID Directory Agent zal een handleiding worden voor gebruikers en beheerders worden opgesteld. Deze handleiding wordt toevoegt aan onze bestaande documentatie portaal: http://docs.helloid.com

Globale planning

Hieronder staan de activiteiten in een globale planning verwerkt. Het afstudeertraject loopt van week 36 tot en met week 52. De aanloopfase zal in week 36 en 37 plaatsvinden. Daarna zullen de cyclische fase met daarin de iteratieve sprint van start gaan. Tijdens het afstudeertraject loopt dit van week 37 tot en met week 51.

Activiteit	36	37	38 tot en met 49	50	51	52
Requirements verzamelen						
Ontwerpen						
Ontwikkelen						
Testen						
Demo-omgeving inrichten						
Handleiding schrijven						

Mijlpaal producten

- Requirementsrapport;
- Algemeen testplan;
- Documentatie van systeemarchitectuur en database ontwerp;
- Gerealiseerde applicatie: De HelloID Directory Agent;
- Handleiding;
- Ingerichte demonstratie omgeving;
- Rapportage van unit tests.

BIJLAGE F: REQUIREMENTSRAPPORT

VERSIEBEHEER

Versie	Auteur	Datum	Omschrijving
1.0	Robbert Brussaard	09-09-2016	Eerste versie opgezet
1.1	Robbert	17-09-2016	Uitbreiden van requirements en stellen van prioriteiten.

VERZENDLIJST

Wie	Contactgegevens
George Bakker	G.Bakker@tools4ever.com
Tjeerd Seinen	T.Seinen@tools4ever.com
Sebastiaan Otten	S.Otten@tools4ever.com
Aad Lutgert	A.Lutgert@tool4ever.com
Peter Vos	P.Vos@tools4ever.com
Robbert Brussaard	R.Brussaard@tools4ever.co
Arie Toet	A.J.Toet@hhs.nl
Paul Smit	P.R.Smit@hhs.nl

UITGANGSPUNTEN

- Interviews met Aad Lutgert, George Bakker;
- Plan van aanpak;

INLEIDING

Dit document bevat de requirementsspecificatie van het te ontwikkelen HelloID Directory Agent (HDA). Het document heeft als doel om alle betrokkenen bij het HDA-ontwikkeltraject duidelijk te maken wat voor systeem wordt ontwikkeld moet worden en welke redenen daaraan ten grondslag liggen. Hiernaast zal duidelijk worden gemaakt uit welke requirements HDA bestaat.

BEDRIJFSVISIE

HUIDIGE SITUATIE

Aan het huidige systeem, de AD-Connector, ontbreken belangrijke functionaliteiten zoals: het synchroniseren van gebruikersgegevens, het mappen van velden en het ondersteunen van andere identiteitsbronnen dan Active Directory (AD). Dit leidt tot problemen bij klanten die een andere identiteitsbron hebben, of klanten die een groot aantal gebruikers hebben die gesynchroniseerd moeten worden met HelloID.

- Het komt voor dat klanten zelf een synchronisatiemechanisme moeten schrijven. Hierdoor is het product minder aantrekkelijk om aan te schaffen.

- Op dit moment is voor de klant niet te bepalen hoe en waarnaartoe brongegevens gesynchroniseerd worden binnen HelloID (mappen van velden). Hierdoor is het product niet flexibel genoeg om brongegevens te gebruiken in bijvoorbeeld losstaande inloggegevenssets, waardoor handelingen door gebruikers of beheerders handmatig moet worden verricht.

- Bij een intake van een klant blijkt nogal eens dat zij een andere identiteitsbron dan AD gebruiken. Hierdoor kan de klant bij de intake al beslissen dat zij het product niet aanschaffen.

Onder andere door deze problemen is het product HelloID minder aantrekkelijk om aan te schaffen en loopt Tools4ever omzet mis.

GEWENSTE SITUATIE

Het proces om ervoor te zorgen dat gebruikers correct en efficiënt uit welke bron dan ook in HelloID komen te staan moet verbeteren. Klanten moeten geen noodzaak hebben om zelf een synchronisatiemechanisme te schrijven. Hierdoor moet de dienst van HelloID stabiel blijven en de klant er niet van weerhouden om het product aan te schaffen. Om het bovenstaande te verbeteren wordt een nieuwe service ontwikkeld de: HelloID Directory Agent. In dit document staan de requirements beschreven om dit te bewerkstelligen.

BUSINESSREQUIREMENTS

Tools4ever wil het systeem ontwikkelen want:

B1) De opdrachtgever wil dat klanten niet zelf een synchronisatiemechanisme hoeven te schrijven;

B2) De beheerder wil eenvoudig zelf kunnen bepalen hoe en waarnaartoe gegevens worden gesynchroniseerd tussen het bronsysteem en HelloID.

B3) De opdrachtgever wil dat een klant niet afhaakt bij een intake omdat hun identiteitsbron niet kan worden ondersteund;

B4) De klant wil snel en eenvoudig kunnen inloggen met dezelfde gebruikersgegevens als dat hij gebruikt in zijn bedrijfsnetwerk.

BELANGHEBBENDE/ACTOREN

EINDGEBRUIKERS

GEBRUIKERS

Gebruikers kunnen via hun HelloID-portaal inloggen. Waar zij gebruik kunnen maken van de diensten dat dit platform biedt. Zij zijn personeelsleden van een organisatie dat HelloID heeft afgenomen. De HelloID softwareontwikkelaar George Bakker zal deze gebruikers vertegenwoordigen in het project.

BEHEERDERS/CONSULTANTS

De beheerders zorgen voor het beheer van hun HelloID-portaal. Zij beheren de gebruikers, groepen, applicaties en meer. Zij zijn ook personeelsleden van een organisatie dat HelloID heeft afgenomen. Zij zijn vaak de personen die weten of hun huidige infrastructuur aansluit op HelloID, met in combinatie van de huidige AD-connector. George Bakker en Aad Lutgert zullen de beheerders vertegenwoordigen in het project.

GEBRUIKERSOMGEVING

Alle instanties van het HelloID-portaal hebben de mogelijkheid om gebruik te maken van de huidige AD-connector en in de toekomst de nieuwe HelloID Directory Agent (HDA). Bijna (ongeveer 80%) zal daadwerkelijk gebruik gaan maken van de nieuwe HDA. HDA zal door de beheerders worden beheerd en geïnstalleerd binnen hun Windows-infrastructuur.

SYSTEEM

CONTEXT

De belangrijkste functionaliteiten van het nieuwe connector systeem zijn: synchroniseren van gebruikers, authentiseren tegen identiteitsbronnen, de flexibiliteit om te bepalen hoe gegevens worden gesynchroniseerd (velden mappen) en het ondersteunen van verschillende identiteitsbronnen. Het correct, efficiënt en veilig synchroniseren van gegevens bij alle functionaliteiten zijn hierbij kritische succesfactoren.

HelloID is een verzamelnaam voor de dienst dat wordt aangeboden. Het HelloID-portaal waarop gebruikers inloggen en waarmee beheerders onder andere gebruikers en groepen beheren is onderdeel van de dienst HelloID. De nieuwe connector systeem is ook onderdeel van deze dienst. Hieronder is het contextdiagram van HDA weergegeven.

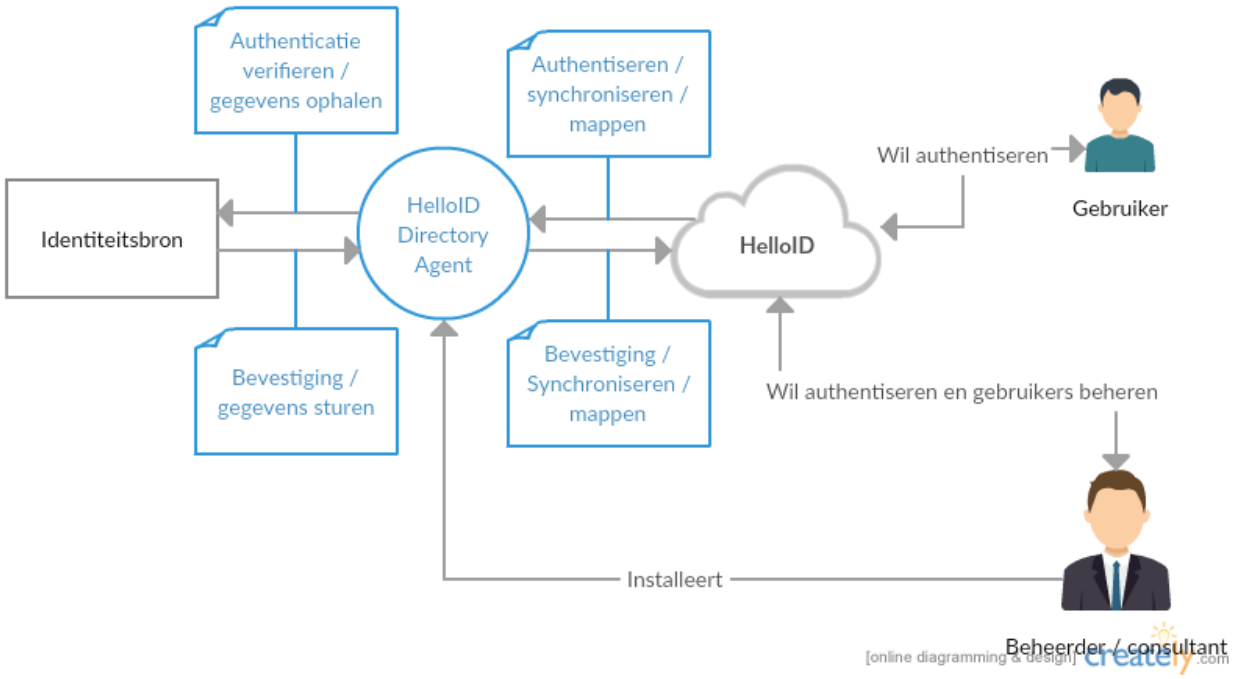


FIGURE 1 CONTEXTDIAGRAM

REQUIREMENTS

GEBRUIKERSREQUIREMENTS

Hieronder staan alle functionele requirements wat een gebruiker met het systeem wil doen. Deze requirements beschrijven de functionele onderdelen dat het product in de gewenste situatie moet bevatten. De requirements zijn geschreven in de vorm van user story's:

<Identificatie code>	**<Titel>**	**<prioriteit>**	**<traceability>**
Als <actor> wil ik <eis/wens>.			

Requirements zijn gegroepeerd op belanghebbende/actoren. Requirements zijn geprioriteerd aan de hand van de MoSCoW-methode. Zie voor meer uitleg over de MoSCoW-methode het plan van aanpak. Als bijlage heb ik een op prioriteiten gesorteerde lijst van requirements toegevoegd.

GEBRUIKERS

GF0	**Inloggen op het HelloID-portaal met de identiteit die ik gebruik bij mijn organisatie**	**Must have**
Als gebruiker wil ik inloggen op het HelloID-portaal met dezelfde identiteitsgegevens die ik gebruik in het netwerk van mijn organisatie. Zo hoef ik maar een inloggegevens-set te onthouden.		

GF1	**Mijn gegevens inzien**	**Must have**
Als gebruiker wil ik inloggen op het HelloID-portaal en via mijn profiel mijn gegevens inzien. Zo kan ik zien of een veld gecorrigeerd moet worden in het bronsysteem.		

GF2	**Mijn gegevens wijzigen**	**Could have**
Als gebruiker wil ik inloggen op het HelloID-portaal en via mijn profiel mijn gegevens wijzigen. Als twee weg synchronisatie aanstaat kan ik zelf mijn wijziging doorvoeren naar het bronsysteem. Zo hoeft een beheerder niet handmatig of met een ander systeem mijn gegevens te wijzigen.		

BEHEERDERS/CONSULTANTS

Configuratie wizard

BFW0	**Configuratie wizard in het HelloID-portaal**	**Must have**
Als beheer wil ik eenvoudig het configuratie proces doorlopen. Ik wil dat het HelloID-portaal mij hierbij helpt door middel van een configuratie-wizard. Hierdoor zal ik minder fouten maken tijdens het installatie en configuratie proces.		

BFW1	**Directory Agent downloaden**	**Must have**
Als beheerder of consultant wil ik dat ik in de eerste stap de Directory agent kan downloaden. Zo wordt direct duidelijk gemaakt dat ik eerst dit stuk software nodig heb voordat ik verder kan gaan.		

BFW2	**Genereren van een ticket**	**Must have**
Als beheerder of consultant wil ik dat HelloID een ticket voor mijn Directory Agent genereert. Zodat niet een kwaadwillend persoon een Directory Agent naar onze HelloID omgeving kan wijzen en hierop vervolgens kan inloggen.		

BFW3	**Installeren van de agent via ticket**	**Must have**
Als beheerder of consultant wil ik de Directory agent installeren in mijn omgeving. Bij het installeren moet ik de ticket opgeven, zodat de agent weet waarmee hij moet verbinden.		

BFW4	**Directory agent automatisch verbinden naar HelloID**	**Must have**
Als beheerder of consultant wil ik dat ik de verbinding door HDA na installatie automatisch wordt opgezet. Zodat ik geen extra instellingen hiervoor hoeft te doen.		

BFW5	Status van verbinding tonen in wizard	Should have
Als beheerder of consultant wil ik dat wanneer de verbinding door de directory agent is opgezet de status hiervan wordt getoond in de configuratie wizard. Zo weet ik of de verbinding goed is opgezet of helemaal niet.		

BFW6	De wizard annuleren	Must have
Als beheerder of consultant wil ik dat ik het proces kan annuleren. Dit zodat ik eenvoudig terug kan gaan naar het overzicht scherm en dat eventuele lopen processen worden uitgeschakeld.		

BFW7	Velden mappen in de laatste stap	Must have
Als beheerder of consultant wil ik dat via een interface gegevens die uit het bronsysteem komen in de juiste velden in HelloID terechtkomen. Het mappen van velden zou ook beschikbaar moeten zijn in de configuratie wizard. Zo weet ik bij het configureren gelijk hoe velden worden gesynchroniseerd.		

Field mapping

BFM0	Velden mappen via interface	Should have
Als beheerder of consultant wil ik dat via een interface gegevens die uit het bronsysteem komen in de juiste velden in HelloID terechtkomen. Dit zodat gebruikers niet zelf hun gegevens hoeven te corrigeren		

BFM1	Data manipulatie	Won't have
Als beheerder of consultant wil ik in de mapping kunnen aangeven of de data tijdens het synchroniseren moet worden gemanipuleerd. Bijvoorbeeld: In HelloID moeten usernames worden gebaseerd op de eerste letter van de voornaam en daaraan toegevoegd de achternaam. Hierdoor hoeft in bronsystemen geen wijzigingen worden gemaakt.		

BFM2	Instellen van twee weg synchronisatie	Should have
Als beheerder of consultant wil ik kunnen instellen of de gegevens van gebruikers zowel van de bron naar HelloID mogen worden gesynchroniseerd, maar ook andersom. Bijvoorbeeld de gebruiker wijzigt zijn gegevens, dan wil ik bij twee weg synchronisatie dat dit ook wordt bijgewerkt in het bronsysteem.		

BFM3	Mapping afdwingen	Must have
Als beheerder of consultant wil ik dat de mapping die is opgegeven wordt afgedwongen tijdens het synchroniseren van gegevens. Dit kan zijn tijdens inloggen of wanneer een set van gegevens naar de server worden gestuurd voor synchronisatie. Hierdoor hoeft ik zelf geen synchronisatiemechanisme te schrijven.		

Directory overzicht

BFD0	Overzicht van directory agents	Must have
Als beheerder of consultant wil ik via de userinterface een overzicht van mijn directory agents zien. Zodat ik weet welke systemen verbinding maken met HelloID.		

BFD1	Status controleren	Must have
Als beheerder of consultant wil ik via de userinterface de status van de agent controleren binnen HelloID. Dit zodat ik weet of iets mis is met de verbinding.		

BFD2	Verbinding verwijderen	Must have
Als beheerder of consultant wil ik verbindingen die ik heb aangemaakt ook weer kunnen verwijderen. Dit zodat ik later een andere verbinding kan configureren en overbodige verbindingen kan opruimen.		

Instellingen directory

BFI0	Kiezen van identiteitsbron	Should have
Als beheerder of consultant wil ik via een userinterface kunnen kiezen welke identiteitsbron HDA raadpleegt. Dit maakt het voor mij eenvoudig om in te stellen.		

BFI1	Offline modus voor authenticatie	Could have
Als beheerder of consultant wil ik via een userinterface kunnen instellen dat wanneer de directory agent offline is alsnog kan worden ingelogd op het HelloID-portaal. Hiervoor wordt een local login voor de gebruiker aangemaakt. Dit zodat ik als beheerder continuïteit kan aanbieden aan gebruikers.		

BFI2	Opgegeven van identiteitsbron specifieke AD instellingen	Must have
Als beheerder of consultant wil ik de specifieke AD-instellingen van een AD-identiteitsbron kunnen instellen via de userinterface.		

BFI3	Opgegeven van identiteitsbron specifieke LDAP instellingen	Should have
Als beheerder of consultant wil ik de specifieke LDAP-instellingen van een LDAP-identiteitsbron kunnen instellen via de userinterface.		

BFI4	Opgegeven van identiteitsbron specifieke SQL instellingen	Should have
Als beheerder of consultant wil ik de specifieke SQL-instellingen van een SQL-identiteitsbron kunnen instellen via de userinterface.		

BFI5	Opgegeven van identiteitsbron specifieke E-SSOM instellingen	Should have
Als beheerder of consultant wil ik de specifieke E-SSOM-instellingen van een E-SSOM-identiteitsbron kunnen instellen via de userinterface.		

BFI6	Selecteren welke OU/groepen mogen worden geraadpleegd door de Directory Agent [AD specifieke instelling]	Must have
Als beheerder of consultant wil ik in een userinterface kunnen selecteren welke OU/groepen gesynchroniseerd mogen worden. Zo kan ik als beheerder voorkomen dat bepaalde groepen en leden uit groepen niet worden gesynchroniseerd.		

BFI7	Kiezen voor Just in time synchronisatie	Must have
Als beheerder of consultant wil ik in een userinterface kunnen aangeven of ik gebruik wil maken van just in time provisioning (JIT), dat de gebruiker pas aangemaakt en gesynchroniseerd wanneer deze wil inloggen in het systeem.		

BFI8	Optie om altijd synchroniseren bij JIT aan en uit te zetten	Should have
Als beheerder of consultant wil ik in een userinterface kunnen aangeven of bij het inloggen de gebruiker altijd moet worden gesynchroniseerd of dat dit alleen tijdens synchroniseren van alle gebruikersgegevens gebeurt.		

BFI9	Opgeven van synchronisatie interval	Must have
Als beheerder of consultant wil ik in een userinterface kunnen aangeven om de hoeveel tijd de directory agent opnieuw met synchroniseren met HelloID.		

Synchronisatie

BFS0	Directory agent synchroniseert gebruikersgegevens uit AD	Must have
Als beheerder of consultant wil ik dat HDA de gebruikersgegevens uit AD naar HelloID synchroniseert. Zo hoeven wij als bedrijf zelf geen synchronisatiemechanisme te schrijven.		

BFS1	Directory agent synchroniseert groepsgegevens uit AD	Must have
Als beheerder of consultant wil ik dat HDA de groepsgegevens uit AD naar HelloID synchroniseert. Zo hoeven wij als bedrijf zelf geen synchronisatiemechanisme te schrijven.		

BFS2	Directory agent synchroniseert groepslidmaatschappen uit AD	Must have
Als beheerder of consultant wil ik dat HDA de groepslidmaatschappen uit AD naar HelloID synchroniseert. Zo hoeven wij als bedrijf zelf geen synchronisatiemechanisme te schrijven.		

BFS3	Directory agent synchroniseert gebruikersgegevens uit LDAP	Should have
Als beheerder of consultant wil ik dat HDA de gebruikersgegevens uit LDAP-systemen naar HelloID synchroniseert. Zo hoeven wij als bedrijf zelf geen synchronisatiemechanisme te schrijven.		

BFS4	Directory agent synchroniseert gebruikersgegevens uit SQL	Should have
Als beheerder of consultant wil ik dat HDA de gebruikersgegevens uit SQL-systemen naar HelloID synchroniseert. Zo hoeven wij als bedrijf zelf geen synchronisatiemechanisme te schrijven.		

BFS5	Directory agent synchroniseert credentials uit E-SSOM	Should have
Als beheerder of consultant wil ik dat HDA de credentials van AD-users uit E-SSOM naar HelloID synchroniseert. Zo hoeven wij als bedrijf zelf geen synchronisatiemechanisme te schrijven.		

Installatie eisen

BFIS20	Geen poorten openzetten	Must have
Als beheerder wil ik bij de installatie van de directory agent geen poorten openzetten in mijn huidige infrastructuur.		

SOFTWAREREQUIREMENTS

Hieronder staan de kwaliteitseigenschappen opgesomd waaraan HDA moet voldoen. Deze eigenschappen zijn gegroepeerd op basis van de kwaliteitseigenschappen uit de ISO 9126 standaard. De eigenschappen beschrijven hoe het systeem moet werken en niet wat het systeem zou moeten doen.

FUNCTIONALITEIT

Juistheid

NF1	**Juistheid van gegevens**
De directory agent moet ervoor zorgen dat de juiste gegevens aan de juiste gebruiker wordt gekoppeld. Hierin mag geen fout inzitten (fouten van het bronsysteem uitgezonderd).	

NF2	**Juistheid van verbinding**
De directory agent moet ervoor zorgen dat deze met het juiste HelloID-portaal verbinding maakt.	

Beveiligbaarheid

NF3	**Onderschepping van gegevens niet mogelijk**
HDA en HelloID moeten bij het overzenden van gegevens onderschepping door derden onmogelijk maken.	

NF4	**Gebruik maken van een veilige verbinding**
De verbinding tussen HDA en HelloID moet over een veilige verbinding lopen.	

BETROUWBAARHEID

Foutbestendigheid

NF6	Foutbestendig tegen invalide data
De directory agent mag niet uitvallen bij het ontvangen of verzenden van invalide data.	

NF7	Foutbestendig tegen uitgevallen bronsystemen
De directory agent mag niet uitvallen indien een achterliggend bronsysteem is uitgevallen. Het raadplegen van gegevens hoeft dan niet mogelijk te zijn. De beheerder zou hiervoor als fallback de offline modus kunnen aanzetten in HelloID.	

NF19	Verwijderen van corrupte data
De agent moet in het geval van corrupte data deze verwijderen. Hiermee moet worden voorkomen dat als corrupte data is ontvangen, en niet kan worden verwerkt dat dit oneindig vaak wordt geprobeerd om te verwerken.	

Herstelbaarheid

NF7	Direct opnieuw verbinden
De directory agent moet bij het verliezen van de verbinding naar HelloID direct opnieuw verbinding maken.	

NF8	Bij herstart van server automatisch opstarten
Indien bij de klant de server opnieuw heeft moeten opstaren dan moet de directory agent zo snel mogelijk weer automatisch gestart worden.	

BRUIKBAARHEID

Begrijpelijkheid

NF9	**Het begrijpen van de installatie**
Een beheerder moet binnen enkele minuten begrijpen hoe de installatie van de directory agent werkt.	

NF10	**Het begrijpen van de instellingen**
Een beheerder moet binnen enkele minuten begrijpen hoe hij de directory agent naar zijn wensen kan instellen op het HelloID-portaal.	

NF11	**Het begrijpen van mappen van velden**
Een beheerder moet binnen enkele minuten begrijpen hoe de velden uit het bronsysteem kan ‘mappen’ met de velden in HelloID.	

Gebruiksgemak

NF12	**Tonen van activiteit**
HelloID moet tonen wanneer een Directory Agent verbinding probeert te maken.	

EFFICIENCY

Snelheid

NF13	**Snelheid van inloggen**
De dienst, HelloID en HDA samen, moet inlogacties zo snel mogelijk afhandelen.	

Midden beslag

NF15	**Comprimeren van gegevens**
Om ervoor te zorgen dat het dataverkeer minimaal blijft moeten de gegevens worden gecomprimeerd.	

NF21	**Meerdere verbindingen openen naar HelloID**
De directory agent moet meerdere verbindingen openen naar HelloID om schaalbaarheid te creëren.	

NF20	**Redundant HelloID installeren op verschillende server**
De agent moet op verschillende servers kunnen worden geïnstalleerd zodat verbindingen naar HelloID redundant kunnen worden uitgevoerd. Dit is handig omdat hierdoor de verbinding horizontaal kan schalen, maar ook indien een server uit de lucht gaat de verbinding kan worden opgepakt door een andere server.	

ONDERHOUDBAARHEID

Wijzigbaarheid

NF16	**Uitbreidbaarheid van software**
De software van de directory agent moet eenvoudig uit te breiden zijn. De agent moet in een korte tijd met een extra identiteitsbron uitgebreid kunnen worden. Hoelang het precies duurt is afhankelijk van de identiteitsbron.	

OVERDRAAGBAARHEID

Installeerbaarheid

NF17	**Installeren van HDA**
De directory agent moet door de klant zelf binnen een 10 minuten geïnstalleerd kunnen worden.	

NF18	**Downloaden van HDA**
De directory agent moet door de klant zelf vanaf een HelloID-portaal of docs.helloid.com gedownload kunnen worden.	

PRIORITEITEN VAN REQUIREMENTS

Identificatie code	Titel	Prioriteit
GF0	Inloggen op het HelloID-portaal met de identiteit die ik gebruik bij mijn organisatie	Must have
GF1	Mijn gegevens inzien	Must have
BFW0	Configuratie wizard in het HelloID-portaal	Must have
BFW1	Directory Agent downloaden	Must have
BFD0	Overzicht van directory agents	Must have
BFD1	Status controleren	Must have
BFD2	Verbinding verwijderen	Must have
BFI2	Opgegeven van identiteitsbron specifieke AD instellingen	Must have
BFI6	Selecteren welke OU/groepen mogen worden geraadpleegd door de Directory Agent [AD specifieke instelling]	Must have
BFI7	Kiezen voor Just in time synchronisatie	Must have
BFI9	Opgeven van synchronisatie interval	Must have
BFS0	Directory agent synchroniseert gebruikersgegevens uit AD	Must have
BFS1	Directory agent synchroniseert groepsgegevens uit AD	Must have
BFS2	Directory agent synchroniseert groepslidmaatschappen uit AD	Must have
BFIS20	Geen porten openzetten	Must have
BFW2	Genereren van een ticket	Must have
BFW3	Installeren van de agent via ticket	Must have
BFW4	Directory agent automatisch verbinden naar HelloID	Must have
BFM3	Mapping afdwingen	Must have
BFW6	De wizard annuleren	Must have
BFW7	Velden mappen in de laatste stap	Must have
BFM0	Velden mappen via interface	Should have
BFW5	Status van verbinding tonen in wizard	Should have
GF2	Mijn gegevens wijzigen	Should have
BFM2	Instellen van twee weg synchronisatie	Should have
BFI0	Kiezen van identiteitsbron	Should have

Identificatie code	Titel	Prioriteit
BFI3	Opgegeven van identiteitsbron specifieke LDAP instellingen	Should have
BFI4	Opgegeven van identiteitsbron specifieke SQL instellingen	Should have
BFI5	Opgegeven van identiteitsbron specifieke E-SSOM instellingen	Should have
BFI8	Optie om altijd synchroniseren bij JIT aan en uit te zetten	Should have
BFS3	Directory agent synchroniseert gebruikersgegevens uit LDAP	Should have
BFS4	Directory agent synchroniseert gebruikersgegevens uit SQL	Should have
BFS5	Directory agent synchroniseert credentials uit E-SSOM	Should have
BFI1	Offline modus voor authenticatie	Could have
BFM1	Data manipulatie	Won't have

BIJLAGE G: ONTWERPDOCUMENTATIE

VERSIEBEHEER

Versie	Auteur		Omschrijving
1.0	Robbert Brussaard		Eerste versie opgezet
1.1	Robbert Brussaard		Communicatie model toegevoegd
1.2	Robbert Brussaard		Klassendiagrammen aangepast, Module API uitgewerkt.

VERZENDLIJST

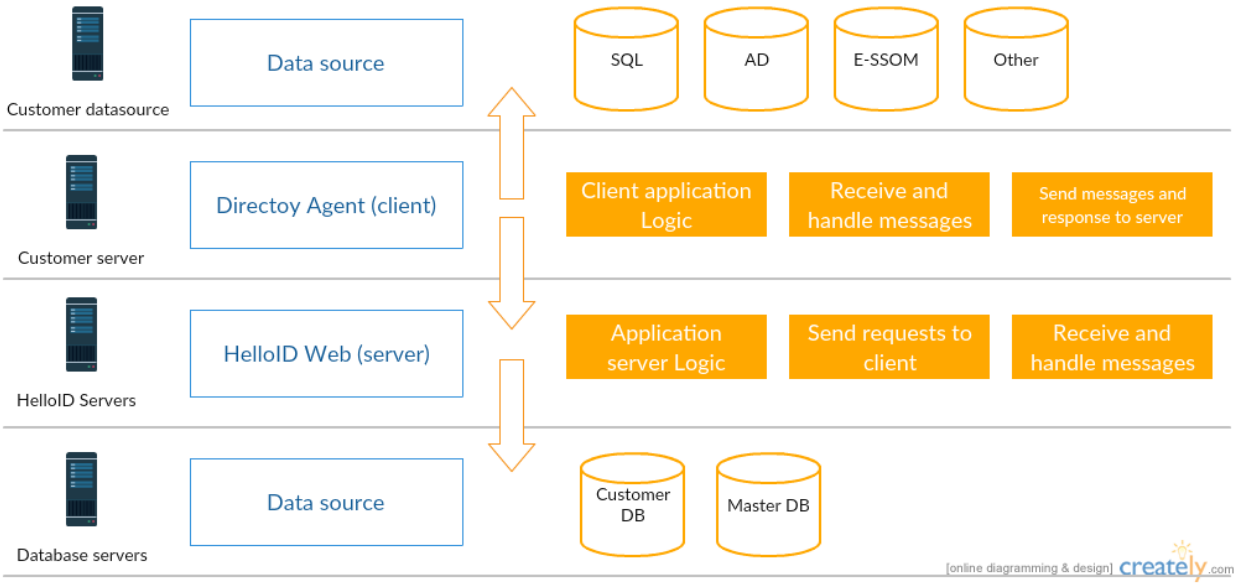
Wie	Contactgegevens
George Bakker	G.Bakker@tools4ever.com
Aad Lutgert	A.Lutgert@tool4ever.com
Robbert Brussaard	R.Brussaard@tools4ever.co
Arie Toet	A.J.Toet@hhs.nl
Paul Smit	P.R.Smit@hhs.nl

SAMENVATTING

Dit document bevat de ontwerpdocumentatie van het de HelloID Directory Agent en de gerelateerde wijzigingen in HelloID. Het document heeft als doel om alle betrokkenen bij de Directory Agent duidelijk te maken hoe het ontwikkelde systeem in elkaar zit.

ARCHITECTURALE STRUCTUUR

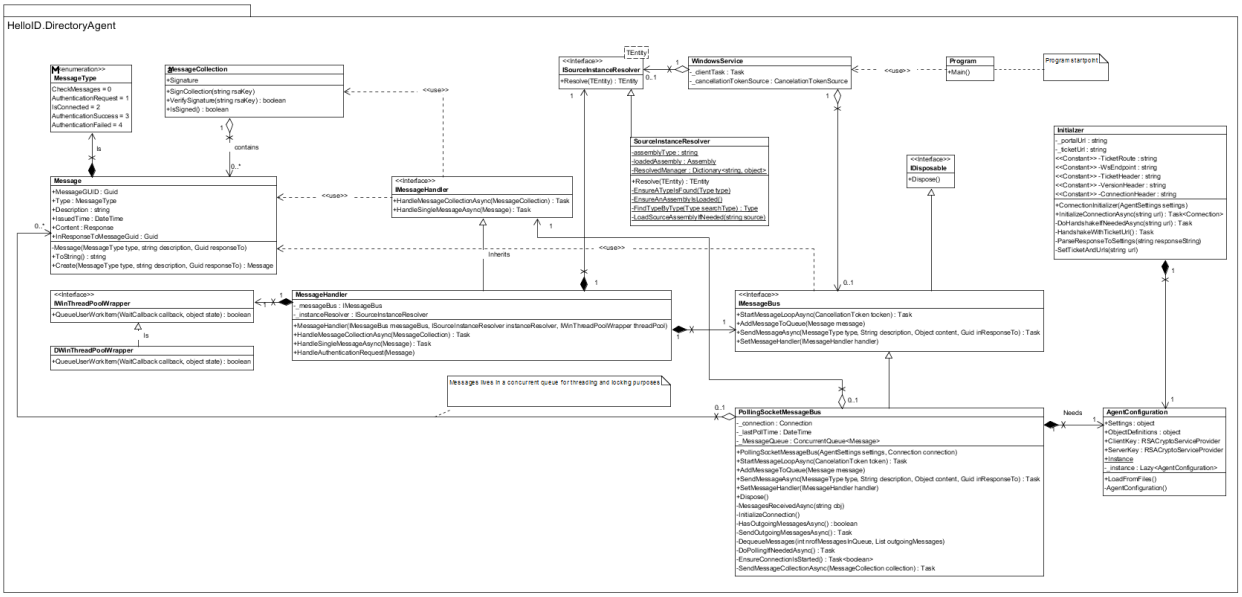
Het systeem is gebaseerd op een Multi-tier (lagen) client-server architectuur. Wat wil zeggen dat meerdere clients (lees Directory Agents) kunnen verbinden met een centrale server waar een deel van de applicatie logica draait en de verbinding met een database. De cliënt zelf bestaat uit twee lagen: Een laag met applicatie logica en verbinding met externe bron, bijvoorbeeld Active Directory of een database. Hieronder wordt de lagenstructuur van de HelloID Directory Agent afgebeeld in Figuur 1.



FIGUUR 32 LAGENSTRUCTUUR

KLASSENDIAGRAM DIRECTORY AGENT

CORE PACAKAGE



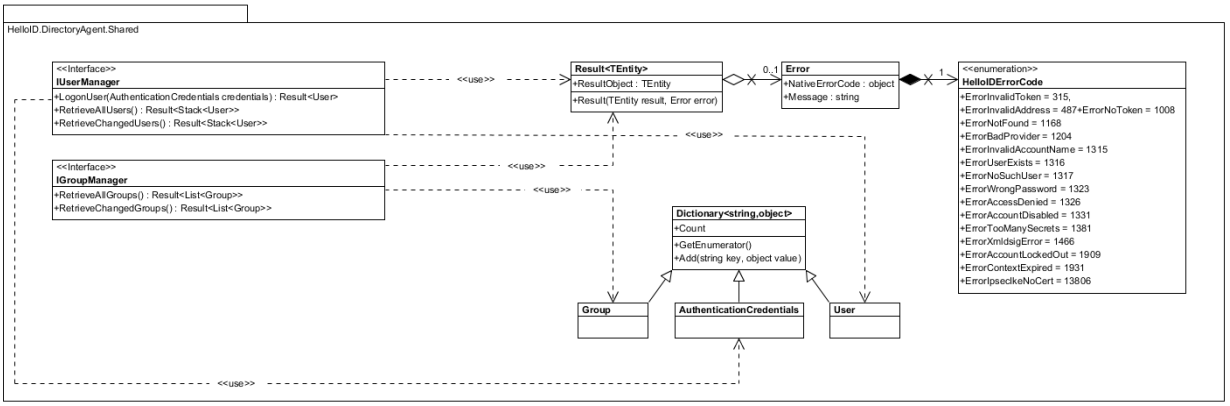
FIGUUR 33 KLASSENDIAGRAM DIRECTORY AGENT

Hierboven staat in Figuur 2 Klassendiagram Directory Agent beschreven hoe de interne structuur is opgebouwd van de Windows service de Directory Agent. Hierin zichtbaar zijn de relaties tussen de componenten en welke attributen en operaties een component heeft.

De core package is verantwoordelijk voor het draaiend houden van de Windows Service, het inladen van de identiteitsbron, het opzetten van de verbinding naar HelloID (Zie ook Sequentie diagram: verbinding opzetten) en het correct afhandelen van berichten. Bij het afhandelen van de berichten wordt indien nodig het identiteitsbronsysteem aangesproken door middel van een assembly die gebruik maakt van de publieke API uit de Shared package (zie ook Module API).

Een betere weergave van dit diagram is te zien in bijlage 1: Klassendiagram Directory Agent.

SHARED PACKAGE



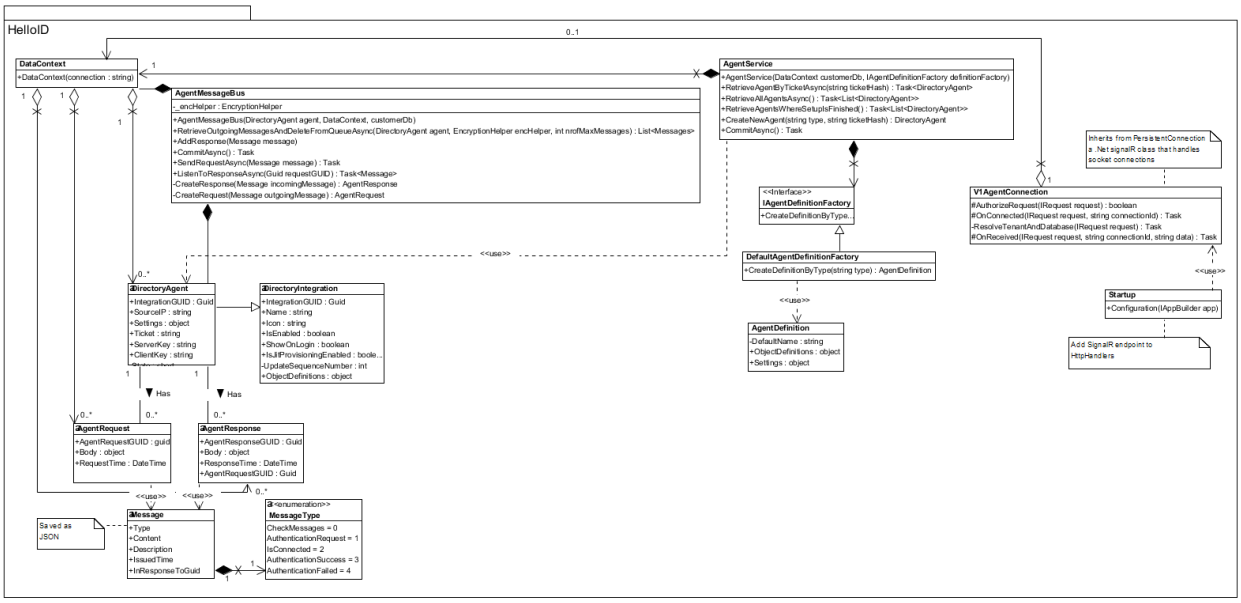
FIGUUR 34 SHARED PACKAGE

Hierboven staat in Figuur 3 Klassendiagram Shared Package Directory Agent beschreven hoe de interne structuur is opgebouwd van de Windows service de Directory Agent. Hierin zichtbaar zijn de relaties tussen de componenten en welke attributen en operaties een component heeft.

De shared package geeft informatie hoe de API van een identiteitsbron eruit moet zien (IUserManager en IGroupManager). Daarnaast biedt het publieke elementen om resultaten op te bouwen die correct door het core package kan worden verwerkt.

Een betere weergave van dit diagram is te zien in bijlage 2: Shared package.

KLASSENDIAGRAM HELLOID



FIGUUR 35 KLASSENDIAGRAM HELLOID

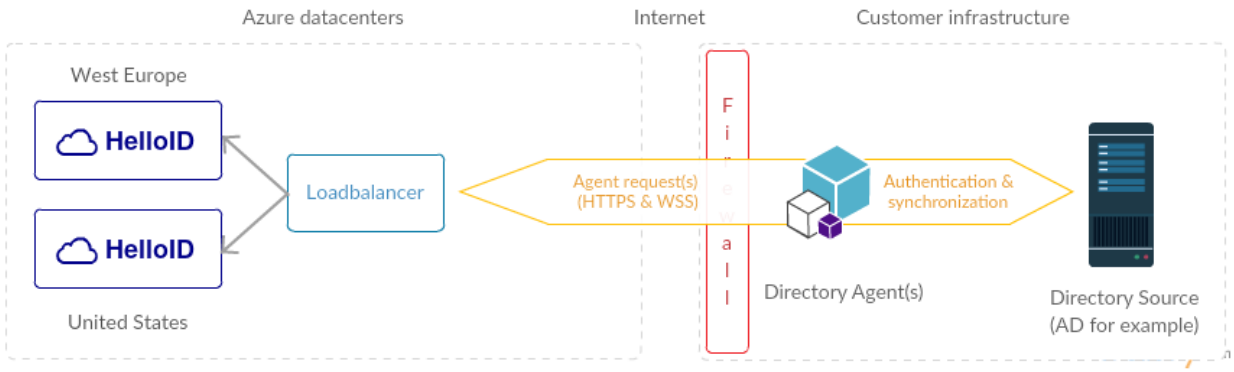
Hierboven staat in Figuur 4 Klassendiagram HelloID beschreven hoe de interne structuur is opgebouwd van de aanpassingen die zijn verricht aan HelloID om een Directory Agent aan te maken, een verbinding op te zetten met een Directory Agent en aanvragen aan te maken die door een agent kunnen worden verwerkt. In deze klassendiagram is zichtbaar welke de relaties bestaan tussen de componenten en welke attributen en operaties een component heeft.

Binnen HelloID zijn meer wijzigingen van toepassing, deze hebben vooral betrekking op het domein model van HelloID. Deze staan beschreven in het hoofdstuk Domeinmodel HelloID. Waarin ook de database wijzigingen staan beschreven.

Een betere weergave van dit diagram is te zien in bijlage 3: Klassendiagram HelloID.

COMMUNICATIE TUSSEN AGENT EN HELLOID

CONTEXT DIAGRAM



FIGUUR 36 CONTEXT DIAGRAM

Om te faciliteren dat gebruikers op HelloID kunnen inloggen met gegevens uit een ander systeem en gegevens te synchroniseren tussen dat systeem en HelloID wordt deze Directory Agent ontwikkeld. De agent moet in de omgeving van de klant worden geïnstalleerd. De agent zet een verbinding via HTTPS [11]op naar een HelloID instantie (deze wordt bepaald door de loadbalancer[12]) als de verbinding is goedgekeurd dan wordt het omgezet naar een WSS [13]verbinding. Vervolgens kan over deze verbinding berichten worden uitgewisseld.

De directory agent vraagt door middel van een polling[14] mechanisme of berichten aanwezig zijn die verwerkt moeten worden. De berichten geven aan of een authenticatie aanvraag heeft plaatsgevonden, of dat gegevens moeten worden gesynchroniseerd. Deze berichten worden verwerkt en indien nodig wordt het achterliggende bronsysteem geraadpleegd om de gegevens te controleren of op te halen. Berichten die verwerkt zijn worden teruggestuurd naar HelloID[15]. Later in dit hoofdstuk is door middel van een activity diagram uitgelegd hoe een authenticatie aanvraag wordt verwerkt.

[11] https://nl.wikipedia.org/wiki/HyperText_Transfer_Protocol_Secure
[12] https://nl.wikipedia.org/wiki/Load_balancing
[13] https://en.wikipedia.org/wiki/WebSocket#Overview
[14] https://nl.wikipedia.org/wiki/Polling_(techniek)
[15] https://www.tools4ever.nl/software/helloid-idaas-cloud-single-sign-on/

ELEMENTEN

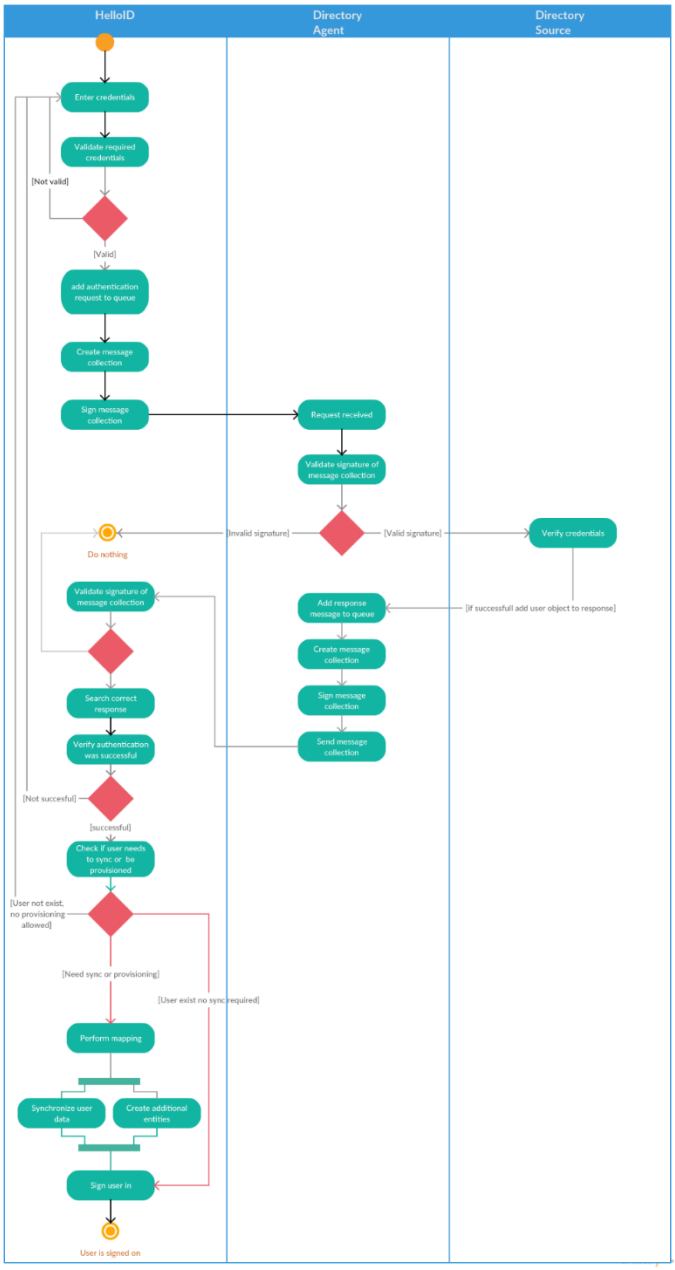
Element naam	Beschrijving
Customer infrastructuur	De omgeving van de klant. Alle elementen die binnen dit element vallen zijn in het beheer van de klant.
Azure datacenters	Alle elementen die binnen dit element vallen zijn in het beheer van Tools4ever, maar worden wel ge-host in de cloud omgeving van Microsoft: Azure[16].
Directory Agent(s)	Dit systeem is door Tools4ever ontwikkeld. Het draagt zorg voor het opzetten van een verbinding vanuit de omgeving van de klant naar een HelloID instantie. Over deze verbinding worden authenticatie aanvragen verwerkt en synchronisatie acties uitgevoerd tussen een bronsysteem en HelloID.
Directory Source	Dit is het bronsysteem van de klant. Zoals in het diagram staat kan dit Active Directory zijn (AD), echter kunnen dit ook andere bronsystemen zijn zoals SQL of E-SSOM.
Loadbalancer	Dit is een component dat in Azure wordt ge-host om werk te verdelen tussen verschillende computers en processen. Bij Tools4ever wordt hierin een keuze gemaakt o.b.v. de geografische locatie van de server of computer waar de Directory Agent op is geïnstalleerd.
HelloID	HelloID is de cloud gebasseerde Single Sign On [17]oplossing van Tools4ever. Dit is het doelsysteem van de Directory Agent. Hier worden bijvoorbeeld de authenticatie vragen aangemaakt en door de openstaande verbinding door de agent verwerkt en geverifieerd tegen het bronsysteem.
Firewall	Een firewall is een systeem dat het netwerk van de klant kan beschermen tegen ongewenst verkeer over het netwerk en

[16] https://nl.wikipedia.org/wiki/Windows_Azure

[17] https://www.tools4ever.nl/software/enterprise-sso-single-sign-on-authentication-management/wat-is-single-sign-on/

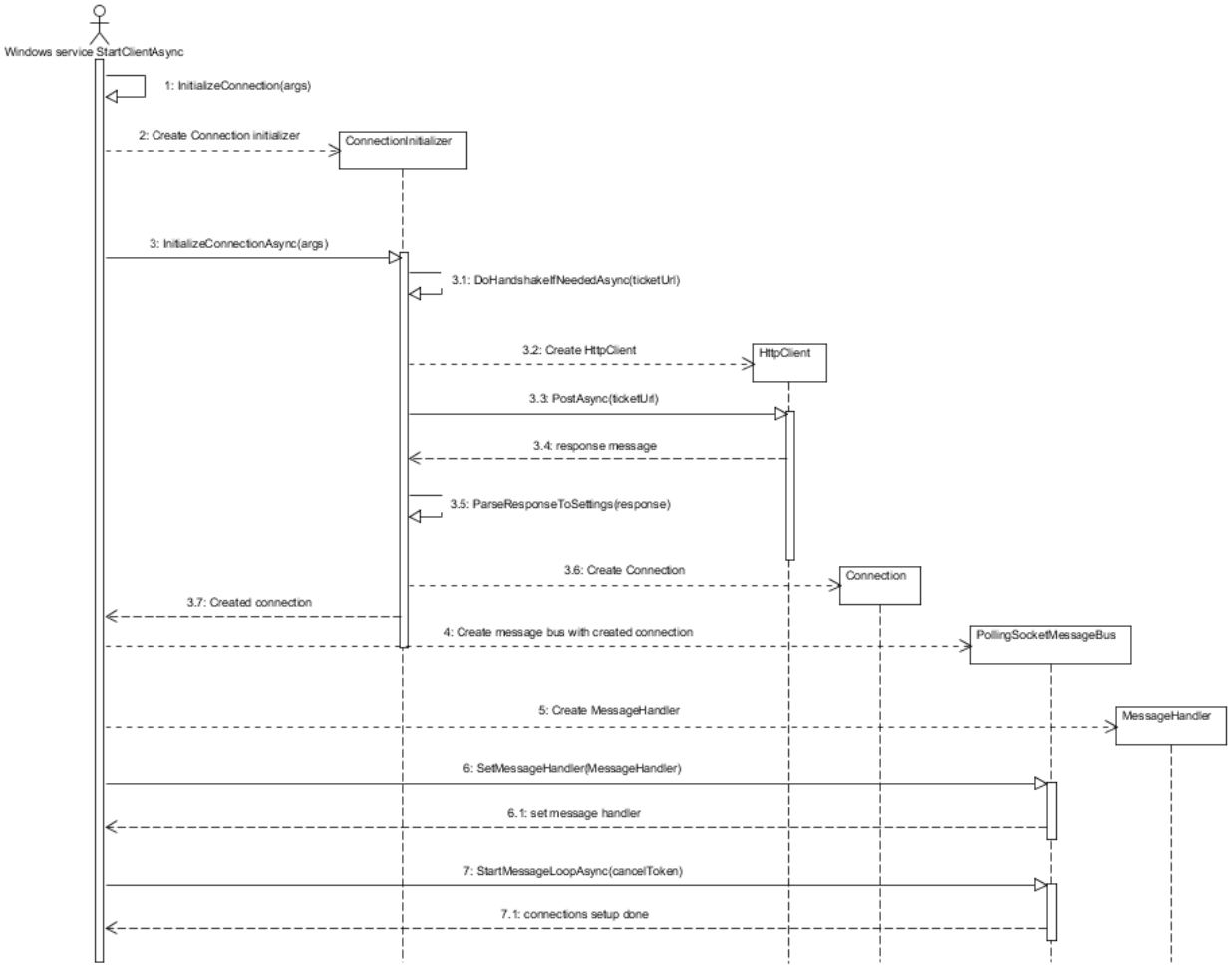
Element naam	Beschrijving
	misbruik van buitenaf.

ACTIVITY DIAGRAM: FLOW VAN EEN AUTHENTICATIE AANVRAAG



FIGUUR 37 ACTIVITY DIARAM: FLOW VAN AUTHENTICATIE

SEQUENTIE DIAGRAM: VERBINDING OPZETTEN

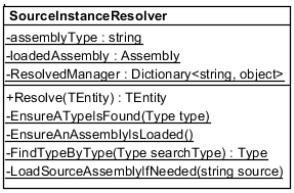


FIGUUR 38 VERBINDING OPZETTEN

Hierboven wordt weergegeven welke stappen worden ondernomen in de directory agent om een verbinding op te zetten. Eerst wordt een handshake (kennismaking tussen de systemen) moet plaatsvinden. Hiervoor wordt een HttpClient geïnstantieerd en een POST request naar HelloID gestuurd. Dit request heeft een informatie over de klant, server en agent. Deze gegevens worden verwerkt en opgeslagen. Een nieuw connection object wordt aangemaakt een doorgegeven aan de WindowsService. Dit connection object wordt vervolgens gebruikt in de MessageBus, die dan gelijk de message loop start.

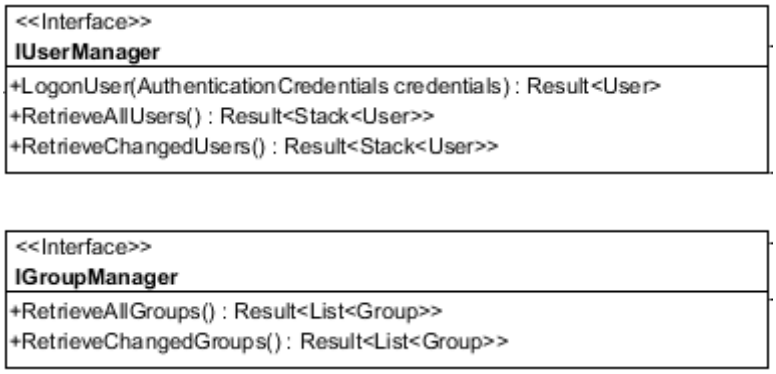
MODULE API

De Directory Agent is modulair opgezet om ervoor te zorgen dat voor elke identiteitsbron apart een assembly kan worden geschreven en vervolgens ingeladen in de Agent. Bijvoorbeeld een Directory Agent wordt geconfigureerd voor Active Directory, bij het starten van de agent wordt de juiste DLL opgezocht en ingeladen zodat de implementatie van de DLL kan worden gebruikt om de bron te raadplegen.



FIGUUR 39 SOURCEINSTANCERESOLVER

Voor dit proces is de klasse SourceInstanceResolver verantwoordelijk (deze is terug te vinden in figuur 3). Deze klasse leest uit de bin folder in de root folder van de agent alle assemblies die voldoen aan de naamgeving: HelloID.DirectoryAgent.Sources.<SourceType>. Als de Resolve method uit wordt gevoerd dan zoekt hij alleen klassen die overerven van interfaces uit de shared package, de shared package bevat alle publieke elementen om een identiteitsbron te maken die door de agent kan worden gebruikt.



FIGUUR 40 PUBLIEKE API SHARED

MODULE RESOLVEMENT

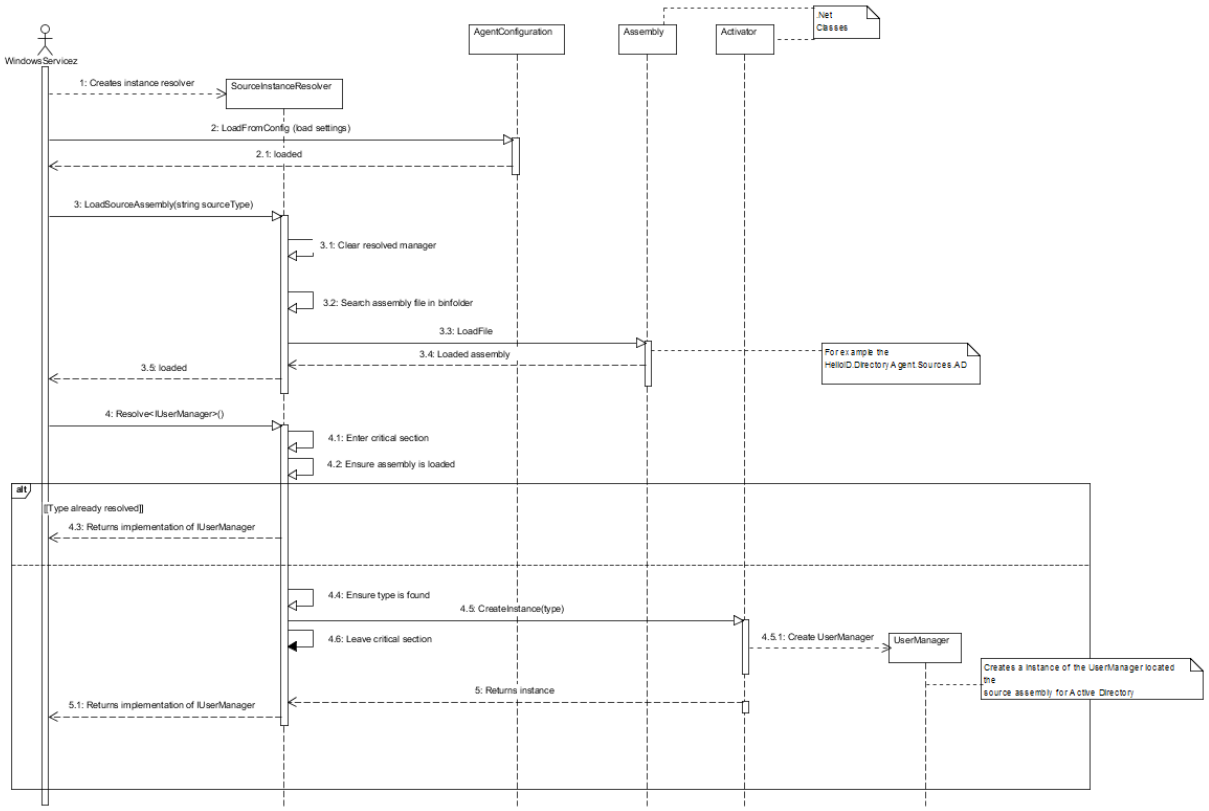
Om ervoor te zorgen dat de juiste module wordt ingeladen moet de module (lees assembly) aan de volgende eisen voldoen:

- Het volgt de naamgeving HelloID.DirectoryAgent.Sources.<SourceType>, waar source type de unieke identifier is. Bijvoorbeeld voor de Active Directory: HelloID.DirectoryAgent.Sources.AD;
- Het bevat een implementatie voor IUserManager;
- Het bevat een implementatie voor IGroupManager;

Om te bepalen welke module wordt ingeladen moet dit in de settings zijn opgenomen. De setting: ServerSettings>SourceType wordt bij de installatie van de agent via het HelloID-portaal doorgegeven aan de agent in JSON format. Het is mogelijk om deze ook handmatig aan te passen. De agent settings staan in het bestand: agent.config. Voorbeeld van setting:

```
"ServerSettings": {
       "SourceType": "AD",
}
```

SEQUENCTIE DIAGRAM: RESOLVEN VAN IUSERMANAGER

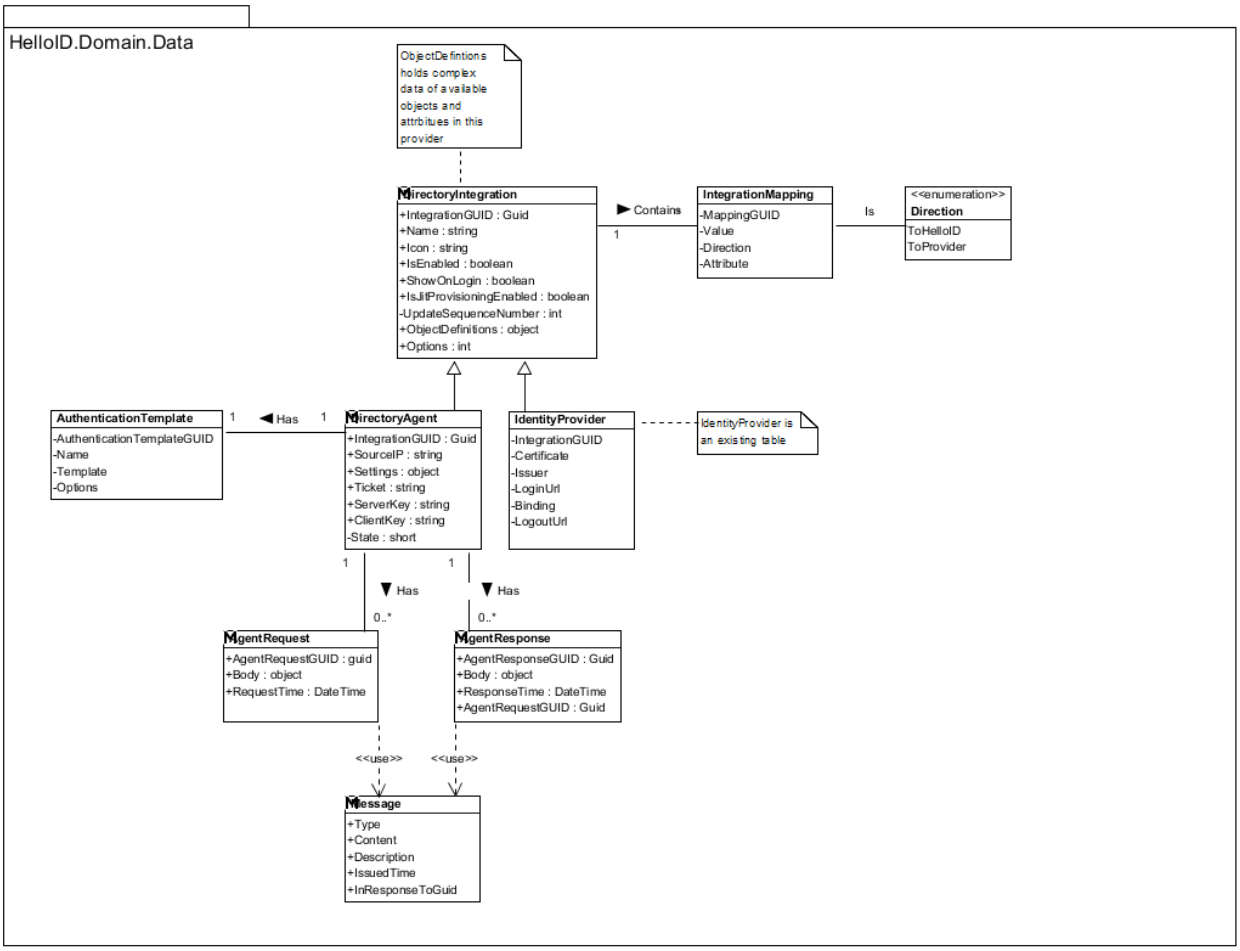


FIGUUR 41 RESOLVING IUSERMANAGER

In het bovenstaande voorbeeld is te zien hoe een implementatie voor de IUserManager van een module wordt ingeladen. De Windows Service instantieert de SourceInstanceResolver. Dan laad de service de configuratie in, deze configuratie bevat een type dat aangeeft welke source assembly moet worden gebruikt als een klasse wordt opgezocht en ingeladen. De service roept de methode load source assembly en deze zoekt de assembly op in de juiste folder en laad deze in. De service roept (dit kan overigens op elke willekeurige plek in de code zijn) de methode Resolve aan met de aangegeven type van de interface. Deze methode stapt een critical section in (zorgt ervoor dat andere threads dit niet tegelijktijdig kunnen uitvoeren en moeten wachten). Dan zorgt de methode ervoor dat alles correct is ingeladen en roept de Activator aan om een instantie aan te maken en geeft deze terug aan de service.

DOMEINMODEL HELLOID

ANALYSE KLASSENDIAGRAM



FIGUUR 42 ANALYSE KLASSENDIAGRAM HELLOID

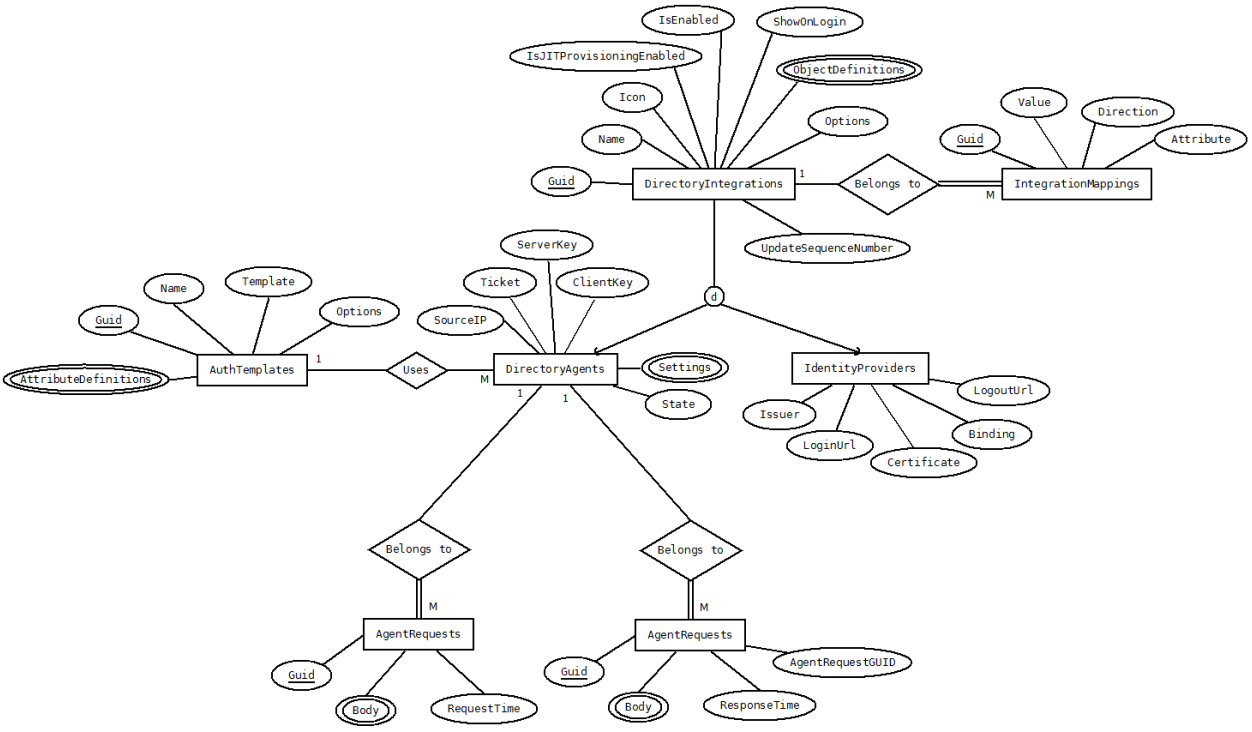
BESCHRIJVING VAN KLASSEN

Klasse/Enumeratie	Omschrijving
AuthenticationTemplate	Deze klasse bevat informatie over een inlogscherm. Hierin wordt de template (HTML[18]) van een inlogscherm opgeslagen. Standaard zal HelloID twee templates opnemen. Een die is afgeleid van de standaard van HelloID, maar ook een die het inlogscherm van Windows imiteert.
DirectoryIntegration	DirectoryIntegration is een basis klasse voor de DirectoryAgent en IdentityProvider. Die klassen dienen als provider van gegevens. Op dit niveau is opgenomen of de provider mag worden gezien als authentication provider, wat aangeeft dat gebruikers via de provider mogen inloggen op het HelloID portaal.
DirectoryAgent	De klasse DirectoryAgent bevat gegevens over de bron van gegevens (Directory binnen deze context) en gegevens om met de agent die bij de klant geïnstalleerd staat te communiceren. Indien het inlogscherm afwijkt van de standaard in HelloID kan een eigen authentication template worden gebruikt.
IdentityProvider	Dit is al een bestaande klasse binnen HelloID. Het representeert de leveranciers van identiteit door middel van protocollen zoals SAML[19][20].
IntegrationMapping	Deze klasse dient voor het vastleggen van hoe waarden vanuit de provider naar HelloID worden gecommuniceerd en vice versa. Om de richting aan te duiden wordt gebruik gemaakt van de Direction enumeratie. Mappings vallen onder een provider.

[18] https://nl.wikipedia.org/wiki/HyperText_Markup_Language
[19] https://nl.wikipedia.org/wiki/Security_Assertion_Markup_Language
[20] https://www.tools4ever.nl/blog/2016/saml-wat-is-het-en-wat-doet-het/

Klasse/Enumeratie	Omschrijving
Direction	De volgende waarden zijn beschikbaar in deze enumeratie: ToHelloID(0), ToProvider(1)
AgentRequest	In deze klasse worden aanvragen die naar de agent moeten worden doorgezet vastgelegd. De aanvragen worden maar tijdelijk opgeslagen, na lezen worden deze aanvragen direct verwijderd.
AgentResponse	In deze klasse worden de reacties vanuit de agent op aanvragen vastgelegd. Wanneer een response aan de serverkant wordt gelezen wordt deze direct verwijderd. De AgentRequestGUID is de identifier die aangeeft waar reactie op wordt gegeven. Deze identifier is alleen tijdelijk bij de server bekend.
Message	Deze klasse houdt het bericht vast wat door een request of response wordt meegegeven.

ENHANCED ENTITY-RELATIONSHIP MODEL (EERM)



FIGUUR 43 EER MODEL

RELATIONEEL REPRESENTATIEMODEL (RRM)

AuthTemplates(AuthTemplateGUID, Name, Template, AttributeDefinitions, Options)

DirectoryIntegrations(IntegrationGUID, Name, Icon, IsJITProvisioningEnabled, IsEnabled, ShowOnLogin, ObjectDefinitions, UpdateSequenceNumber,Options)

DirectoryAgents(*IntegrationGUID*, SourceIP, Settings, Ticket, ServerKey, ClientKey, State, *AuthTemplateGUID*)

FK(ProviderGUID) verwijst naar Providers NULL **niet** toegestaan.

FK(AuthTemplateGUID) verwijst naar AuthenticationTemplates NULL toegestaan.

IdentityProviders(*IntegrationGUID*, Issuer, LoginUrl, Certificate, Binding, LogoutUrl)

FK(IDPGUID) verwijst naar Providers NULL **niet** toegestaan.

IntegrationMappings(MappingGUID, Value, Direction, Attribute*, Options, IntegrationGUID*)

CONSTRAINT(Direction) 0 = ToHelloID OF 1 = ToProvider, NULL niet toegestaan

FK(ProviderGUID) verwijst naar Providers NULL **niet** toegestaan

---

AgentRequests(*AgentRequestGUID*, Message, RequestTime, *DirectoryAgentGUID*)

FK(DirectoryAgentGUID) verwijst naar AgentProviders NULL niet toegestaan.

AgentResponses(*AgentResponseGUID*, Message, ResponseTime, AgentRequestGUID *DirectoryAgentGUID*)

FK(DirectoryAgentGUID) verwijst naar AgentProviders NULL **niet** toegestaan.

RELATIONEEL IMPLEMENTATIE MODEL (RIM)

```
CREATE TABLE [AuthTemplates]
(
	[AuthTemplateGUID] uniqueidentifier NOT NULL,
	[Name] nvarchar(128) NOT NULL,
	[Template] nvarchar(MAX) NOT NULL,
	[Options] int NOT NULL DEFAULT(0), --SystemDefined(1)
	PRIMARY KEY([AuthTemplateGUID])
)
GO

CREATE TABLE [DirectoryIntegrations]
(
	[IntegrationGUID] uniqueidentifier NOT NULL,
	[Name] nvarchar(128) NOT NULL,
	[Icon] nvarchar(MAX) NULL,
	[IsEnabled] bit NOT NULL DEFAULT(0),
	[ShowOnLogin] bit NOT NULL DEFAULT(0),
	[IsJITProvisioningEnabled] bit NOT NULL DEFAULT(0),
	[UpdateSequenceNumber] int NULL CHECK ([UpdateSequenceNumber] IS NULL OR
[UpdateSequenceNumber] >= 0), --greater or equal to zero
	[ObjectDefinitions] nvarchar(MAX) NULL,
	[Options] int NOT NULL DEFAULT(0), --Enabled(1) --ShowOnLogin(2) --
IsJITProvisioningEnabled(4)
	PRIMARY KEY([IntegrationGUID])
)
GO

CREATE TABLE [DirectoryAgents]
(
	[IntegrationGUID] uniqueidentifier NOT NULL,
	[SourceIP] nvarchar(2000) NULL,
	[Settings] nvarchar(MAX) NULL,
    [Ticket] nvarchar(128) NOT NULL,
	[ServerKey]  nvarchar(max) NULL,
	[ClientKey]  nvarchar(max) NULL,
	[State] smallint NOT NULL DEFAULT(0), --HasFinishedConnection(1) --
HasFinishedSetup(2)
	[AuthTemplateGUID] uniqueidentifier NULL,
	PRIMARY KEY([IntegrationGUID])
)
GO

ALTER TABLE [DirectoryAgents]		ADD CONSTRAINT [FK_AgentProviders_TPT_Providers]
	FOREIGN KEY([IntegrationGUID])	REFERENCES
[DirectoryIntegrations]([IntegrationGUID])			ON UPDATE CASCADE ON
DELETE CASCADE
GO
```

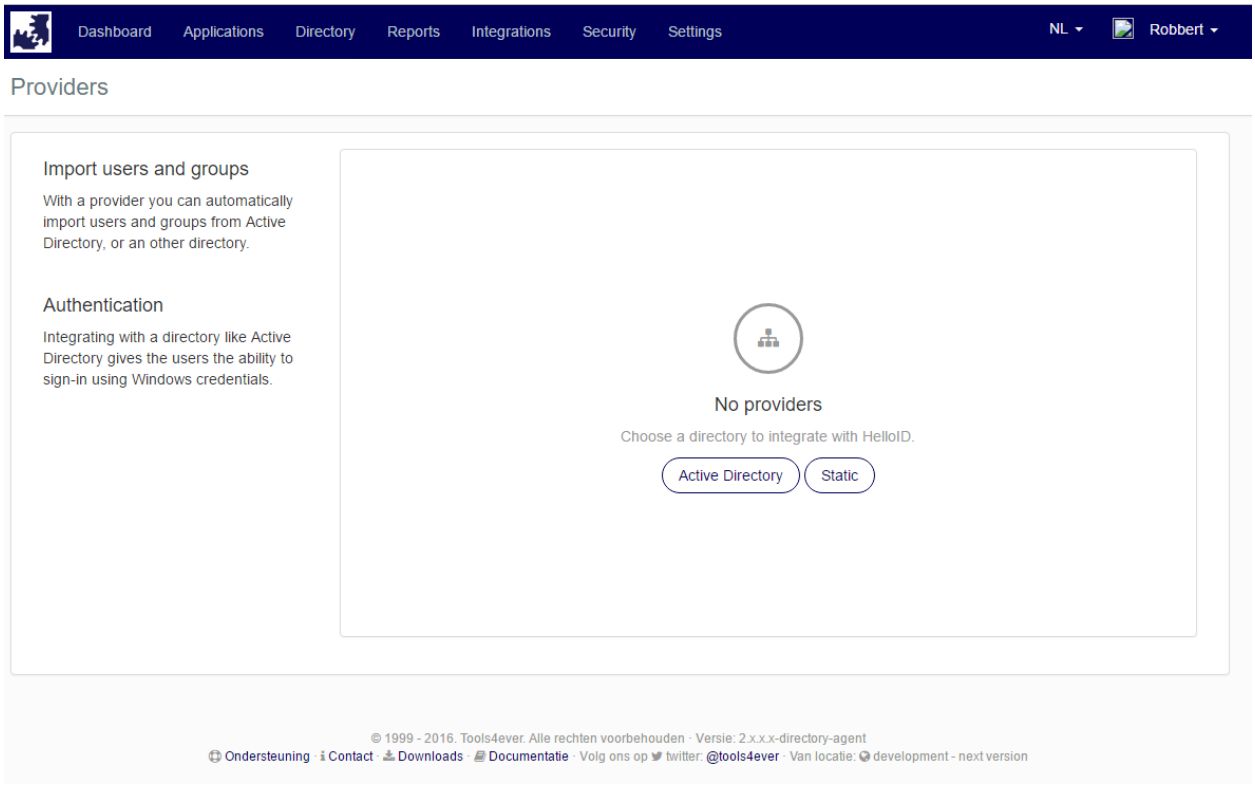
```
ALTER TABLE [DirectoryAgents]     ADD CONSTRAINT [FK_AgentProviders_AuthTemplate] FOREIGN
KEY([AuthTemplateGUID])                 REFERENCES [AuthTemplates]([AuthTemplateGUID])
       ON UPDATE CASCADE ON DELETE CASCADE
GO

CREATE TABLE [IdentityProviders]
(
       [IntegrationGUID] uniqueidentifier NOT NULL,
       [Issuer] nvarchar(2000) NULL,
       [LoginUrl] nvarchar(1000) NULL,
       [Certificate] nvarchar(255)  NULL,
       [Binding] nvarchar(255) NULL,
       [LogoutUrl] nvarchar(1000) NULL,
       PRIMARY KEY([IntegrationGUID])
)
GO

ALTER TABLE [IdentityProviders]          ADD CONSTRAINT
[FK_IdentityProviders_TPT_Providers]     FOREIGN KEY([IntegrationGUID])    REFERENCES
[DirectoryIntegrations]([IntegrationGUID])                          ON UPDATE CASCADE ON
DELETE CASCADE


CREATE TABLE [IntegrationMappings]
(
       [IntegrationMappingGUID] uniqueidentifier NOT NULL,
       [Value] nvarchar(512) NULL,
       [Direction] smallint NOT NULL,
       [Attribute] nvarchar(512) NULL,
       [Options] int NULL DEFAULT(0),
       [ProviderGUID] uniqueidentifier NOT NULL,
       PRIMARY KEY([IntegrationMappingGUID]),
       CHECK([Direction] = 0 OR [Direction] = 1),
)
GO

ALTER TABLE [IntegrationMappings]        ADD CONSTRAINT [FK_Mappings_Provider]    FOREIGN
KEY([ProviderGUID]) REFERENCES [DirectoryIntegrations]([IntegrationGUID])
       ON UPDATE CASCADE ON DELETE CASCADE
GO


CREATE TABLE [AgentRequests]
(
       [AgentRequestGUID] uniqueidentifier NOT NULL,
       [Body]  nvarchar(max) NOT NULL,
       [RequestTime]  datetime NOT NULL,
       [DirectoryAgentGUID] uniqueidentifier NOT NULL,
       PRIMARY KEY([AgentRequestGUID])
)
GO
```

```
ALTER TABLE [AgentRequests]       ADD CONSTRAINT [FK_AgentRequest_DirectoryAgent] FOREIGN
KEY([DirectoryAgentGUID])  REFERENCES [DirectoryAgents]([IntegrationGUID])
       ON UPDATE CASCADE ON DELETE CASCADE
GO

CREATE TABLE [AgentResponse]
(
       [AgentResponseGUID] uniqueidentifier NOT NULL,
       [Body]  nvarchar(max) NOT NULL,
       [ResponseTime]  datetime NOT NULL,
       [DirectoryAgentGUID] uniqueidentifier NOT NULL,
       [AgentRequestGUID] uniqueidentifier NOT NULL,
       PRIMARY KEY([AgentResponseGUID])
)
GO

ALTER TABLE [AgentResponse]       ADD CONSTRAINT [FK_AgentResponse_DirectoryAgent]
       FOREIGN KEY([DirectoryAgentGUID]) REFERENCES [DirectoryAgents]([IntegrationGUID])
                     ON UPDATE CASCADE ON DELETE CASCADE
```

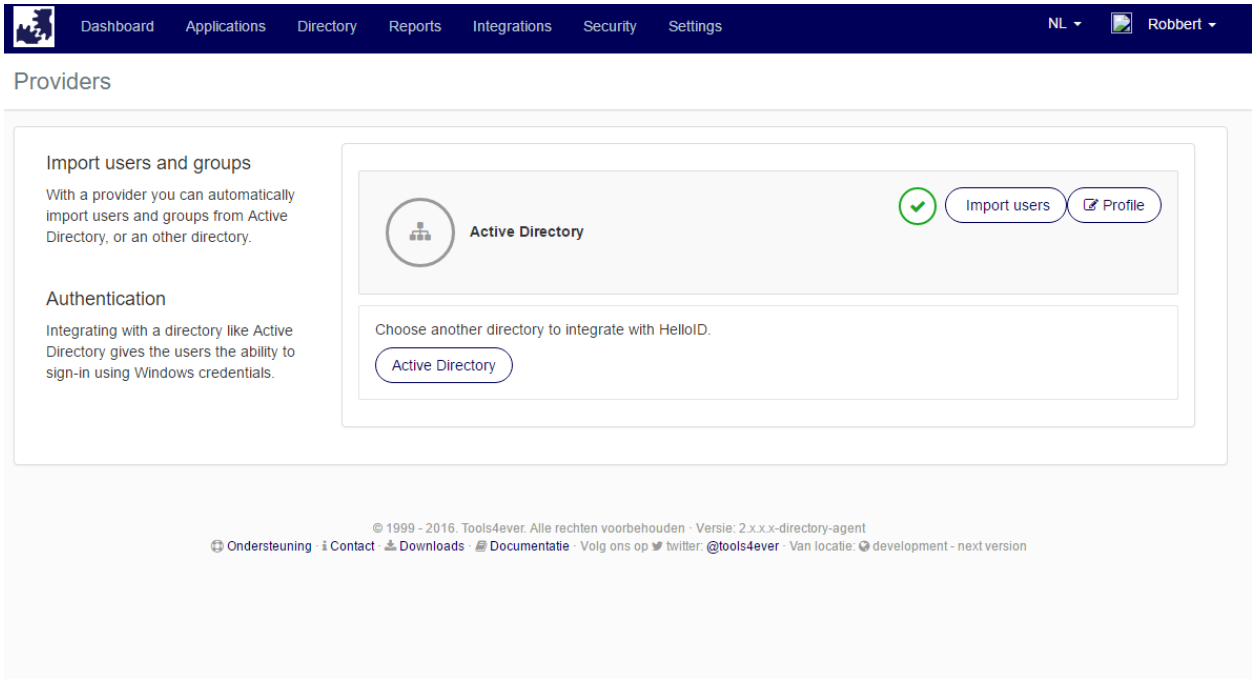
GRAFISCHE WEERGAVE IN HELLOID

AGENT OVERVIEW

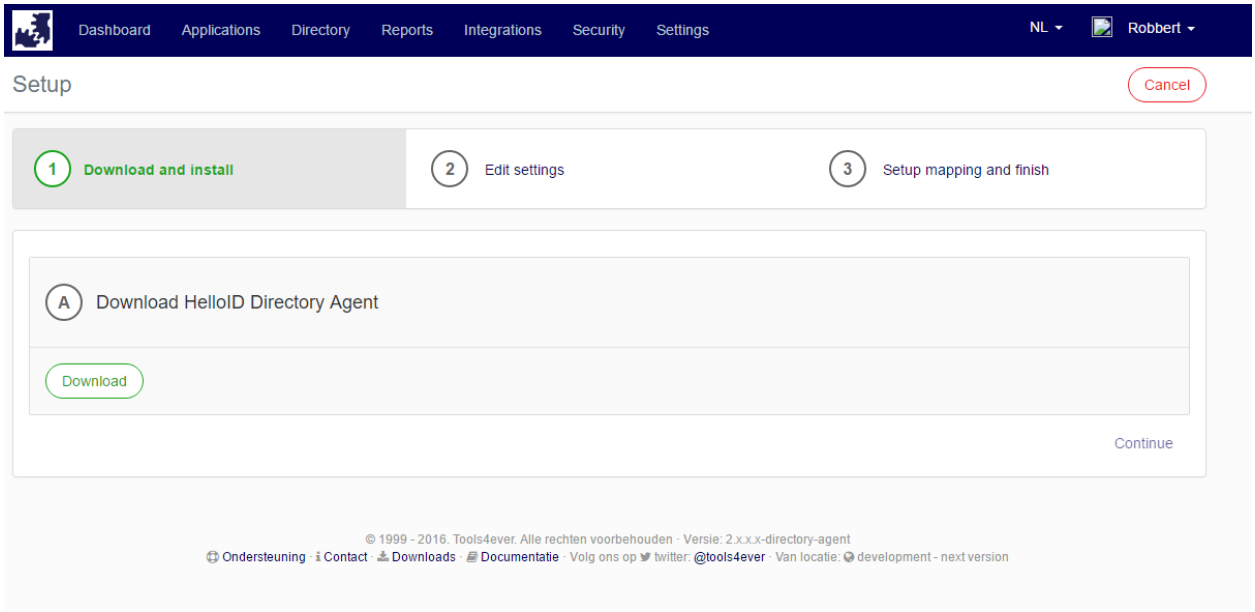
GEEN PROVIDERS AANWEZIG

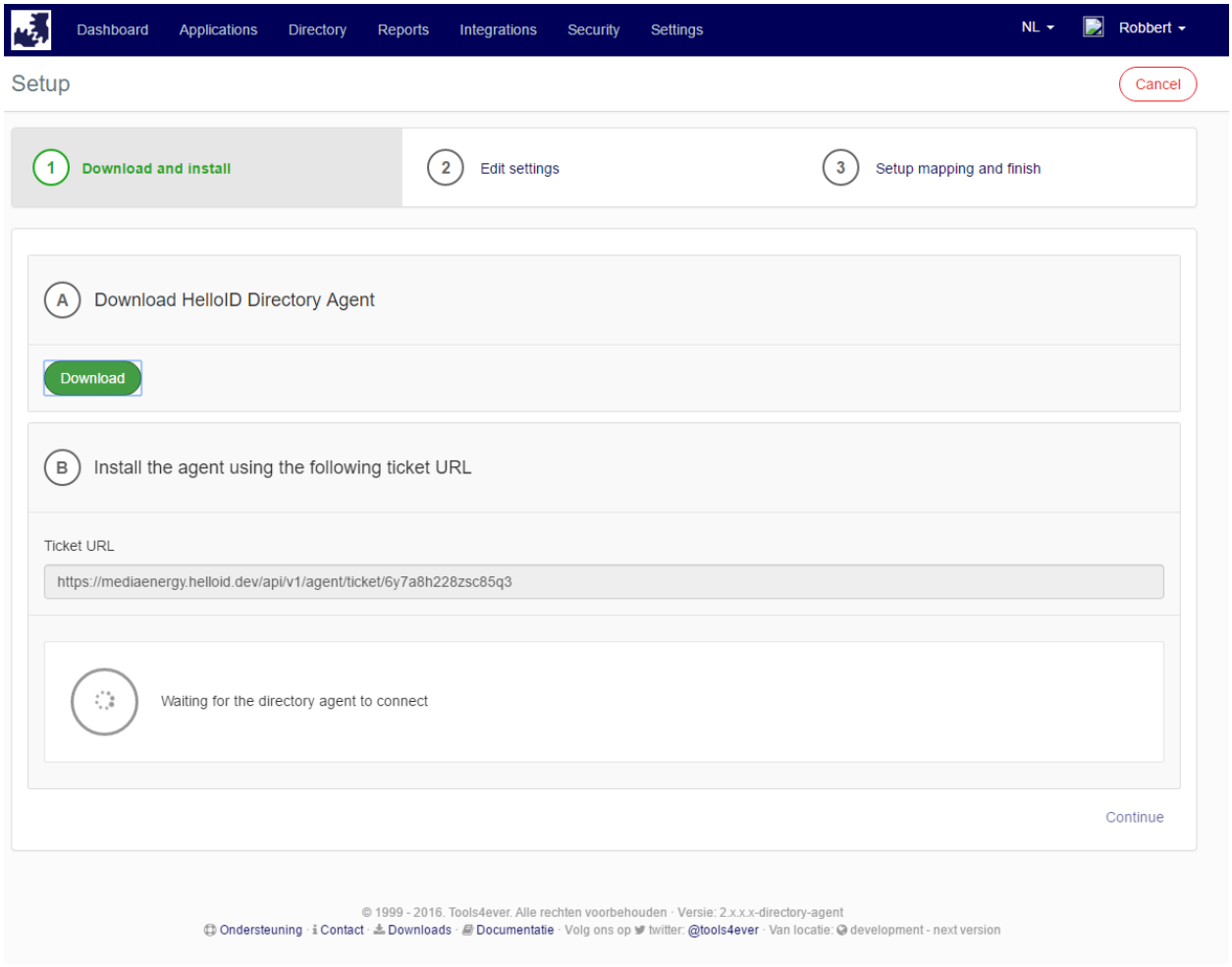


PROVIDER AANWEZIG



AGENT WIZARD




Dashboard
Applications
Directory
Reports
Integrations
Security
Settings
NL
Robbert
Setup
Cancel
1 Download and install
2 Edit settings
3 Setup mapping and finish
A Download HelloID Directory Agent
Download
B Install the agent using the following ticket URL
Ticket URL
https://mediaenergy.helloid.dev/api/v1/agent/ticket/6y7a8h228zsc85q3
Waiting for the directory agent to connect
Continue
© 1999 - 2016. Tools4ever. Alle rechten voorbehouden · Versie: 2.x.x.x-directory-agent
Ondersteuning · Contact · Downloads · Documentatie · Volg ons op twitter: @tools4ever · Van locatie: development - next version

BIJLAGE H: TESTPLAN

VERSIEBEHEER

Versie	Auteur	Omschrijving
1.0	Robbert Brussaard	Eerste versie opgezet.
1.1	Robbert Brussaard	Test basis en test procedures toegevoegd.
1.2	Robbert Brussaard	Structuur aangepast.

VERZENDLIJST

Wie	Contactgegevens
George Bakker	G.Bakker@tools4ever.com
Aad Lutgert	A.Lutgert@tool4ever.com
Robbert Brussaard	R.Brussaard@tools4ever.co
Arie Toet	A.J.Toet@hhs.nl
Paul Smit	P.R.Smit@hhs.nl

INTRODUCTIE

Het doel van dit document is om alle betrokkenen bij het project een overzicht te bieden van de soorten testen, activiteiten en taken omtrent het testen van de HelloID Directory Agent.

DOMEIN

De onderstaande onderdelen vallen onder het domein van dit testplan:

- Het verzekeren van code kwaliteit;
- Het schrijven en uitvoeren van unit tests;
- Het schrijven en uitvoeren van integratie tests;
- Het elimineren en verminderen van bugs;
- Het testen van arbitraire input in de userinterface;

TEST BASIS

Hieronder wordt de testbasis beschreven. De testbasis is een totaalverzameling van documenten, systeembeschrijvingen waaraan het testobject moet voldoen.

DOCUMENTEN

Naam	Versie	Datum	Omschrijving
Requirementsrapport	1.1	17-09-2016	
Ontwerpdocumentatie	1.1	23-10-2016	

APPLICATIES

Naam	Versie	Datum	Omschrijving
HelloID Directory Agent	1.0		Het te ontwikkelen systeem
HelloID	2.X		Alleen het toegevoegde dat betrekking heeft op de directory agent.

VERANTWOORDELIJKHEDEN

Persoon	Rol	Verantwoordelijkheid
Robbert Brussaard	Programmeur	Opstellen van module en integratie testen, oplossen van bugs en fouten, rapportage maken van white-box tests.
Aad Lutgert	Tester	Uitvoeren van black-box tests, melden van fouten in Jira.

AANPAK

WHITE-BOX TESTS

Zowel de module tests als de integratie tests, deze tests worden geïmplementeerd met kennis van het gehele systeem. Het doel van deze tests is het testen van de interne structuren en werking. Deze white-box tests zullen continue tijdens het ontwikkelen van het project worden uitgevoerd. Dit omdat gebruikt wordt gemaakt van Test Driven Development[21].

MODULE TESTS

Module tests zijn kleine stukken code die onafhankelijke stukken programmatuur testen. Deze tests mogen geen externe bronnen of andere componenten aanspreken. Dit om te zorgen dat zij de correcte van de werking en structuur van een individueel component testen. Bij elke wijziging kunnen deze tests opnieuw worden uitgevoerd om eventuele problemen (bugs) te detecteren en te waarborgen of functionaliteit blijft werken zoals dat bedoeld is.

Microsoft Visual Studio 2013 en 15 worden gebruikt om deze tests op te stellen. Hierbij wordt gebruik gemaakt van de standaard unit test library van Microsoft. Hiernaast wordt de extensies Reshaper en DotCover van de partij JetBrains gebruikt. Met Reshaper wordt de programmeur geholpen om programmatuur te herstructureren, om hiermee de leesbaarheid en onderhoudbaarheid te verbeteren (Refactoring[22]). DotCover wordt ingezet om rapportage van de module testen te genereren. Deze rapporten worden in het versiebeheer gearchiveerd.

De module testen zullen in een apart project worden ondergebracht. Dit om ervoor te zorgen dat de tests niet in de productie-code terechtkomen.

INTEGRATIE TESTS

Net als module tests zijn integratie tests kleine stukken code, maar in tegendeel van module tests, testen zij juist de samenwerking tussen individuele componenten en het gebruik van externe bronnen. Wederom wordt gecontroleerd op correctheid van de werking, maar hiernaast hebben de tests als doel: het verifiëren van de betrouwbaarheid van de samenwerking tussen componenten.

Deze tests zullen op dezelfde manier worden opgesteld als de module tests. Ook de rapportage zal op dezelfde manier verlopen. Deze tests zullen ook in een apart project worden ondergebracht, gescheiden van de module tests.

[21] https://nl.wikipedia.org/wiki/Test-driven_development
[22] https://nl.wikipedia.org/wiki/Refactoren

BLACK-BOX TESTS

Tijdens het ontwikkelen van dit project wordt ook gebruik gemaakt van een black-box test. Het belangrijkste doel van deze tests in dit project is het testen op arbitraire input in de userinterface. Wij werken in een Scrum-team en een functionele tester zal wanneer een user story (requirement) gereed is deze testen.

ERROR GUESSING

Zoals de naam al impliceert: error guessing is het gissen naar fouten. De waarde hiervan ligt vooral in het onverwachte. Op basis van ervaring gaat de tester tijdens de usability tests op zoek naar foutgevoelige plekken in het systeem en bedenkt passende tests hiervoor tijdens de testuitvoering. Net als de usability tests zullen fouten worden gemeld in Jira en wordt geen andere rapportage opgesteld.

BIJLAGE I: VOORBEELD TESTRAPPORTAGE

12-12-2016 tests for ad source.htm

Tests (58 tests) [0:12.697] Success

HelloID.DirectoryAgent.Sources.AD.Integration.Tests (46 tests) [0:11.489] Success

HelloID.DirectoryAgent.Sources.AD.Tests (46 tests) [0:11.489] Success

Internal (20 tests) [0:00.021] Success

LdapErrorTests (5 tests) [0:00.000] Success

LdapError_ErrorCode_CorrectErrorCodesOfAccessDenied [0:00.000] Success

LdapError_ErrorCode_CorrectErrorCodesOfAccountDisabled [0:00.000] Success

LdapError_ErrorCode_CorrectErrorCodesOfAccountLockedOut [0:00.000] Success

LdapError_ErrorCode_CorrectErrorCodesOfNoSuchUser [0:00.000] Success

LdapError_ErrorCode_UknownErrorShouldGiveAccessDenied [0:00.000] Success

LdapUtilsTests (7 tests) [0:00.018] Success

LdapUtils_BindLdapString_WithLdapStringShouldGetLdapString [0:00.000] Success

LdapUtils_BindLdapString_WithOutLdapStringShouldRaiseException [0:00.001] Success

LdapUtils_GetDefaultDnLdapString_ShouldNotBeEmptyAndShouldContainDCAndLdapPrefix [0:00.005] Success

LdapUtils_GetLdapPrefix_WithServerShouldGetLdapAndServer [0:00.000] Success

LdapUtils_GetLdapPrefix_WithoutServerShouldGetLdapOnly [0:00.000] Success

LdapUtils_NamingContext_ShouldReallyBeDefaultNamingContext [0:00.000] Success

LdapUtils_SearchDnFromRoot_ShouldNotBeEmptyAndShouldContainDC [0:00.012] Success

SuffixListTests (4 tests) [0:00.001] Success

SuffixList_ShouldAddItemsToListThatAreValid [0:00.000] Success

SuffixList_ShouldGetItemByNameFirstPartOfSuffix [0:00.000] Success

SuffixList_ShouldNotAddItemsToListThatAreInvalid [0:00.000] Success

SuffixList_ShouldNotFailWhenInstantiatingWithOutSuffixList [0:00.000] Success

SuffixTests (4 tests) [0:00.001] Success

Suffix_Empty_EmptyMeansNoNameAndNoDomainName [0:00.000] Success

Suffix_Equals_WithInvalidValuesShouldNotMatchByNameAndDomainName [0:00.000] Success

Suffix_Equals_WithValidValuesShouldMatchByNameAndDomainName [0:00.000] Success

Suffix_Equals_WithValidValuesShouldMatchByNameAndDomainNameIgnoredByCase [0:00.000] Success

GroupManagerTests (1 test) [0:00.019] Success

GroupManager_RetrieveAllGroups_ShouldBeMoreThenZero [0:00.019] Success

LdapSpecificUserSearchSetTests (15 tests) [0:00.001] Success

LdapSpecificUserSearchSet_Constructor_StructMustBeEmptyWhenUsingNullParams [0:00.000] Success

LdapSpecificUserSearchSet_GetDomainName_GetCorrectByDefaultDomainName [0:00.000] Success

LdapSpecificUserSearchSet_GetDomainName_GetCorrectBySuffixList [0:00.000] Success

LdapSpecificUserSearchSet_GetDomainName_GetCorrectBySuffixListWithOutDcSpec [0:00.000] Success

LdapSpecificUserSearchSet_GetDomainName_GetCorrectByUpn [0:00.000] Success

LdapSpecificUserSearchSet_GetLdapBindString_GetCorrectByDefaultDomainName [0:00.000] Success

LdapSpecificUserSearchSet_GetLdapBindString_GetCorrectBySuffixList [0:00.000] Success

LdapSpecificUserSearchSet_GetLdapBindString_GetCorrectByUpn [0:00.000] Success

LdapSpecificUserSearchSet_GetUserName_GetCorrectByDefaultDomainName [0:00.000] Success

LdapSpecificUserSearchSet_GetUserName_GetCorrectByDefaultDomainNameAndPrefix [0:00.000] Success

LdapSpecificUserSearchSet_GetUserName_GetCorrectBySuffixList_ShouldBeSamAccount Name [0:00.000] Success

LdapSpecificUserSearchSet_GetUserName_GetCorrectByUpn [0:00.000] Success

LdapSpecificUserSearchSet_GetUserSearchFilter_GetCorrectByDefaultDomainName [0:00.000] Success

LdapSpecificUserSearchSet_GetUserSearchFilter_GetCorrectBySuffixList [0:00.000] Success

LdapSpecificUserSearchSet_GetUserSearchFilter_GetCorrectByUpn [0:00.000] Success

UserManagerTests (10 tests) [0:11.447] Success

UserManager_LogOnUser_GetErrorForLoginWithEmptyValues [0:00.000] Success

UserManager_LogOnUser_GetErrorForLoginWithoutdomain [0:00.042] Success

UserManager_LogOnUser_GetErrorWhenUserIsInvalidUserWithUpnLogin [0:00.026] Success

UserManager_LogOnUser_GetUserObjectWhenUserIsAValidUserWithDefaultLogin [0:00.028] Success

UserManager_LogOnUser_GetUserObjectWhenUserIsAValidUserWithNetBiosLogin [0:00.033] Success

UserManager_LogOnUser_GetUserObjectWhenUserIsAValidUserWithSuffixNetBiosStyle [0:02.288] Success

UserManager_LogOnUser_GetUserObjectWhenUserIsAValidUserWithSuffixUpnStyle [0:00.212] Success

UserManager_LogOnUser_GetUserObjectWhenUserIsAValidUserWithUpnLogin [0:00.026] Success

UserManager_RetrieveAllChangedUsers_GetUsersWithValidDomainData [0:04.342] Success

UserManager_RetrieveAllUsers_GetUsersWithValidDomainData [0:04.445] Success

HelloID.DirectoryAgent.Tests (11 tests) [0:01.174] Success

HelloID.DirectoryAgent.Tests.Internal (11 tests) [0:01.174] Success

Messaging (8 tests) [0:01.167] Success

MessageHandlerTests (6 tests) [0:01.162] Success

MessageHandler_HandleMessageCollecionAsync_DoesNothingWhenNullIsPassed [0:00.001] Success

MessageHandler_HandleMessageCollectionAsync_HandleMultipleMessageIsAuthenticatedSuccessAndSuccessResponseIsQueued [0:00.812] Success

MessageHandler_HandleSingleMessageAsync_DoesNothingWhenNullIsPassed [0:00.001] Success

MessageHandler_HandleSingleMessageAsync_HandlesMessageIsAuthenticatedFailedResponseIsQueued [0:00.103] Success

MessageHandler_HandleSingleMessageAsync_HandlesMessageIsAuthenticatedSuccessResponseIsQueued [0:00.144] Success
MessageHandler_HandleSingleMessageAsync_HandlesMessageIsConnectedResponseIsSentBack [0:00.099] Success

MessageTests (2 tests) [0:00.004] Success

Message_ToString_ShouldMatchTypeAuthenticationSuccessAndDescriptionAndLongDate [0:00.000] Success

Message_ToString_ShouldMatchTypeIsConnectedAndDescriptionAndLongDate [0:00.003] Success

ConnectionInitializerTests (3 tests) [0:00.006] Success

ConnectionInitializer_ParseResponseToSettings_NullValueShouldRaiseArgumentExceptio
n [0:00.006] Success

ConnectionInitializer_ParseResponseToSettings_ValueIncorrectShouldRaiseArgumentExc
eption [0:00.000] Success

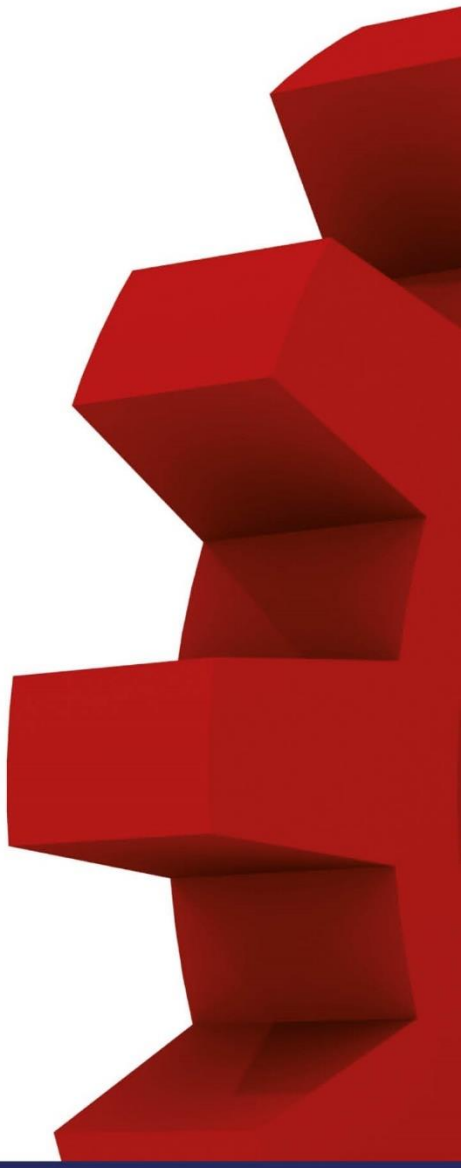
ConnectionInitializer_SetTicketAndUrls_NullValueShouldRaiseArgumentException
[0:00.000] Success

HelloID.DirectoryAgent.Shared.Tests (1 test) [0:00.034] Success

HelloID.DirectoryAgent.Shared.Tests.Entities (1 test) [0:00.034] Success

UserTests (1 test) [0:00.034] Success

User_CreateUser_AfterAddSimpleValueStillExists [0:00.034] Success 2/2



Amalialaan 126c
3743 KJ Baarn
Nederland

T +31 (0) 35 54 832 55 F +31 (0) 35 54 327 36

Informatie info@tools4ever.com
Sales sales@tools4ever.com
Support isupport@tools4ever.com