Afstudeerverslag

Handsfree Deuren Openen

Yael Keemink

Student: Yael Keemink
Student nummer: 13007491
Eerste examinator: G.A. Mijnarends
Tweede examinator: E.M. van Doorn
Opdrachtgever: Martijn Ermers
School: De Haagse Hogeschool
Plaats: Veenendaal

Afstudeerverslag

Titel	Afstudeerverslag
Project/Onderwerp	Handsfree deuren openen
Versie	1.0.0
Status	Definitief
Datum	02-06-2017
Bestand	Afstudeerverslag v1.0.0.docx
Bedrijf	Info Support

Info Support B.V. Kruisboog 42, 3905 TG, Veenendaal, Tel. +31 (0) 318 55 20 20 - Fax +31 (0) 318 55 23 55
Kenniscentrum De Smalle Zijde 39, 3903 LM Veenendaal, Tel. +31 (0) 318 50 11 19 - Fax +31 (0) 318 51 83 59

Historie

Versie	Status	Datum	Auteur	Verandering
0.1	Concept	09-02-2017	Yael Keemink	Creatie
0.2	Concept	10-02-2017	Yael Keemink	Organisatie toegevoegd en aanpak toegevoegd
0.3	Concept	13-02-2017	Yael Keemink	Hoofdstuk eerste week
0.4	Concept	14-02-2017	Yael Keemink	Sprint 1 toegevoegd
0.5	Concept	21-02-2017	Yael Keemink	Sprint 2 toegevoegd
0.6	Concept	02-03-2017	Yael Keemink	Sprint 3 toegevoegd
0.7	Concept	10-03-2017	Yael Keemink	Sprint 4 toegevoegd
0.8	Concept	13-03-2017	Yael Keemink	Feedback verwerkt
0.9	Concept	17-03-2017	Yael Keemink	Sprint 5 toegevoegd
0.10	Concept	21-03-2017	Yael Keemink	Sprint 6 toegevoegd
0.11	Concept	30-03-2017	Yael Keemink	Sprint 7 toegevoegd en feedback verwerkt
0.12	Concept	03-04-2017	Yael Keemink	Sprint 8 toegevoegd
0.13	Concept	10-04-2017	Yael Keemink	Sprint 9 toegevoegd
0.14	Concept	24-04-2017	Yael Keemink	Feedback examinator verwerkt en sprint 11 t/m 12 toegevoegd
0.15	Concept	18-05-2017	Yael Keemink	Feedback examinator verwerkt
0.16	Concept	24-05-2017	Yael Keemink	Sprint 13 en 14 toegevoegd
0.17	Concept	29-05-2017	Yael Keemink	Sprint 15 toegevoegd

Distributie

Versie	Status	Datum	Aan
1.0	Definitief	02-06-2017	De Haagse Hogeschool, Info Support

Referaat

Keemink, Yael

Het onderzoeken, ontwerpen en bouwen van een handsfree toegangscontrole systeem binnen Info Support.
Veenendaal, Info Support BV, 2017

Afstudeerverslag van Yael Keemink, geschreven in het kader van het afstuderen bij de opleiding Informatica aan De Haagse Hogeschool.

In het verslag wordt het proces dat is doorlopen tijdens het afstuderen van Yael Keemink bij Info Support BV te Veenendaal. Deze opdracht stond in het teken van het selecteren van een geschikte identificerende techniek en hier een proof of concept voor te maken om te bewijzen dat deze techniek door Info Support BV te gebruiken is. Naast het onderzoek moest er tijdens het ontwikkelen van de applicatie op worden gelet dat de juiste architectuur gekozen werd zodat deze gebruikt kon worden om de definitieve applicatie mee te maken.

Descriptoren

- Afstudeeropdracht
- C#
- .NET core
- Microservice architecture
- Event bus
- Docker
- Domain driven design
- Onion architecture
- Test driven development
- Swagger
- Entity framework core
- RFID
- Toegangscontrole systeem

Voorwoord

Dit is het afstudeerverslag 'Handsfree deuren openen'. Dit verslag heb ik geschreven in het kader van het afstuderen aan de opleiding Informatica aan De Haagse Hogeschool.

Ik heb van 30 januari tot 2 juni gewerkt aan een project om aan te tonen of er een mogelijkheid is, om binnen Info Support de deuren 'handsfree' te kunnen openen.

Ik wil graag mijn begeleiders (Martijn Ermers en Laila van de Hoef) en mijn manager (Bas Meerman) bedanken voor de uitstekende begeleiding tijdens mijn afstudeerperiode en voor het bedenken van nieuwe uitdagingen, waardoor ik nieuwe inzichten verkreeg.

Yael Keemink
Veenendaal, 2 juni 2017

© Info Support, Veenendaal 2017

Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook, zonder voorafgaande toestemming van Info Support.

No part of this publication may be reproduced in any form by print, photo print, microfilm or any other means without written permission by Info Support.

Prijsopgaven en leveringen geschieden volgens de Algemene Voorwaarden van Info Support B.V., gedeponeerd bij de K.v.K. te Utrecht onder nr. 30135370. Een exemplaar zenden wij u op uw verzoek per omgaande kosteloos toe.

Inhoudsopgave

1 INLEIDING 1

2 CONTEXT BESCHRIJVING 2

2.1 Organisatiebeschrijving 2
2.2 Mijn werkomgeving 4
2.3 Het probleem 4
2.4 Doelstelling 4

3 AANPAK 6

3.1 De keuze voor scrum 6
3.2 Scrum in een eenmansproject 7
3.3 Scrum met een vaste tijd, vast budget en vaste set aan features 7
3.4 Te gebruiken tools, talen en frameworks 8
3.5 Schatten van backlogitems 8
3.6 Risico management 8
3.7 Planning 9

4 DOEN VAN HET ONDERZOEK 10

4.1 Sprint 1, beginnen vooronderzoek 10
4.2 Sprint 2, beginnen onderzoek 14
4.3 Sprint 3, achterhalen functionaliteit 16
4.4 Sprint 4, afronding onderzoek 19

5 BEPALEN VAN DE ARCHITECTUUR 23

5.1 Sprint 5, grote architecturale keuzes 23
5.2 Sprint 6, eerste oefening met RabbitMQ en Docker 33

6 PROGRAMMEREN EN TESTEN 35

6.1 Sprint 7, bouwen RDIF integration service 35
6.2 Sprint 8, logging 37
6.3 Sprint 9, gebruikers identificeren 38
6.4 Sprint 10, gebruikers autoriseren 40
6.5 Sprint 11, onderzoek naar logging 43
6.6 Sprint 12, logging service ontwerpen 46
6.7 Sprint 13, logging service maken 47
6.8 Sprint 14, delay inbouwen 49
6.9 Sprint 15, integratie en acceptatie tests schrijven 51

7 PRODUCT EVALUATIE 53

7.1 Plan van aanpak 53
7.2 Onderzoeksrapport 53
7.3 Architectuur document 53
7.4 Ontwerpen 53
7.5 Software 54

8 PROCES EVALUATIE 55

8.1 Externe afhankelijkheden 55
8.2 Onderzoek 55
8.3 Ontwikkelen 55
8.4 Scrum 55
8.5 Beroepstaken 55

LITERATUUR **58**

EXTERNE BIJLAGES **59**

Bijlage A – Plan van aanpak 59
Bijlage B – Onderzoeksplan 59
Bijlage C – Onderzoeksrapport 59
Bijlage E – Architectuur 59
Bijlage F – Ontwerpen 59
Bijlage G – Acceptatie tests 59

1 Inleiding

Ik heb een project bij Info Support gedaan om te onderzoeken of er een mogelijkheid is om het huidige toegangscontrole systeem uit te breiden, zodat je geen handen meer hoeft te gebruiken om een deur te openen. Het huidige systeem werkt door een pasje tegen een paslezer te houden en dat zorgt momenteel voor irritaties bij de werknemers, doordat ze soms meerdere pogingen nodig hebben om een deur open te krijgen met volle handen.

Om dit te bereiken heb ik:

- Een onderzoek gedaan naar welke identificerende techniek het best gebruikt zou kunnen worden binnen Info Support.
- Een architectuur rapport beschreven waarin ik al mijn architecturale keuzes heb uitgelegd.
- Een toegangscontrole systeem gebouwd op basis van een microservice architectuur.

Het is me gelukt om een toegangscontrole systeem te bouwen dat werkt tot een afstand van twee meter. Daarmee is het doel behaald en kunnen deuren dus 'handsfree' geopend worden. Dit systeem zal naast het huidige toegangscontrole systeem draaien.

Aan het einde van dit verslag zal ik uiteenzetten wat er goed en wat er beter kon bij het doen van dit project.

2 Context beschrijving

2.1 Organisatiebeschrijving

Info Support is een bedrijf dat voornamelijk ICT consultants levert. Deze consultants zullen advies leveren aan de klant en software schrijven voor de klant. Bij Info Support staat kwaliteit hoog in het vaandel.

2.1.1 Info Support BV

Info Support BV is een dochteronderneming van Info Support Group. Info Support BV heeft vier business units.

1. Business unit finance (BUF)
2. Business unit handel en industrie (BUI)
3. Business unit overheid (BUO)
4. Business unit Zorg en verzekeringen (BUZ)

Deze business units bedienen elk een eigen groep klanten. Naast de business units heb je ook unit overschrijdende afdelingen:

1. Business intelligence
2. Kenniscentrum
3. Mangaged IT services
4. Professional development center (PDC)

2.1.1.1 PDC

Ik heb mijn opdracht gedaan binnen de afdeling professional development center (PDC). Ik was bij PDC geplaatst, omdat mijn opdrachtgever ook bij PDC zit.

Het PDC zorgt ervoor dat software ontwikkelaars goed kunnen werken. Dit doen ze door werkomgevingen in te richten en te beheren, maar ook door de nieuwste technieken binnen software ontwikkeling te onderzoeken. Wanneer er een nieuwe techniek onderzocht is en nuttig wordt bevonden, zal het PDC ondersteuning bieden bij de klanten om de techniek te gebruiken.

Een voorbeeld hiervan is Docker. Docker is een tool waarmee je gemakkelijk een applicatie op elke machine kan deployen. Veel klanten van Info Support zagen de potentie van Docker, maar durfden het niet te gebruiken door gebrek aan kennis over deze techniek. Doordat het PDC deze kennis wel had en deze beschikbaar stelde aan de klant, konden deze overgaan tot de implementatie van Docker.

Het PDC biedt niet alleen ondersteuning aan ontwikkelaars van Info Support maar ook aan programmeurs van andere bedrijven. Deze nemen dan een contract af bij het PDC voor de ondersteuning.

2.1.2 Info Support NV

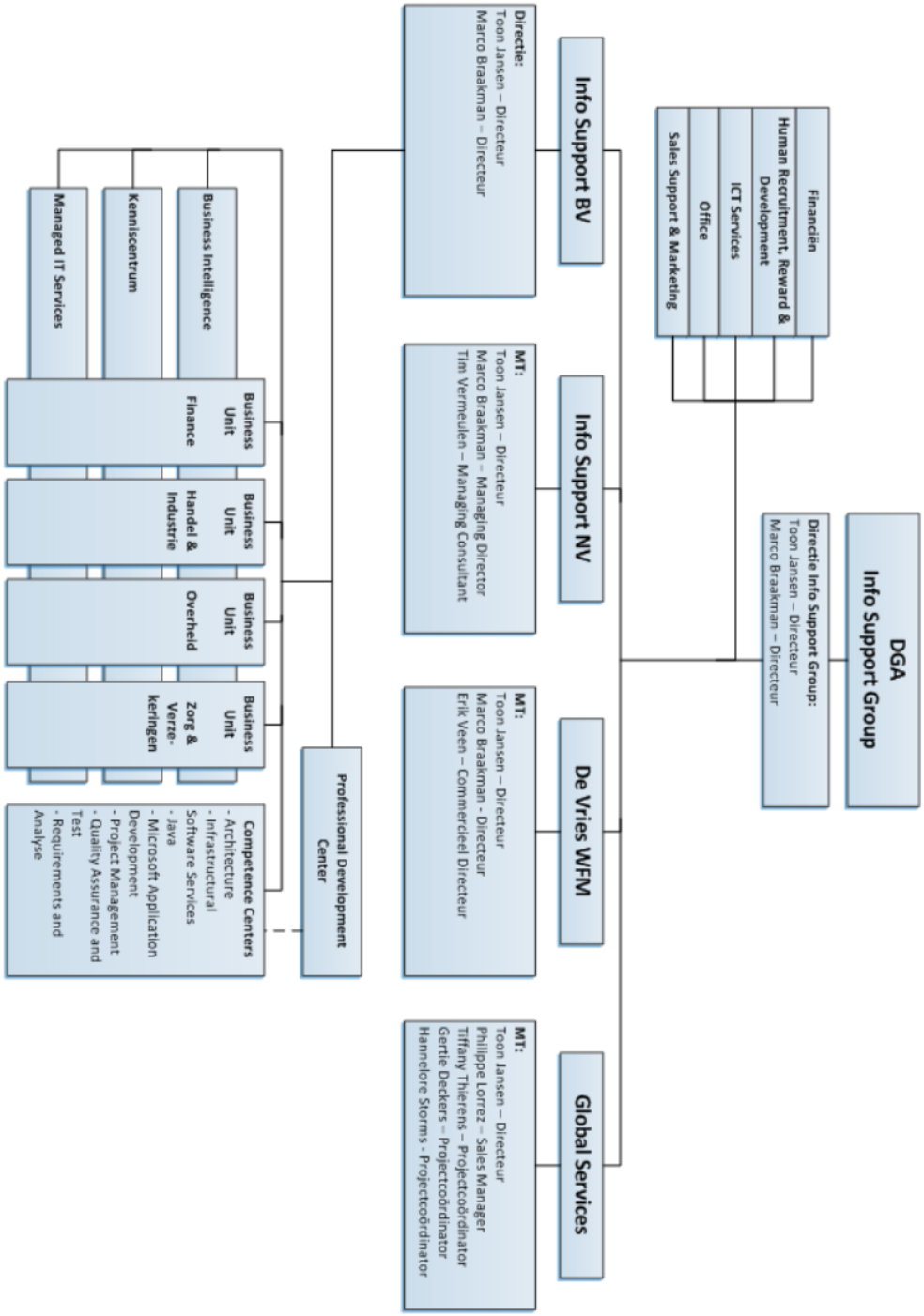
Info Support NV is de Belgische tak van Info Support BV. Info Support NV voert dezelfde soort werkzaamheden uit als Info Support BV.

2.1.3 De vries WFM

De Vries Workforce Management is een Europees softwarebedrijf, met het hoofdkantoor in Veenendaal en is marktleider voor Workforce Management oplossingen in de Retail. Per 1 april 2010 is De Vries overgenomen door Info Support International Group.

2.1.4 Global services

Global Services – GS Training & Knowledge Management' is sinds 1996 de ITopleidingspartner van KMO's (kleine of middelgrote ondernemingen) en multinationals in meer dan 20 landen.



Figuur 1: de organisatie

2.2 Mijn werkomgeving

2.2.1 De begeleiding

Binnen Info Support heb ik begeleiding gehad van een tweetal medewerkers. Ik ben begeleid door een procesbegeleider en een opdrachtgever.

De procesbegeleider, Laila van de Hoef, heeft mij begeleid bij het gehele afstudeertraject. Zij zorgde ervoor dat mijn afstudeertraject in goede banen verliep en dat, wanneer ik problemen had, deze problemen ook opgelost zouden worden.

De opdrachtgever, Martijn Ermers, was de bedenker van de opdracht. Ik kon met al mijn technische vragen terecht bij hem.

2.3 Het probleem

2.3.1 Aanleiding

Veel medewerkers van Info Support vinden het storend dat ze om een deur te ontgrendelen eerst een pasje voor een scanner moeten houden om vervolgens binnen enkele seconden de deurklink naar beneden te moeten trekken. Onder normale omstandigheden zal dit niet voor problemen, echter wanneer een medewerker met volle handen de deur moet ontgrendelen kan dit voor irritaties zorgen.

2.3.2 Probleemstelling

Wanneer dat een medewerker veel spullen in zijn handen heeft zal hij deze even op de grond moeten leggen om zijn pasje te scannen. Vervolgens zal hij zijn spullen weer op moeten pakken en de deur open moeten doen en dit allemaal binnen een paar seconden omdat de deur anders weer in het slot valt.

Dit zorgt bij veel werknemers voor irritaties en soms zelfs tot het permanent openhouden van de deur, waardoor iedereen naar binnen kan.

2.4 Doelstelling

De doelstelling voor dit project was:

> *Met het uitvoeren van deze opdracht moet voor Info Support duidelijk worden of, en zo ja hoe, de deuren anders open gemaakt kunnen worden, zodat de werknemers hier niet meer door worden gehinderd. De oplossing moet ervoor blijven zorgen dat de medewerkers van Info Support alleen op locaties in het gebouw kunnen komen waar ze mogen komen.*

2.4.1 Resultaat

Er is een onderzoek gedaan naar de mogelijkheden om de deuren handsfree te openen. Dit onderzoek resulteerde in een onderzoeksrapport. Deze is in de bijlage mee geleverd.

Er is ook een proof of concept gemaakt waarmee werd bewezen dat het mogelijk is om de deuren veilig en handsfree te openen. Hiernaast is er ook een architectuur rapport opgeleverd welke al mijn architecturale beslissingen bevat.

2.4.2 Effect

Met het resultaat moet duidelijk worden of en hoe het huidige systeem verbeterd/vervangen kan worden om de irritaties bij de medewerkers weg te nemen en toch de veiligheid te blijven garanderen.
Ook zal de afstudeerder voldoende moeten kwaliteit hebben getoond in voldoende competenties om te slagen.

3 Aanpak

In dit hoofdstuk zal ik uitleggen welke keuze ik heb gemaakt voor de ontwikkelmethode, welke tools ik gebruikt heb en wat de globale planning was.

3.1 De keuze voor scrum

In de eerste week van mijn afstuderen heb ik een aantal presentaties gevolgd. Waaronder een presentatie over scrum en een presentatie over het maken van een plan van aanpak. Deze presentatie volgde ik, omdat in elk bedrijf scrum anders geïmplementeerd wordt en ik wilde weten hoe Info Support vindt dat scrum gebruikt moet worden.

Info Support houdt zich heel strikt aan de scrum guide (Schwaber & Sutherland, 2016). Hierdoor wijken ze vrijwel niet af van de manier waarop ik scrum op school geleerd heb.

Wel kwam tijdens de presentatie over scrum een discussie op of dit te gebruiken is voor je onderzoek. Uit deze discussie kwam naar voren dat dit mogelijk is door scrum op één van de de volgende twee manieren te gebruiken.

De eerste manier om scrum te gebruiken als methode voor je onderzoek is door elke sprint een klein stukje onderzoek te doen en vervolgens een Proof of Concept (PoC) te maken voor dat kleine stukje onderzoek.

De tweede manier is door je opdracht in twee fases op te delen. De eerste fase is dan je onderzoeksfase en de tweede fase de ontwikkelfase. Tijdens de eerste fase zal je aan het einde van elke sprint in plaats van software een deel van je onderzoek afronden. En in de ontwikkelfase lever je stukjes software op.

Ik koos ervoor om de tweede manier van scrum toe te passen op mijn opdracht. Ik heb hiervoor gekozen, omdat ik geen tijd had om voor elke te onderzoeken techniek een PoC te maken.

Ik heb voor scrum gekozen, omdat het werken met iteraties en de vele contactmomenten met de opdrachtgever mij erg handig leken. Door elke iteratie een planning, retrospective en een refinement te houden zorg je ervoor dat het product dat je maakt ook daadwerkelijk aan de eisen van de opdrachtgever voldoet. Met waterval was het gevaar dat ik aan het begin de requirements verkeerd begreep en daar pas aan het einde van het project achter kom. Met scrum heb ik dat risico niet, omdat er elke sprint wordt gekeken naar mijn increment.

Een andere reden voor het kiezen van scrum als methode was dat dit binnen Info Support veel gebruikt wordt. Om mee te kunnen maken hoe er binnen Info Support gewerkt wordt, heb ik ervoor gekozen om ook dezelfde ontwikkelmethode te gebruiken.

Het laatste voordeel dat scrum zal geven ten opzichte van waterval voor dit project, is dat ik aan het einde van het project daadwerkelijk ook iets kan laten zien. Het zou kunnen voorkomen dat, wanneer ik waterval had gebruikt, ik bleef hangen bij de ontwikkelfase en dat het product waar ik mee bezig was dus nog niet werkt en nog niet getest was.

Ik had afgesproken met de opdrachtgever om sprints van één week te houden. Deze keuze was gemaakt, omdat ik op die manier elke week iets kan laten zien aan de opdrachtgever en hij daar direct feedback op kan geven. Door deze manier van werken zorgde ik ervoor dat ik zo min mogelijk werk hoefde aan te passen wanneer iets niet naar zijn wens was.

3.2 Scrum in een eenmansproject

Door de keuze voor scrum kwam ik voor een probleem te staan, namelijk dat scrum bedoeld is voor teams van minimaal 3 personen. Hierdoor was het het meest logisch om de rollen van product owner, scrum master en development team op me te nemen. Ik vertegenwoordig dus eigenlijk het gehele scrum team.

Ik heb ervoor gekozen om zelf de product owner te zijn in plaats van mijn opdrachtgever de product owner te laten zijn, omdat de product owner de belangen van alle stakeholders moet vertegenwoordigen. De enige die de eisen van alle stakeholders goed kan vertegenwoordigen ben ik zelf.

De opdrachtgever kan de belangen van De Haagse Hogeschool en die van mij wel vertegenwoordigen maar zal dit minder goed kunnen doen dan ik, omdat ik veel beter weet wat er vanuit De Haagse Hogeschool verwacht wordt. Ook is voor de opdrachtgever het project al geslaagd als de deuren handsfree geopend kunnen worden, maar dit hoeft echter niet te betekenen dat ik ook geslaagd ben voor mijn opleiding.

De daily standup wordt normaal gesproken gedaan met het hele scrum team. Omdat ik het hele scrum team vertegenwoordig heb ik samen met mijn opdrachtgever en een paar andere afstudeerders, besloten om de daily standup samen te houden. Op deze manier kon ik alsnog problemen die ik had voorleggen aan anderen. Uiteindelijk heeft dit niet heel goed gewerkt, omdat de andere afstudeerders maar twee dagen op het kantoor in Veenendaal waren.

Ook de retrospective was een punt van aandacht, dit is ook iets wat met het hele team gebeurt. De retrospective deed ik samen met mijn opdrachtgever. Ik kon dan met hem bespreken wat er goed ging en wat er beter kon.

3.3 Scrum met een vaste tijd, vast budget en vaste set aan features

Het is niet mogelijk om het tijdsbestek waarin een project afgerond moet zijn, het budget dat beschikbaar is voor het project en de functionaliteiten die af moeten zijn aan het einde van het project vooraf vast te leggen. Dit is echter wel het geval in dit project. Ik wist van tevoren wanneer ik mijn afstudeerverslag moest inleveren en ik het project dus moest afronden. Tevens wist ik dat ik het gedurende die periode alleen zou moeten doen en ik wist grofweg welke functionaliteiten in het product moesten zitten.

Om te voorkomen dat er geen werkend product opgeleverd kon worden aan het eind van dit project, ging ik als eerst de minimal viable product (mvp) maken. In de mvp zit het minimale waarmee bewezen kan worden dat het product zou kunnen werken. Het product hoeft dan dus nog niet daadwerkelijk functioneel te zijn. Na de mvp kan er verder gebouwd worden aan het product om het functioneel te maken.

Daarnaast zorgde ik ervoor dat de verwachtingen van het uiteindelijke product reëel bleven. Dit deed ik door elke sprint review een verwachting uit te spreken van welke onderdelen waarschijnlijk af zouden komen.

Door op deze manier om te gaan met de vaste tijd, budget en functies, zorgde ik ervoor dat er toch nog een vorm van flexibele functionaliteit kwam

3.4 Te gebruiken tools, talen en frameworks

Tijdens het maken van het plan van aanpak wist ik nog niet welke techniek ik wilde gebruiken en daardoor ook niet welke taal en welk framework ik wilde gebruiken. Wat ik al wel kon bepalen was met welke tools ik mijn uitvoering zou gaan ondersteunen.

Ik heb in overleg met mijn opdrachtgever besloten om zoveel mogelijk te werken met de tools die binnen Info Support gebruikt worden. Dit heeft als voordeel dat er geen extra licenties aangevraagd hoeven te worden en dat er kennis over deze tools binnen Info Support is en ik dus met al mijn vragen terecht kon bij mijn collega's.

Omdat ik met C# ging werken en ik de tools die binnen Info Support gebruikt worden ging gebruiken, ging ik gebruik maken van TFS voor mijn scrumboard en mijn git repository. TFS is een applicatie van Microsoft waar je een backlog kan bijhouden, een scrumboard hebt, je code repository kan plaatsen en een buildstraat kan opzetten. Deze applicatie wordt binnen Info Support veel gebruikt door C# ontwikkelaars, omdat deze goed is geïntegreerd met de code editor voor .NET, visual studio.

3.5 Schatten van backlogitems

Om een goede sprintplanning te maken moet er geschat worden hoeveel moeite elk backlogitem kost. Om in te kunnen schatten hoeveel tijd een backlogitem kost, heb ik gebruik gemaakt van story points. Story points zijn punten die je aan een backlogitem kan geven. Deze punten geven aan hoeveel moeite elk backlogitem kost.

Het bepalen van hoeveel elk backlogitem kost begint bij het bepalen hoeveel moeite één of twee punten kosten. Dat wordt gedaan door te kijken naar een gemakkelijke backlog en deze te schatten. Vervolgens kan er geschat worden aan de hand van hoeveel meer of minder moeite een backlog item kost.

3.6 Risico management

Bij het maken van het plan van aanpak heb ik een aantal risico's onderkend en daar passende maatregelen voor bedacht.

1. **De hardware voor het PoC is niet beschikbaar.**
   Dit risico kan twee oorzaken hebben. Ten eerste kan het zo zijn dat de hardware die ik nodig heb te duur is en ik dus geen geld hiervoor krijg en ten tweede kan het zo zijn dat de levertijd van de hardware te lang is en de hardware dus niet op tijd aankomt.

   Er kan bij dit risico tussen twee maatregelen gekozen worden.
   Ik kan ervoor kiezen om andere hardware te kiezen waarmee ik het PoC ga maken of ik kies ervoor om het PoC te mocken. Bij het mocken van de paslezer zorg ik ervoor dat er een applicatie komt die doet alsof hij de paslezer is. Hierdoor kan alsnog bewezen worden of de gekozen oplossing werkt.

2. **De beschreven planning blijkt niet haalbaar.**
   Dit kan komen doordat de planning te strak gepland is. Hier wordt rekening mee gehouden door een week uitloop in te plannen.

3.7 Planning

			Januari	Februari				Maart				April				Mei				
			Sprint 0	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9	Sprint 10	Sprint 11	Sprint 12	Sprint 13	Sprint 14	Sprint 15	Sprint 16	Sprint 17
Taak	**Start**	**Af**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
Onderzoeksfase																				
Plan van aanpak	1	2																		
Storyboard	3	4																		
Onderzoeksplan	2	2																		
Onderzoeksrapport	2	5																		
Ontwikkelingsfase																				
Story board	6	18																		
UML diagrammen	6	18																		
Proof of concept	6	18																		
Testen	6	18																		
Testrapport	6	18																		
Fase overstijgende activiteiten																				
Afstudeerverslag	2	20																		
Backlog bijwerken	3	18																		
Architectuur																				
Trainingen																				

Figuur 2: de planning

In de tweede week was het plan om het plan van aanpak af te maken en te beginnen met het onderzoeksplan. Het onderzoek zou tot week 6 lopen. Vanaf week 6 zal vervolgens aan het ontwikkelen van het proof of concept gewerkt worden. Hierbij zou het architectuurrapport elke sprint bijgewerkt worden naar de nieuwe inzichten.

4 Doen van het onderzoek

4.1 Sprint 1, beginnen vooronderzoek

Deze sprint had ik de volgende story's gepland uit de backlog:

Story	Punten
Plan van aanpak maken.	5
Onderzoeksplan maken.	6
Eerste week beschrijven	6
Onderzoeksrapport opzetten	3
Eerste opzet architectuur rapport maken.	8
Totaal aantal punten	28

Deze story's zouden helpen bij het behalen van het opgestelde sprintdoel: *'Te weten komen wat er onderzocht zal worden'*.

4.1.1 voorbereiden van het onderzoek

Voor mijn onderzoeksplan had ik een hoofdvraag en een aantal deelvragen nodig. Daarom had ik de volgende vragen opgesteld:

Hoofdvraag	**Wat is een goede identificerende techniek, waarbij er geen handen nodig zijn, om binnen Info Support te herkennen wie iemand is, zodat bepaalt kan worden of diegene bevoegd is om door de deur te mogen?**
Deelvraag 1	Wat zijn de belangrijkste eisen en wensen van de Info Support medewerkers?
Deelvraag 2	Wat zijn de belangrijkste eisen en wensen van systeembeheer?
Deelvraag 3	Hoe werkt het huidige systeem?
Deelvraag 4	Welke identificerende technieken kunnen gebruikt worden?
Deelvraag 5	Wat zijn de voor en nadelen per techniek?
Deelvraag 6	Wat zijn de kosten per techniek?

Tijdens een gesprek met de vervangend opdrachtgever is naar boven gekomen dat deze hoofdvraag niet helemaal correct is. Hij vond dat het gedeelte na de laatste komma niet bij dit onderzoek hoort, omdat het bij dit onderzoek vooral om de identificerende technieken zal gaan en niet zozeer over het autoriseren.

Ook vond hij de deelvragen één tot en met vier niet helemaal juist. Dit was wel informatie die je moest weten, maar het was niet zozeer informatie die in het onderzoek onderzocht moest worden. Deze informatie zou ik ook in een vooronderzoek kunnen vergaren, zodat ik deze al heb voordat ik aan het onderzoek begin. Hij stelde voor om elk voor- en nadeel als deelvraag op te nemen en daarop mijn hoofdvraag te baseren.

Dit heb ik gedaan en daar is de volgende hoofdvraag en deelvragen uitgekomen:

Hoofdvraag	**Welke identificerende techniek voldoet het beste aan de eisen van Info Support?**
Deelvraag 1	Wat is het gebruiksgemak per techniek?
Deelvraag 2	Wat is de betrouwbaarheid per techniek?
Deelvraag 3	Wat zijn de kosten per techniek?

Om erachter te komen wat de eisen vanuit Info Support zijn, heb ik in het vooronderzoek de vraag gesteld: *"Wat zijn de eisen van de opdrachtgever en systeembeheer?"* De eisen van de opdrachtgever waren grotendeels al bekend, deze hoefden alleen nog uitgewerkt te worden. Om achter de eisen van systeembeheer te

komen had ik een aantal vragen opgesteld. Met deze vragen ben ik naar systeembeheer gegaan.
Systeembeheer en de opdrachtgever hadden de volgende eisen hebben aan het systeem:

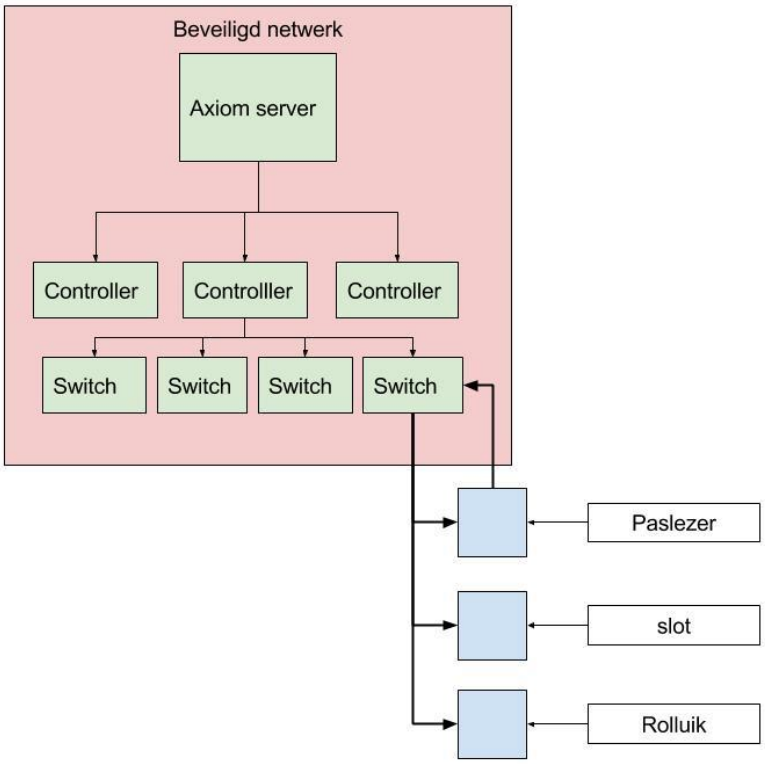
- Deuren mogen niet onnodig open gaan.
- Deuren mogen niet open gaan voor mensen zonder het juiste beveiligingsniveau.
- Het identificatiemiddel moet moeilijk te kopiëren zijn.
- Het identificatiemiddel mag geen privé apparaat zijn.
- De deur moet zonder gebruik van de handen, voor het identificeren en het open doen van de deur zelf, open gaan.

Daarna ben ik op zoek gegaan naar geschikte technieken om als identificatie techniek binnen Info Support te fungeren. Ik heb gezocht naar technieken die zouden kunnen voldoen aan de eisen van Info Support. Ik heb gekozen voor gezichtsherkenning, stemherkenning, irisscanner en radio frequency identification (RDID). Bij al deze technieken is het identificatie middel niet of moeilijk te kopiëren en kan de techniek in principe gebruikt worden om deuren handsfree te openen.

4.1.2 Werking van het huidige systeem

Voor het eerste gedeelte van het architectuur rapport moest ik ook bij systeembeheer zijn. Ik wilde weten hoe het huidige systeem om de deuren te ontgrendelen in elkaar zit en of er een API beschikbaar was die ik zou kunnen gebruiken.

Systeembeheer legde uit dat het systeem niet alleen de deuren ontgrendelt, maar ook het brandalarm en het inbraakalarm regelt. Om die reden heeft het systeem dan ook geen API die ik kan gebruiken. Wanneer ik direct tegen dat systeem aan kon praten, zou ik ook het beveiligingssysteem kunnen omzeilen. Wel kon hij uitleggen hoe het huidige systeem werkt en hoe mijn systeem er dus uit moest komen te zien.



Figuur 3: de huidige architectuur

Het systeem bestaat uit 3 delen. Je hebt de server, deze autoriseert of een geïdentificeerd persoon ergens naar binnen mag. Je hebt een controller, deze zorgt ervoor dat de input van alle verschillende apparaten op de goede plek komt. Als laatst heb je de switch, hierop worden alle apparaten, zoals sloten en paslezers, aangesloten. Deze kan bijvoorbeeld een signaal versturen naar de deur dat hij moet ontgrendelen en hij kan een signaal ontvangen van de paslezer met daarin het id van een pasje.

Op dit moment werkt het systeem met behulp van NFC. Door een pasje van Info Support voor een paslezer te houden zal deze een signaal geven aan een switch, deze stuurt het signaal door naar een controller en de controller stuur het signaal weer door naar de server. Deze server zal kijken of het id in dat signaal geautoriseerd is om door de deur te mogen. Wanneer dit het geval is zal hij een signaal terug sturen aan de controller dat de deur open mag. De controller zal vervolgens een elektrische puls sturen naar het slot van de deur welke open zal gaan.

4.1.3 Opzetten van het architectuur rapport

Voor het opzetten van de structuur van het architectuur rapport heb ik als referentie materiaal gebruik gemaakt van de lessen en het project in blok H. Hier koos ik voor omdat ik vond dat hier zeer goede richtlijnen in zaten van wat er in een architectuur rapport zou kunnen staan. Ik heb besloten hier alleen uit te gebruiken wat mij nodig leek. Het leek mij handig om de volgende onderwerpen in het rapport op te nemen:

- **Algemene architectuur**, dit is handig om een globaal overzicht te geven van hoe het systeem in elkaar zit

- **Functional viewpoint**, om een goed beeld te krijgen wat voor functionaliteit in de verschillende onderdelen hebben is het handig dit onderdeel op te nemen in het rapport
- **Information viewpoint**, om de verschillende eventstromen in kaart te brengen en om de inhoud van de databases weer te geven is ook dit een handig onderwerp
- **Development viewpoint**, dit onderdeel is handig om duidelijk te maken welke technieken er gebruikt worden en om de endpoints van de API's te beschrijven
- **Regulations perspective**, er is wetgeving waar deze applicatie aan moest voldoen, daarom is het handig om dit te onderkennen en duidelijk te maken dat daar rekening mee gehouden is
- **Evolution perspective**, om duidelijk te maken dat er rekening gehouden is met een eventuele uitbreiding is het handig om dit ook op te nemen in je architectuur rapport, hierdoor kan je voorkomen dat je niet denkt aan eventuele uitbreidingen van het systeem

4.1.4 Sprintreview

Deze sprint was het helaas niet gelukt om aan mijn afstudeerverslag te werken. Dit werd dan ook weer opgenomen in de volgende sprint. De overige story's waren wel afgekomen.

4.2 Sprint 2, beginnen onderzoek

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Vooronderzoek doen	8
Inleiding onderzoeksrapport	4
Deelvraag 1, Wat is het gebruiksgemak per techniek?, beantwoorden	9
Eerste week beschrijven in het afstudeerverslag	2
Deelvraag 2, Wat is de betrouwbaarheid per techniek?, beantwoorden	9
Sprint 1 beschrijven in het afstudeerverslag	4
Totaal aantal punten	36

Tijdens deze sprint wilde ik informatie inwinnen over de verschillende technieken en wilde ik de eerste twee deelvragen beantwoorden.

4.2.1 De verschillende technieken

Na het vooronderzoek was het voor mij duidelijk welke technieken bruikbaar waren en wat die technieken inhouden. Ik koos ervoor om de volgende technieken met elkaar te vergelijken:

- Gezichtsherkenning
- Stemherkenning
- Irisscanner
- RFID
- Beacons
- QR-codes

Tijdens het vooronderzoek bleek al snel dat een QR-code geen goede oplossing was. De gebruikers zouden dan allemaal een QR-code op de kleding moeten dragen waardoor je het risico loopt dat een QR-code kwijtraakt of op een verkeerde hoogte hangt voor de reader. Ook is het te makkelijk om een foto te nemen van een QR-code en deze vervolgens voor een scanner te houden, waardoor er mogelijk een veiligheidslek ontstaat.

Beacons had ik aan het begin van mijn onderzoek niet onderzocht. Ik zag te grote nadelen bij beacons waardoor ik deze had afgeschreven. Hier kwam ik echter in sprint drie, na een gesprek met de privacy adviseur, op terug en besloot beacons alsnog te onderzoeken. Dit wordt in sprint drie verder uitgelegd.

Van de technieken die ik wel meenam in het onderzoek zal ik hieronder even kort uitleggen hoe de technieken werken.

4.2.1.1 Irisscanner

Een irisscanner is een infraroodcamera die een scan maakt van je oog. Vervolgens zal er een iriscode worden gegenereerd. Deze iriscode wordt vergeleken met andere iriscodes in de database en wanneer er een match is gevonden zal hij een signaal geven dat de persoon geïdentificeerd is (Meus, 2017).
De iriscode wordt opgebouwd uit unieke kenmerken van een iris zoals:

- Groeven in de iris
- Putjes in de iris
- Strepen in de iris

4.2.1.2 Gezichtsherkenning

Gezichtsherkenning werkt op een vergelijkbare manier als een irisscan. Er wordt een foto gemaakt van het gezicht. Hier worden bepaalde kenmerken uitgehaald zoals:

- Dikte van de lippen
- Plaatsing van de neus
- Afstand tussen de ogen
- Diepte van de oogkassen
- Botstructuur
- Breedte van de mond
- Lengte van de kaaklijn
- Afstand ogen tot de neus
- Afstand tussen de oren

Al deze kenmerken bij elkaar worden opgeslagen als een template in een database. Deze template kan nu gebruikt worden om iemand te identificeren.

4.2.1.3 Stemherkenning

Stemherkenning wordt gedaan door een voicemodel te maken op basis van je stem. Dit voicemodel wordt net als bij de vorige technieken opgeslagen in een database om later die persoon te kunnen identificeren.

4.2.1.4 RFID

RFID is een techniek die met behulp van radiogolven data kan versturen. RFID kan op afstanden tot 100 meter data verzenden. RFID kan op twee verschillende manieren gebruikt worden (Impinj, z.j.). Er kan gebruik gemaakt worden van passieve en actieve RFID apparaten.

De passieve zullen de energie halen uit een reader en de actieve heeft een batterij bij zich waarmee hij zichzelf kan voorzien van energie. De techniek zal een id verzenden door korte pulsen op een bepaalde radio frequentie te genereren. Deze pulsen worden opgevangen door een reader welke deze pulsen weer kan omzetten naar een id. Met dat id kan bepaald worden wie jij bent en tot welke deuren jij toegang hebt.

4.2.1.5 Beacons

Beacons zijn kleine apparaatjes die constant een bluetooth signaal uitzenden. In dit signaal staat welke beacon het is en hieruit kan de afstand tot de beacon berekend worden (Nijland, 2014). Er zijn twee manieren waarop de beacons gebruikt kunnen worden.

1. Er kan een applicatie gemaakt worden, welke op de telefoon van de gebruiker draait, die luistert naar de beacon signalen. Wanneer deze een signaal van een beacon binnenkrijgt kan de applicatie een bericht naar de server doorsturen met daarin zijn eigen id en het id van de beacon. Hierop kan het backend systeem gaan controleren of de deur open zou mogen.
2. De medewerkers krijgen een beacon die ze bij zich kunnen dragen. De ontvangers staan dan bij de deuren. Deze ontvangers kunnen een bericht naar het backend systeem sturen zodat die kan bepalen of de deur open mag gaan.

4.2.2 Wat is het gebruiksgemak per techniek?

Om antwoord te krijgen op de eerste deelvraag, wat is het gebruiksgemak per techniek, stelde ik bij elke techniek een aantal vragen:

- Zijn er speciale handelingen nodig om de techniek te kunnen gebruiken?

- Wat is de reactie snelheid?
- Wat moet er gebeuren om een nieuwe medewerker in het systeem te krijgen?

Hieruit kwam de volgende tabel:

	Irisscanner	**Gezichts-herkenning**	**Stem-herkenning**	**RFID**	**Beacons**
Speciale handelingen	In camera kijken	In camera kijken	Praten in microfoon	Geen	App aanzetten of geen
	Letten op plaatsing camera				
Snelheid	< 1 seconden	2 seconden (Fingertec, 2009)	Niet gevonden	< 1 seconden	< 1 seconden bij hoge frequentie
	2-3 seconden bij geen identificatie				Tot 10 seconden bij 100 ms frequentie
Enrollment	Iris scannen	Foto maken	Stem opnemen	Pas uitgeven	App installeren
					Beacon uitgeven

Hieruit werd duidelijk dat de RFID het best naar voren kwam. De RFID heeft een reactietijd van minder dan 1 seconden en de gebruiker hoeft helemaal niets te doen om een deur open te krijgen. De andere technieken hebben allemaal wel een gebruikershandeling nodig om een persoon te kunnen identificeren. Dit zorgt ervoor dat op het gebied van gebruiksvriendelijkheid RFID de beste keus is.

4.2.3 Sprintreview

In deze sprint is het helaas niet gelukt om ook een antwoord te vinden op de tweede deelvraag. Het vooronderzoek duurde langer dan ik verwacht had waardoor ik pas later kon beginnen met mijn onderzoek.

4.3 Sprint 3, achterhalen functionaliteit

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	**Punten**
Deelvraag 2, wat is de betrouwbaarheid per techniek, beantwoorden	2
User story map maken	8
Sprint 1 beschrijven in het afstudeerverslag	4
Sprint 2 beschrijven in het afstudeerverslag	4
Inleiding onderzoeksrapport schrijven	4
Eerste week beschrijven in het afstudeerverslag	2
Deelvraag 3, wat zijn de kosten per techniek, beantwoorden	8
Hoofdvraag beantwoorden	6
Totaal aantal punten	36

Tijdens deze sprint wilde ik alvast duidelijk hebben welke functionaliteit het systeem grofweg moest krijgen. Om hierachter te komen maakte ik gebruik van de techniek user story mapping.

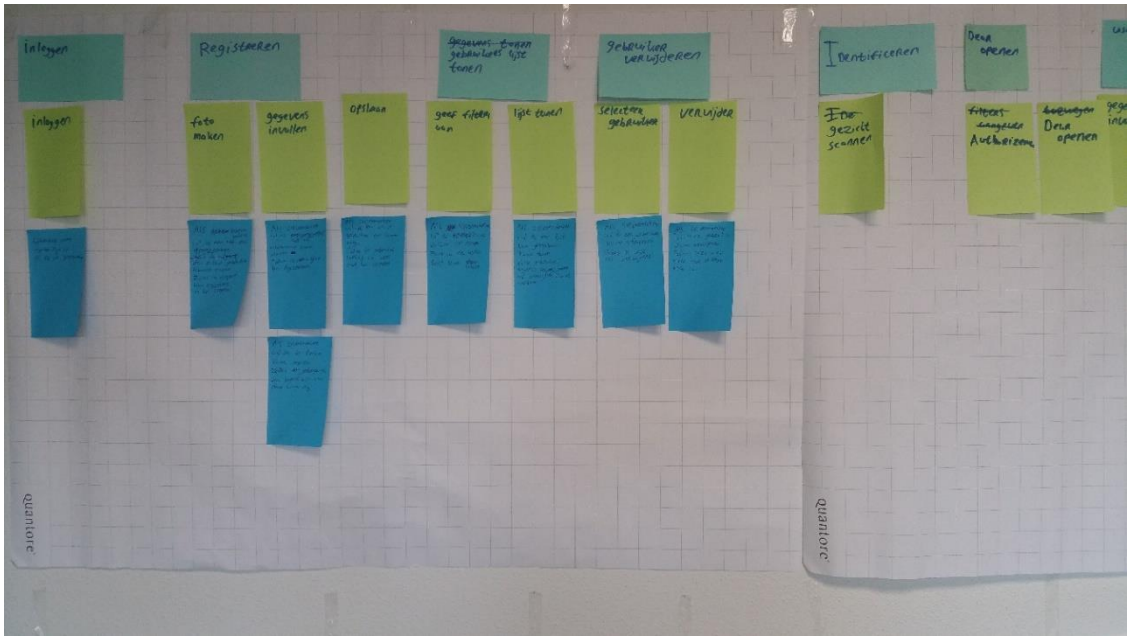
4.3.1 User story map

Bij een user story map maak je op verschillende niveaus duidelijk wat je systeem moet kunnen. Helemaal bovenaan hang je briefjes op om een heel grof beeld te

krijgen van welke acties het systeem moet kunnen. In mijn geval heb ik daar functies hangen als inloggen en registreren.

Daaronder hangen briefjes waarmee je duidelijk maakt welke stappen er nodig zijn om deze functionaliteiten te kunnen uitvoeren. Om iemand te registreren binnen een systeem met gezichtsherkenning, moet je bijvoorbeeld eerst een foto maken, vervolgens de persoonsgegevens invoeren en als laatste de gegevens opslaan. Al deze stappen hangen onder het briefje registreren.

Als je dan nog een stap dieper gaat kijken ga je per actie definiëren welke requirements gesteld worden aan die acties. Wanneer dit helemaal gedaan is heb je een duidelijk beeld van hoe je applicatie moet gaan werken. Deze map mag gaandeweg uitgebreid worden met meer acties of functionaliteiten.



Figuur 4: user story mapping

Bij het lezen van een user story map moet er eigenlijk een verhaal verteld worden. De functionaliteiten en de acties moeten chronologisch opgesteld zijn. Het zou bijvoorbeeld onlogisch zijn om eerst een deur open te doen en vervolgens die persoon te registreren. Het idee daarachter is dat een opdrachtgever of een product owner die geen of weinig technische kennis heeft toch gemakkelijk mee kan praten over de requirements. Hierdoor verhoog je het wederzijds begrip en zorg je ervoor dat je minder snel iets bouwt waar de product owner niks aan heeft.

Ik heb twee user story maps gemaakt. Eén om ervoor te zorgen dat een gebruiker geregistreerd kan worden en één om een gebruiker toegang te kunnen verlenen tot een ruimte.

Bij het maken van de user story maps heb ik in overleg met de opdrachtgever besloten dat het PoC geen systeem hoeft te krijgen dat gebruikersgegevens kan wijzigen en verwijderen. Het toevoegen van een gebruiker zal alleen worden gerealiseerd wanneer daar tijd voor is. Daarom heb ik ervoor gekozen een user story map te maken voor het identificeren en autoriseren van een gebruiker en voor het registreren van een gebruiker.

4.3.2 Wat is de betrouwbaarheid per techniek?

Om te kijken naar de betrouwbaarheid van de technieken, heb ik gekeken naar wat de False acceptence rate (FAR) en de false rejection rate is (FRR). Deze cijfers geven aan hoe groot de kans is dat iemand foutief geaccepteerd wordt of foutief geweigerd wordt. Hierbij geldt dat hoe hoger het rechter getal hoe minder vaak dit gebeurt.

Uit het antwoord op deelvraag twee, wat is de betrouwbaarheid per techniek, kwam de volgende tabel:

Identificatiemiddel	False acceptance rate	False rejection rate
Irisscanner	1 op 151.000 (Daugman, 1993)	1 op 128.000 (Daugman, 1993)
Gezichtsherkenning	1 op 1.000.000 (Fingertec, 2009)	1 op 100 (Fingertec, 2009)
Stemherkenning	1 op 7 (Beritelli & Spadaccini, 2012)	1 op 4 (Beritelli & Spadaccini, 2012)
RFID	0	0
Beacons	0	0

Uit de onderzoeksgegevens blijkt dat de irisscanner een kleine kans heeft dat er iemand foutief wordt herkend of foutief word geweigerd. Bij gezichtsherkenning valt op dat de FAR laag is maar dat de FRR wel vrij hoog uitvalt.

Verder werd duidelijk dat stemherkenning in ieder geval af zou vallen. Eén op de zeven wordt door stemherkenning foutief toegelaten en één op de vier zal foutief niet toegelaten worden. Door de lage nauwkeurigheid van stemherkenning heb ik besloten dat deze optie niet verder onderzocht zal worden.

RFID is, net als bij de vorige deelvraag, de beste oplossing. Dat komt doordat wanneer er een id verzonden word deze één op één vergeleken kan worden met het id in de database. Hierdoor zou het in principe nooit fout mogen gaan.

4.3.3 Het privacy probleem

Ik had ook een gesprek met de privacy adviseur gepland. Ik had deze afspraak gemaakt, omdat ik, op aanraden van een mede-afstudeerder, wilde kijken welke maatregelen ik moest nemen zodat het systeem zou gaan voldoen aan de wet bescherming Persoonsgegevens.

Tijdens dit gesprek werd duidelijk dat het in ieder geval zeer moeilijk zou worden om aan de wet te voldoen. De adviseur legde uit dat biometrische gegevens in de zwaarste categorie vallen, categorie drie. Dit houdt in dat deze in principe niet mogen worden verwerkt behalve onder hele strenge voorwaarden.

1. Er moet een noodzaak zijn of er moet toestemming worden gegeven om iemands persoonsgegevens te gebruiken. Omdat mijn systeem in een bedrijf gebruikt gaat worden en de gebruikers werknemers zijn, mag er geen toestemming gevraagd worden. Dat komt doordat de relatie tussen een werknemer en een werkgever niet gelijk aan elkaar is. Daardoor kan het zijn dat wanneer de werknemer geen toestemming geeft dat hij niet aangenomen wordt of dat hij geen opslag krijgt.
   Er zal dus een noodzaak moeten zijn waarom de biometrische gegevens verwerkt moeten worden. Gebruiksgemak zal niet aanvaard worden als een geldige reden om de noodzaak te verantwoorden en er zal dus een andere reden bedacht moeten worden waarom deze gegevens nodig zijn.

2. Er is geen andere techniek mogelijk die minder inbreuk maakt op iemand zijn privacy. In mijn geval is er wel een techniek beschikbaar welke minder inbreuk maakt op iemand zijn privacy dan een irisscanner, namelijk RFID. Hierbij zijn geen biometrische gegevens nodig.
3. De beveiliging van de gegevens moeten goed geregeld zijn. Dit is een probleem dat goed op te lossen is.

Na overleg met mijn opdrachtgever hebben we besloten om door te gaan met het onderzoek, maar wel een extra deelvraag in het onderzoek op te nemen. In de volgende sprint moest ik een deelvraag over privacy bedacht hebben. Wel zou ik het gesprek met BPI door laten gaan. Hier had ik voor gekozen, omdat zij misschien een oplossing hebben voor het privacy probleem. Verder had ik afgesproken dat ik het vooronderzoek zou uitbreiden. Ik zou gaan kijken of beacons of QR-codes niet alsnog meegenomen konden worden in het onderzoek.

4.3.4 Sprintreview

Om de derde deelvraag, wat zijn de kosten per techniek, te beantwoorden moest ik een indicatie van de prijzen voor de verschillende technieken weten. Veel prijzen waren online te vinden, alleen die van de irisscanners waren moeilijker te vinden. Hierom besloot ik een bedrijf te contacteren om de prijzen te achterhalen. Ik heb het Nederlandse bedrijf BPI gecontacteerd en ze wilden me graag verder helpen. Ze wilden het product graag laten zien en hier bij Info Support gaan kijken naar de huidige situatie. Daarom had ik een afspraak gemaakt voor de week daarop.

Dit had echter wel gevolgen voor mijn sprint. Deze kon ik hierdoor niet afkrijgen, omdat ik de deelvraag over de kosten niet kon beantwoorden en daardoor dus ook de hoofdvraag niet kon beantwoorden.

4.4 Sprint 4, afronding onderzoek

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Hoofdvraag beantwoorden	6
Sprint 3 beschrijven in het afstudeerverslag	3
Onderzoeksplan bijwerken	1
Vooronderzoek uitbreiden	8
Architectuur scope bepalen	2
Architectuur requirements beantwoorden	2
Privacy deelvraag beantwoorden	8
Totaal aantal punten	32

4.4.1 Demonstratie van de irisscanner

In deze sprint heb ik een gesprek gehad met een vertegenwoordiger van BPI. Dit bedrijf verkoopt allerlei identificatie hardware en software. Ik had een gesprek aangevraagd om wat meer te weten te komen over de irisscanner. Tijdens dit gesprek heeft de vertegenwoordiger een demonstratie gegeven van de irisscanner. De snelheid waarmee hij iemand herkend was sneller dan ik dacht. Het herkennen van een persoon duurt nog geen halve seconde, wanneer iemand niet herkend wordt duurt het echter een stuk langer. De irisscanner blijft dan twee seconden zoeken waarna hij een signaal geeft dat de gebruiker niet herkend wordt.

In dit gesprek kwam ook een mogelijke oplossing naar voren waarmee ik mogelijk wel biometrische gegevens mag gebruiken. Hij stelde voor om de biometrische gegevens op een pasje of op iemand zijn telefoon te zetten. Hiermee blijft de gebruiker eigenaar van zijn gegevens en zou het volgens hem mogen.

Dit wilde ik natuurlijk verifiëren bij de privacy adviseur van Info Support. Hij vertelde dat het dan alsnog niet mag, omdat de gegevens nog worden verwerkt door Info Support zelf.

4.4.2 Uitbreiding van het vooronderzoek

Naar aanleiding van het eerste gesprek met de privacy adviseur heb ik gekeken of er nog andere mogelijkheden waren om iemand te identificeren. Ik keek naar technieken die ik op het begin niet in het onderzoek mee had genomen. Het gaat dan om de technieken beacons en QR-codes.

Ik heb opnieuw gekeken of er niet alsnog een mogelijkheid is om een van deze technieken te gebruiken. Hierdoor kwam ik tot de conclusie dat QR-codes echt niet werkbaar zijn, maar dat beacons wel een optie kunnen zijn. Het nadeel dat je een actief apparaat bij je moet dragen zal dan maar voor lief genomen moeten worden.

Doordat ik een nieuwe te onderzoeken techniek heb opgenomen in mijn onderzoek, moest ik ook de deelvragen bijwerken. Hieruit kwam naar voren dat de beacons op het gebied van gebruiksvriendelijkheid minder goed scoorde dan RFID. Dit kwam voornamelijk doordat beacons een interval hebben van tien seconden. Deze interval kan sneller, maar dat betekent dat de batterij minder lang mee zal gaan. Dit is weer niet handig omdat de levensduur van een batterij in de beacons ongeveer één jaar is. Op het gebied van betrouwbaarheid scoort beacons even goed als RFID.

De privacy deelvraag heeft duidelijk gemaakt dat wanneer dit systeem geïmplementeerd moet gaan worden, alleen een RFID of een beacon oplossing gebruikt zou kunnen worden. Dit komt doordat de RFID en beacon oplossing minder privacygevoelige informatie nodig hebben dan de biometrische herkenning. Daardoor zal volgens de wet automatisch de minder privacy schendende manier gekozen moeten worden.

4.4.3 Wat zijn de kosten per techniek?

De laatste deelvraag ging over de kosten van de technieken. Hierbij is alleen rekening gehouden met de aanschaf kosten per deur. Twee technieken hebben naast deze kosten ook nog kosten per medewerker. Hierbij moet gedacht worden aan pasjes of aan draagbare beacons. Hieruit kwam de volgende tabel:

Identificatiemiddel	Per deur	Per medewerker
Irisscanner	€1750,-	€0,-
Gezichtsherkenning	€30,-	€0,-
RFID	€450,-	€3,-
Beacons	€20,-	€0-5,-

Uiteindelijk heb ik in samenspraak met mijn opdrachtgever gekozen voor RFID. De keuze voor RFID kwam doordat ten eerste de irisscanner, gezichtsherkenning en stemherkenning al af waren gevallen, omdat deze volgens de wet niet mogen. Daardoor werd de keuze beperkt tot RFID of beacons.

4.4.4 Gesprek met Nedap

Het is natuurlijk niet wenselijk dat het RFID signaal op een afstand van 100 meter uitgelezen kan worden. Dit zou namelijk betekenen dat alle deuren binnen het Info Support gebouw open gaan wanneer er iemand zich in het pand bevind.

De readers die er nu hangen kunnen slechts op enkele centimeters afstand pasjes uitlezen. Daarom ben ik gaan zoeken naar RFID-readers die op een grotere afstand dan enkele centimeters pasjes kunnen lezen, maar een kleinere leesafstand hebben dan 100 meter. Hierdoor kwam ik uit bij Nedap, dit bedrijf kon een reader leveren dat pasjes kon uitlezen tot twee meter afstand.

Bij Nedap heb ik een demonstratie gekregen van deze reader. Daaruit werd duidelijk dat het systeem heel gevoelig was voor externe factoren. Wanneer het pasje vastgehouden werd of wanneer het pasje in je broekzak zat, was het bereik al veel minder groot. Het bereik slinkt dan tot enkele centimeters, verder werkte het systeem goed. Hierdoor besefte ik mij dat wanneer dit systeem in gebruik genomen gaat worden de pasjes wel op een bepaalde manier gedragen moeten worden.

4.4.5 Conclusie van het onderzoek

Zoals eerder al gezegd is heeft beacons een heel groot nadeel. Er kan geen gebruik gemaakt worden van de mobiele telefoon van de werknemers, omdat systeembeheer als eis had gesteld dat het identificatiemiddel geen privé apparaat mocht zijn. Daardoor moesten de werknemers een beacon bij zich gaan dragen. Deze draagbare beacons hebben een batterijlevensduur van tussen de twee en drie maanden bij een interval van 10 seconden. Dat betekent dat de beacons elke twee maanden vervangen moeten worden en daarnaast kan het betekenen dat je in het slechtste geval 10 seconden moet wachten totdat een deur open gaat.

Omdat dit niet erg handig is en na verloop van tijd ook duurder dan RFID is er gekozen voor RFID. Wel moest er nog een ander probleem opgelost worden met deze techniek.

Systeembeheer had namelijk als eis gesteld dat de deuren niet zonder reden open mochten gaan. Dit zou bij het gebruik van RFID mogelijk wel gebeuren. Wanneer er een pasje binnen de leesstraal van de reader komt zou de deur in principe open gaan. Dit is echter niet altijd gewenst. Het kan voorkomen dat iemand gewoon langs een deur loopt om naar een andere deur te gaan. In dit geval zullen uiteindelijk beide deuren open gaan terwijl de gebruiker alleen door de laatste deur wil.

Om dit probleem op te lossen is er gekozen voor een tweevoudige oplossing. Als eerste moest er een vertraging in het systeem gebouwd worden. Deze vertraging zal ervoor zorgen dat wanneer iemand gezien wordt door de reader er niet gelijk geïdentificeerd zal gaan worden, maar dat er gewacht wordt. Wanneer die persoon na enkele seconden nog steeds zichtbaar is voor de reader, zal hij waarschijnlijk door de deur willen en zal het proces om dat voor elkaar te krijgen in werking gezet kunnen worden.

Als tweede maatregel om dit probleem te voorkomen heb ik besloten om een camera op te hangen die kan detecteren of er een gezicht te zien is. Het idee is dat de persoon die naar binnen wil even omhoog zal moeten kijken om aan te geven dat hij naar binnen wil.

Omdat ik twijfelde of ik van de privacy wetgeving gebruik mocht maken van een camera had ik aan de privacy adviseur gevraagd of het misschien mogelijk was om

een camera te gebruiken om te kijken of iemand ergens naar binnen wil en de identificatie via een RFID-pasje te laten verlopen. Dit was geen probleem zolang de beelden van de camera maar niet opgeslagen zouden worden.

5 Bepalen van de architectuur

5.1 Sprint 5, grote architecturale keuzes

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Sprint 4 beschrijven in het afstudeerverslag	4
Privacy deelvraag beantwoorden	2
Architectuur, functional viewpoint maken	1
Onderzoek RFID	4
Event bus opzetten	2
Architectuur, information viewpoint maken	3
Buildstraat aanmaken	2
Ontwerp maken voor de front-end identificatie service	4
Development viewpoint maken	8
Totaal aantal punten	30

In deze sprint wilde ik het architectuur rapport zover af hebben dat ik wist welke tools ik moest gebruiken en hoe het gehele plaatje eruit zou komen te zien. Vervolgens wilde ik een ontwerp gaan maken voor het gedeelte dat de pasjes gaat registreren en wanneer er dan nog tijd over was, wilde ik dit gedeelte ook daadwerkelijk gaan bouwen.

5.1.1 Keuze voor de architectuur

De applicatie moest een robuuste applicatie worden. Het is namelijk zeer vervelend wanneer de deuren niet meer open gaan, doordat mijn applicatie niet werkt. Ook moest de applicatie gemakkelijk uitbreidbaar zijn. De applicatie moest uit te breiden zijn met bijvoorbeeld een systeem dat bijhoud waar alle werknemers zich bevinden.

Ik keek naar een microservice architecture, Enterprise Service Bus (ESB) en een monoliet.

5.1.1.1 Monoliet

Een monoliet is een applicatie die de gehele workflow kan uitvoeren. In het geval van dit project zou het registreren en aanpassen van gebruikers, het identificeren van een gebruiker, het autoriseren van een gebruiker en het open doen van een deur in één systeem zitten.

Dit maakt het systeem erg groot en lastig uitbreidbaar, omdat er bij een kleine aanpassing of uitbreiding rekening gehouden moet worden met al deze componenten. Je zal niet gemakkelijk een component kunnen toevoegen zonder dat je daarvoor wijzigingen moet maken in de rest van het systeem.

5.1.1.2 Microservice architecture

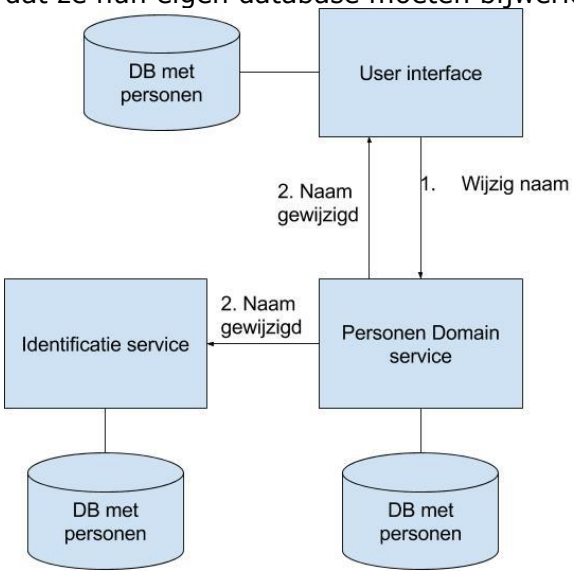
Er is voor Microservice architecture is geen duidelijke beschrijving van wat het inhoud. Daarom hield ik voor dit project de definitie van Fowler en Lewis (2014) aan. Fowler en Lewis geven aan dat een microservice architectuur vaak gebruik maakt van:

- Scheiding van processen
- Gedecentraliseerd databeheer
- Services zijn autonoom en onafhankelijk
- Slimme endpoints en domme pijplijnen

Bij een microservice architectuur is de applicatie opgedeeld in verschillende kleinere applicaties, genaamd services. Elke service heeft een eigen taak. Hierdoor krijg je een aparte applicatie die bijvoorbeeld iemand kan identificeren en een applicatie die bepaald of iemand de rechten heeft om door een deur te mogen. Door deze scheiding van processen kan er gemakkelijk een proces geüpdatet of vervangen worden, zonder dat de gehele applicatie geüpdatet hoeft te worden. Dat resulteert weer in een grotere beschikbaarheid van de applicatie in zijn geheel.

Ook is er een scheiding tussen services die data beheren en services die data verwerken. Zo heb je bijvoorbeeld een service welke de gebruikers data beheerd en een service die de gebruikers data gebruikt om iemand te identificeren. Elke service heeft zijn eigen database. Dit geeft als voordeel dat de services niet meer kunnen draaien wanneer een database niet bereikbaar is. Ook is het zo dat je dan verschillende database technologieën kan gebruiken binnen verschillende services. Zo kan je bijvoorbeeld in de ene service een NoSQL database gebruiken en in de andere service een relationele database.

Om ervoor te zorgen dat niet elke database die gebruikers gegevens bevat deze data kan aanpassen en je dus niet meer weet welke data nou correct is, zal er een aparte service gemaakt kunnen worden die verantwoordelijk is voor het beheren van de data. Deze service wordt ook wel domain service genoemd. Wanneer er een service is die bijvoorbeeld een naam van een persoon wil aanpassen, zal hij een reqeust moeten sturen naar de domain service van personen. Deze domain service zal de data in zijn eigen database aanpassen en vervolgens een bericht sturen naar de andere services dat ze hun eigen database moeten bijwerken. Dit is te zien in figuur 5.



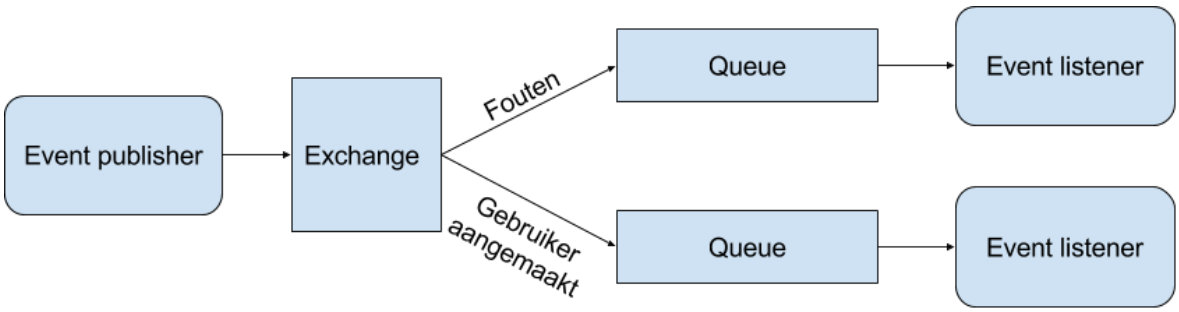
Figuur 5: wijzigen van data binnen microservices

Voor de communicatie tussen de verschillende services zal er een lichtgewicht protocol nodig zijn. Volgens Fowler wordt er vaak gekozen tussen twee verschillende protocollen, HTTP request-response, zoals REST, en een event bus. Hetgeen wat deze twee protocollen gemeen hebben is dat er zo min mogelijk logica in het communicatie mechanisme is.

Een event bus is een stukje software dat ervoor zorgt dat alle berichten op de goede plek aankomen. Bij een event bus kunnen er vier onderdelen onderscheiden worden. Een event publisher, een exchange, een queue en een event listener.

De event publisher is een stuk software dat je zelf zal moeten schrijven. Hiermee zorg je ervoor dat je applicatie events naar de event bus kan sturen. De exchange zit in de event bus. In de exchange komen alle events binnen en hij zal ze vervolgens naar de juiste queues routen. Dit kan doordat elke queue kan aangeven naar welk type events ze willen luisteren. In figuur 6 kan je zien dat er één queue naar events met type 'fouten' luisterd en de andere naar de events 'gebruiker aangemaakt'.

Wanneer er een event in de queue staat kan hij weer uit de queue gehaald worden door een event listener. Deze event listener is een stuk software dat zelf geschreven moet worden in de ontvangende applicatie. Deze listener zal bepalen wat er moet gebeuren wanneer er een bepaald event binnen komt.



Figuur 6: werking event bus

Het voordeel van een event bus ten opzichte van HTTP request-response is dat alles asynchroon gebeurd. Je hebt dus geen downtime van je applicatie omdat deze een request aan het verwerken is. Het nadeel is dat dit erg veel complexiteit met zich meebrengt. Hoe moet je bijvoorbeeld omgaan met services die niet beschikbaar zijn, of wat moet er gebeuren wanneer er een exceptie optreedt binnen een andere service tijdens zo'n asynchrone taak.

5.1.1.3 Enterprise Service Bus (ESB)

Een ESB architectuur lijkt erg op die van de microservice architectuur. Je hebt verschillende services die in verschillende talen geschreven mogen zijn. Het grote verschil met de Microservice architectuur is dat je bij een ESB je communicatie via een service bus laat gaan. Deze service bus verschilt met de event bus doordat er veel logica in zit. Dit maakt de service bus wat zwaarder dan de event bus.

De service bus doet in ieder geval de volgende zaken extra:

- Een ESB zorgt voor de beveiliging van de communicatie
- Een ESB regelt de transformatie van de data
- Een ESB monitort de serviceaanvragen en rapporteert hierover

Een ESB vergt veel configuratie doordat hij heel veel doet. Door al deze configuratie wordt het onderhouden van de applicatie een stuk complexer. Er zal veel rekening gehouden moeten worden met de verschillende services. Wanneer er een nieuwe service gemaakt is, zal deze geregistreerd moeten worden bij de ESB.

5.1.1.4 Conclusie

Een monoliet is lastiger uit te breiden is dan een microservice architecture en genoot daardoor niet mijn voorkeur. Ook een ESB is lastiger uit te breiden dan een

microservice architecture, omdat wanneer je een nieuwe service toevoegt aan een ESB je ook de configuratie moet aanpassen. Ik wilde dat het systeem gemakkelijk uitbreidbaar is, omdat er nog andere systemen binnen Info Support bestaan die ook gebruik kunnen gaan maken van de informatiestromen die binnen dit systeem gebruikt worden.

Zo wordt er door een systeem dat bijhoudt waar iedereen zich in het Info Support gebouw bevindt, gebruik gemaakt van de paslezer bij de deuren. Wanneer een pas gescand wordt bij een deur wordt dit ook in een aanwezigheidsregistratiesysteem geregistreerd. Dit systeem houdt bij waar jij je in het gebouw bevindt. Wanneer het systeem dat ik gebouwd heb in gebruik genomen wordt, zal het aanwezigheidsregistratiesysteem ook gekoppeld moeten worden aan mijn systeem. Met microservices is dat gemakkelijk te doen door dat systeem te laten luisteren naar bepaalde events die hij nodig heeft.

Uiteindelijk viel mijn keuze dus op de microservice architecture, omdat deze gemakkelijk uit te breiden is en

5.1.2 Communicatie tussen services

Zoals ik eerder heb vermeld, kan er gekozen worden tussen twee protocollen voor de communicatie tussen de verschillende microservices.

- HTTP request-response
- Een event bus

Bij HTTP request-response kan men denken aan REST. Het voordeel van REST is dat het een synchrone actie is en je dus kan wachten op de response, omdat je meestal direct een response zal krijgen. Voor mijn project zal dat als voordeel geven dat de gebruiker zo snel mogelijk te weten krijgt dat hij niet naar binnen mag wanneer een service niet bereikbaar is.

De event bus werkt asynchroon en kan dus meerdere berichten tegelijk afhandelen. Dit geeft als voordeel dat de verschillende services altijd bereikbaar zijn als ze draaien. Er wordt simpelweg een nieuwe thread gestart om een bericht te kunnen afhandelen. Een ander voordeel van de event bus is dat de verschillende services helemaal geen kennis van elkaar hoeven te hebben. Hierdoor hoeft er in de services zelf geen rekening gehouden te worden met het format request dat je stuurt.

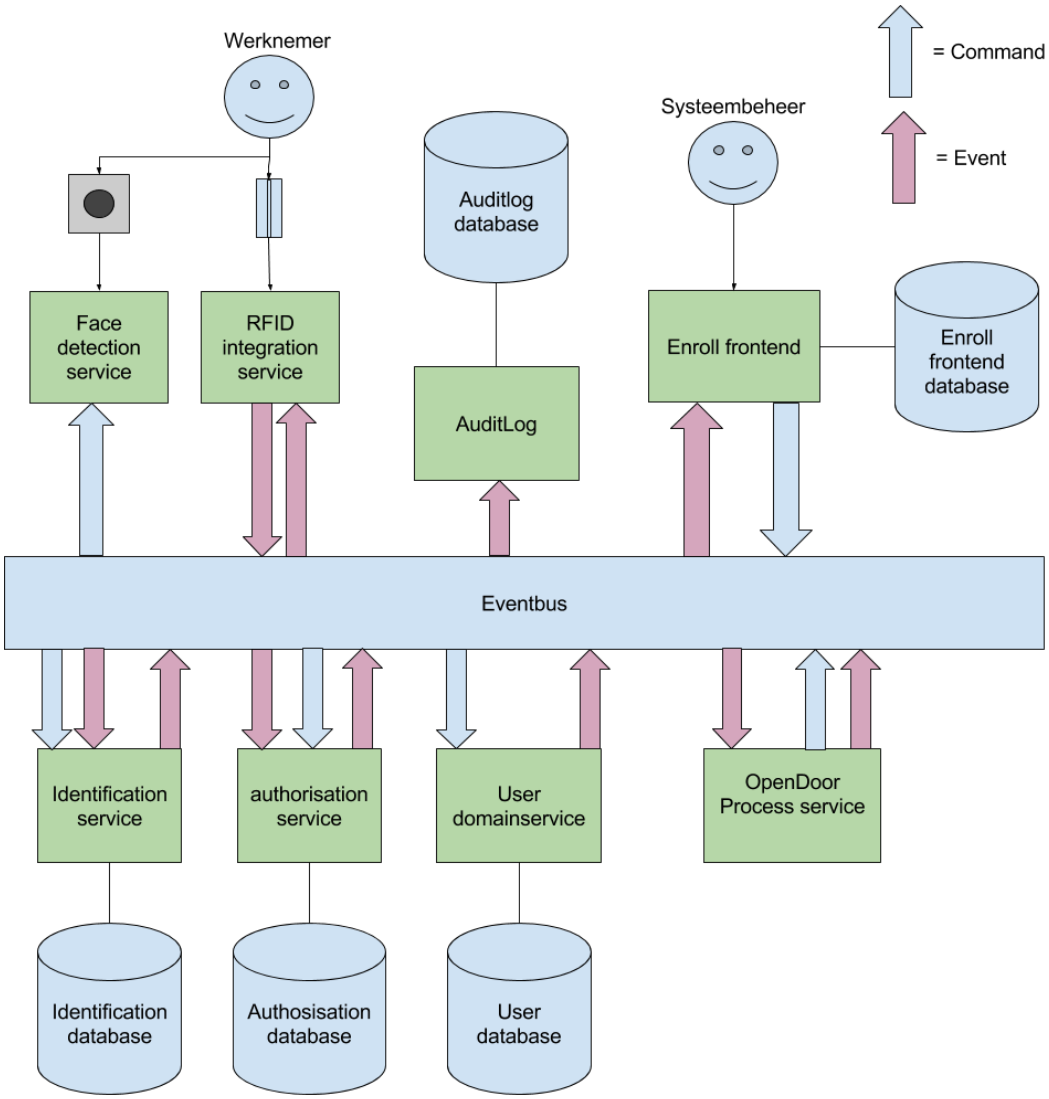
Dit betekent echter niet dat je zomaar een event aan mag passen. Wanneer je dit doet bestaat namelijk de kans dat er onderdelen van het systeem niet meer werken, doordat hij het event niet meer begrijpt. Om dit te voorkomen is het de bedoeling dat events niet meer aangepast mogen worden nadat ze in productie zijn genomen. Wanneer er toch een event aangepast moet worden, zal er een nieuw event aangemaakt moeten worden en het oude event moet vervolgens uit gefaseerd worden. Op die manier zorg je ervoor dat alle onderdelen van je systeem blijven werken.

Omdat ik ervoor wilde zorgen dat het systeem gemakkelijk uit te breiden is, vooral met het oog op het andere systeem dat registreert waar werknemers zich bevinden, wilde ik ervoor zorgen dat dit zo gemakkelijk mogelijk kon. Hierdoor wilde ik werken met de event bus, omdat deze ervoor zorgt dat de services geen kennis van elkaar hoeven te hebben.

Bij de communicatie via een event bus kan onderscheid gemaakt worden tussen events en commands. Een event geeft aan dat er iets gebeurd is en een command geeft aan dat er iets moet gebeuren. Een voorbeeld van een command is bijvoorbeeld 'CreëerPersoon' met in het command alle gegevens van de persoon. Een command is gericht aan één service en zet vervolgens op die service de actie om een persoon toe te voegen aan een service in werking. Een event is bedoeld aan alle services die geïnteresseerd zijn in de data die in dat event staat en wordt verstuurd wanneer er iets is gebeurd. Een voorbeeld hiervan is 'PersoonGecreëerd' met daarin alle informatie van de persoon. Wanneer er een persoon gecreëerd is in een bepaalde service zal dit event op de event bus gezet worden. Dit kan vervolgens opgepakt worden door de services die een gedeelte of alle informatie uit dat event willen.

5.1.3 De initiële architectuur

Van systeembeheer mocht ik geen gebruik maken van het huidige systeem. Ik moest dus een compleet nieuw systeem bedenken en realiseren. Ik ben begonnen met het maken van het functional viewpoint binnen mijn architectuur rapport. Hiervoor ben ik gaan nadenken welke services ik zal gaan onderkennen in het systeem en wat deze services dan zouden moeten doen.



Figuur 7: initiële architectuur

Ik had een enroll front-end erin gezet voor het beheer van de gebruikers van het systeem. Hierin kan je nieuwe gebruikers toevoegen, verwijderen en rechten toewijzen aan gebruikers.

De auditlog is een service die alle events opvangt, hij zal ze allemaal opslaan in zijn database. Deze data kan uitgelezen worden als er een nieuwe instantie, van bijvoorbeeld de user domain service, wordt opgestart. Hij kan dan alle events naar die service sturen zodat hij zijn database kan opbouwen uit al deze events.

De RFID integration service zal een integratie service zijn die voor de verbinding met de RFID reader zal zorgen. Hij zal de berichten die uit de reader komen omzetten naar berichten die begrijpelijk zijn voor dit systeem.

De Face detection service zal moeten kijken wanneer er een signaal van de RFID reader is dat er iemand gevonden is, of er een gezicht te zien is. Hiermee moet aangegeven worden of er iemand ook daadwerkelijk naar binnen wil of dat die persoon gewoon toevallig langsloopt.

De openDoor process service zal zorgen dat een persoon geïdentificeerd zal worden, geautoriseerd wordt en vervolgens de deur open doen. Hij zal dus de identificatie service en de authenticatie service aanroepen.

Als laatste heb je de identificatie service en de autorisatie service. De identificatie service zorgt voor het identificeren van een gebruiker en de autorisatie service zorgt voor het autoriseren van gebruikers. Deze services zullen aangeroepen worden door de Open Door process service. Ik heb de identificatie en autorisatie van gebruikers van elkaar gescheiden, omdat dit twee aparte handelingen zijn.

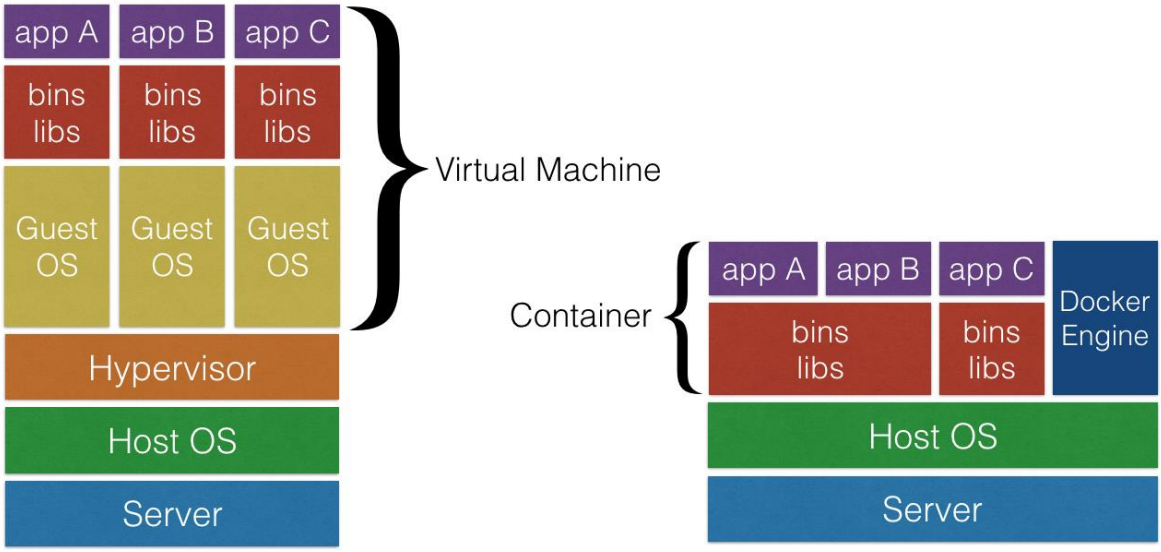
5.1.4 Docker

Ik wist nog niet op wat voor server mijn applicatie zou gaan draaien en of het altijd op eenzelfde soort server zal blijven draaien. Om te voorkomen dat mijn applicatie restricties legt aan het OS waarop het kan draaien koos ik ervoor om gebruik te maken van Docker.

Docker is een vorm van virtualisatie. Het zorgt ervoor dat je geen rekening meer hoeft te houden met de onderliggende server en softwareversies. Tevens het zorgt ervoor dat er een geïsoleerde omgeving, ook wel container genoemd, wordt opgezet waarbinnen je je applicatie kan draaien. Het voordeel van deze containers is dat je in elke container een andere softwareversie kan draaien zonder problemen.

Je kan bijvoorbeeld met Docker SQL-server 2014 en SQL-server 2015 in verschillende containers zonder problemen draaien. In mijn project kan het handig zijn omdat elke service een database kan krijgen op basis van de behoefte van de service.

Zoals ik eerder al zei is Docker een vorm van virtualisatie. Ondanks dat het veel weg heeft van een virtuele machine, verschilt het toch op een aantal punten. Zo heb je voor meerdere containers maar één Docker nodig. Er hoeven dus niet meerdere virtuele machines te worden aangemaakt om verschillende containers te draaien. Ook probeert Docker ervoor te zorgen dat er zoveel mogelijk library’s gedeeld worden. Dit zorgt ervoor dat er minder resources nodig zijn om de virtualisatie tot stand te laten komen.



Figuur 8: Werking van Docker

In figuur 8 is te zien wat het verschil is tussen virtuele machines en een Docker container. Waar de virtuele machine (VM) niks deelt na de hypervisor, deelt Docker zoveel mogelijk tussen verschillende Docker containers. Wanneer jij voor twee verschillende containers SQL-server 2014 nodig hebt, zal Docker slechts éénmaal de binaries opslaan. Deze zijn vervolgens toegankelijk voor alle containers die deze database gebruiken. Ook is er geen extra OS nodig per VM, alle Docker containers draaien op dezelfde Docker engine. Hierdoor heeft de host minder resources nodig en is Docker dus een stuk minder zwaar voor de server.

5.1.5 Development viewpoint

In het development viewpoint van mijn architectuur rapport wilde ik opnemen welke taal ik zou gaan gebruiken, welke tools en frameworks ik wilde gebruiken en welke componenten mijn services zouden krijgen.

5.1.5.1 Taal en framework

Ik koos ervoor om als taal C# in .NET CORE te gaan gebruiken. .NET Core is een taal die erg lichtgewicht is, doordat er in de basis niet veel bij zit. Je zal al je library's bij elkaar moeten verzamelen. Hierdoor krijg je wel dat je applicaties kan bouwen die niet meer dan het nodige bevatten en dus kleiner kunnen zijn dan andere applicaties geschreven in het oudere .NET framework of in JAVA.

Het nadeel van .NET CORE is dat deze taal erg nieuw is en het dus mogelijk is dat sommige functies missen die in het vorige framework van Microsoft, ASP .NET MVC 5, wel zitten. Hierom heb ik eerst gekeken of de library's waarvan ik van te voren wist dat ik die nodig had voor het project, al beschikbaar waren voor .NET CORE. Dit bleek het geval en dus besloot ik dat het risico dat er functionaliteiten zouden gaan missen niet opwoog tegen het voordeel van .NET CORE voor de kleinere footprint. Wanneer er toch iets miste moest ik ervoor zorgen dat ik de functionaliteit zelf schreef of ik moest er omheen proberen te werken.

5.1.5.2 Event bus

Als implementatie van de event bus heb ik gekozen voor RabbitMQ. RabbitMQ is een event bus die JSON berichten distribueert naar verschillende queues. Doordat RabbitMQ JSON gebruikt is het mogelijk om verschillende talen via RabbitMQ te laten

communiceren. Om objecten gemakkelijk te kunnen serialiseren en vooral te deserialiseren heb ik gebruik gemaakt van een laag om RabbitMQ heen, EasyNetQ. EasyNetQ is een laag die, binnen .NET, een verbinding tussen je applicatie en de event bus kan opzetten en ervoor zorgt dat je berichten goed geserialiseerden gedeserialiseerd worden.

5.1.6 Ontwerp voor RFID-integration service

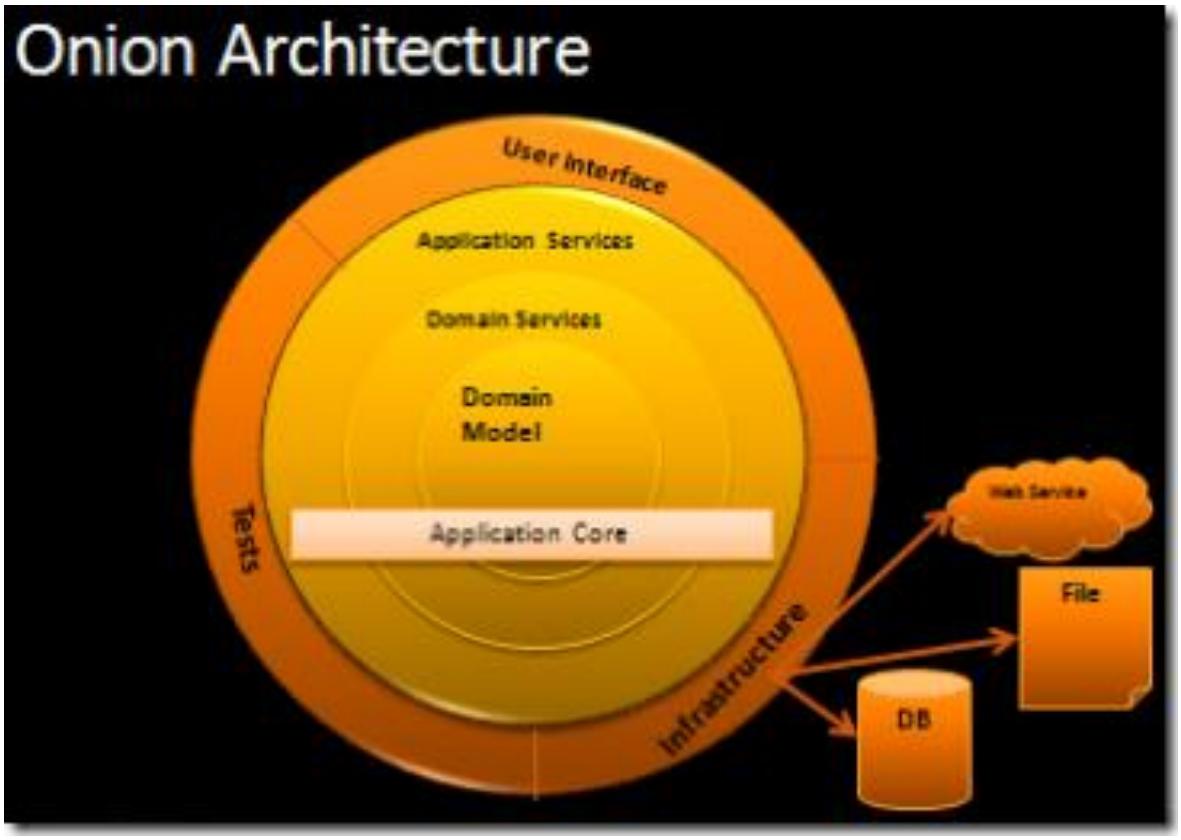
Ik maakte in deze sprint ook een ontwerp voor de RFID integration service. Het gaat dan in het architectuur plaatje van figuur 7 om het volledige blokje genaamd RFID-interface. Deze service moet een event te gooien wanneer hij een signaal krijgt van de RFID reader.

Om ervoor te zorgen dat mijn applicatie goed te testen was wilde ik zoveel mogelijk weg kunnen mocken. Hiervoor had ik het design pattern inversion of controll nodig. Bij inversion of controll wordt de instantiering van klassen uit de desbetreffende klasse gehaald. Alle klassen die ik gebruik binnen een bepaalde klasse, behalve entiteiten, wilde ik via dependency injection injecteren. Om gebruik te kunnen maken van dependency injection, moet alles dat je wil injecten een interface bezitten. Daarom heb ik een IRegisterService aangemaakt voor RegisterService.

Ook heb ik het singleton pattern gebruikt. Ik wilde voorkomen dat er elke keer dat er een nieuwe connectie met de event bus geopend en gesloten wordt, omdat dit voor vertraging kan zorgen in de applicatie. Het singleton pattern bood hiervoor de uitkomst. Hierdoor heb ik maar één instantie van het gedeelte dat de event bus connectie regelt. Deze blijft bestaan totdat de applicatie afgesloten werd.

Als laatste design pattern dat geïmplementeerd werd in elke service, en dus ook in de RFID-integration service, is het Façade pattern. Om te voorkomen dat de bedrijfslogica direct aangesproken kan worden zal er één centraal aanspreekpunt moeten zijn. Dit zorgt ervoor dat wanneer er een wijziging in deze functionaliteit komt en bijvoorbeeld het return type veranderd je niet gelijk alle services hoeft aan te passen welke deze functionaliteit gebruikt. Je hoeft dan alleen nog maar het centrale aanspreekpunt aan te passen. Elke service krijgt een apart project dat als façade dient. Hier zal alle communicatie met de andere services dienen.

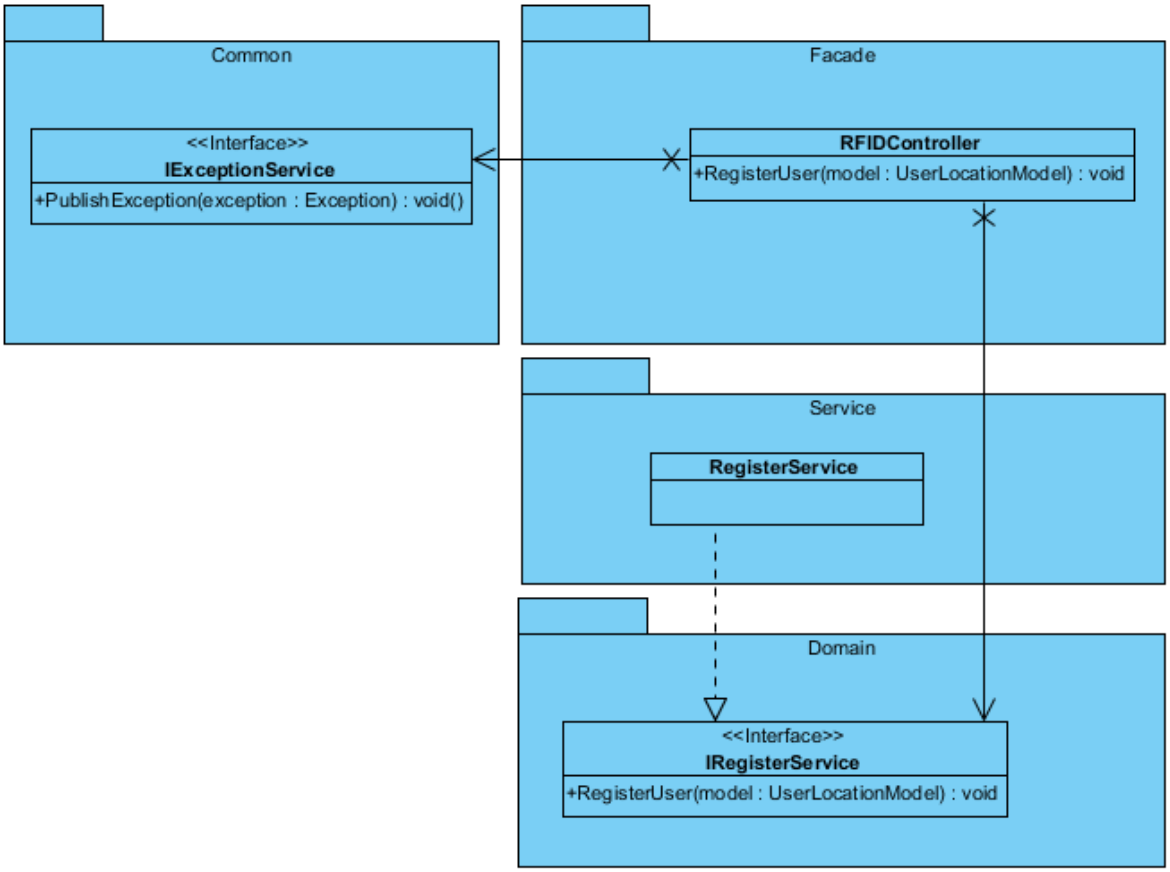
Ik koos ervoor om de microservices op te delen in verschillende delen gebaseerd op de onion architectuur. Dit model verdeelt een microservice in verschillende onderdelen. Het idee is dat er verschillende lagen in je applicatie ontstaan, hierdoor krijg je een scheiding tussen verschillende delen. Elke laag kent de laag die zich onder die laag bevindt, maar niet andersom.



Figuur 9: onion architectuur (Palermo, 2008)

In figuur 9 zijn de verschillende lagen goed te zien. Hier zie je in de buitenste laag, de User interface, infrastructure en de tests. Daaronder de application services, domain services en als laatst het domain model.

In het midden zit het domain model. Daar zullen alle entiteiten en interfaces geplaatst worden. Daaromheen zitten twee service lagen, deze zullen in de microservices samengevoegd worden omdat een microservice vrij klein is en geen twee service lagen nodig heeft. In deze service laag zal alle bedrijfslogica gebeuren. In de infrastructure laag komt de database repository. En als laatst is de user interface. Via deze laag zal de gebruiker kunnen communiceren met de applicatie. Dit model zal niet overal hetzelfde geïmplementeerd worden. Sommige services zullen bijvoorbeeld geen service laag bevatten, omdat er in de service laag niets hoeft te gebeuren.



Figuur 10: ontwerp RFID-integration service

In figuur 10 is te zien hoe ik het onion model in mijn microservices heb geïmplementeerd. Ik heb helemaal bovenaan een Façade, dit is het aanspreek punt voor andere services of in dit geval voor de RFID-reader. Daaronder heb je de service laag. In deze laag zit de implementatie van de service interface. Hierin bevindt zich alle bedrijfslogica. In dit geval zal daar dan dus het omzetten van de data die je binnen krijgt van de RFID-reader naar een event plaats vinden en dit event op de event bus zetten.

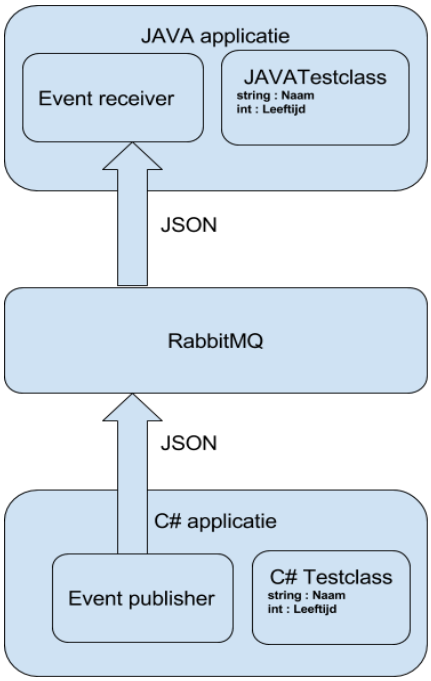
Helemaal onderaan zit het domain. Hierin zit de interface die gebruikt wordt voor de dependency injection. Als laatst is ook nog een Common package te zien. Dit is een zelfgemaakte library die ervoor zorgt dat bepaalde functionaliteiten gedeeld kunnen worden over verschillende services. Zo staat in deze library een klasse met een methode die ervoor zorgt dat de excepties die gegooid worden binnen mijn applicatie op de event bus gezet worden, dit is een keuze die ik pas in sprint 11 heb gemaakt. Hier dus meer over in sprint 11. Ook bevat de common package alle events. Meer over de common package in hoofdstuk 6.1.4.

5.2 Sprint 6, eerste oefening met RabbitMQ en Docker

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Sprint 5 beschrijven in het afstudeerverslag	4
Proof of concept maken RabbitMQ	6
User story map bijwerken	2
Feedback verwerken	4
Onderzoeksrapport definitief maken	4
Architectuur bijwerken met nieuwe microservice	2
Regulations perspective toevoegen	4
Evolution perspective toevoegen	4
Totaal aantal punten	32

5.2.1 Proof of concept RabbitMQ



Figuur 11: Architectuur PoC RabbitMQ

Mijn begeleider had de ervaring dat niet alle event bussen goed om konden gaan met het gebruik van verschillende talen voor verschillende services. Het werd dan in de ene taal zo geserialiseerd dat de andere service in de andere taal deze niet goed meer kon deserialiseren. Dit gaf als probleem dat een event niet goed of soms zelfs helemaal niet aankwam bij de andere service.

Als event bus had ik juist om die reden voor RabbitMQ gekozen. Volgens de documentatie van RabbitMQ kon deze event bus door verschillende talen tegelijk gebruikt worden. Daar wilde mijn opdrachtgever bewijs voor en vroeg mij om hier een proof of concept voor te maken.

Daarom wilde ik deze sprint met behulp van een proof of concept bewijs leveren dat RabbitMQ gebruikt kan worden om de communicatie te regelen tussen een JAVA en C# applicatie. Zoals te zien is in figuur 11, zal de C# applicatie een object parsen naar JSON en deze versturen naar de JAVA applicatie via RabbitMQ. Vervolgens zal de JAVA applicatie deze JSON moeten deserialiseren naar een object dat hij kent, waar dezelfde properties in zitten.

5.2.1.1 Werken met docker

Om ervoor te zorgen dat de applicatie in Docker ging draaien was het nodig om een Dockerfile aan te maken. Deze Dockerfile zorgt ervoor dat er een image gemaakt kan worden van mijn applicatie. Vervolgens wilde ik ervoor zorgen dat mijn applicatie in een netwerk zou komen. Door een netwerk aan te maken creëerde ik de mogelijkheid om verschillende Docker containers met elkaar te laten communiceren. Dit heb ik gedaan in een Docker-compose file, omdat je daarin kan definiëren met welk netwerk je verbinding wil maken. Dit kan ook buiten de Docker-compose om, door het bijvoorbeeld via een script te doen. Het voordeel van je volledige configuratie in de

Docker-compose te zetten is dat je maar een plek hebt om je configuratie aan te passen.

Met een Docker-compose file kan je alle microservices die je nodig hebt opstarten. Ook kan je hierin je configuratie voor je applicatie kwijt. Hierdoor hoef je niet elke microservice stuk voor stuk op te starten. In deze Docker-compose file kan je ook een netwerk aangeven. Met dit netwerk kan je vervolgens je verschillende Docker-containers met elkaar laten communiceren.

Om de event bus op te zetten maakte ik gebruik van Docker. Doordat ik Docker gebruikte, hoefde ik alleen maar in een Docker-compose file aan te geven welke image ik wilde downloaden en welke inloggegevens ik wil gebruiken om in te kunnen loggen op de event bus. Hierdoor zal Docker gaan zoeken op de website van Docker naar deze image en zal hij automatisch een container voor mij starten met daarin de event bus.

5.2.2 Sprintreview

Met het proof of concept over RabbitMQ bewees ik dat er geen problemen waren met de communicatie tussen een C# applicatie en een JAVA applicatie via de event bus RabbitMQ. De berichten werden in C# geserialiseerd en in JAVA weer op de goede manier gedeserialiseerd.

Doordat mijn JAVA nogal roestig was geworden, moest ik hier meer tijd aan besteden en was ik zoveel tijd kwijtgeraakt dat ik mijn sprint niet meer zou gaan halen. Het was me uiteindelijk niet meer gelukt om de story’s sprint 5 te beschrijven, regulations perspective toevoegen en evolution perspective toevoegen. Na overleg met de opdrachtgever leek het ons het handigst als deze items in de volgende sprint opgenomen zouden worden. Het was namelijk niet handig om het afstudeerverslag te ver achter te laten lopen, omdat op een gegeven moment de besluiten die je genomen hebt kunnen vervagen. Hierdoor wordt het moeilijker om het verslag goed te schrijven. Ook de perspectives werden in de volgende sprint opgenomen.

6 Programmeren en testen

6.1 Sprint 7, bouwen RDIF integration service

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Sprint 5 beschrijven in het afstudeerverslag	4
Sprint 6 beschrijven in het afstudeerverslag	6
Maken RFID integration service	4
Onderzoeksrapport definitief maken	4
Regulations perspective toevoegen	4
Evolution perspective toevoegen	4
Totaal aantal punten	28

6.1.1 Test driven development

Bij het bouwen van de applicatie koos ik ervoor om dit met behulp van test driven development (TDD) te doen. TDD is een programmeer techniek, waarbij je eerst een unit test schrijft en daarna pas de functionaliteit die bij de unit test hoort. Door TDD te programmeren wilde ik ervoor zorgen dat alle functionaliteiten getest worden om daarmee de kwaliteit van mijn code te waarborgen.

Figuur 12: TDD

Zoals te zien in figuur 12 heeft TDD drie fasen. Als eerste schrijf je een test, de test run je en deze moet vervolgens falen. Door de test te laten falen weet je zeker dat je test goed is. Vervolgens schrijf je zo min mogelijk code om de test te laten slagen. Als laatst refactor je de code, zodat de code ook begrijpelijk is.

Test driven programmeren was op het begin heel erg wennen. Ik was gewend om eerst mijn functionaliteit te maken en daarna pas mijn tests te schrijven. Nu moest ik van tevoren nadenken wat ik wilde testen en daarna zo min mogelijk code schrijven om mijn test te laten slagen. Een goed voorbeeld van TDD in mijn project is te vinden in tests die de database logica (repository) test, deze zal in hoofdstuk 6.8.2 gegeven worden.

6.1.2 Mocken van klassen

Om tests te schrijven had ik gekozen voor het test framework van Microsoft, MS Test. Om te voorkomen dat de gehele applicatie gaat werken wanneer ik één enkele functie probeer te testen, moest ik ervoor zorgen dat andere klassen niet gebruikt zouden worden. Ik moest er dus voor zorgen dat de methode dacht dat hij een bepaalde klasse had, maar in werkelijkheid een dummy klasse. Een dummy klasse wordt ook wel een mock genoemd.

Hiervoor gebruikte ik het mocking framework MoQ. MoQ kan op basis van interfaces een mock aanmaken. Ik kan zelf bepalen welke methodes aangeroepen kunnen worden door een test. Wanneer er een andere methode toch aangeroepen wordt, zal er een fout optreden en faalt mijn test. Het is handig om te controleren of alleen de methodes die jij verwacht te gebruiken ook daadwerkelijk aangeroepen worden.

Naast het aangeven welke methodes gebruikt mogen worden, kan je ook controleren hoe vaak een methode is aangeroepen en kan je aangeven met welke argumenten een methode aangeroepen kan worden. Door te controleren of de methode die je test ook daadwerkelijk alle methodes even vaak aanroept als jij verwacht, krijg je meer zekerheid dat je applicatie het verwachte gedrag vertoond.

Het aangeven met welke argumenten een methode aangeroepen kan worden, zorgt ervoor dat je zeker weet dat de methode aangeroepen wordt met de argumenten die jij verwacht.

6.1.3 Overload van de conversion operator

Binnen .NET heb je conversion operatoren. Je kan deze conversion operatoren overloaden. Hierdoor kan je entiteiten naar elkaar toe converteren. Je hebt twee verschillende conversion operatoren, implicit en explicit. Wanneer je een implicit operator hebt geschreven om bijvoorbeeld class Persoon naar class Mens te converteren, kan je vervolgens in je code het volgende doen:

```
Persoon persoon = new Persoon();
Mens mens = persoon;
```

Wanneer je een explicit operator hebt geschreven voor dezelfde class, moet je expliciet neerzetten dat je de conversie wil maken van Persoon naar Mens. Dit doe je met behulp van een cast:

```
Persoon persoon = new Persoon();
Mens mens = (Mens)persoon;
```

In de controller van mijn RFID-integration service krijg ik een model binnen. Ik wilde dat dit model impliciet geconverteerd werd naar een event, zodat ik deze direct op de eventbus kon zetten. Ik wilde dit met een implicit conversion doen zodat er bij een eventuele uitbreiding op de code deze conversie niet opnieuw hoeft te schrijven.

6.1.4 Common library

Bij het maken van een event bedacht ik mij dat ik op elke plek waar dat event opgevangen wordt, diezelfde class moest maken. Dit zou voor veel herhaling van code zorgen. Dat wilde ik voorkomen en bedacht daarom om een aparte library te maken met daarin alle events. Op deze manier hoefde ik alleen deze library te importeren in het desbetreffende project waardoor deze gelijk alle events kent. Ook wilde EasyNetQ dat exact dezelfde class gebruikt werd over verschillende services, zodat hij weet hoe hij de objecten moet deserialiseren.

6.1.5 Requlations perspective

In de regulations perspective heb ik de privacywetgeving opgenomen. Ik heb hierin een advies geschreven over hoe er voldaan kan worden aan de privacywetgeving. Dit was nodig omdat er in dit systeem met persoonsgegevens gewerkt zal gaan worden. Om dit te mogen moet er aan een aantal eisen voldaan worden. Ik gaf als advies dat het pasje zichtbaar gedragen moest worden en dat medewerkers aangespoord moesten worden om personen zonder een zichtbaar pasje aan te spreken. Hiermee zorg je ervoor dat de noodzaak, het buitenhouden van onbevoegde personen, ook legitiem is. Wanneer je dit niet zou doen kan iedereen door simpelweg achter iemand anders aan te lopen binnen komen en kan je je afvragen of het systeem wel nodig is.

6.1.6 Evolution perspective

In het evolutions perspective heb ik een blik geworpen op de toekomst van het systeem wanneer het ook daadwerkelijk gebruikt gaat worden. Ik heb hierbij vooral gekeken naar wat de mogelijkheden zijn van het uitbreiden van het systeem. Hier kwam uit dat door de microservice architectuur in principe alles aangesloten kan worden op het systeem.

6.1.7 Sprintreview

Tijdens deze sprint heb ik een werkende RFID integration service gemaakt. Deze RFID integration service kan een deur id en een gebruikers id ontvangen en dit omzetten naar een event dat verstuurd wordt naar de event bus. Doordat ik de RFID reader nog niet had moest ik dit gedeelte mocken. Dat mocken deed ik door een REST call te maken naar de service.

Naast de RFID integration service had ik ook mijn architectuur rapport bijgewerkt. Ik had het regulations perspective en het evolution persopective toegevoegd.

6.2 Sprint 8, logging

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Sprint 7 beschrijven in het afstudeerverslag	4
Sprint 7 feedback verwerken in het afstudeerverslag	1
Logging werkend krijgen op RFID-integration service	6
Open door process service toevoegen aan het Architectuur Rapport (optioneel)	4
Totaal aantal punten	15

6.2.1 Het implementeren van een logger

Tot nu toe was het niet mogelijk om fouten te loggen binnen mijn services. Om ervoor te zorgen dat problemen met een service terug gevonden konden worden, moest ik een logger implementeren. Hiervoor koos ik voor het serilog framework.

Ik wilde de logger via dependency injection aan de controllers en services geven en ik heb dan ook de logger geregistreerd in de startup file. Vervolgens heb ik in de controller gevraagd om de interface van de logger. En als laatst heb ik een Json file aangemaakt met daarin de configuratie van de logger. Ik heb hierin neergezet vanaf welk logging level hij logs moet opslaan, wat het type sink is en waar hij de files moet opslaan.

Ik koos voor een Json file voor mijn configuratie, omdat deze niet mee gecompileerd wordt met mijn code. Hierdoor kan ik gemakkelijk de configuratie aanpassen zonder dat de code opnieuw gecompileerd hoeft te worden. Dit geeft als voordeel dat je de configuratie kan aanpassen terwijl de applicatie al in de productie omgeving staat. Dit zorgde ervoor dat mijn applicatie beter te beheren was.

6.2.2 Sprintreview

De problemen met de logger duurde ongeveer anderhalve dag en besloeg ook een deel van sprint 9. Daardoor had ik het sprintdoel niet gehaald. Wel was het nog gelukt om de feedback te verwerken en sprint 7 te beschrijven.

6.3 Sprint 9, gebruikers identificeren

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Sprint 8 beschrijven in het afstudeerverslag	2
Sprint 8 feedback verwerken in het afstudeerverslag	5
Voortgangsverslag school maken	3
Bijwerken van architectuur met nieuwe inzichten voor commands	1
Documenten updaten afstudeerportal	1
Logging werkend krijgen op RFID-integrations service	6
Open door processservice toevoegen aan architectuur rapport	3
Identification service toevoegen aan architectuur rapport	3
Iemand kunnen identificeren	12
Totaal aantal punten	37

In deze sprint wilde ik ervoor zorgen dat gebruikers geïdentificeerd kunnen worden. Dit houdt in dat een id verzonden kan worden naar de RFID-integration service, en dat er in een andere service gekeken wordt of dit id in de database staat.

6.3.1 Herziening besluit voor communicatie

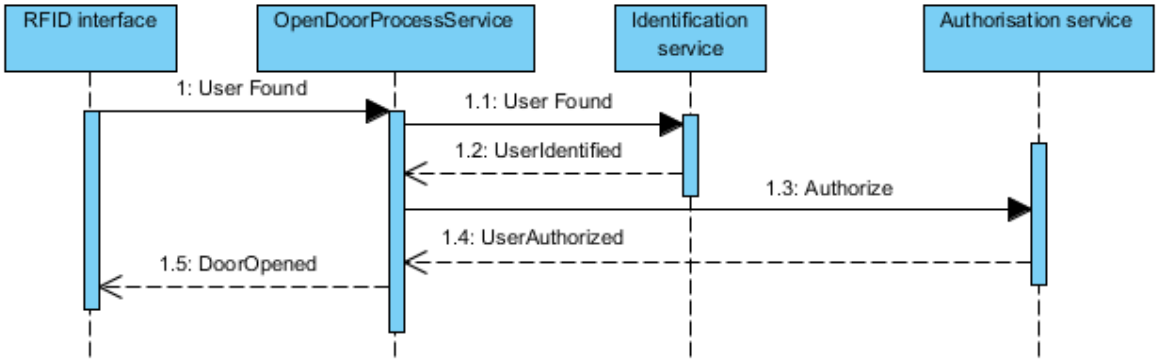
Ik koos ervoor om ook gelijk een gedeelte van de open door process service te maken. Deze process service zou de events op gaan vangen die gestuurd worden door de RFID-service. Vervolgens moet deze de identification service gaan aanroepen.

Na overleg met mijn opdrachtgever had ik besloten om mijn besluit om de commands over de event bus te sturen aan te passen. De voornaamste reden waarom ik de commands over de event bus wilde was, dat ik niet wilde dat de gebruiker een foutmelding te zien kreeg als een service niet beschikbaar is. Het nadeel dat hier wel mee komt, is dat wanneer een service niet draait het command op de event bus gezet wordt en er pas afgehaald wordt wanneer de service weer draait. Dit kan ervoor zorgen dat na bijvoorbeeld 10 minuten de service weer draait en dat daardoor alle deuren, waar personen doorheen probeerden te komen, open gaan.

De opdrachtgever stelde mij de vraag of het wel zo erg is om een foutmelding terug te krijgen en vroeg mij om daarover na te denken. Ik kwam tot de conclusie dat het in deze situatie juist gewenst is om een foutmelding terug te krijgen. Je kan dan aan de gebruiker tonen dat hij niet naar binnen mag. Dat is beter dan wanneer je eerst tien minuten moet wachten voordat de service weer bereikbaar is.

Om iemand te kunnen identificeren moest er eerst nagedacht worden over hoe het proces moest gaan verlopen vanaf het registreren van een pasje tot het open doen

van een deur. Hieruit is het sequence diagram van figuur 13 gekomen.



Figuur 13: verloop van events

Er zal eerst een gebruiker gezien worden door de RFID-reader. De RFID-Reader zal een signaal sturen naar de RFID-integration service. Dit signaal wordt als event op de event bus gezet en opgevangen door de OpenDoorProcessService. Daarna zal deze service een command sturen naar de identification service welke een user terug zal sturen met daarin de usergroup. Deze usergroup en het id van de deur dat vanaf de reader wordt gestuurd zal met een command gestuurd worden naar de authorisation service. Deze kan vervolgens bepalen of de persoon geautoriseerd is om door de deur te mogen en zal een boolean terugsturen. Daarop zal de Open Door process service een signaal sturen dat de deur open gemaakt mag worden.

6.3.2 Swagger

Bij het maken van het gedeelte dat vanuit de open door process service de identification service zal aanroepen (agent) heb ik gebruik gemaakt van swagger. Swagger is een tool waarmee ik heel eenvoudig documentatie en agents kan genereren. Swagger doet dat met behulp van een swaggerfile. Een swaggerfile is een Json file die beschrijft welke data je API verwacht, welke endpoints er zijn en welke data je terug kan verwachten. Deze swaggerfile wordt gegenereerd door te kijken naar attributes die boven je methodes staan en de parameter waar je methode om vraagt.

```
[HttpGet("{userId}")]
[SwaggerOperation("Identify")]
[ProducesResponseType(typeof(UserDTO), (int)HttpStatusCode.OK)]
[ProducesResponseType(typeof(IdentifyException), (int)HttpStatusCode.BadRequest)]
2 references | 2/2 passing | Yael keemink, 5 days ago | 1 author, 4 changes | 0 requests | 0 exceptions
public IActionResult Identify(int userId)
{
```

Figuur 14: swagger in een controller

In het figuur 14 zie je een voorbeeld van een API endpoint. Swagger zal op basis van deze data een gedeelte van een swaggerfile maken. Hij weet nu dat de methode Identify heet, dat deze methode twee return types heeft, namelijk een OkResult met daarin een UserDTO of een BadRequest met daarin een IdentifyException en dat de methode een integer nodig heeft.

6.3.3 Identificeren van gebruikers

Wanneer er iets fout gaat in de Identification service, wilde ik dat de service een exception terug zou sturen. Ik had daarvoor een eigen exception aangemaakt die overerft van Exception. Bij het testen of hij ook daadwerkelijk een exception terug zou sturen kreeg ik de melding dat de service niks terug stuurde. Hierna probeerde ik om de exceptie dan maar als een string terug te sturen. Dat werkte helaas ook niet. Ik kreeg weer geen reactie vanuit de service.

Omdat dit weer niet het gewenste resultaat gaf wilde ik gaan kijken of het aan iets in mijn exceptie lag of dat het aan iets binnen de Exception class lag. Dit wilde ik testen door eerst mijn class niet meer over te laten erven van Exception. Dit ging goed. Hierdoor wist ik dat het aan iets binnen de Exception class lag. De stacktrace kan erg lang worden en kan vreemde tekens kan bevatten, mijn vermoeden was dat deze daardoor niet meer geparsed kon worden naar Json en dat de service daarom niks terug stuurde. Door de stacktrace leeg te laten werd het probleem inderdaad verholpen. De service gaf de foutmelding die ik verwachtte te krijgen.

Het nadeel was wel dat ik daardoor geen stacktrace meer had. Deze stacktrace wil je wel hebben om te bekijken waar het precies fout ging. Daarop had ik een eigen versimpelde versie van een stacktrace geïntroduceerd. Ik voegde bij mijn exceptie een string toe die gevuld zou worden met de plek waar de fout gegooid wordt.

Swagger genereert ook de models die nodig zijn om meet te sturen naar de API. Deze models komen op dezelfde plek te staan als waar de aanroep naar de API plaats vindt staat. Omdat mijn eigen entiteiten in het Domain project zitten hebben die geen weet van het model waarvandaan gemapped zal moeten worden om te kunnen verwerken. Daardoor is het niet mogelijk een impliciete mapper aan te maken.

Omdat ik toch een goed herbruikbare manier wilde om het model naar de entiteit te mappen heb ik gebruik gemaakt van extension methods. Door een extenstion method aan te maken is het mogelijk om de mapping methode achter het object te zetten zodat deze naar het goede type gemapped wordt.

6.4 Sprint 10, gebruikers autoriseren

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	**Punten**
Nakijken van documenten op taalfouten	3
Feedback week 8 verwerken	5
Feedback week 9 verwerken	8
Sprint 9 beschrijven	3
Authorisation service toevoegen aan het architectuur rapport	3
Bad request terugsturen wanneer iemand niet geïdentificeerd is	3
Gebruikers kunnen autoriseren	8
Autorisatie een bad request laten sturen bij niet geautoriseerd persoon	3
Totaal aantal punten	36

6.4.1 Entity Framework Core

Het autoriseren van een gebruiker gebeurt door te kijken of de usergroup waar de gebruiker toe behoort de juiste deur bevat. Wanneer dit zo is, is een gebruiker geautoriseerd om door de deur te gaan. Voor de database connectie had ik gekozen

voor Entity Framework Core. Dit framework zorgt ervoor dat je gemakkelijk CRUD functionaliteiten kan maken, databases kan creëren op basis van je code (code first), of code kan genereren op basis van een database (database first).

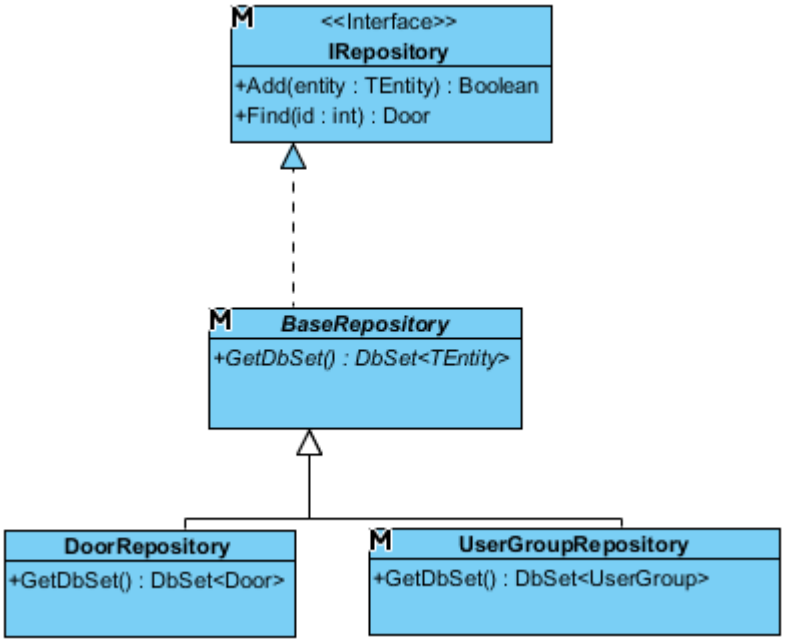
Ik maakte gebruik van code first, hiervoor moest ik eerst de verschillende entiteiten aanmaken en daarna deze entiteiten in een database set zetten. In de authorisation service heb ik gebruik gemaakt van twee entiteiten. Een Door en een UserGroup. Deze hebben een veel op veel relatie met elkaar.

Om duidelijk te maken voor het Entity Framework dat deze entiteiten een veel op veel relatie met elkaar hebben moet je normaal gesproken in beide entiteiten een IEnumerable opnemen van het type van de andere entiteit. Dit werkte echter nog niet in Entity Framework Core. Entity Framework Core had op dat moment nog geen ondersteuning voor een veel op veel relatie. Om toch te krijgen wat ik wilde moest ik een andere entiteit toevoegen aan mijn project die als tussentabel zou dienen. Vervolgens moest ik een één op veel relatie duidelijk maken door in de UserGroup en Door IEnumerable op te nemen van het type van de tussentabel en in de tussentabel een property van het type UserGroup en Door toe te voegen.

6.4.2 Gebruikte design patterns in de repository

In de authorisation service heb ik nog twee design patterns gebruikt. Als eerste heb ik het template pattern gebruikt. Volgens dit pattern moest een abstracte klasse aanmaken welke al de functionaliteit bezit die de subklassen moeten kunnen gebruiken. De subklassen kunnen deze functionaliteiten overschrijven als dat nodig is. Dit heb ik gerealiseerd door een abstracte BaseRepository te maken en deze IRepository te laten implementeren. Deze bevat de methodes die door de subklasse gebruikt moeten worden en een extra abstracte methode waarmee de subklasse kan aangeven welke database set gebruikt moet worden.

Het template pattern gebruikte ik om te voorkomen dat er veel dubbele code gemaakt moest worden. Elke repository zal een add en een find methode krijgen en deze zullen er ongeveer hetzelfde uitzien. Door het gebruik van het template pattern, zorgde ik ervoor dat ik zo veel mogelijk code kon hergebruiken.



Figuur 15: Template en repository pattern

Als tweede pattern is er gebruik gemaakt van het repository pattern. Hiermee heeft de buisiness logica niet direct verbinding met de database, maar praat deze tegen een database acces laag. In deze database acces laag zit een repository. Een repository kan van één entiteit database wijzigingen doorvoeren, zoals insert, find en update. Hierdoor zorg je ervoor dat er niet op verschillende plekken en manieren database wijzigingen doorgevoerd worden. Dit verhoogd de onderhoudbaarheid van de applicatie.

6.4.3 Generics

Een andere techniek die ik gebruikt heb in dit project zijn generics. Bij generics weet je niet van tevoren wat voor type de generic wordt. Dit kan gebruikt worden om bijvoorbeeld een generieke methode te schijven zodat deze met verschillende entiteiten gebruikt kan worden. Zo kan je dus een methode Add schrijven, maar je weet nog niet wat je gaat toevoegen. Dit wordt in een afgeleide klasse of bij het instantiëren van de klasse bepaalt. Ik heb dit gebruikt in mijn abstracte base class baseRepository.

Ik heb voor een baserepository gekozen zodat in een later stadium wanneer de verschillende repository's meer methodes krijgen zoals bijvoorbeeld insert, delete en update er niet veel werk verzet hoeft te worden. Je hoeft dan alleen de methode in de base repository aan te maken en de methode is beschikbaar in beide repository's. Omdat de repository's verschillende entiteiten verwerken kan je niet gewoon een bepaalde entiteit in de base repository zetten, maar zal je de base repository zo moeten maken dat je op een later moment kan zeggen dat die van een bepaald type is. Dat doe je dan in de userGroup en de Door repository.

Om aan te kunnen geven welk type er gezocht moet worden heb ik gekozen voor generics. Bij generics zeg je eigenlijk dat je in een later stadium bepaalt van welk type iets is. Dit kan bijvoorbeeld in de overervende klasse of in de initialisatie van die

klasse. In .NET, en dus in mijn BaseRepository, moet dat op de volgende manier:

```
3 references | Yael keemink, 12 days ago | 1 author, 3 changes
public abstract class BaseRepository<TEntity>
    : IRepository<TEntity>
    where TEntity : class
```

Figuur 16: BaseRepository met generics

Je ziet hier drie dingen. Als eerste zie je naast de klasse naam <TEntity>. Die TEntity is niks anders dan een placeholder en zegt alleen maar dat er op een later moment een type gedefinieerd wordt. Vervolgens zie je dat hij overerft van IRepository en daaraan het type TEntity meegeeft. Ook IRepository wil weten met welk type hij dingen kan doen. Als laatste is er een where statement te zien, hierin kan je aangeven aan welke eisen TEntity moet voldoen. In dit geval moet TEntity een class zijn en mag het dus geen struct zijn.

TEntity kan gebruikt worden in methodes. Dit kan bijvoorbeeld door hem als returntype op te geven of mee te geven aan een klasse die ook een type wil zoals een List of in dit geval een DbSet.

```
protected abstract DbSet<TEntity> GetDbSet();

14 references | Yael keemink, 19 days ago | 1 author, 1 change | 0 exceptions
public virtual TEntity Find(int id)
{
    return GetDbSet().Find(id);
}
```

Figuur 17: Generics in methodes

Wanneer ik een repository laat afleiden van BaseRepository, bijvoorbeeld PersonenRepository, moet ik een type meegeven. Dit type zal vervolgens TEntity vervangen.

```
public class PersoonRepository
    : BaseRepository<persoon>
{
```

Figuur 18: TEntity is vervangen door 'Persoon'

6.5 *Sprint 11, onderzoek naar logging*

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Feedback week 9 verwerken	8
Feedback week 10 verwerken	4
Documentatie schrijven code	5
Context beschrijving afstudeerverslag	3
Onderzoeken op welke manier er gelogd zal worden	2
Logging service in architectuur opnemen	3
Totaal aantal punten	25

6.5.1 Onderzoek logging methode

Tijdens het review gesprek met mijn opdrachtgever in de vorige sprint kwam naar boven dat er ook gelogd kan worden vanuit een andere service. Hij vroeg mij om te onderzoeken of het voor dit project misschien niet beter is om dat te doen in plaats dat elke service zelf aan logging doet.

Ik heb van een aantal cases onderzocht hoe de loggers zouden reageren en of het het systeem gecompliceerder maakt.

Case	**Logger service**	**interne logger**
Service start op	Wordt gelogd en bijgehouden welke services draaien	Wordt gelogd, maar er is nergens bekend welke services draaien
Service sluit af met een melding	Wordt gelogd	Wordt gelogd
Service sluit af zonder melding	Wordt gelogd	Niet gelogd
Er gebeuren veel dingen, zoals veel fouten overal en veel mensen die tegelijk door deuren proberen te gaan	Kan problemen geven met de event bus. De event bus is het snelst wanneer de queues leeg zijn. Hoe meer events in de queue staan hoe minder snel hij wordt	Geeft geen problemen voor de event bus. alles blijft even snel draaien
Systeem wordt opnieuw opgestart na een stroomstoring	Deze service zal als eerste gestart moeten worden. Hier zal rekening mee gehouden moeten worden. Dit moet omdat er anders logging data verloren kan gaan.	Geen bijzonderheden
Event bus sluit af	Wordt gelogd	Wordt gelogd
Loggerservice sluit af	Niet gelogd	-

Uit de cases is duidelijk geworden dat een logger service vooral handig is om te loggen of als een applicatie onverwacht afsluit en om een overzicht te kunnen tonen welke services nog draaien. Wel zorgt hij ervoor dat het opstarten van het systeem complexer wordt, doordat deze service als eerste opgestart moet zijn zodat de andere services zich kunnen aanmelden. Dit is echter alleen van toepassing wanneer de event bus ook opnieuw opgestart moet worden, omdat de queue van de logger service dan niet meer bestaat.

Hiernaast heb ik onderzocht hoe gemakkelijk je data analyses kan verrichten. Het is met verschillende interne loggers een stuk moeilijker om een goede analyse te kunnen maken van wat er allemaal achter elkaar fout gaat als er een bepaalde error opkomt. Dit komt doordat ze allemaal in een eigen log file wegschrijven en geen weet van elkaar hebben.

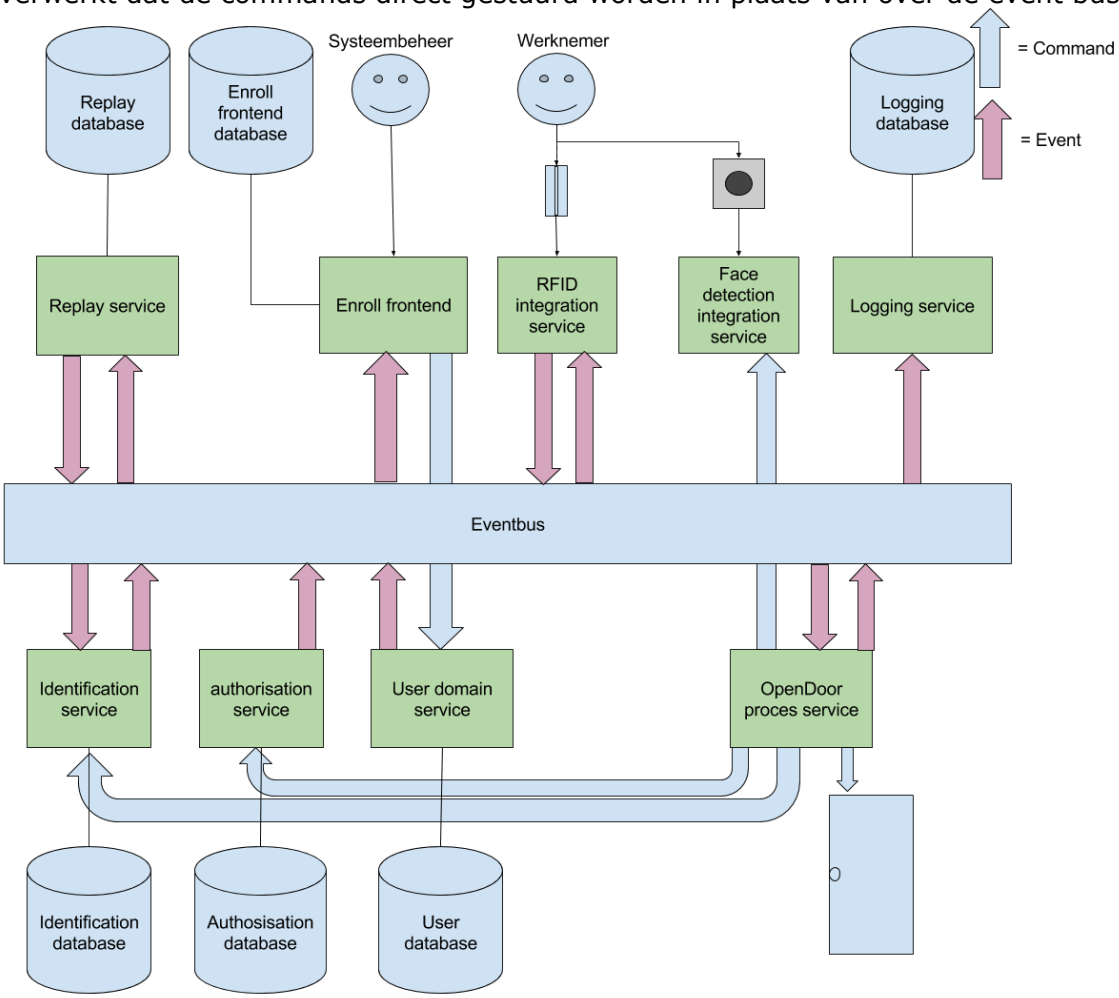
Bij een logger service worden alle logs centraal opgeslagen, waardoor er gemakkelijk gekeken kan worden naar welke fouten vaak achter elkaar optreden. Vervolgens kan er gekeken worden of er een mogelijkheid is om het systeem zichzelf te laten herstellen.

Als laatste voordeel heeft de loggerservice dat wanneer de logger service generiek gemaakt is, hij in principe niet meer aangepast hoeft te worden wanneer er een nieuwe service bijkomt voor bijvoorbeeld het bijhouden van de locatie van de werknemers. Bij de andere oplossing zal er per microservice logging geïmplementeerd moeten worden.

Ik was van mening dat de nadelen van een interne logger per microservice niet opwogen tegen de voordelen van een logger service. Daarom heb ik besloten om een logger service te gaan bouwen.

6.5.2 Logging service in de architectuur opnemen

Omdat er een microservice bij was gekomen moest ik ervoor zorgen dat mijn architectuur weer up-to-date was. Om verwarring te voorkomen tussen de logger service en de auditlog, hernoemde ik de auditlog naar replay service. Ook is erin verwerkt dat de commands direct gestuurd worden in plaats van over de event bus.



Figuur 19: aangepaste functionele architectuur

Voor het bijhouden van welke services nog beschikbaar zijn waren meerdere opties. Ik kon elke service een API geven en de logger service laten pollen of de service nog beschikbaar is. Of ik leg de verantwoordelijkheid bij de services zelf, zodat de services zelf een event sturen dat ze online zijn. Dit zouden ze dan bijvoorbeeld elk kwartier kunnen doen, waardoor ik weet dat als ik na 16 minuten nog steeds niks heb gehoord van een van de online services dat die service niet meer online is en mogelijk foutief is afgesloten. Bij die oplossing zou dan wel ook een event gegooid moeten worden wanneer er een service gepland gestopt wordt.

Omdat de laatste optie minder complex is en hetzelfde effect =heeft besloot ik om voor die optie te gaan.

6.6 *Sprint 12, logging service ontwerpen*

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Feedback docent begeleider tot hoofdstuk 5 verwerken	8
Sprint 11 beschrijven in het afstudeerverslag	4
Logging voor project maken	5
Totaal aantal punten	17

Deze sprint stond volledig in het teken van het ontwerpen en bouwen van de logging service en het verwerken van de feedback die ik had gekregen van mijn docent begeleider op mijn afstudeerverslag.

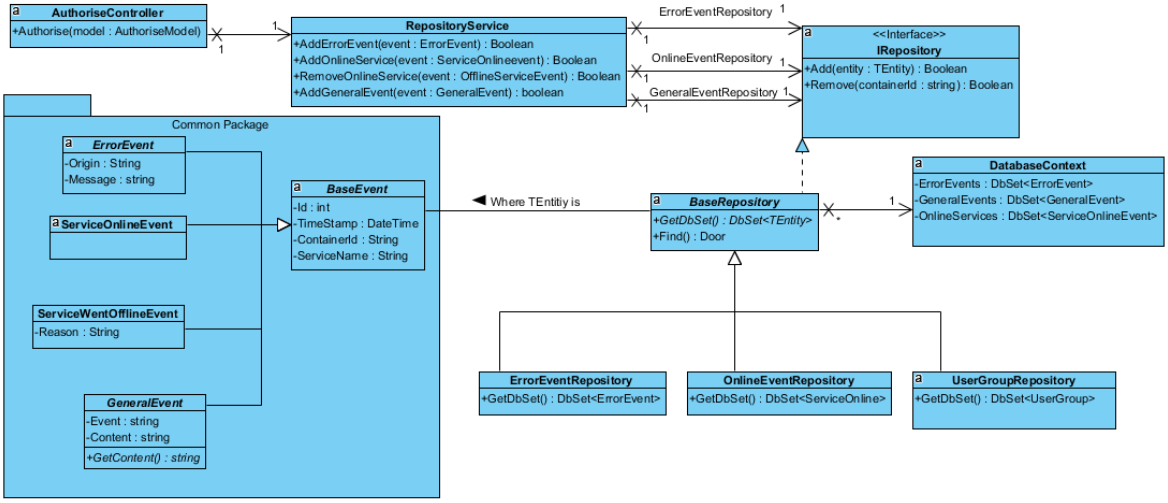
6.6.1 Ontwerpen van de loggerservice

Het ontwerpen van de logger service was erg lastig. Ik wilde ervoor zorgen dat de logger service zo generiek was dat hij niet meer aangepast hoeft te worden zodra er een nieuwe service bijkomt. Om dit voor elkaar te krijgen besloot ik om een BaseEvent aan te maken waar alle andere events van zouden overerven. Ik wilde alleen wel een onderscheid kunnen maken tussen ‘normale’ events en ‘error’ events. Ik wilde namelijk dat de errorevents in een aparte tabel opgeslagen zouden worden. Daarom besloot ik twee verschillende events aan te maken waarin je alle overige events zou kunnen opdelen: GeneralEvent en ErrorEvent.

Omdat je niet wil dat de informatie die een specifiek event als UserFoundEvent bevat verloren gaat, moest ik een property opnemen in het GeneralEvent. Dit werd het property EventContent. Dit is een abstract read-only property. Hiermee zorgde ik ervoor dat deze property niet meer aangepast kan worden na het maken van het event.

Het ErrorEvent had deze constructie niet nodig, omdat het ErrorEvent zelf al alles bevatte wat er in de afgeleide klasse terecht zou komen.

Ook wilde ik dat de logger service zou kunnen bijhouden welke services online zijn. Daarom zorgde ik ervoor dat er een aparte tabel kwam voor alleen de online services. Verder moesten er twee aparte events komen om een service aan te kunnen melden en een service weer af te kunnen melden. Hiervoor had ik het OnlineEvent en het OfflineEvent gemaakt.



Figuur 20: technisch ontwerp logger service

6.7 Sprint 13, logging service maken

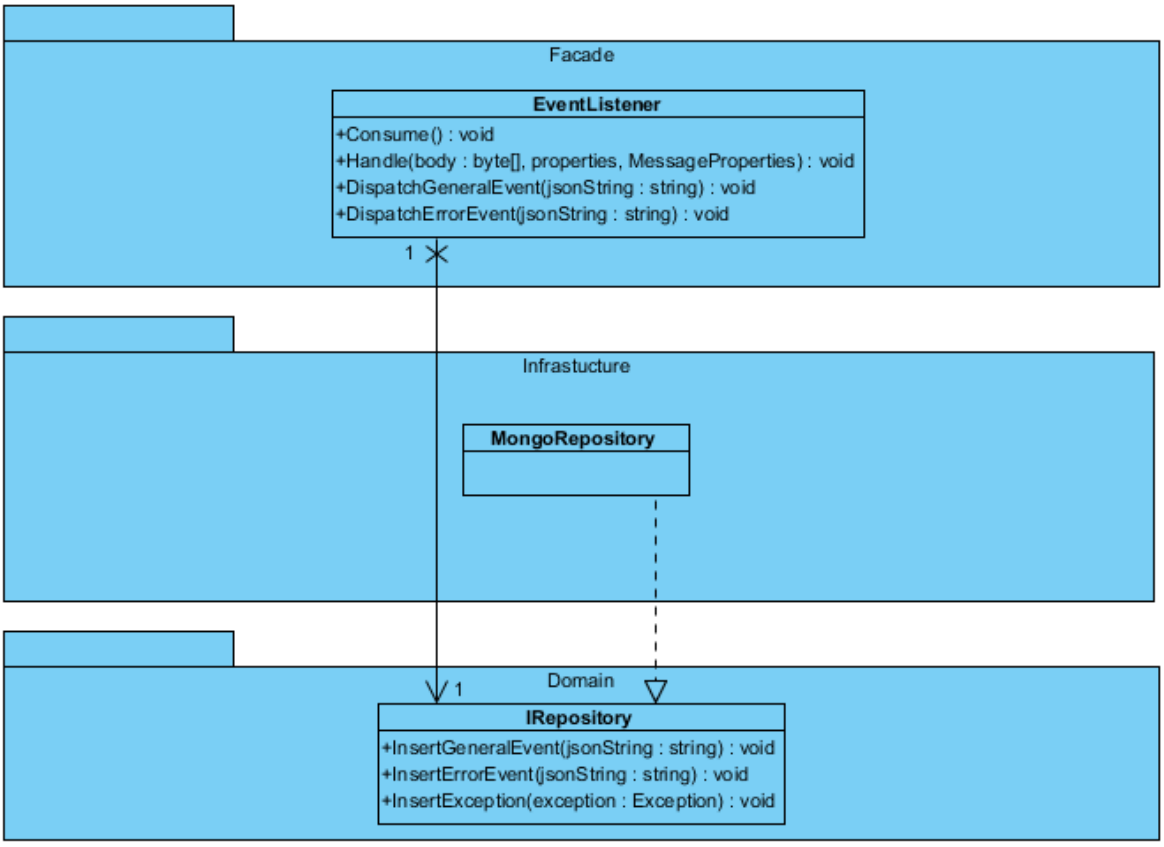
In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Services exceptions op de eventbus laten zetten	2
Sprint 10 beschrijven in het afstudeerverslag	3
Feedback docent begeleider verwerken	5
Sprint 11 beschrijven in het afstudeerverslag	3
Sprint 12 beschrijven in het afstudeerverslag	3
Logging voor project maken	12
Evaluatie proces beschrijven	8
Evaluatie product beschrijven	4
Totaal aantal punten	40

In de vorige sprint had ik een ontwerp gemaakt voor de logging service en ik was dan ook klaar om te beginnen met het bouwen van de logging service. Omdat ik dit ontwerp toch erg complex vond wilde ik toch nog kijken of er niet een andere manier was waarmee ik de complexiteit van de logger service kon verlagen. Mijn begeleider rade me aan om naar MongoDB databases te kijken.

6.7.1 Wijziging van logger service ontwerp

MongoDB is een NoSQL database waar je BSON documenten in kan stoppen. BSON documenten zijn binaire JSON documenten. Door MongoDB te gebruiken zou ik in theorie de JSON die ik binnen krijg van mijn event bus kunnen parsen naar een BSON document en direct doorzetten naar de database. Dit geeft als voordeel dat de logger service geen kennis hoeft te hebben over de objecten en dus ook niet meerdere database repositories hoeft te hebben. Omdat ik met MongoDB de complexiteit van de service kan verlagen, koos ik er ook voor om een nieuw ontwerp te maken waarbij ik MongoDB als database gebruik. Hieruit is Figuur 21 gekomen.



Figuur 21: Technisch ontwerp logger service versie 2

In figuur 21 is te zien dat er drie methodes in de repository zitten, wat erop duidt dat er toch een onderscheid is gemaakt tussen de verschillende soorten events. Toch kent de logger service geen van alle events. Dat er toch een onderscheid gemaakt kan worden komt door de MessageProperties die meegestuurd worden met elk event. Hierin kan je onder andere neerzetten wat het type van het event is. Hiermee maakte ik onderscheid tussen twee verschillende events, namelijk een GeneralEvent en een error event. Onder het ErrorEvent vallen alle events die ontstaan wanneer er een exceptie optreed in één van de verschillende microservices. Alle overige events zijn general events.

Dit onderscheid wilde ik maken omdat de general events tabel vooral als auditlog zou functioneren en de error events tabel vooral als fouten log zou functioneren. Hierdoor zal het doorzoeken van de databases wat makkelijker kunnen gaan. Je kan dan door te kijken naar de tabel snel zien welke fouten er achter elkaar ontstonden en op die manier een goede diagnose stellen.

Als laatste methode in de repository is een methode om excepties te loggen. Dit gaat dan niet om excepties die uit andere microservices komen, maar om excepties te loggen die binnen de logger service zelf voorkomen. Op die manier kan hij van zichzelf bijhouden of er dingen fout gaan bij bijvoorbeeld het parsen van een event naar een BSON document.

Wanneer je figuur 21 en figuur 20 met elkaar vergelijkt is te zien dat er behoorlijk veel is weggevallen en er geen gebruik meer hoeft te worden gemaakt van overerving, wat de complexiteit sterk heeft verminderd.

6.8 Sprint 14, delay inbouwen

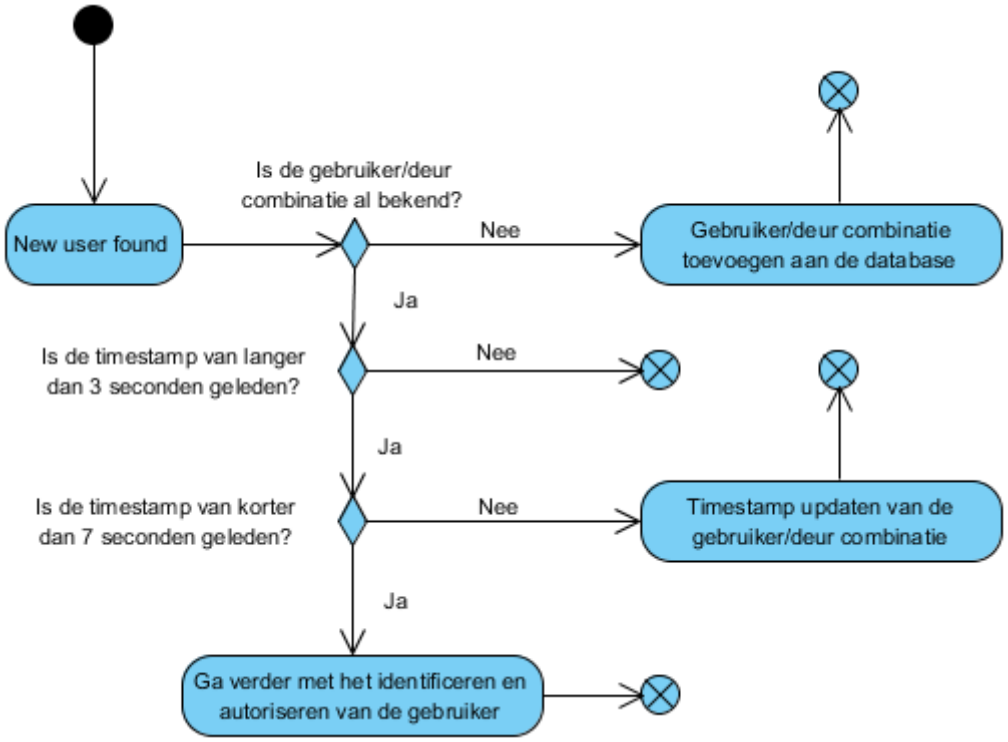
In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Feedback docenten verwerken tot aanpak	16
Sprint 13 beschrijven in het afstudeerverslag	2
Services exceptions op de event bus laten zetten	4
Delay inbouwen	2
Totaal aantal punten	24

In deze sprint wilde ik ervoor zorgen dat deuren niet direct open gaan zodra er een persoon in de buurt is. Ook wilde ik ervoor zorgen dat de logger service zijn werk goed kon doen. Dit deed ik door bij alle andere services de fouten op te vangen en deze op de event bus te zetten.

6.8.1 Inbouwen van de vertraging

Om ervoor te zorgen dat deuren niet direct open gaan wanneer er een persoon langsloopt met het juiste pasje, wilde ik een vertraging inbouwen. Deze vertraging zou drie seconden bedragen, dat is de tijd die er gemiddeld nodig is om langs een deur te lopen. Wanneer er na drie seconden nog een signaal wordt afgegeven dat datzelfde pasje in de buurt is, zal de deur alsnog open gaan.



Figuur 22: activity diagram vertraging

Zoals te zien in figuur 22, worden er een aantal stappen gevolgd vanaf het moment dat er een nieuwe gebruiker gevonden is door een RFID-reader. Als eerste wordt er gecontroleerd of de open door process service de combinatie van gebruikers id met prefix en id van de RFID-reader, kan vinden in zijn database. Wanneer dit niet het geval is, zal hij dit id in zijn database zetten met een timestamp.

Wanneer dit id wel wordt herkend kan hij verder met de volgende stap, controleren of de timestamp langer dan 3 seconden geleden gemaakt was. Hiermee wordt gekeken

of iemand ook daadwerkelijk ergens naar binnen wil. Wanneer dit niet het geval is zal het proces niet verder gaan. Wanneer het wel langer dan drie seconden geleden is, zal hij een volgende controle uitvoeren.

Wanneer de timestamp meer dan 3 seconden geleden gemaakt was, is de kans groot dat iemand ook daadwerkelijk naar binnen wil. Hierop gaat het proces verder met het identificeren en het autoriseren van de persoon.

Er zal ook gecontroleerd worden of de timestamp korter dan 7 seconden gemaakt was. Dit wordt gedaan om te voorkomen dat een deur direct open ging wanneer iemand op twee ver uiteen liggende tijden langs een deur liep. Wanneer de timestamp meer dan 7 seconden geleden gemaakt was, update de service de timestamp van de gebruiker/deur combinatie zodat er opnieuw gekeken kan worden of iemand naar binnen wil.

6.8.2 Test Driven Development

De functie die de vertraging veroorzaakt is natuurlijk een belangrijke functie om goed te testen. Zoals eerder al vermeld, maakte ik gebruik van test driven development. Hierdoor wist ik zeker dat al de code geraakt zou worden door mijn tests.

Als eerste had ik een test gemaakt die test of de methode 'ProcessCanStart' false teruggeeft. Ik had gekozen om false standaard terug te geven, omdat ik wilde dat mijn applicatie standaard niet verder gaat met identificeren tenzij aan alle voorwaarden voldaan werd. Na het runnen van mijn test zag ik dat mijn test faalde. Dat klopte omdat ik de methode nog niet geïmplementeerd had. Nu kon ik de bijbehorende code maken. Het enige dat ik hoefde te doen, was 'return false;' in de methode te zetten. Dit is de minimale hoeveelheid code die nodig was om mijn test te laten slagen.

Omdat dit natuurlijk niet het gewenste resultaat gaf, ben ik mijn testmethode gaan uitbreiden. Ik wilde nu kijken of hij false returned wanneer hij het id dat hij binnen kreeg, niet in zijn database kon vinden. De methode om een id te vinden in de database, zal null teruggeven als hij het id niet kan vinden. Ik wilde dus kijken of ten eerste deze methode aangeroepen werd en ten tweede of hij false returned.

Om dit voor elkaar te krijgen moest ik de klasse en methode die gebruikt werd om het id te zoeken mocken. Ik zorgde ervoor dat de methode om een id te zoeken, Find(), altijd null returned, hierdoor wist ik zeker dat ik het gewenste resultaat uit deze methode kreeg. Vervolgens heb ik de test laten controleren of deze methode aangeroepen werd en of er false teruggegeven werd door de methode die ik aan het testen was.

Deze test faalde, omdat de Find methode niet aangeroepen werd. Ook dit was weer naar verwachting en dus kon ik beginnen met het implementeren van de minimale hoeveelheid code die nodig was om de test te laten slagen. Dit was in dit geval enkel het aanroepen van de Find methode. Hierdoor werd aan alle voorwaarden van de test voldaan.

Vervolgens wilde ik een nieuwe test schrijven om te controleren of de methode ook true teruggeeft wanneer er aan alle voorwaarden voldaan werd. De test die ik hiervoor schreef leek erg op de test om te controleren of er false teruggegeven werd, alleen werd er in dit geval geen null teruggegeven door de Find methode. De Find methode moest nu een object teruggeven. Dit regelde ik weer in het mock object. Nadat ik mijn

test weer draaide zag ik dat mijn eerste test goed ging, maar dat mijn nieuwe test fout ging. Dit kwam doordat er altijd false teruggegeven werd. Om dit op te lossen moest ik in mijn methode een if statement neerzetten welke controleert of het object dat teruggegeven werd null was of niet. Na dit if statement toegevoegd te hebben slaagde mijn beide tests. Op deze manier ben ik door gegaan tot de functionaliteit naar behoren werkte.

6.9 Sprint 15, integratie en acceptatie tests schrijven

In de sprintplanning zijn de volgende backlogitems opgenomen:

Story	Punten
Acceptatie tests schrijven	16
Feedback Docent begeleider verwerken	8
Feedback sprint 14 verwerken	4
Sprint 14 beschrijven	2
Sprint 15 beschrijven	2
Integratie tests schrijven	8
Totaal aantal punten	40

6.9.1 Acceptatie tests schrijven

Om aan te tonen dat het systeem werkt zoals de opdrachtgever het verwacht, wilde ik acceptatie tests schrijven. Toen ik dit voorstelde vond hij dit een goed plan, maar hij wilde graag geautomatiseerde acceptatie tests. Dit wilde hij omdat hij graag elke week, of zelfs elke dag, een nieuwe versie wilde kunnen publiceren. Hij wilde dan niet elke keer weer handmatig door de applicatie heen gaan, maar dit gewoon als een unit test kunnen draaien. In de bijlagen zijn de gemaakte acceptatie tests te vinden.

6.9.1.1 Specification by example

Om acceptatie tests te automatiseren wilde ik gebruik maken van specflow. Specflow is een open source framework waarmee je tests kan schrijven met behulp van 'specification by example'. Specification by example is een manier van tests schrijven, waarbij de test begrepen kan worden door een gebruiker. Er wordt samen met een gebruiker of met de opdrachtgever een test opgesteld. Dit gebeurd door het invullen van een aantal stappen.

Als eerste heb je de stap 'given'. In deze stap worden de voorwaarden waar de test aan moet voldoen voor deze kan draaien gezet. Je kan zoveel 'given' neerzetten als nodig is. Daarna moet je de 'when' invullen. Hierin komt je actie te staan en als laatst heb je de 'then'. Hierin moet je de gewenste uitkomst neerzetten.

Bijvoorbeeld:
Given I have entered 50 into the calculator
Given I have also entered 70 into the calculator
When I press add
Then the result should be 120 on the screen

Wanneer dit is ingevuld heb je een test scenario, vanuit dit testscenario kan je methodes laten genereren die corresponderen met de stappen in je scenario. In deze methodes kan je de test uitvoeren. Elke stap krijgt zijn eigen methode en in elke methode zal je dus ook alleen dat gedeelte wat in de bijbehorende stap beschreven wordt moeten zetten.

6.9.1.2 Probleem met .NET core

Helaas was de library voor specflow nog niet beschikbaar voor .NET core. Hierdoor kon ik geen gebruik maken van deze library. Om toch nog acceptatie tests te kunnen draaien, besloot ik om een stappenplan te maken, waarin beschreven wordt welke stappen je moet ondernemen om de test uit te voeren en welke resultaten je kan verwachten. Op deze manier kon er alsnog een acceptatie test gedaan worden, er werd alleen niet voldaan aan de vraag naar geautomatiseerde acceptatie tests.

6.9.2 Integratie tests

Om de kwaliteit per service aan te tonen wilde ik integratie tests schrijven. Bij het schrijven van de integratie tests wilde ik niet dat er daadwerkelijk events op de event bus gezet werden of dat er een command verstuurd werd naar een andere service. Daarom zorgde ik ervoor dat de service bus en de response op een command gemocked werd. Vervolgens wilde ik ervoor zorgen dat er geen server hoefde te draaien. Hiervoor maakte ik gebruik van een library van Microsoft die een testserver kan aanmaken.

Met deze testserver had ik echter een probleem waar een lange periode overheen ging. De testserver stuurde op al mijn requests een 404 terug, terwijl ik een 200 verwachte. Op internet waren veel antwoorden te vinden op vergelijkbare problemen. Al deze oplossingen hielpen mij helaas niet.

Omdat ik er zelf niet uitkwam, heb ik hulp gevraagd aan een ervaren .NET ontwikkelaar binnen Info Support. Hij maakte een nieuw project om te kijken of hij het probleem kon reproduceren. Bij hem werkte de test server echter wel en dus ging ik zijn code vergelijken met mijn code. Het verschil zat hem in de url die gebruikt werd. Mijn url bevatte aan het einde ‘controller’ en de andere code bevatte dit niet. Hierdoor besefte ik mij dat .NET van de naam van een controller het ‘controller’ gedeelte weg haalde. Dit houdt in dat bij een controller naam ‘ValuesController’ de naam van de controller voor de url verkort wordt naar Values.

Door deze testserver te gebruiken kon ik gemakkelijk de API endpoints aanspreken vanuit mijn tests. Ik heb per service elke keer twee testcases gemaakt. Eén waarbij alles goed moet gaan en één waarbij de flow niet goed gaat. Op deze manier test ik niet alleen of de service zijn werk goed doet wanneer alles goed gaat, maar ook of de service zijn werk doet wanneer er iets niet goed gaat.

7 Product evaluatie

7.1 Plan van aanpak

In het plan van aanpak heb ik beschreven wat er zou gebeuren, welke risico's daaraan zaten en welke producten ik op zou leveren. Hierdoor werd voor mij en mijn opdrachtgever duidelijk wat er gedaan zou worden en hoe dat zou gebeuren.

7.2 Onderzoeksrapport

In het onderzoeksrapport heb ik onderzocht welke techniek het best aansloot op de eisen van Info Support. Hiervoor heb ik de volgende technieken onderzocht:

- Irisscanner
- Gezichtsherkenning
- Stemherkenning
- RFID
- Beacons
- QR-code

Het onderzoek is volgens plan verlopen, er is een goed advies uit voortgekomen en dit advies is ook overgenomen door mijn opdrachtgever.

7.3 Architectuur document

In mijn architectuur document heb ik een aantal views en perspectives opgenomen. Ik heb gekeken naar:

- Welke functionaliteiten waar komen, dit staat in het functional viewpoint.
- Welke tools en talen de developer moet gebruiken, dit staat in het development viewpoint.
- Hoe de data opgeslagen moet worden, information viewpoint, welke wetgeving relevant is voor mijn applicatie, dit staat in het regulations perspective.
- Wat de verwachting is voor eventuele uitbreidingen van deze applicatie, dit staat in het Evolution perspective.

Ik heb hierin onder andere gekozen voor een microservice architectuuren het versturen van commands met behulp van REST en events over een event bus.

De meeste keuzes hebben goed uitgepakt. Zo heeft de keuze voor microservices ervoor gezorgd dat de onderhoudbaarheid van mijn systeem erg hoog was. Je kon hierdoor gemakkelijk aanpassingen maken aan één service zonder dat andere services daar last van hadden. Ook had ik hierdoor de mogelijkheid om verschillende databases te gebruiken voor verschillende services.

De keuze voor een logging service zorgde ervoor dat er bij een eventuele uitbreiding een gedeelte ingebouwd kan worden welke controleert welke services nog draaien en welke onverwacht gestopt zijn.

Mijn opdrachtgever en ik zijn tevreden over de gemaakte keuzes.

7.4 Ontwerpen

Mijn ontwerpen zien er simpel uit. Dit was ook de bedoeling, omdat hoe complexer het systeem is, hoe lastiger het systeem aan te passen en uit te breiden is. Microservices hebben er zeker aan bijgedragen het systeem niet al te complex te maken. Door microservices zijn de grote functionaliteiten gescheiden van elkaar en zijn de ontwerpen dus ook kleiner en makkelijker te begrijpen geworden. Ik heb ontwerpen gemaakt voor al mijn geschreven services. Hierbij heb ik rekening gehouden met het

Onion model en met verschillende design patterns. Ook heb ik geprobeerd om de complexiteit van de services zo laag mogelijk te houden.

Ik en mijn opdrachtgever zijn tevreden over het resultaat.

7.5 Software

Ik heb veel services gemaakt. Ik heb in totaal 5 microservices, en een library gemaakt.

- RFID-integration service
- Open door process service
- Identification service
- Authorisation service
- Logger service
- Common package

Elke microservice is goed getest door unit tests en er zijn acceptatie tests geschreven om aan de opdrachtgever aan te tonen dat mijn applicatie goed werkt.

Ik heb een service gemaakt welke een RFID-Reader kan uitlezen en de data hiervan op een event bus kan zetten, RFID-integration service. Ook heb ik een service die op basis van de gegevens van de RFID-integration service iemand kan identificeren en een service die op basis van de gegevens van de identificatie service en de RFID-Integration service kan kijken of die persoon ook daadwerkelijk een bepaalde deur door mag.

Daarnaast heb ik nog een service gemaakt die het gehele proces, vanaf het moment dat er een event op de bus is gezet met daarin dat er iemand gezien is door de RFID-reader tot aan het open doen van de deur, kan regelen. Hierin zit ook een functionaliteit verwerkt die ervoor zorgt dat de deur niet gelijk open gaat zodra er een persoon met het juiste ID langsloopt.

Als laatste is er nog een logger service gemaakt. Deze loggerservice zal alle fouten en alle events loggen. Deze logger service is zo gemaakt dat hij gemakkelijk uit te breiden is om hem ook te laten controleren op welke services online zijn en welke onverwachts offline zijn gegaan.

De opgeleverde software voldoet aan alle Info Support code conventies en het grootste gedeelte van mijn code is goed getest. Hierdoor kan ik de kwaliteit van mijn code waarborgen. De opgeleverde software voldoet ook aan alle functionaliteitseisen en mijn opdrachtgever en ik zijn dan ook tevreden over het systeem.

8 Proces evaluatie

8.1 Externe afhankelijkheden

Tijdens dit project heb ik te maken gehad met een aantal externe afhankelijkheden. Ik heb bijvoorbeeld moeten omgaan met een RFID-Reader welke te laat werd geleverd en met library's die nog niet beschikbaar waren op .NET core. Hierdoor heb ik de noodzaak van het onderkennen van risico's en daar van tevoren oplossingen voor bedenken ondervonden. Ik heb beide risico's van tevoren zien aankomen en ik had voor deze risico's ook een oplossing klaar liggen. Hierdoor liep mijn planning geen vertraging op. Ik had namelijk al voorzorgsmaatregelen genomen door de RFID-reader vanaf het begin af aan te mocken.

8.2 Onderzoek

Ik vond het lastig om een goede onderzoeksvraag en deelvragen te bedenken. Dit heeft veel tijd gekost welke ik ook had kunnen besteden aan het onderzoek zelf. Ik had hier niet heel veel anders kunnen doen dan ik heb gedaan en vind ook dat ik goed heb gehandeld. Wel kan ik de kennis die ik heb opgedaan door het doen van het onderzoek meenemen in een eventuele volgende keer dat ik een onderzoek zal uitvoeren.

8.3 Ontwikkelen

Het ontwikkelen van de applicaties ging niet altijd helemaal volgens de planning. Ik had sommige delen zoals het bedenken van een goed ontwerp voor een loggerservice beter kunnen plannen. Ik had deze onderschat waardoor ik enkele sprints niet haalde. In plaats van het in één grote story te houden had ik de story op moeten delen in drie kleinere, zoals onderzoeken van de goede logging techniek voor dit product, ontwerpen van de logging service en het bouwen van de logging service. Hierdoor had ik beter inzicht gehad in de werkzaamheden en dus beter kunnen plannen.

Het test driven werken werkte wel erg fijn. Door alles eerst te testen voordat je het bouwt kan je met meer zekerheid zeggen dat de kwaliteit van je software hoog ligt.

8.4 Scrum

Het was een goed idee om scrum te kiezen als methode voor het gehele afstudeer traject. Het zorgde voor veel overzicht van wat er gedaan moest worden en hoeveel werk er in een sprint af kon komen. Ook de keuze om zelf scrum master te zijn beviel goed. Scrum hield ook de druk om iets op te leveren hoog door korte sprints van één week te houden.

8.5 Beroepstaken

Voordat ik begon met afstuderen heb ik een aantal competenties gekozen, waarvan ik zou aantonen dat ik die competenties op een bepaald niveau heb. Dit waren de volgende competenties:

1. 3.1 Ontwerpen software architectuur (niveau 3)
2. 3.2 Ontwerpen systeemdeel (niveau 3)
3. 3.3 Bouwen applicatie (niveau 4)
4. 3.5 Uitvoeren van en rapporteren over het testproces (niveau 4)

8.5.1 Ontwerpen software architectuur

Bij het ontwerpen van de architectuur van dit systeem is rekening gehouden met eventuele uitbreiding van het systeem op een later moment. Dit is gedaan door een

microservice architectuur te kiezen. Op deze manier hoeft niet het gehele systeem geüpdatet te worden wanneer er een uitbreiding klaar is. Je kan gewoon een nieuwe container in Docker opstarten die luistert op de event bus zonder dat de rest van de applicatie daar last van heeft (hoofdstuk 5.1).

Ook is er rekening gehouden met de wetgeving door een advies te schrijven in het architectuur rapport (hoofdstuk 4.4). En heb ik de architectuur vanuit verschillende viewpoints bekeken.

Het ontwerpen van softwarearchitectuur heeft het complexiteitsniveau 'complex', omdat de architectuur begrijpelijk beschreven is voor de stakeholders, er rekening is gehouden met verschillende stakeholders (gebruikers en systeembeheer), en er rekening is gehouden met de wetgeving.

8.5.2 Ontwerpen systeemdeel

Bij het ontwerpen van dit systeem is rekening gehouden met de uitbreidbaarheid door abstracte classes, generics en interfaces te gebruiken. Hierdoor wordt het relatief eenvoudig om nieuwe functionaliteiten toe te voegen (hoofdstuk 6.4.3). Daarnaast is het systeem gemakkelijk te testen doordat er veel gewerkt wordt met interfaces en dependecy injection. Daardoor kunnen delen van het systeem gemakkelijk gemocked worden, waardoor het systeem goed testbaar is. Ook wordt er zoveel mogelijk code hergebruikt. Dat is te zien aan de Common package en de BaseRepository gebruikt in de Authorisation service (figuur 15 en hoofdstuk 6.1.4).

Voor het ontwerpen van de systeem delen is gebruik gemaakt van de tooling 'visual paradigm'.

Het ontwerpen van een systeemdeel heeft minimaal complexiteitsniveau 'lastig', omdat ik rekening heb gehouden met uitbreidbaarheid, testbaarheid en hergebruik. Ook heb ik gebruik gemaakt van enkele design patterns, zoals het façade pattern (de identification en authorisation service hebben een API waartegen gepraat kan worden), singleton pattern(event bus is een singleton) en het repository pattern (figuur 15).

8.5.3 Bouwen applicatie

Bij het bouwen van de applicaties heb ik gebruik gemaakt van verschillende geavanceerde concepten van de programmeertaal gebruikt, zoals generics (hoofdstuk 6.4.3), test driven development (hoofdstuk 6.1.1), impliciete operatoren en het maken van extensie methoden. Ook heb ik vrij veel opgeleverd, namelijk 5 applicaties en één gedeelde library.

Het bouwen van een applicatie heeft minimaal complexiteitsniveau 'lastig', omdat ik rekening gehouden heb met uitbreidbaarheid, testbaarheid en hergebruik. Ook is er gebruik gemaakt van versie beheer (TFS) en is er gebruik gemaakt van het framework entity framework en .NET core.

8.5.4 Uitvoeren van en rapporteren over het testproces

Om mijn applicatie te testen heb ik Test driven geprogrammeerd. Door TDD te werken zorgde ik ervoor dat mijn functies al getest was voor ik de functie had gemaakt. Deze tests heb ik vervolgens aan mijn opdrachtgever laten zien ter verificatie. Naast het maken van unit tests heb ik acceptatie tests en automatische integratie tests geschreven. Deze acceptatie tests zullen de applicatie van voor tot achter testen of

eruit komt wat je kan verwachten. De acceptatie tests zullen met behulp van een testdraaiboek gedaan moeten worden, omdat alle services opgestart zullen moeten worden en Docker op de juiste manier geconfigureerd moet zijn op de acceptatie omgeving.

Ondanks dat dit een andere manier van testen is (zonder testplan, testbasis en conventioneel testrapport) vind ik dat ik deze competentie op complexiteitsniveau 'lastig' heb gehaald. Dit vind ik omdat ik met verschillende testframeworks heb gewerkt en alles test driven heb gemaakt. Bij test driven development hoef ik geen rapport op te stellen over de kwaliteit van de applicatie, omdat alle tests moeten slagen voor ik verder kan met de volgende functie. Er is bij deze manier van ontwikkelen geen scheiding tussen tester en programmeur, omdat de programmeur zelf de tester is en er is dus ook geen feedback vanuit de tester naar de programmeur met wat er beter kan. Het testrapport is dus eigenlijk het feit dat alle tests slagen. Wel is de gehele applicatie per publieke methode en met alle services samen getest.

Literatuur

Biotechno. (z.j.). Wat is een irisscan. Geraadpleegd op 07 februari, 2017, van http://www.educatie-deverbinding.nl/biotechno/docenten/biometrie/biometrie_irisscan.html

Daugman, J. (1993). *High confidence visual recognition of persons by a test of statistical independence*. Geraadpleegd van http://www.cl.cam.ac.uk/~jgd1000/PAMI93.pdf

Fingertec. (2009, augustus). Face recognition technology white paper. Geraadpleegd van http://www.fingertec.com/download/tips/whitepaper-02.pdf

Impinj. (z.j.). The different types of RFID systems. Geraadpleegd op 07 februari, 2017, van http://www.impinj.com/resources/about-rfid/the-different-types-of-rfid-systems.aspx

Meus, H. (2017, 05 januari). De biometrische irisscan, iriscode en irisherkenning. Geraadpleegd op 07 februari, 2017, van http://wetenschap.infonu.nl/recht-en-wet/178040-de-biometrische-irisscan-iriscode-en-irisherkenning.html

Moret, M. (2009, 3 april). Stemherkenning: biometrie op afstand. Geraadpleegd van https://www.security.nl/posting/24772/Stemherkenning:+biometrie+op+afstand

Nijland, N. (2014, 18 augustus). Wat zijn ibeacons en wat kun je ermee. Geraadpleegd op 07 februari, 2017, van https://www.mediamoose.nl/blog/innovatie/wat-zijn-ibeacons-en-wat-kun-je-ermee/

Beritelli, F., & Spadaccini, A. (2012). Performance Evaluation of Automatic Speaker Recognition Techniques for Forensic Applications. Geraadpleegd van https://cdn.intechopen.com/pdfs-wm/40309.pdf

Fowler, M., & Lewis, J. (2014, 25 maart). Microservices. Geraadpleegd van https://martinfowler.com/articles/microservices.html

Palermo, J. (2008, 29 juli). The Onion Architecture : part 1. Geraadpleegd van http://jeffreypalermo.com/blog/the-onion-architecture-part-1/

Schwaber, K., & Sutherland, J. (2016, juli). The Scrum Guide. Geraadpleegd van http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100

Externe bijlages

Bijlage A – Plan van aanpak

Bijlage B – Onderzoeksplan

Bijlage C – Onderzoeksrapport

Bijlage E – Architectuur

Bijlage F – Ontwerpen

Bijlage G – Acceptatie tests