Afstudeerverslag

Info Support Butler

De (mobiele) persoonlijke assistent

Auteur M. Stuivenberg
Studentnummer 12072494

Datum 21 maart 2016

Opleiding De Haagse Hogeschool - Informatica

Afstudeerverslag

Auteur	M. Stuivenberg
Studentnummer	12072494
Onderwijsinstelling	De Haagse Hogeschool Johanna Westerdijkplein 75 2521 EN, Den Haag
Opleiding	Informatica
Begeleidende examinator	Dhr. van Doorn
Tweede examinator	Mw. Maas
Afstudeerbedrijf	Info Support bv Kruisboog 42 3905 TG, Veenendaal
Afstudeerbegeleider	Mw. Hijl
Opdrachtgever	Dhr. Kuiper
Technisch begeleider	Dhr. Gorter

Titel	Afstudeerverslag
Project/Onderwerp	Info Support Butler
Versie	1.0
Status	Definitief
Datum	21-03-2016
Bestand	Afstudeerverslag
Bedrijf	Info Support bv

Historie

Versie	Status	Datum	Auteur	Verandering
0.1	Concept	14-12-2016	Maikel Stuivenberg	Creatie
0.2	Concept	12-02-2016	Maikel Stuivenberg	- Indeling; - Hfst 6.
0.3	Concept	26-02-2016	Maikel Stuivenberg	- InfoSupport stijl; - Typ-, spel- en grammaticafouten; - Feedback verwerkt; - Hfst 4.2 en 7.3; - Deel van evaluatie.
0.4	Concept	03-03-2016	Maikel Stuivenberg	- Feedback verwerkt n.a.v. TTA; - Hfst 5.3, 4.3.2.
0.5	Concept	09-03-2016	Maikel Stuivenberg	- Ontwikkeling & evaluatie afgerond.
1.0	Definitief	16-03-2016	Maikel Stuivenberg	Document definitief

Distributie

Versie	Status	Datum	Aan
0.1	Concept	11-02-2016	De Haagse Hogeschool
0.2	Concept	22-02-2016	Marco Kuiper, John Gorter
0.3	Concept	29-02-2016	De Haagse Hogeschool (TTA)
0.4	Concept	08-03-2016	Marco Kuiper, John Gorter
1.0	Definitief	21-03-2016	De Haagse Hogeschool, Info Support

Referaat

Stuivenberg, Maikel

Het onderzoeken, ontwerpen en bouwen van een mobiele persoonlijke assistent (de Info Support Butler) dat herkent wanneer iemand een gebouw binnenloopt/verlaat en afhankelijk daarvan informatie kan weergeven.
Veenendaal, Info Support BV, 2016

Afstudeerverslag van Maikel Stuivenberg, geschreven in het kader van het afstuderen bij de opleiding Informatica aan de academie voor ICT & Media aan de Haagse Hogeschool.

Het verslag behandelt het (ontwikkel)proces dat is doorlopen tijdens de afstudeerperiode van Maikel Stuivenberg bij Info Support BV te Veenendaal. Tijdens deze opdracht is onderzoek gedaan naar het gebruik van informatiebronnen en een pakketselectie gedaan voor een universele connector. Daarnaast zijn er 2 applicaties (frontend en backend) ontworpen en ontwikkeld om dit project te realiseren. Met dit project is er een proof of concept ontwikkeld dat het gebruik van verschillende technieken kan aantonen.

Descriptoren:

- Afstudeeropdracht
- Scrum
- C#
- ASP.NET Web API
- Xamarin.Forms
- Mobiele applicatie (Android/iOS)
- MSSQL
- JSON
- XML
- OAuth
- Beacon
- Bluetooth

Voorwoord

Voor u ligt het afstudeerverslag 'Info Support Butler'. Dit verslag is geschreven in het kader van mijn afstuderen aan de opleiding Informatica aan de Haagse Hogeschool te Den Haag.

Van november 2015 tot en met maart 2016 heb ik bij Info Support de opdracht: 'Info Support Butler' mogen uitvoeren, waarbij een onderzoek gehouden werd naar het bouwen van een universele connector en heb ik een proof of concept mogen bouwen.

Hierbij wil ik mijn begeleiders (Marco Kuiper, John Gorter en Pascalle Hijl) bedanken voor de goede begeleiding en ondersteuning tijdens mijn afstudeerperiode. Daarnaast wil ik ook de medewerkers van het knowNow team bedanken voor het meedenken en waar nodig hulp bieden bij het maken van de Butler applicaties.

Maikel Stuivenberg
Veenendaal, 21 maart 2016

Inhoudsopgave

1 Inleiding

Dit afstudeerverslag is geschreven in kader van mijn studie Informatie aan de Haagse Hogeschool te Den Haag. Het afstudeertraject heeft plaatsgevonden bij Info Support bv te Veenendaal in de periode van 16 november 2016 tot en met 18 maart 2016.

Bij Info Support heb ik gewerkt aan een persoonlijke assistent (in de vorm van een mobiele applicatie) dat kan herkennen wanneer je binnenkomt of het gebouw verlaat. Afhankelijk van de actie kan er in één overzicht informatie getoond worden uit systemen als Nu.nl, Google Maps, OpenWeatherMap en Slack.

Dit verslag is opgedeeld in zeven hoofdstukken, waarbij ik in hoofdstuk twee meer vertel over Info Support en de begeleiding die zij mij hebben geboden.

Hoofdstuk 3 van dit verslag geeft een introductie op de opdracht. Hierin staan de probleemstelling, de concrete opdracht en de bijbehorende eisen vermeld.

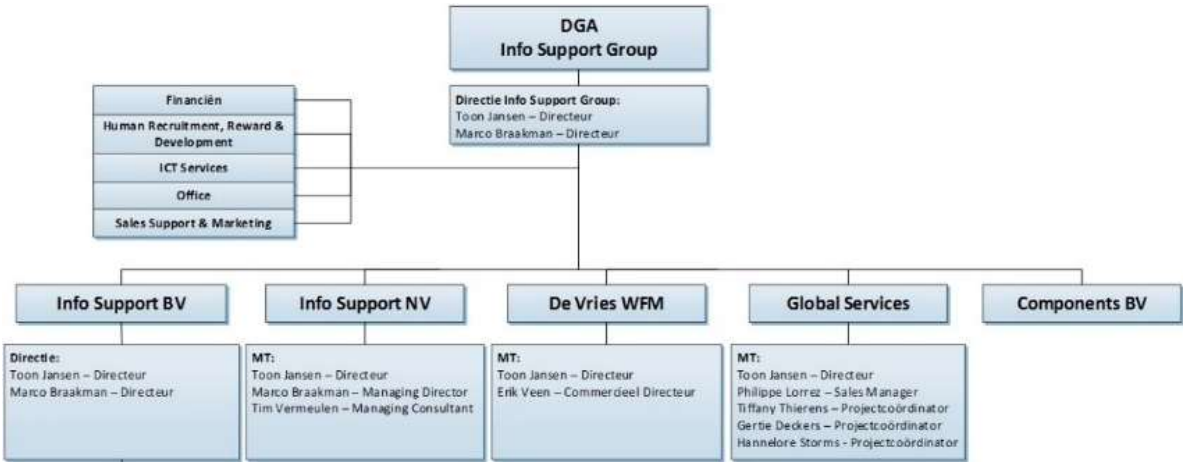
Het 4e hoofdstuk geeft inzicht in de aanpak van dit project. Hierbij heb ik een planning gemaakt voor de gehele afstudeerperiode en de sprints ingedeeld in de laatste 10 weken. Daarnaast staat in dit hoofdstuk ook uitgelegd welke ontwikkelmethode ik ga gebruiken en hoe ik dit toepas binnen dit project.

De aanpak resulteert in twee hoofdstukken, 5 en 6, waarbij in hoofdstuk 5 het onderzoek wordt beschreven en in hoofdstuk 6 het ontwikkelproces van de applicaties wordt beschreven.

Tot slot vind in hoofdstuk 7 een evaluatie plaats, waarin ik beschrijf hoe het gehele proces is verlopen en wat de kwaliteit van de opgeleverde producten is. Daarnaast wordt in dit hoofdstuk ook beschreven hoe ik voldoe aan de opgegeven beroepstaken.

2 Organisatie

De Info Support (International) Group bestaat uit meerdere ondernemingen (zie afbeelding 2-1), waarvan de dienstverlening elkaar aanvult. Hierdoor kan Info Support opdrachtgevers een compleet pakket van oplossingen en diensten aanbieden en waar nodig de juiste partner of het juiste bedrijfsonderdeel aanhaken.



Figuur 2-1 - Info Support Group (zie bijlage A voor de volledige versie)

Info Support BV

Info Support BV is opgericht in 1986 met als visie dat de PC een grote verschuiving zou gaan geven in het IT-landschap (InfoSupport | Geschiedenis Info Support, sd). Nu, 30 jaar later, is Info Support BV uitgegroeid tot een IT-dienstverlener met meer dan 400 medewerkers. Daarnaast is Info Support BV tegenwoordig onderdeel van de Info Support Group, waarbij Info Support BV bedrijven in Nederland voorziet van maatwerk software. Klanten van Info Support BV zijn onder andere ING, NS en de Belastingdienst.

Info Support BV is tevens het gedeelte waar ik, als afstudeerder, mijn werkzaamheden uitvoer. Hierbij ben ik gevestigd in de hoofdvestiging in Veenendaal en heb ik, zoals te lezen in hoofdstuk 2.1, verschillende begeleiders toegewezen gekregen die ook op deze locatie werkzaam zijn.

Info Support NV

Info Support NV is de Belgische tak van Info Support. Zij is opgericht in 1998 en sindsdien uitgegroeid tot een bloeiende onderneming van 25 vaste medewerkers.

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.

Global Services

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

Components BV
Info Support Components bevat softwarecomponenten, ontwikkeld binnen de Info Support group, die voor een bredere doelgroep interessant kunnen zijn. Deze worden klant-en-klaar aangeboden via Info Support Components. Enkele voorbeelden voor deze softwarecomponenten zijn: Kanzi en knowNow.

2.1 Begeleiding

Binnen Info Support ben ik door verschillende medewerkers begeleid in mijn afstudeertraject. In de eerste week zijn verschillende afspraken gemaakt over de rolverdeling tussen deze personen. De rolverdeling van mijn begeleiders is als volgt:

Afstudeerbegeleider
Pascalle Hijl is verantwoordelijk voor de algemene begeleiding binnen het bedrijf. Zij plant onder andere presentaties en trainingen over het plan van aanpak en onderzoek. Daarnaast heb ik ook elke twee weken een gesprek met Pascalle gehad over de voortgang van het Butler project en tips bij het verder ontwikkelen van mijn persoonlijke eigenschappen/vaardigheden.

Technisch begeleider
John Gorter, mijn technisch begeleider, heeft tijdens de afstudeerperiode ondersteuning gegeven bij het maken van de verschillende producten. Hierbij keek hij inhoudelijk naar documenten (klopt de planning, is het een correct onderzoek), of de gemaakte documenten taaltechnisch correct zijn. Daarnaast heeft hij ook de kwaliteit van mijn gemaakte code besproken en advies gegeven voor verbeteringen.

Opdrachtgever
Marco Kuiper is de opdrachtgever van het Butler project. Met deze rol kijkt hij niet technisch naar de opgeleverde producten, maar of alle functionaliteiten kloppen met zijn verwachting. Daarnaast heb ik in samenwerking met de opdrachtgever de vragen opgesteld die in het onderzoek behandeld zijn (zie ook hoofdstuk 5). Om alle voortgang te controleren hebben wij elke twee weken een voortgangsgesprek/review gehouden, waarbij ook de scrum-sprints besproken werd.

3 Opdracht

De Info Support Butler is een opdracht dat als proof of concept moet dienen om het gebruik van verschillende technieken te demonstreren. Hierdoor is er een case opgesteld door Info Support waarin deze technieken toegepast kunnen worden.

In de onderstaande paragrafen vertel ik de probleemstelling, de concrete opdracht en geef ik een voorbeeld van de eisen die ik heb opgesteld in samenwerking met de opdrachtgever.

3.1 Probleemstelling

Elke medewerker van Info Support raadpleegt op een werkdag verschillende informatiebronnen. Dit wordt gedaan om op de hoogte te blijven van gebeurtenissen en om te zien welke werkzaamheden hij heeft. Een aantal voorbeelden van deze bronnen zijn:

- Agenda;
- Build status.

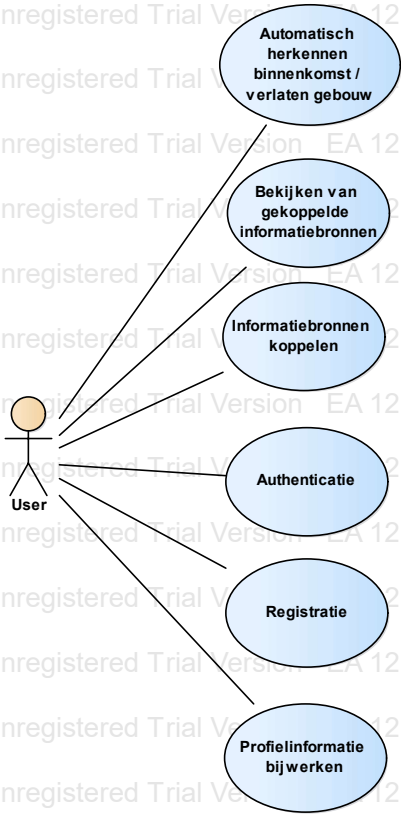
Maar ook andere (niet werkgerelateerde) bronnen worden geraadpleegd op het werk, zoals:

- Nieuws;
- Files;
- Aanwezigheid van een collega.

Al deze verschillende bronnen raadplegen is tijdrovend. Om tijd te besparen zou het handig zijn als alle informatie bij binnenkomst (of verlaten) van het gebouw in één overzicht getoond wordt.

3.2 Concrete opdracht

Aan het eind van dit project wordt er een proof of concept opgeleverd van een persoonlijke assistent (de Info Support Butler) in de vorm van een mobiele applicatie voor Android en iOS. Deze applicatie herkent met behulp van beacons[1] of een medewerker het pand van Info Support binnenkomt of verlaat en geeft, gebaseerd op deze informatie, een scherm met informatie uit verschillende externe bronnen.



Figuur 3-1 Use cases

De gebruiker moet aan de butler kunnen opgeven uit welke bronnen hij informatie wilt zien. Hierdoor zal er naast de mobiele applicatie een universele connector gemaakt worden die de applicatie met de verschillende externe bronnen verbindt.

Doordat op dit moment nog niet bekend is welke systemen gebruikt worden door medewerkers van Info Support, zal dit voorafgaand aan het ontwikkelen onderzocht worden. Daarnaast wordt in dit onderzoek, op basis van de eisen van de verschillende systemen, ook gekeken naar software dat ingezet kan worden als Universele Connector.

[1] Een beacon (ook iBeacon genoemd) is een klein bluetoothapparaat dat een identificatie uitstuurt waarmee een ander apparaat de beacon kan herkennen.

3.3 Producten en eisen

Aan de producten zijn, door de opdrachtgever, verschillende eigenschappen en eisen gesteld. Deze zijn opgeschreven in verschillende documenten zoals het 'Functioneel ontwerp' en het 'Requirements document'.

Zoals op afbeelding 3-1 te zien is, heeft de gebruiker verschillende mogelijkheden in het gebruik van de applicatie. In het functioneel ontwerp (Bijlage G – Functioneel ontwerp) zijn alle use cases uitgewerkt naar use case beschrijvingen. In tabel 3-1 staat de uitwerking van de use case 'Informatiebronnen koppelen' uitgewerkt.

Use Case 'Informatiebronnen koppelen'	
ID	3
Primaire actor	Gebruiker
Secundaire actor	N.v.t.
Requirement	FR04
Precondities	1. De gebruiker moet ingelogd zijn.
Main flow	1. De gebruiker kiest informatiebronnen die hij wilt koppelen. 2. Het systeem slaat deze bronnen op. 3. De gebruiker geeft extra benodigde profielinformatie op. 4. Het systeem slaat de extra informatie op
Resultaat	De gebruiker krijgt de nieuw gekozen systemen te zien bij het binnenkomen of verlaten van Info Support. (Use Case 2)
Alternatieve flow	1. De gebruiker kiest de informatiebronnen die hij wilt koppelen. 2. Het systeem vraagt de gebruiker om zich te authentiseren. 3. Het systeem onthoudt de authenticatie. 4. Het systeem slaat de informatiebronnen op. 5. De gebruiker geeft extra benodigde profielinformatie op. 6. Het systeem slaat de extra informatie op.

Tabel 3-1 Use case beschrijving: UC 3 - Informatiebronnen koppelen

Naast de verschillende use cases zijn in het Requirements document (zie Bijlage F – Requirements), in overleg met de opdrachtgever, verschillende functionele en niet functionele requirements opgesteld. Een aantal voorbeelden van deze requirements zijn:

ID	**Requirement**
FR01	De applicatie moet automatisch herkennen wanneer een gebruiker het pand binnenkomt of verlaat.
FR04	De gebruiker moet systemen kunnen koppelen.
FR06	De gebruiker moet (afhankelijk van de gekozen systemen) informatie kunnen opgeven zoals een woonadres of vertrektijd.
NF04	Na het herkennen van een iBeacon, moet de applicatie een notificatie weergeven.

Tabel 3-2 Voorbeeld van een aantal functionele en niet functionele requirements

4 Aanpak

In de eerste weken van de afstudeerperiode heb ik een plan opgesteld voor het onderzoeken en ontwikkelen van de verschillende applicaties.

In de onderstaande paragrafen staat beschreven hoe de definitieve planning er uit ziet en beschrijf ik hoe ik scrum ga toepassen binnen dit project.

4.1 Planning

In de eerste weken van de afstudeerperiode heb ik een plan van aanpak gemaakt, waarin de planning is gemaakt voor de gehele afstudeerperiode. Doordat het tijdens deze eerste weken steeds duidelijker werd hoe de invulling van de afstudeerperiode zou zijn, wijkt de planning, zoals te zien in tabel 4-1, af van het afstudeerplan.

	November			December				Januari					Februari				Maart	
	16 - 22	23 - 29	30 - 06	07 - 13	14 - 20	21 - 27	28 - 03	04 - 10	11 - 17	18 - 24	25 - 31	01 - 07	08 - 14	15 - 21	22 - 28	29 - 06	07 - 13	14 - 20
Taak	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Sprint								1		2		3		4		5		
Opstart																		
Onderzoek																		
Ontwikkeling																		
Afstuderen																		
Trainingen																		

Tabel 4-1 Planning zoals geschreven in het plan van aanpak

Zoals te zien in tabel 4-1, zijn de eerste drie weken besteed aan het opstarten binnen Info Support. Hierbij is een plan van aanpak gemaakt, is er begonnen met het doen van vooronderzoek en met het maken van een onderzoeksplan.

Week 2 tot en met week 8 is gebruikt voor het uitvoeren en documenteren van het onderzoek. De overige weken (8 t/m 18) zijn gebruikt voor het ontwerpen, documenteren en ontwikkelen van de applicaties.

Tijdens mijn afstuderen heeft Info Support ook trainingen aangeboden om mijn documentatie en code te verbeteren. Een aantal voorbeelden hiervan zijn:

- Schrijven van een plan van aanpak
- Hoe zet ik een onderzoek op?
- Functioneel- en technisch ontwerp
- Xamarin Advanced

4.2 Onderzoek

Zoals in het hoofdstuk 3.2 te lezen is het nog niet bekend welke systemen gebruikt worden door medewerkers van Info Support. Daarnaast is het ook nog niet bekend hoe een universele connector gebouwd kan worden en welke eisen aan de connector gesteld worden.

De eerste 2 weken van de onderzoeksperiode zijn gebruikt om naast het opstartproces, een onderzoeksplan te schrijven (zie ook Bijlage C – Onderzoeksplan). De overige weken (4, 5, 7 & 8) zijn gebruikt om de verschillende vragen te beantwoorden en het onderzoeksrapport op te stellen.

4.3 Ontwikkeling

Voor het ontwikkelen van de applicaties (week 8 t/m 18) heb ik gekozen om de ontwikkelmethode Scrum te gebruiken. Hierbij heb ik in sprints van 2 weken gewerkt aan het Butler project. De laatste week van de afstudeerperiode (week 18) is niet ingepland als sprint, maar kon gebruikt worden bij uitloop van werkzaamheden en het werken aan het afstudeerverslag.

De volgende redenen hebben mij doen besluiten om voor de methode Scrum te kiezen:

- Veelgebruikte methode binnen Info Support;

- Mogelijkheid om iteratief te werken;
  Ik heb elke 2 weken een gesprek met de opdrachtgever gehad waarbij de voortgang besproken werd. Hierbij hebben we ook nieuwe backlog items geïntroduceerd die in de opvolgende sprint gebouwd gaan worden.

- Aan het eind van elke sprint kon een werkend product getoond worden aan de opdrachtgever.

4.3.1 Uitvoering Scrumonderdelen

Binnen Scrum zijn er verschillende taken/onderdelen te onderscheiden. Omdat het project door één persoon wordt uitgevoerd, staat hieronder uitgelegd hoe ik scrum heb toegepast binnen dit project.

Product-owner
De product-owner is de eigenaar van het product. In dit project ging het om de opdrachtgever: Marco Kuiper.

Ontwikkelteam
Het ontwikkelteam bestaat uit één persoon: Maikel Stuivenberg (opdrachtnemer).

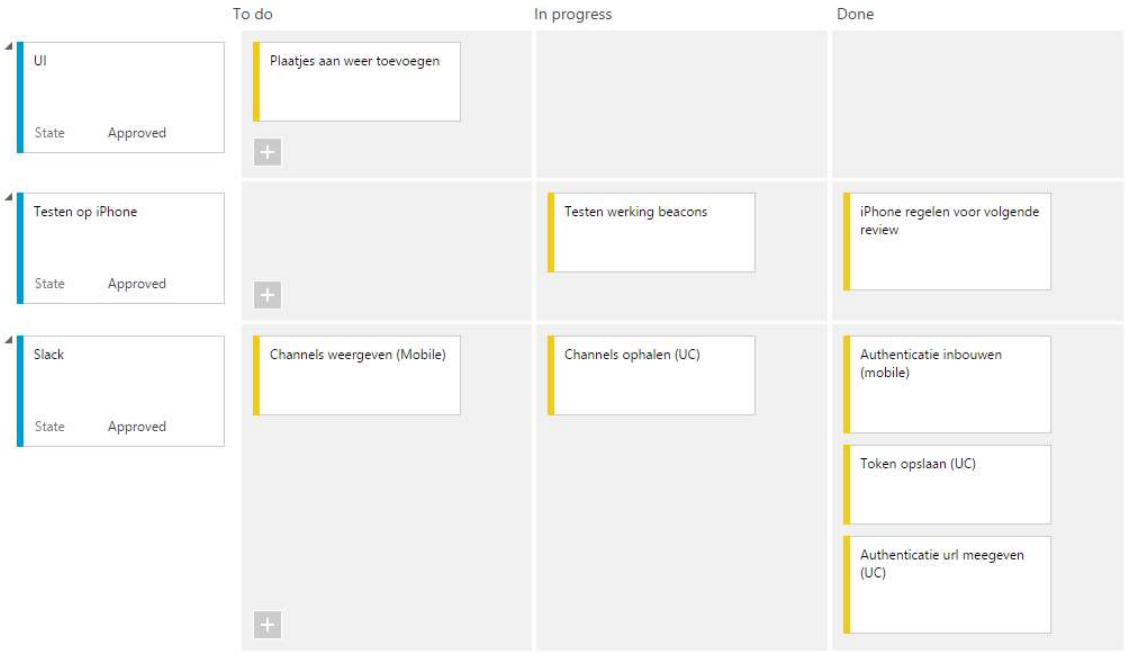
Stand-up
Normaal wordt bij een scrum project dagelijks een stand-up meeting georganiseerd, waarbij medeontwikkelaars bespreken wat er gister is gedaan en wat er vandaag gedaan gaat worden. Hoewel het ontwikkelteam bestaat uit één persoon, heb ik een stand-up meeting gehouden met de ontwikkelaars die in dezelfde kamer zaten als ik (Ontwikkelaars van het knowNow team).

Doordat ik, ondanks het één-persoonsteam, mee deed aan de stand-up meeting ben ik vooraf gaan nadenken over mijn werkzaamheden. Hierdoor werden mijn werkzaamheden beter gepland en kon ik de planning volgen zoals geschreven in het plan van aanpak.

Retrospective
Tijdens een retrospective beoordeelt het team zijn eigen werk en wordt er gekeken naar mogelijke verbeteringen. Doordat een retrospective met een team gehouden wordt, is het bij mij niet mogelijk om dit in de standaard vorm uit voeren.

Als alternatief heb ik met de betrokken personen (zie ook hoofdstuk 2.1) tijdens de gehele afstudeerperiode evaluaties gedaan om te kijken of de ontwikkelingen goed gaan. Hierbij werd naar zowel de applicaties, documentatie als algemene problemen tijdens mijn afstuderen gekeken.

Scrumboard en Backlog
De product-, sprint backlog en het scrumboard zijn bijgehouden via TFS (Visual Studio - Team Foundation Server). Een voorbeeld van de scrumboard is te zien in afbeelding 4-1.



Figuur 4-1 Overzicht van het scrumboard tijdens de 4de sprint.

4.3.2 Versiebeheer

Met versiebeheer is het mogelijk om veranderingen in een bestand (of een groep van bestanden) bij te houden, zodat later specifieke versies opgevraagd kunnen worden. (GIT - Het wat en waarom van versiebeheer, sd)

In het butler project heb ik versiebeheer toegepast met behulp van de TFS omgeving. Hierbij heb ik de code ontwikkeld in een development branch en heb ik na elke sprint een merge gedaan naar de master branch. Hierdoor was de Master branch altijd gevuld met een stabiele versie en heeft de development branch de laatste wijzigingen.

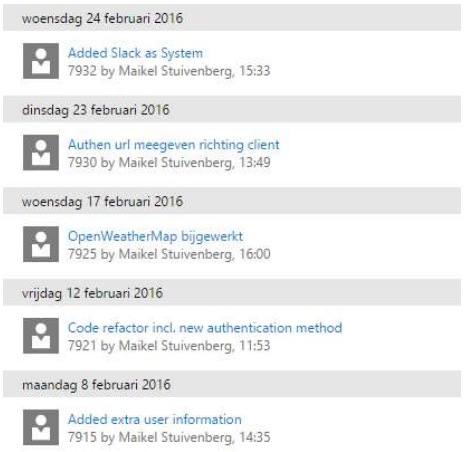
Om nieuwe functionaliteiten uit te proberen, heb ik een enkele keer een extra branch aangemaakt waarin in dit gebouwd werd. Hierdoor kan ik deze tijdelijke branch eenvoudig verwijderen als een oplossing niet blijkt te werken en kan de branch samengevoegd worden met de development branch bij een succesvolle implementatie.

Door versiebeheer te gebruiken binnen mijn project heb ik de volgende voordelen ervaren:

- Mijn code kon op verschillende ontwikkelomgevingen opgehaald en opgeslagen worden.

- Op het moment dat er een aantal aanpassingen gedaan waren die achteraf niet goed werkte, kon eenvoudig de vorige versie van een bestand of het hele project teruggezet worden.

- een commit kan aan een backlogitem gekoppeld worden. Hierdoor was het later mogelijk om te zien welke code aangepast is voor een bepaalde backlogitem.



Figuur 4-2 Overzicht met een aantal commits in de development branch

4.3.3 OTAP omgeving

Tijdens het ontwikkelen van het Butler project heb ik gebruik gemaakt van drie fases uit de OTAP methodiek. Hierbij gaat het om de ontwikkel-, test- en acceptatiefase.

Ontwikkeling

Voor de ontwikkeling van de butler applicaties heb ik gebruik gemaakt van twee verschillende omgevingen:

- Een Windows omgeving waarop de universele connector wordt gebouwd.
- Een OSX omgeving waarin de mobiele applicatie (voor zowel Android als iOS) wordt gebouwd.

Voor de universele connector heb ik gekozen om Windows te gebruiken omdat het gekozen framework (zie hoofdstuk 6.1.2) een product is dat momenteel alleen op Windows werkt.

De mobiele applicatie wordt gebouwd met Xamarin (zie hoofdstuk 6.1.1) wat zowel op Windows als OSX gebruikt kan worden. Doordat mijn applicatie moest werken op iOS en ik op Windows problemen ervaarde met de emulator van Android, heb ik voor OSX gekozen als ontwikkelomgeving voor de mobiele applicatie.

Test

De testomgeving binnen OTAP wordt meestal gebruikt om automatische testen (bijvoorbeeld unit tests) uit te voeren over de gemaakte code. Voor het Butler project heb ik een testomgeving opgezet die ik gebruikte om (handmatig) de universele connector via het internet te testen en te controleren of alle functies in de mobiele applicatie werken.

De testomgeving kan net als de ontwikkelomgeving opgedeeld worden in twee delen:

- Een Azure omgeving voor de universele connector.
- Fysieke mobiele telefoons voor de mobiele applicatie.

Om de mobiele applicaties te testen op fysieke apparaten is het nodig dat de universele connector extern (via een url of IP-adres) benaderbaar is. Naast de testmogelijkheden was het met behulp van de ingebouwde ondersteuning in Visual Studio eenvoudig om de applicatie te publiceren naar Azure.

Doordat er gebruik wordt gemaakt van beacons was het nodig om te testen met behulp van fysieke mobiele telefoons. Zoals in de opdrachtomschrijving (hoofdstuk 3.2) vermeld staat worden twee mobiele besturingssystemen ondersteund, iOS en Android. Hierdoor heb ik gebruik gemaakt van een iPhone (iOS) en een OnePlus (Android) om de gemaakte applicaties te testen.

Acceptatie

Aan het eind van elke sprint is aan de opdrachtgever een werkend product laten zien, zodat de opdrachtgever kan controleren of de gemaakte functionaliteiten overeenkomen met zijn verwachtingen.

Om de mobiele applicatie te laten zien is er, voor het Butler project, gebruik gemaakt van een omgeving die overeenkomt met de bovenstaande testfase. Hierbij zijn de applicaties gedemonstreerd op een Android en iOS toestel.

Productie

De laatste fase van OTAP is de productiefase. Voor het Butler project is er na de acceptatie van de opdrachtgever op dit moment geen extra stap. Hierdoor heb ik geen productieomgeving opgezet.

5 Onderzoek

Voordat de Butler gerealiseerd kon worden, heb ik onderzoek gedaan naar een mogelijke Universele connector. Hiervoor heb ik gekeken naar welke systemen (informatiebronnen) gebruikt worden door medewerkers van Info Support en hoe er informatie uit deze systemen gehaald kan worden.

In de onderstaande paragrafen staat beschreven:

- Hoe ik tot de hoofd- en deelvragen ben gekomen.
- Met welke onderzoekmethodes ik mijn deelvragen heb kunnen beantwoorden.
- Een samenvatting van de resultaten.

5.1 Opstellen hoofd-/deelvragen

Hoewel globaal bekend was wat ik ging onderzoeken, waren nog niet alle details van het onderzoek bekend om tot hoofd- en deelvragen te komen. Hierdoor heb ik vooraf een longlist opgesteld van ongeveer 40 vragen. In deze longlist staan vragen die allemaal gerelateerd zijn aan systemen en de universele connector. Het doel van deze lijst is het vaststellen van de scope van het onderzoek.

Alle vragen uit de longlist zijn voorgelegd aan de opdrachtgever en zijn wel of niet benodigd bevonden voor het onderzoek. Daarnaast was het voor de opdrachtgever ook mogelijk om na elke vraag zelf met nieuwe vragen te komen. Door middel van het samenvoegen van de verschillende vragen ben ik, zoals ook in het onderzoeksplan (Bijlage C – Onderzoeksplan) te zien is, tot drie deelvragen gekomen die ik met dit onderzoek behandel:

1. Welke informatie wordt door een ontwikkelaar van Info Support bekeken?
2. Hoe kan data bij de verschillende systemen opgehaald worden?
3. Met welk framework kan een universele connector gemaakt worden?

Deze deelvragen zijn samen ondergebracht tot de hoofdvraag:

- Hoe kan de universele connector gemaakt worden?

5.2 Gebruikte onderzoekmethoden

De eerste deelvraag “Welke informatie wordt door een ontwikkelaar van Info Support bekeken?” is een vraag die alleen beantwoord kan worden door middel van interviews met medewerkers (ontwikkelaars) van Info Support.

Hiervoor heb ik twee medewerkers uit de kamer waar ik werkzaam ben en de opdrachtgever (ook werkzaam als ontwikkelaar) een aantal vragen gesteld over de informatiebronnen die zij dagelijks raadplegen. Hierbij heb ik nagevraagd hoe ze dit systeem bereiken en op welke momenten dit gedaan wordt. Deze antwoorden zijn gecombineerd en is hieruit een lijst gekomen met 15 systemen die gebruikt worden door medewerkers van Info Support (zie ook Bijlage D – Onderzoeksrapport en hoofdstuk 5.3).

De tweede deelvraag is een vervolg op deelvraag 1, echter moet hierbij meer (technische) informatie opgezocht worden over de verschillende systemen. Door middel van literatuuronderzoek (zoals online documentatie) zijn eigenschappen van de verschillende systemen verzameld.

In de derde deelvraag heb ik alle eisen samengesteld op basis van de gevonden resultaten in deelvraag 2 en eisen voortgekomen uit gesprekken met de opdrachtgever (verwerkt in andere documenten). Op basis van deze eisen heb ik een aantal frameworks opgezocht waarmee een universele connector gemaakt zou kunnen worden en een vergelijking gemaakt tussen deze frameworks.

5.3 Onderzoeksresultaten

Tijdens het onderzoeken van de bovenstaande deelvragen zijn verschillende resultaten naar voren gekomen. In de paragrafen 5.3.1 tot en met 5.3.4 beschrijf ik kort welke resultaten ik heb gevonden en hoe dit toegepast kan worden in het Butler project. Een uitgebreid overzicht van het onderzoek is terug te vinden in Bijlage D – Onderzoeksrapport.

5.3.1 Geraadpleegde informatiebronnen

Uit interviews met een drietal medewerkers van Info Support zijn de onderstaande 15 systemen gekomen die zij dagelijks gebruiken als informatievoorziening. In tabel 5-1 zijn deze informatiebronnen, zoals het nieuws en weer, te vinden. Daarnaast staan in deze tabel ook systemen die voor specifieke projecten gebruikt worden, zoals Kibana of New Relic.

- Buienradar
- Flitsmeister
- Flitsservice
- Google agenda
- Google verkeersinformatie
- Kibana
- knowNow
- New Relic
- NOS
- Nu.nl
- Slack
- TFS Portal
- Tomtom routeinformatie
- Werkagenda
- Werkmail

Voor elke informatiebron is in het interview nagevraagd waar dit voor gebruikt wordt en is gekeken hoe de gebruiker het systeem benaderd. De volgende drie systemen zijn voorbeelden van de uitkomst op deze extra vragen. De complete lijst met uitwerkingen is terug te vinden in Bijlage D – Onderzoeksrapport.

Buienradar

Omschrijving	Met het bekijken van het weer kan er bepaald worden of de paraplu meegenomen moeten worden uit de auto. (Vooral belangrijk voor het einde van de werkdag)
Locatie	http://www.buienradar.nl
Bereikbaarheid	De informatie op deze website is voor iedereen toegankelijk.

Nu.nl

Omschrijving	Nu.nl wordt, net als de NOS website, gebruikt om het laatste nieuws te bekijken.
Locatie	http://nu.nl
Bereikbaarheid	De informatie op deze website is voor iedereen toegankelijk.

Slack

Omschrijving	Slack is een communicatiemiddel die door medewerkers van Info Support gebruikt wordt. Naast de gesprekken van de medewerkers wordt hier ook (automatisch) een statusupdate gepost (in de vorm van een bericht) zodra een build geslaagd/mislukt is.
Locatie	Applicatie op desktop of mobiel, zie ook http://slack.com
Bereikbaarheid	Elke Slack gebruiker heeft een eigen account nodig om gebruik te maken van deze dienst. Daarnaast moet een gebruiker bij 'private channels' geregistreerd zijn om berichten te kunnen bekijken.

5.3.2 Ophalen van gegevens uit systemen

Voor elk systeem uit hoofdstuk 5.3.1 heb ik bekeken of het mogelijk is om door middel van een API[2] informatie op te halen uit een informatiebron. Dit heb ik gedaan met behulp van de productwebsites en online documentatie. Hieruit zijn, voor de uitgewerkte systemen uit hoofdstuk 5.3.1, de volgende resultaten gekomen:

Buienradar

Benodigde informatie	Weer voor de komende dag.
Locatie API	*Niet aanwezig*
Formaat	N.v.t.
Kosten	N.v.t.
Restricties	N.v.t.

Nu.nl

Benodigde informatie	Laatste nieuwsberichten.
Locatie	**Laatste nieuwsberichten** http://www.nu.nl/rss.html
Formaat	XML (Atom)
Kosten	Gratis
Restricties	*Geen*

[2] Een API zorgt voor toegang tot een bepaalde informatiebron. Hoewel het voor een gebruiker mogelijk is om de informatie online te bekijken (bijvoorbeeld via een website of app), kunnen wij in de universele connector weinig met deze data. Met behulp van een API wordt data gestructureerd terug gegeven en kan ik dit verwerken op de back- en/of frontend.

Slack

Benodigde informatie	Laatste berichten in een kanaal (channel).
Locatie	**Laatste berichten in een kanaal (channel)** https://slack.com/api/groups.history?token={xxx}&channel={G000}&count=5
Formaat	JSON
Kosten	Gratis
Restricties	Login door middel van OAuth. Met OAuth kan een Acces Token opgevraagd worden die niet verloopt

Hoewel het voor gebruikers mogelijk is om de verschillende systemen te benaderen is het, zoals hierboven te zien, niet altijd mogelijk om gegevens op te vragen door middel van een API.

Om de gebruiker toch de mogelijkheid te bieden informatie uit bovenstaande systemen te halen, is er een alternatief gezocht voor de systemen die geen API beschikbaar hebben. Dit alternatief is gezocht op basis van de informatie die een gebruiker bekijkt en op de beschikbaarheid van een API. In tabel 5-1 is de volledige lijst met systemen te vinden, na onderzoek of er een API beschikbaar is, en wat een eventuele alternatief is.

Systeem	**API beschikbaar**
Buienradar	Nee, alternatief: OpenWeaterMap
Flitsmeister	Nee, alternatief: Geen
Flitsservice	Nee, alternatief: Geen
Google agenda	Ja
Google verkeersinformatie	Ja
Kibana	Nee, alternatief: Geen
knowNow	Ja
New Relic	Ja
NOS	Nee, alternatief: Nu.nl
Nu.nl	Ja
OpenWeaterMap	Ja
Slack	Ja
TFS Portal	Ja
Tomtom routeinformatie	Ja
Werkagenda	Ja
Werk e-mail	Ja

Tabel 5-1 Overzicht onderzochte systemen

5.3.3 Eisen samenstellen

Op basis van de gegevens uit hoofdstuk 5.3.2. heb ik verschillende eisen opgesteld zoals de benodigde ondersteuning voor JSON en XML. Daarnaast zijn er uit gesprekken met de opdrachtgever en het plan van aanpak (Bijlage B – Plan van aanpak) ook verschillende eisen gekomen.

Eisen voortgekomen uit eerdere deelvragen:

1. Ondersteuning voor JSON en XML.

2. Mogelijkheid tot opslaan van gegevens (zoals API keys voor OAuth authenticatie).

3. Instellen van (custom) headers (Nodig voor bijvoorbeeld Tomtom verkeersinformatie).

Overige eisen door de opdrachtgever:

4. De client (informatiescherm van de gebruiker) moet informatie van verschillende systemen kunnen ontvangen.
   Gegevens van verschillende systemen moeten verzameld worden en terug gestuurd kunnen worden naar de client. De meest voorkomende manier om data terug te sturen naar een client is door middel van een API met JSON of XML als data protocol (IBM | Using Internet data in Android applications, sd)

5. De connector moet data kunnen ontvangen van een client.
   De mobiele applicatie moet deze data kunnen versturen naar de connector. Dit kan net als bij de vierde eis gedaan worden door een webservice op te zetten waar in een JSON of XML formaat data naartoe gestuurd kan worden.

6. De connector moet verbonden kunnen worden met een database (voor het opslaan van gekoppelde systemen, profielen van gebruikers en eventuele andere informatie).

7. De connector moet uitbreidbaar zijn.
   Aan de connector moet door de ontwikkelaar systemen toegevoegd kunnen worden, die vervolgens door de gebruiker gekoppeld/gekozen kunnen worden. (Gamma, Helm, Johnson, & Vlissides, 1994)

8. De universele connector moet om kunnen gaan met verschillende gebruikers.

9. De webservice (zie eis 4 en 5) moet data ontvangen en/of terug geven in verschillende versies.
   Bij wijzigingen aan de webservice moet de oude versie beschikbaar blijven voor de huidige applicaties, terwijl de nieuwe versie voor de nieuwe applicatie moet werken.

5.3.4 Framework selecteren

Met de verschillen eisen als basis (zie hoofdstuk 5.3.3) heb ik voor 6 verschillende frameworks onderzocht aan welk van de gestelde eisen zij voldoen en heb ik in samenspraak met de opdrachtgever gewichten gegeven aan elke eis. In de onderstaande tabel zijn de punten toegekend aan de verschillende frameworks. Hierbij heb ik de helft van de punten toekent in het geval dat een framework aan de eis voldoet waarbij een kleine aanpassing aan de code gedaan moet worden.

Eisen	1. JSON / XML	2. Opslaan van data	3. Instellen van headers	4. Terugsturen van data	5. Data ontvangen	6. Database ondersteuning	7. Uitbreidbaar	8. Meerdere gebruikers	9. Meerdere API versies	Totaal aantal punten
Gewicht	**1**	**3**	**1**	**3**	**3**	**3**	**2**	**2**	**1**	**19**
ASP.NET Web API	1	3	1	3	3	3	2	2	0,5	**18,5**
Slim Framework	1	3	1	3	3	3	2	0	0	**16**
Django REST Framework	1	3	1	3	3	3	2	2	1	**19**
Phalcon	0,5	3	0,5	3	3	3	2	0	0	**15**
Play framework	1	3	1	3	3	3	2	0	1	**17**
Jersey	1	3	1	3	3	3	2	2	0,5	**18,5**

Tabel 5-2 Punten verdeling frameworks

Zoals in tabel 5-2 te zien is, liggen de onderzochte frameworks dicht bij elkaar qua score en ondersteunen alle frameworks de belangrijkste eisen. Als er gekeken wordt naar frameworks die alle gestelde eisen ondersteunen (eventueel met een kleine aanpassing) dan kunnen de volgende framework geselecteerd worden als mogelijke kandidaat:

- ASP.NET Web API;
- Django REST Framework;
- Jersey.

Om tussen één van deze frameworks te kiezen, zou een vervolgonderzoek gehouden kunnen worden, waarbij onderzoek gedaan wordt naar de mogelijke performance verschillen of extra eisen die door o.a. de opdrachtgever gesteld worden.

Voor het Butler project heb ik gekozen om de connector in ASP.NET te bouwen. De redenen voor deze keuze zijn:

- Makkelijker te publiceren naar een testomgeving.
  Om de mobiele applicatie te testen, moet de backend (universele connector) online benaderbaar zijn. De ASP.NET Web API kan met Visual Studio eenvoudig gepubliceerd worden naar een Azure Service, waardoor de testomgeving binnen een paar seconden online is.

- Kortere leercurve.
  De ASP.NET Web API, onderdeel van het .NET framework, kan ontwikkeld worden in C#. Voor mij is deze taal bekender dan Python (voor het Django framework) of Java (voor Jersey), waardoor het leerproces minder lang is. Daarnaast is ook de frontend (zie ook hoofdstuk 6.2.1) in C# waardoor er maar één taal bekend hoeft te zien.

6 Ontwikkeling

Op basis van het onderzoek is de ontwikkeling van het Butler project gestart. Hierbij heb ik twee applicaties gebouwd, de mobiele applicatie en de universele connector.

Voor beide applicaties heb ik een andere ontwikkelomgeving gebruikt. Zoals in hoofdstuk 6.1 te lezen is wordt de mobiele applicatie gebouwd met Xamarin en gaat de universele connector gebouwd worden met Visual Studio.

In hoofdstuk 6.2 wordt weergeven waar de applicaties staan in de architectuur en hoe de applicaties ingedeeld kunnen worden. Hoofdstuk 6.3 zal verder ingaan op de werkzaamheden verdeeld over vijf sprints.

6.1 Ontwikkelomgeving

Voor de ontwikkeling van het Butler project heb ik 2 ontwikkelomgeving gebruikt. Hieronder staat voor zowel de mobiele applicatie als de universele connector de keuzes hierin beschreven.

6.1.1 Mobiele applicatie

Zoals ik in het afstudeerplan vermeld heb, moet de mobiele applicaties gebouwd worden met Xamarin. Hierbij heb ik er voor gekozen om gebruik te maken van Xamarin.Forms.

Met Xamarin is het mogelijk om Native iOS, Android, Windows en Mac applicaties te bouwen in C#. Daarnaast kan er met Xamarin ook een gedeeld project tussen de applicaties opgezet worden, waarmee code gedeeld wordt tussen de verschillende platformen.

In Xamarin zijn er twee mogelijkheden om de applicatie te bouwen. Dit zijn de ‘Classic’ en de ‘Forms’ variant. Beide varianten bieden dezelfde voordelen aan zoals hierboven beschreven is, maar hebben daarnaast ook nog hun eigen eigenschappen:

Xamarin Classic	Xamarin Forms
Apps met platform specifieke taken / API’s	Vereist weinig platform specifieke code
Apps waarbij de UI heel belangrijk is	Geschikt voor apps waar code belangrijker is dan een custom UI
	UI (voor meerdere platformen) kan in één keer gebouwd worden met XAML

Tabel 6-1 Voordelen Xamarin variant (Xamarin Forms, sd)

De mobiele applicatie moet een proof of concept worden voor de opdrachtgever. Hierbij gaat het vooral om het aantonen van de mogelijkheden. Naast het feit de applicatie en proof of concept is, verwachtte ik aan het begin van het ontwikkeltraject weinig code nodig te hebben waar platform specifieke API’s aangeroepen moeten worden.

Met de genoemde voordelen uit tabel 6-1 en de bovenstaande argumenten, heb ik er voor gekozen om tijdens het ontwikkelen van de mobiele applicaties gebruik te maken van Xamarin.Forms. Hierdoor hoeft voor zowel Android als iOS maar één UI geschreven te worden en werken beide applicaties met de zelfde code.

Naast de bovenstaande voordelen, welke gegeven zijn door Xamarin, heb ik tijdens mijn afstudeerperiode ook voor- en nadelen meegemaakt. Op de volgende pagina staan een aantal van deze voor- en nadelen beschreven.

Voordelen gekozen omgeving

Door gebruik te maken van Xamarin.Forms heb ik tijdens mijn afstuderen verschillende voordelen meegemaakt.

Snellere ontwikkeltijd voor de UI

Met Xamarin.Forms hoeft er maar één keer een UI geschreven te worden, dat werkt op alle ondersteunde platformen.

```
<?xml version="1.0" encoding="UTF-8"?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="ISButler.Views.Systems.Nu">
  <ContentPage.Content>
        <ListView x:Name="listView" ItemsSource="{Binding newsItems}" RowHeight="120">
            <ListView.ItemTemplate>
                <DataTemplate>
                    <ViewCell>
                        <StackLayout Padding="10,10,10,10">
                          <Label Text="{Binding title}" FontAttributes="Bold" FontSize="Large"/>
                          <Label Text="{Binding description}"  />
                        </StackLayout>
                    </ViewCell>
                </DataTemplate>
            </ListView.ItemTemplate>
        </ListView>
    </ContentPage.Content>
</ContentPage>
```

Figuur 6-1 XAML Voorbeeld

De UI kan met Xamarin.Forms ontwikkeld wordt in XAML. Dit is een op XML gebaseerde opmaaktaal, waarmee de lay-out van de pagina's bepaald kan worden. In afbeelding 6-1 is een voorbeeld van één van de XAML pagina's te zien.

Meer gedeelde code t.o.v. de classic variant

Xamarin biedt ondersteuning om een gedeeld project aan te maken, waarin code geschreven kan worden dat gedeeld wordt met alle overige projecten (Android, iOS etc.). Naast deze gedeelde code kan ook de UI en de logica achter de UI gedeeld worden tussen alle platformen. Hierdoor wordt het ook mogelijk om (als voorbeeld) alle OnClick methodes en het laden van informatie in de UI in een gedeeld code project te schrijven.

Nadelen gekozen omgeving

Naast de voordelen die ik tijdens mijn afstudeerperiode heb meegemaakt, zijn er tijdens het ontwikkelingsproces ook nadelen gebleken aan het Xamarin Forms platform.

Ondersteuning voor beacons in een achtergrond proces

Doordat elk platform op een andere manier omgaat met beacons en achtergrondprocessen heeft Xamarin.Forms hier (nog) geen ondersteuning voor. Hierdoor is het lastig om een implementatie te maken dat werkt op alle ondersteunde platformen.

In dit project heb ik in overleg met de opdrachtgever ervoor gekozen om de beacon herkenning uit te voeren op de voorgrond. Dit is gedaan zodat het binnenkomst / verlaten gedrag wel uitgevoerd kan worden.

Methodes als OnFocus op een textbox ontbreken
In Xamarin.Forms ontbreekt ondersteuning voor verschillende methodes zoals een OnFocus of OnFocusLost bij gebruik van bijvoorbeeld een textbox.

Het is met Xamarin.Forms wel mogelijk om deze methodes te implementeren met eigen geschreven objecten, echter was het niet nodig om dit verder uit te zoeken.

6.1.2 Universele Connector

Voor de ontwikkeling van de universele connector heb ik ervoor gekozen om dit met het Web API framework van ASP.NET te maken.

Uit het onderzoek, zie ook hoofdstuk 5 en Bijlage D – Onderzoeksrapport, blijkt dat er geen standaardproduct is dat kan dienen als een universele connector voor het Butler project. Echter was het met de gestelde eisen wel mogelijk om, met behulp van een framework, zelf de universele connector te bouwen.

In het onderzoek kwamen drie frameworks naar voren die het meest geschikt zijn voor het maken van universele connector:

- ASP.NET Web API;
- Django REST Framework;
- Jersey.

Zoals in hoofdstuk 5.3.4 te lezen is, heb ik gekozen om het ASP.NET Web API framework te gaan gebruiken voor de universele connector. Door deze keuze heb ik mijn front- en backend beide in C# geschreven en had ik minder tijd nodig om de taal aan te leren. Ook was het eenvoudig om mijn omgeving te publiceren in Azure.

Voordelen gekozen omgeving

Net als bij de ontwikkelomgeving voor de mobiele applicatie, heb ik ook tijdens het bouwen van de universele connector verschillende voordelen ervaren:

Eenvoudige ondersteuning voor verschillende API adressen
In een ApiController biedt Microsoft (fabrikant van ASP.NET) eenvoudige ondersteuning voor verschillende API's aan. Bij het aanmaken van de methode in een controller kan aangegeven worden op welke API-route er geluisterd moet worden. Daarnaast is het ook mogelijk om de type request (zoals een GET, POST of DELETE) mee te geven.

```
[Route("Api/User/System")]
[HttpPost]
[Authorize]
0 references | Maikel Stuivenberg, 25 days ago | 1 author, 4 changes | 1 work item
public HttpResponseMessage AddSystem([FromBody]string system)
{
    if (!Models.User.GetUser(User.Identity.Name).AddSystem(system))
        return Request.CreateResponse(HttpStatusCode.BadRequest);

    return Request.CreateResponse(HttpStatusCode.OK);
}
```

Figuur 6-2 API Voorbeeld

Ingebouwde ondersteuning voor authenticatie
Zoals in afbeelding 6-2 te zien is, is er ook een attribuut 'Authorize' aanwezig. Hiermee kan aangegeven worden dat de desbetreffende API alleen aangeroepen worden als een user ingelogd is. Deze herkenning van een (niet) ingelogde user wordt onderwater door het Web API framework uitgevoerd.

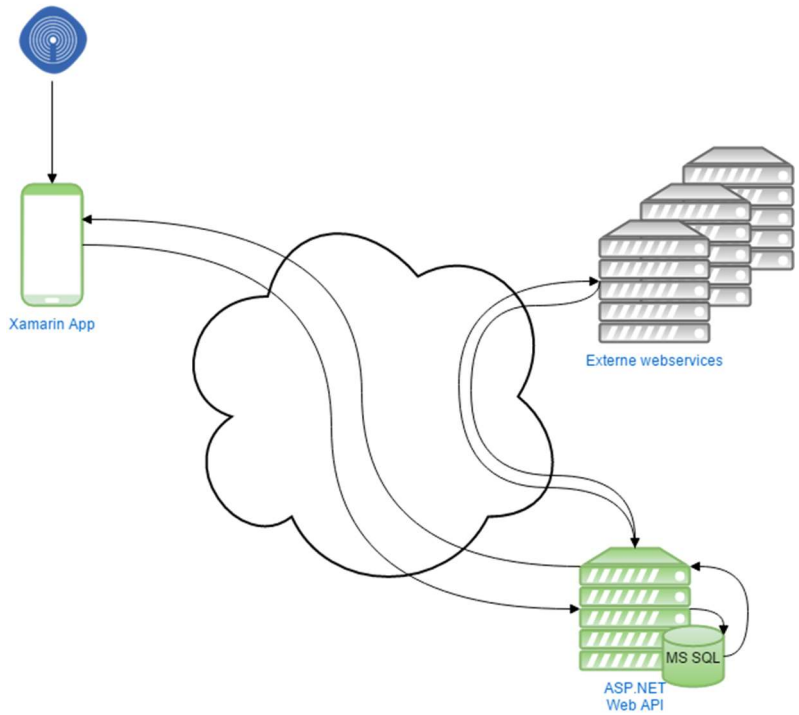
6.2 Globaal overzicht

Zoals in eerdere hoofdstukken gelezen kon worden zijn er verschillende systemen betrokken bij het Butler project:

- Beacons;
- Mobiele applicatie;
- Universele connector;
- Externe informatiebronnen (webservices).

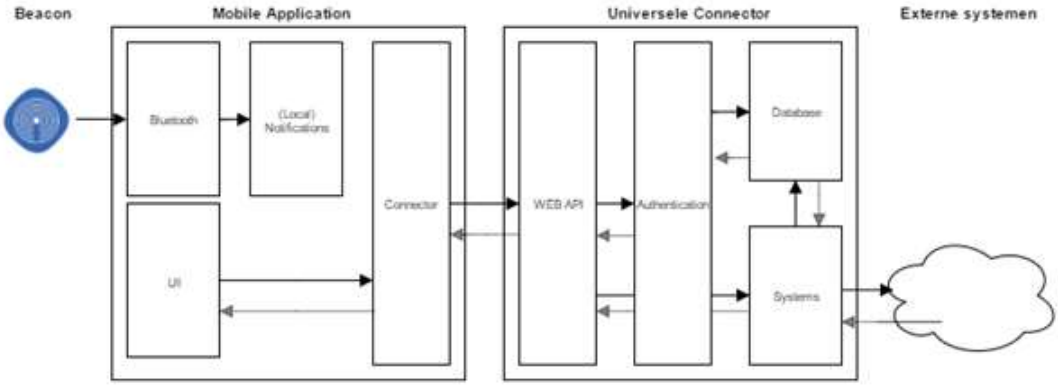
Binnen dit project heb ik twee van deze producten ontworpen en gebouwd: een frontend applicatie (voor Android en iOS) en een backend applicatie (de universele connector). Beide producten staan met elkaar in verbinding door middel van het internet. (zie ook afbeelding 6-3)

De overige onderdelen (beacons en externe informatiebronnen) zijn bestaande systemen die buiten de scope van dit project vallen (zie ook Bijlage B – Plan van aanpak)



Figuur 6-3 Architectuur overzicht

In het overzicht van afbeelding 6-4 heb ik de 2 onderdelen, die door mij gemaakt zijn uitgewerkt in verschillende modules.



Figuur 6-4 Architectuur overzicht

In de volgende paragrafen wordt dieper ingegaan op de verschillende modules die in afbeelding 6-4 te zien zijn.

Allereerst wordt de universele connector beschreven (hoofdstuk 6.3) met als onderdelen:

- Web API;
- Authenticatie;
- Database;
- Systemen.

Vervolgens heb ik de mobiele applicatie beschreven met verschillende UI voorbeelden, de werking van het herkennen van beacons en de verbinding met de universele connector.

6.3 Werkzaamheden

In de volgende paragrafen wordt per sprint uitgelegd welke functionaliteiten ik heb gebouwd, hoe ik dit heb aangepakt en tegen welke problemen ik ben aangelopen.

Voor elke sprint geldt dat ik eerst een ontwerp heb gemaakt (technisch ontwerp, functioneel ontwerp) en op basis van dit ontwerp ben gaan ontwikkelen.

6.3.1 Sprint 1

In de eerste sprint heb ik een basis neergezet van de mobiele applicaties en de universele connector.

Met de opdrachtgever heb ik afgesproken om in deze sprint een applicatie te bouwen waarmee een beacon herkent kan worden. Vervolgens moet, na het herkennen van een beacon, informatie opgehaald worden uit het systeem 'Nu.nl'.

Herkennen van één beacon

Selecteren van externe package

Xamarin.Forms heeft geen ingebouwde ondersteuning voor bluetooth apparaten. Met behulp van NuGet packages[3] is het wel mogelijk om deze ondersteuning toe te voegen aan het project.

Doordat er geen ingebouwde ondersteuning is en gebruik gemaakt moet worden van externe packages, was het nodig om uit te zoeken welk pakket hiervoor gebruikt kon worden. Sommige pakketten werken namelijk niet goed op zowel Android en iOS of worden niet meer ondersteund / geüpdatet.

Om de juiste package te selecteren heb ik meerdere kleine proof of concept projecten gemaakt, waarmee ik keek of er beacons gevonden werden. Uit deze proof of concepten kwam naar voren dat de Monkey.Robotics plug-in het beste kon omgaan met Xamarin.Forms en de gebruikte beacons. Problemen die voortkwamen uit andere gevonden plug-ins waren onder anderen:

- Niet beschikbaar voor zowel Android als iOS
- Missende functionaliteiten voor bepaalde platformen
    - Bijvoorbeeld het ontbreken van RSSI (signaalsterke) waarde

Platform specifieke code

Zoals geschreven in hoofdstuk 6.1.1 heeft Xamarin.Forms weinig platform specifieke code nodig. Om te zorgen dat de Robotics plug-in werkt in Xamarin.Forms is er één regel platform specifieke code nodig:

Android – MainActivity.cs

```
ISButler.App.adapter = new Robotics.Mobile.Core.Bluetooth.LE.Adapter ();
```

iOS – AppDelegate.cs

```
ISButler.App.adapter = new Robotics.Mobile.Core.Bluetooth.LE.Adapter.Current;
```

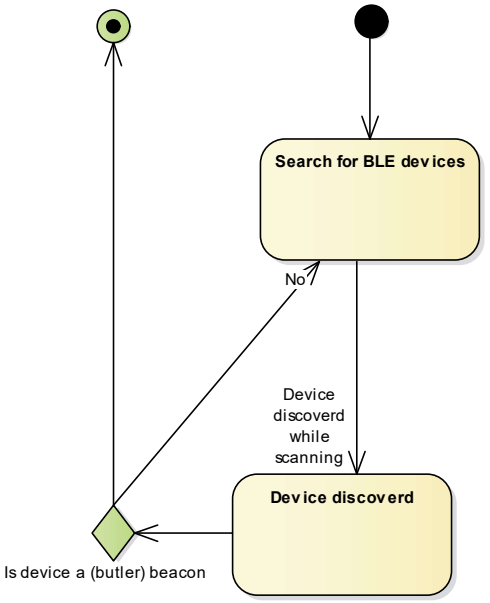
Hiermee wordt aan het Xamarin.Forms project (`ISButler.App`) de Android of iOS variant van de Robotics plug-in meegegeven.

[3] Een NuGet package is een soort plug-in, meestal gemaakt door een 3e partij, waarmee extra functionaliteit toegevoegd kan worden aan de applicatie.

Controleren van een beacon

Op het moment dat de gebruiker in de buurt komt van een beacon moet de applicatie dit herkennen. Om dit te herkennen wordt, zoals in afbeelding 6-5 te zien is, bij het opstarten van de applicatie het zoekproces gestart.

Het zoeken naar beacons heb ik zo ontworpen, dat het een continue zoekproces blijft totdat er een beacon gevonden is. Op het moment dat er een beacon gevonden is wordt er, voor deze eerste sprint, een tekst op het scherm van de gebruiker getoond waarin de gevonden beacon staat.



Figuur 6-5 Activity diagram - Herkennen van één beacon

Bekijken van informatie uit systeem 'Nu.nl'

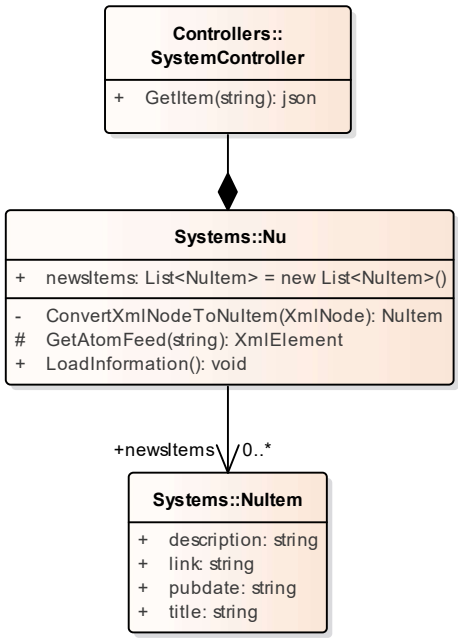
Naast het herkennen van beacons heb ik in de eerste sprint ook het ophalen en weergeven van informatie afkomstig van Nu.nl gerealiseerd. Hierbij heb ik geprobeerd de applicaties zo op te zetten dat het eenvoudig is om in de opvolgende sprints uit te gaan breiden.

Ophalen van data

De mobiele applicatie zal via de universele connector informatie krijgen uit Nu.nl. Hiervoor is de connector zo opgezet dat door middel van een aanroep naar

http://universalconnector.azurewebsites.net/Api/System/Nu

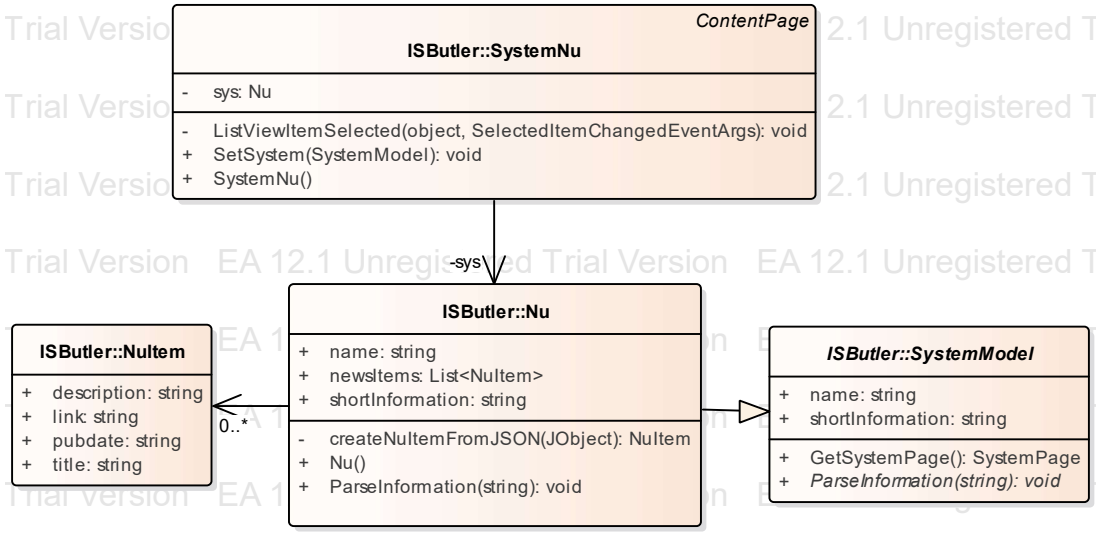
data wordt gegeven uit deze informatiebron. Het ontwerp in afbeelding 6-6 geeft de eerste variant van het ophalen van deze informatie weer.



Figuur 6-6 Class diagram Universele Connector - Ophalen van Nu.nl

Weergeven van data
De mobiele applicatie kan, via de universele connector, de data van Nu.nl ophalen en tonen aan de gebruiker. Het voordeel van deze oplossing is dat de mobiele applicatie alleen een bepaald formaat (JSON) data af hoeft te handelen. Daarnaast hoeft de mobiele applicatie ook geen kennis te hebben over de protocollen van de verschillende systemen.

In afbeelding 6-7 is te zien hoe de pagina (SystemNu) gegevens krijgt van de Nu class. Deze class kan informatie ophalen van de universele connector en de gebruiker een lijst met nieuwsberichten (NuItem) laten zien.



Figuur 6-7 Class diagram mobiele applicatie - Weergeven van data

6.3.2 Sprint 2

In de tweede sprint heb ik een extra informatiebron toegevoegd (OpenWeatherMap). Daarnaast heb ik de gebruiker de mogelijkheid gegeven om zich te kunnen registreren en aan zijn profiel systemen te koppelen.

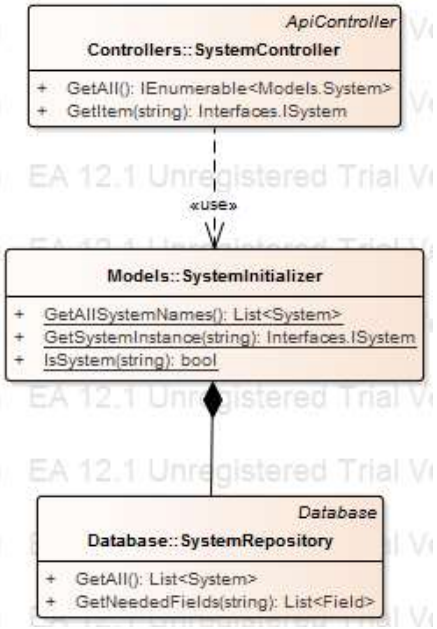
Informatie weergeven uit 'OpenWeatherMap'

Het ophalen van informatie van OpenWeatherMap is op de zelfde manier opgebouwd als Nu.nl (zie sprint 1). In de universele connector zijn er 2 classes bijgekomen waarin de data opgehaald kan worden.

Hoewel dit extra systeem geen consequenties heeft voor de onderhoudbaarheid van de applicatie, kan dit in een later stadium (bij veel systemen) wel problemen op gaan leveren:

- Code om informatie op te halen, van de universele connector, wordt mogelijk dubbel geschreven.
- In de controller moeten bij elk nieuw systeem de class opgegeven worden.

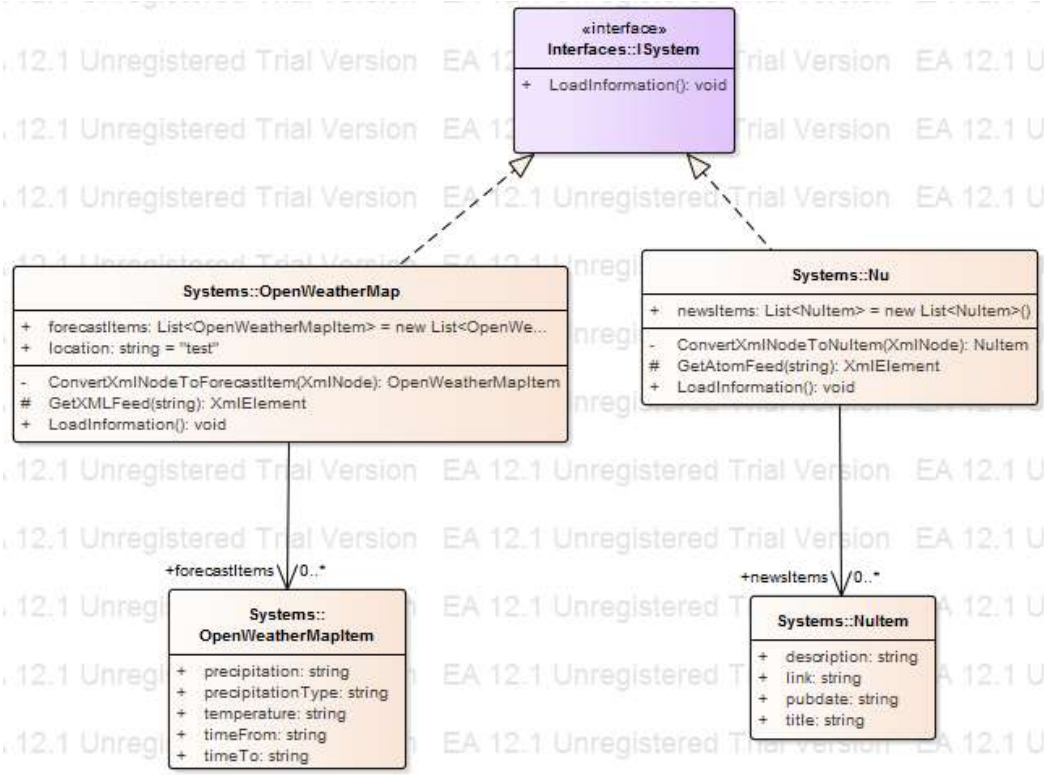
Om deze problemen tegen te gaan heb ik een class genaamd 'SystemInitializer' aangemaakt welke, op basis van een naam, de geïnitieerde class kan teruggeven.



Figuur 6-8 Class diagram universele connector – SystemInitializer

De 'SystemInitializer' class in afbeelding 6-8 vraagt via de 'SystemRepository' class op of het opgegeven systeem bestaat. Vervolgens wordt er een class geïnstantieerd dat het ISystem interface implementeert.

Diagram 6-9 laat een voorbeeld zien hoe de systemen in de nieuwe situatie zijn geïmplementeerd.



Figuur 6-9 Class diagram universele connector - systeem implementatie

Inloggen en gebruikersregistratie

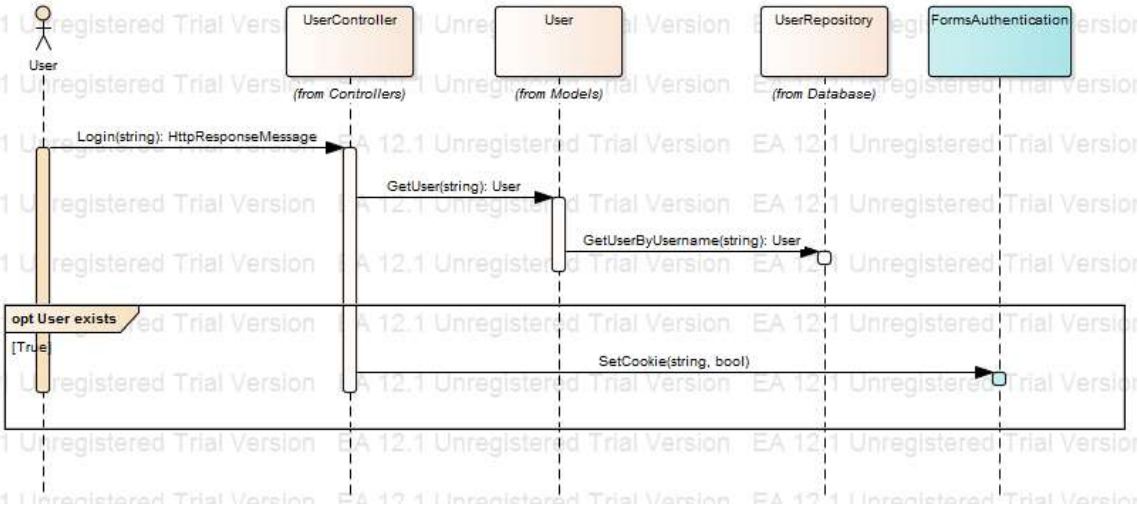
In deze sprint krijgt de gebruiker ook de mogelijkheid om systemen te koppelen. De opdrachtgever heeft hierbij gekozen om dit te koppelen aan een account. Hierdoor wordt een koppeling ook overdraagbaar naar andere apparaten (waar met het zelfde account is ingelogd).

Om te zorgen dat inloggen mogelijk werd, moest aan de universele connector een kleine aanpassing gedaan worden. In de settings file kan 'forms' authenticatie aangezet worden.
Met deze ASP.Net Forms authenticatie kan er een cookie gezet worden als de gebruiker correct ingelogd is. Deze cookie kan vervolgens meegegeven worden bij elke nieuwe request om de gebruiker te herkennen in de universele connector.

Voor de eerste request naar de universele connector moet de mobiele applicatie inloggen bij de connector. Om dit te doen kan er een request gestuurd worden naar:

http://universalconnector.azurewebsites.net/Api/User/Login?username=Maikel

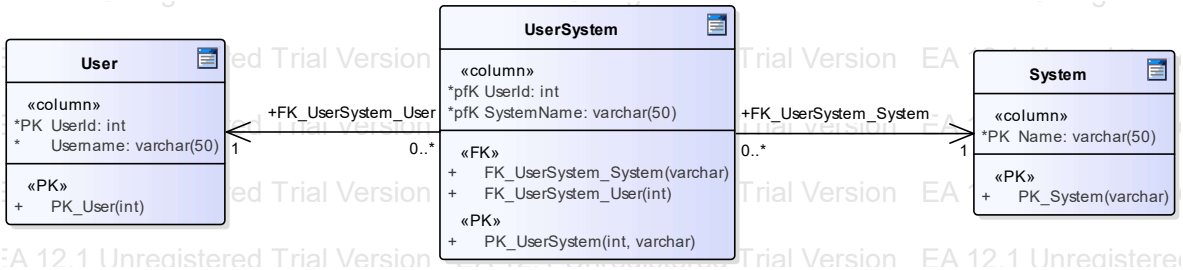
Bij het aanroepen van deze API wordt het proces, zoals afgebeeld in afbeelding 6-10, uitgevoerd in de connector.



Figuur 6-10 sequence diagram universele connector – Login

Koppelen van informatiebronnen

In de voorgaande paragrafen is al een UserRepository en SystemRepository geïntroduceerd welke informatie op kunnen halen uit een database. Deze informatie wordt opgehaald uit de tabellen User en System, zoals te zien in afbeelding 6-11.



Figuur 6-11 Database ontwerp - Koppelen van systemen aan gebruikers

Zoals in de opdrachtomschrijving (hoofdstuk 3.2) te lezen is, moet de gebruiker informatiebronnen kunnen koppelen. In sprint 2 hebben wij ervoor gekozen om deze functionaliteit toe te voegen, waarbij de koppeling niet actie afhankelijk is. De koppeling zit hierbij niet vast aan binnenkomst of verlaten, maar geldt voor beide acties.

Om dit te realiseren is er een meer op meer relatie aangelegd tussen User en System (een gebruiker heeft meerdere systemen gekoppeld en een systeem kan bij meerdere gebruikers horen). Hierdoor is in de database een situatie zoals te zien in afbeelding 6-11 ontstaan.

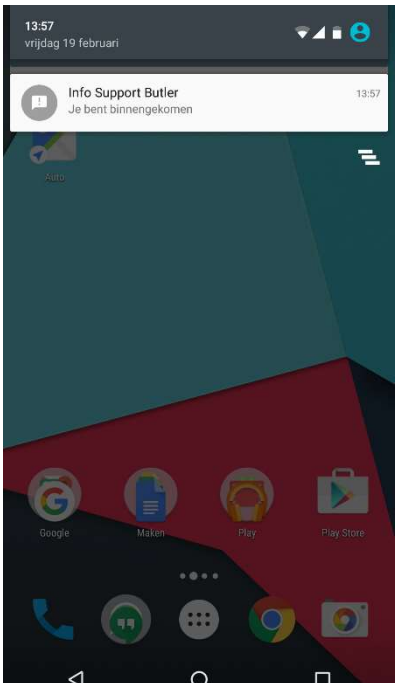
Voor de gebruiker is het mogelijk om door middel van schakelaars een systeemkoppeling aan of uit te zetten. Een voorbeeld hiervan is te zien in afbeelding 6-12.

Figuur 6-12 Aan/uit schakelen koppeling met informatiebron

6.3.3 Sprint 3

Een van de onderdelen die in het Butler project gebouwd moet worden, is het herkennen van binnenkomst / verlaten van het gebouw. In een eerder sprint is al een herkenning van beacons ingebouwd. Om een betere herkenning te bouwen, ga ik in deze sprint hier verder op in.

Daarnaast heb ik in sprint 3 ook de applicatie meer stijl gegeven en de gebruiker mogelijkheden gegeven om extra profielinformatie op te geven.



Figuur 6-13 Notificatie bij herkennen binnenkomst

Herkennen van binnenkomst/verlaten

Zoals in de opdrachtomschrijving (hoofdstuk 3.2) te lezen is, kan het herkennen van binnenkomst/verlaten gedaan worden met behulp van beacons. Hierbij zijn verschillende oplossing te kiezen, zoals het gebruik van één of twee beacons.

Met behulp van een beacon kan een mobiele telefoon herkennen of een gebruiker in de buurt van zo'n beacon is. Nadat een mobiele telefoon een beacon heeft herkent, kan er een actie getriggerd worden zoals het laten weergeven van een notificatie aan de gebruiker. (Afbeelding 6-13)

Zoals hierboven vermeld zijn er meerdere mogelijkheden voor het herkennen van een richting. Hierbij werd in eerste instantie gedacht aan één beacon. Echter kan er bij gebruik van één beacon tegen verschillende problemen aangelopen worden:

- Wat gebeurt er als de gebruiker langs de buitenkant van het gebouw loopt?
- Wat gebeurt er als een actie wordt gemist (door een telefoon die leeg is / uit staat)
- Hoe reageert het systeem als binnen korte tijd (bijvoorbeeld een paar meter verderop) opnieuw de beacon herkend wordt?

Al deze problemen kunnen opgelost worden als er twee beacons gebruikt worden voor de herkenning van een actie (binnenkomen / verlaten). Met twee beacons kan de applicatie zo ontworpen worden, dat de applicatie kijkt naar de volgorde van de beacons. Door te controleren of alleen de volgende acties mogelijk zijn, worden alle bovenstaande problemen gefilterd door de applicatie.

Met twee beacons kan de richting op de volgende manier geregistreerd worden:

- Beacon 1 (buiten) -> Beacon 2 (binnen)
  De gebruiker is naar binnen gegaan
- Beacon 2 (binnen) -> Beacon 1 (buiten)
  De gebruiker is naar buiten gegaan

In de tabel 6-2 staat uitgewerkt wat de verschillen zijn in de reactie van een mobiele applicatie bij één en twee beacons.

Gebruikers actie	1 beacon	2 beacons
Binnenkomst	Binnen	Binnen
Verlaten	Buiten	Buiten
Binnenkomst	Binnen	Binnen
2 meter verder gelopen	**Buiten**	
Verlaten	**Binnen**	Buiten
Langs de ingang lopen	**Buiten**	
Binnenkomst	Binnen	Binnen
Verlaten	Buiten	Buiten
Binnenkomst	Binnen	Binnen
Verlaten (mobiel uit)		
Binnenkomst	**Buiten**	Binnen

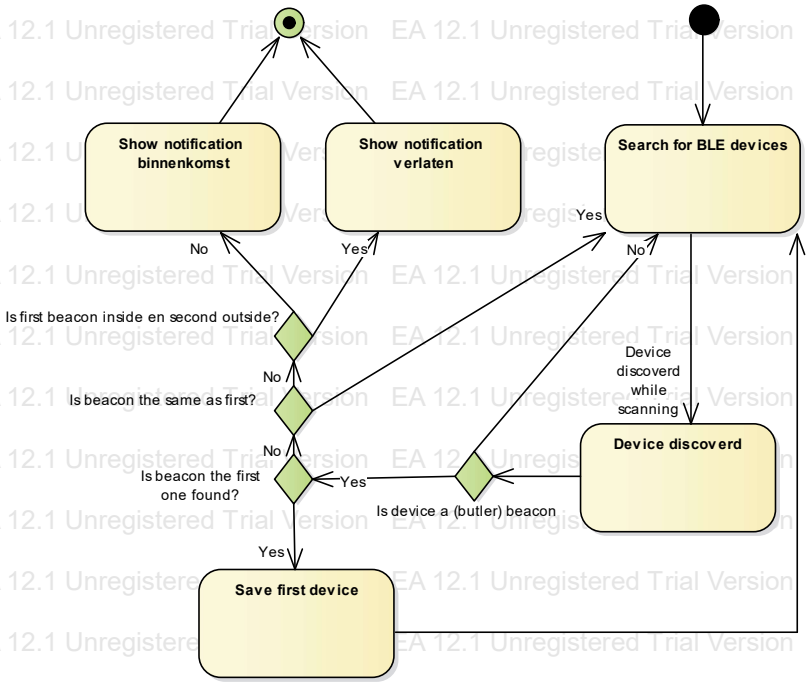
Tabel 6-2 Beacon herkenning

Zoals hierboven te zien is, reageert de applicatie bij twee beacons in de bovenstaande gevallen altijd correct en kan bij één beacon het herkenningsproces verstoord worden door een foutieve actie. Hierdoor heb ik ervoor gekozen om in sprint 3 de applicatie aan te passen naar 2 beacons.

Bepalen wanneer actie plaats vindt

Een gebruiker kan op verschillende manieren langs de in en uitgang lopen. Om te zorgen dat de juiste actie gekozen wordt en de juiste notificatie getoond wordt heb ik dit proces uitgetekend in een activity diagram.

Zoals in afbeelding 6-14 te zien is, wordt de scan naar BLE apparaten altijd gestart. Op moment dat er een apparaat gevonden wordt, gaat het resterende proces van start.



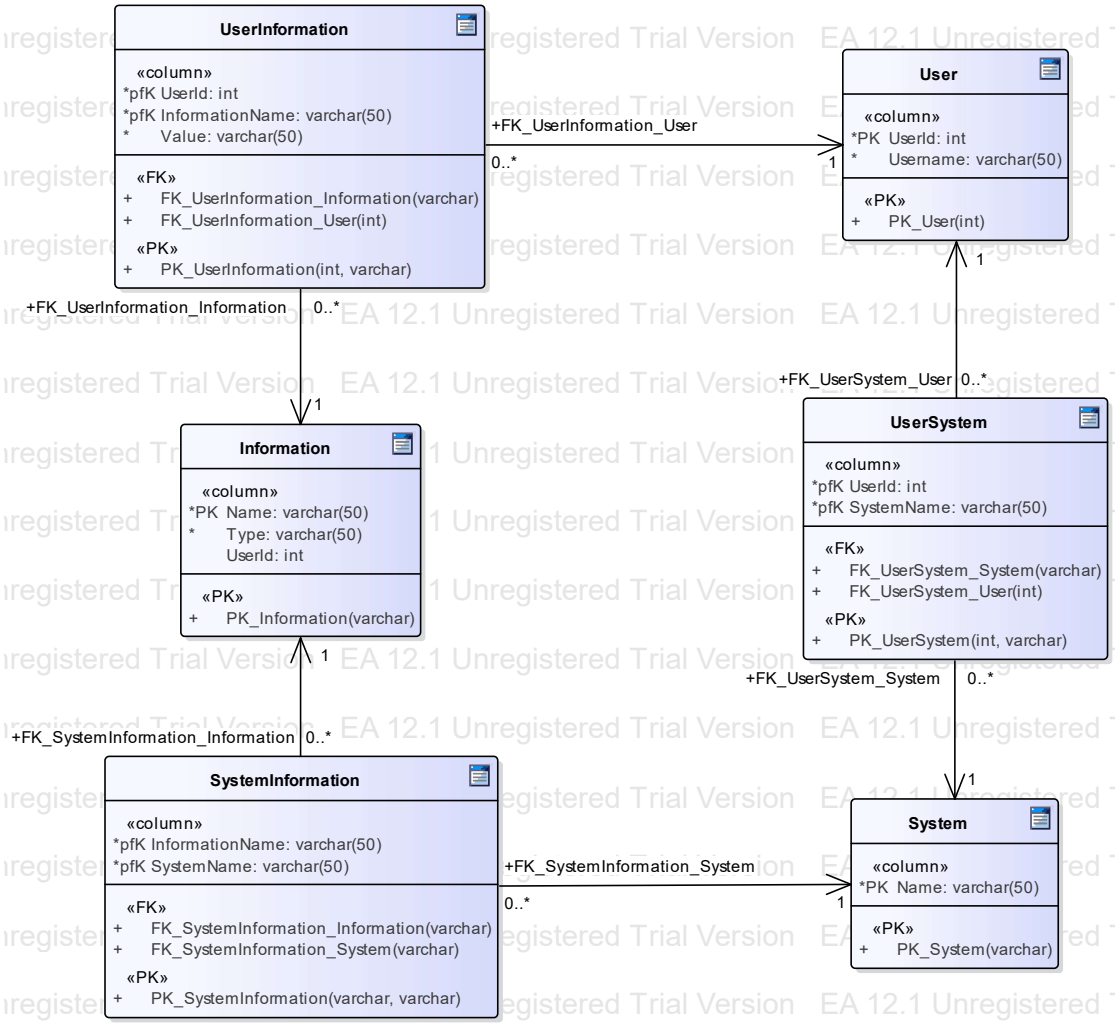
Figuur 6-14 Activity diagram - Beacon herkenning

Extra profielinformatie koppelen

Een aantal systemen hebben gegevens van/over de gebruiker nodig om gegevens op te kunnen halen. Hierbij kan het ook voorkomen dat meerdere systemen bepaalde informatie (bijvoorbeeld een thuisadres) nodig hebben. Een aantal voorbeelden van deze extra informatiegegevens zijn:

- Routeinformatie (Bestemmingsgegevens)
- Weerinformatie (Tijdstip)

Om te zorgen dat informatie niet dubbel opgeslagen wordt en er relaties aanwezig zijn tussen informatiegegevens en systemen is de volgende database diagram ontworpen.



Figuur 6-15 Database ontwerp - uitbreiding met profielinformatie

Om te zorgen dat de mobiele applicatie niet hoeft te weten hoe alle specifieke velden getoond moeten worden aan de gebruiker, heb ik elk informatieveld een type gegeven. Enkele voorbeelden van types zijn: Adres, datum en tijdstip. Hierdoor kunnen meerdere informatievelden gebruik maken van een bepaald type veld. De mobiele applicatie hoeft hierdoor alleen kennis te hebben van de verschillende types, waardoor bij nieuwe systemen eenvoudig nieuwe velden geïntroduceerd kunnen worden.

6.3.4 Sprint 4

De vierde sprint bestaat uit het verbeteren van de UI, zorgen dat de applicatie voor beide platformen (Android en iOS) gelijk is en het toevoegen van het systeem 'Slack'.

Hoewel het verbeteren van de UI vooral kleine verbeteringen waren waardoor de applicaties er een stuk interessanter uitzagen, was voor het toevoegen van het systeem 'Slack' meer tijd nodig. Voor het ophalen van informatie uit 'Slack' was het nodig om extra authenticatie in te bouwen.

UI wijziging

Om te zorgen dat de applicatie aantrekkelijker wordt voor zowel de opdrachtgever en gebruikers, heb ik in sprint 4 verschillende wijzigingen doorgevoerd op UI vlak.

- Elke systeem heeft een bijpassend icoontje (zie afbeelding 6-16)
- De applicatie heeft een eigen logo gekregen
- De witruimtes rondom teksten zijn aangepast, zodat het rustiger oogt.

Figuur 6-16 - Mobiele applicatie - UI wijziging

Naast deze punten is ook gekeken naar de verschillen tussen de iOS en Android applicatie. Ik heb hierbij wat extra aanpassingen gedaan, zodat het op beide besturingssystemen er overzichtelijk uitziet.

Informatie weergeven van 'Slack'

In eerdere sprints zijn 2 informatiebronnen toegevoegd aan de applicaties: Nu.nl en OpenWeatherMap. Met 'Slack' wordt er een nieuwe informatiebron geïntroduceerd met andere eigenschappen dan de voorgaande systemen.

Bij Slack kan alleen informatie opgehaald worden over een specifieke gebruiker. Hiervoor is het noodzakelijk dat de gebruiker de applicatie toestemming geeft dat wij zijn gegevens mogen gebruiken. Slack gebruikt voor deze toestemming het OAuth protocol. Waarmee wij een token krijgen om gegevens op te halen en geen gebruikersnaam of wachtwoord nodig te hebben.

Om een systeem als Slack mogelijk te maken in de Butler zijn er verschillende wijzigen toegepast welke hieronder beschreven staan.

Wijzigingen in de mobiele applicatie

Zoals besproken met de opdrachtgever moet de mobiele applicatie een authenticatiescherm tonen na het activeren van het desbetreffende systeem.

Dit authenticatiescherm is een internetpagina van het systeem waar wij informatie vandaan willen hebben. De URL van deze pagina bestaat, naast het standaard gedeelte, uit een API key en de benodigde permissies. Deze URL wordt opgeslagen in de database (afbeelding 6-15) in de 'System' tabel, waarbij er een extra kolom gecreëerd is. Door deze URL op te slaan in de database, wordt de onderhoudbaarheid

van de applicatie verhoogd. Daarnaast hoeft er ook geen code geüpdatet te worden indien de URL gewijzigd moet worden.
Nadat een gebruiker het authenticatiescherm volledig is doorlopen, zal de externe webservices (Slack) een accesstoken terug geven. Die accesstoken wordt afgevangen door de mobiele applicatie en (op de achtergrond) doorgegeven aan de universele connector.

Wijzigingen in de universele connector
Naast de mobiele applicatie heeft ook de universele connector te maken met een aantal wijzigingen om authenticatie mogelijk te maken. Een aantal wijzigen zoals de extra database kolom en het ontvangen van een accesstoken zijn hierboven al besproken. Andere wijzigingen die benodigd zijn:

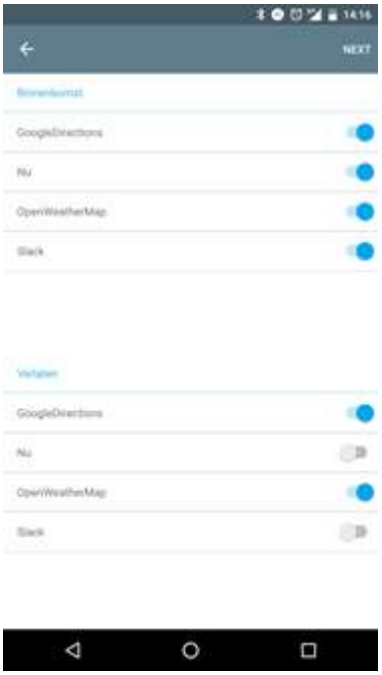
- Accesstoken omzetten naar accesscode;
- Ophalen van data met behulp van de accesscode.

De accesstoken is een tijdelijke token, waarmee een definitieve accescode (binnen 30 minuten) opgevraagd kan worden bij Slack. Deze definitieve accesscode moet vervolgens in de database opgeslagen worden en gebruikt worden bij het ophalen van gegevens bij Slack.

6.3.5 Sprint 5

In sprint 2 is de functionaliteit 'koppelen van systemen' gebouwd. Hierbij geldt dat de koppeling voor zowel binnenkomst als verlaten van het gebouw geldig is. Nadat ik in sprint 3 het herkennen van de verschillende acties (binnenkomst / verlaten) heb gemaakt, kan ik in de laatste sprint het koppelen van systemen aan acties bouwen.

Naast deze wijzigingen heb ik ook gekeken hoe eenvoudig het is om een extra informatiebron toe te voegen aan de applicaties. Hierbij heb ik routeinformatie van Google toegevoegd aan de Butler.



Figuur 6-17 Mobiele applicatie - Systemen koppelen aan verschillende acties

Koppelen aan een actie

Om de bestaande code te wijzigen van algemeen koppelen naar het koppelen aan een actie, waren een paar wijzigingen nodig aan de applicaties:

- Extra kolom in de database;
- API aanpassen;
- Mobiele applicatie een dubbele lijst geven.

Hieronder staat uitgelegd wat de wijziging aan de database is en hoe de nieuwe API aangeroepen moet worden.

Database
Aan het databasemodel van sprint 3 (afbeelding 6-15) moet één kolom toegevoegd worden om de koppeling aan een actie plaats te laten vinden.

De tabel die gebruikt is voor de koppeling 'UserSystem' krijgt hierdoor de kolom 'Actie' (type: varchar(3)), waardoor in deze kolom de waardes 'IN' of 'OUT' gebruikt kunnen worden.

API aanpassen
De API van de universele connector kon alleen doorgeven welk systeem er gekoppeld moest worden. Daarom heb ik de methode aangepast, zodat het twee waardes kan accepteren:

```
public HttpResponseMessage AddSystem(string system, string action){ ... }
```

Hierdoor kan nu de API aangeroepen worden met een systeem en een actie in de URL. Een voorbeeld van het koppelen van Nu.nl bij binnenkomst is:

http://universalconnector.azurewebsites.net/Api/User/System?system=Nu&action=IN

Toevoegen van informatiebron 'Google routeinformatie'

Met het toevoegen van 'Google routeinformatie' in de laatste sprint, heb ik uitgeprobeerd hoe eenvoudig het is om een nieuwe informatiebron te introduceren.

Zoals verwacht waren er weinig aanpassingen nodig om een nieuw systeem toe te voegen. Wijzigingen die hiervoor in de universele connector toegepast moeten worden zijn:

- Twee classes toevoegen
  - Ophalen van data
  - Structuur opgeven waarin data geretourneerd wordt naar de client
- Nieuwe regels aanmaken in de database
  - Weergeven van een nieuw veld (tabel 'Information')
  - Toevoegen van een nieuw systeem (tabel 'System')
  - Koppeling tussen beide tabellen ('SystemInformation')

In de mobiele applicatie was het nodig de volgende aanpassingen te doen aan code:

- Toevoegen van classes
  - Het ophalen van gegevens
  - Opgeven van structuur waarin data verkregen wordt (komt overeen met de class uit de universele connector)
- Maken van een ContentPage met de XAML opmaak voor dit systeem.

Tijdens het gehele project heb ik nagedacht over toekomstige features en de onderhoudbaarheid van de applicatie. Hierdoor was het mogelijk om de toevoeging van deze nieuwe informatiebron binnen één uur te ontwerpen, realiseren en te testen.

7 Evaluatie

7.1 Productevaluatie

Tijdens mijn afstudeerperiode heb ik verschillende (tussen)producten opgeleverd aan mijn opdrachtgever en technisch begeleider. In de onderstaande paragrafen staat beschreven welke producten er zijn opgeleverd. waar deze producten voor hebben gediend en of hiermee het doel van het project is behaald.

7.1.1 Plan van aanpak

Het doel van het plan van aanpak is het inzichtelijk krijgen van het gehele butler project:

- Wat moet er gedaan worden?
- Welke risico's zijn hier verbonden?
- Hoe ga ik het project aanpakken?
- Wanneer worden producten opgeleverd?

Met behulp van dit document is voor mij, mijn opdrachtgever en technisch begeleider de opdracht vastgelegd, zodat het voor iedereen duidelijk was wat mijn doelen waren voor dit project.

Zoals te lezen in hoofdstuk 4.1 is de planning, zoals gemaakt in het plan van aanpak, ook mijn uiteindelijk planning geweest. Hiermee is dit document een goede houvast geweest om te controleren of mijn afstudeerproject op schema liep. Daarnaast heeft het mij ook meer inzicht gegeven in de werkzaamheden die gedaan moesten worden en heeft het ervoor gezorgd dat de opdrachtgever en ik op de zelfde lijn zaten.

7.1.2 Onderzoek

Het doel van het onderzoek (zie ook hoofdstuk 5) was het kijken naar welke informatiebronnen geraadpleegd worden en wat de mogelijkheden zijn omtrent het maken van een Universele Connector. De interviews hebben mij en de opdrachtgever meer inzicht gegeven in de verschillende systemen die gebruikt worden en de manieren waarop deze worden benaderd.

Met het onderzoeksrapport (Bijlage D – Onderzoeksrapport) heb ik een lijst met bronnen achterhaald en uitgezocht welk framework gebruikt kan worden voor het maken van een universele connector. Op basis van deze informatie heb ik een connector gebouwd in één van de onderzochte frameworks. Ook zijn er, in overleg met de opdrachtgever, verschillende informatiebronnen toegevoegd aan de connector.

7.1.3 Documentatie applicaties

Zoals in het afstudeerplan beschreven is, worden aan het eind van het afstudeertraject de volgende documentatie-producten opgeleverd:

- Architectuur document (Bijlage E – Architectuur)
- Requirements document (Bijlage F – Requirements)
- Functioneel ontwerp (Bijlage G – Functioneel ontwerp)
- Technisch ontwerp (Bijlage H - Technisch ontwerp)

Door voorafgaand aan het ontwikkelen een opzet te maken van de verschillende ontwerpen, ben ik bewuster gaan nadenken over de te maken onderdelen. Daarnaast hebben (een aantal van) deze documenten ook geholpen bij de uitleg van de applicaties aan de opdrachtgever en technisch begeleider.

7.1.4 Applicaties

In het Butler project zijn verschillende applicaties opgeleverd:

- Android applicatie
- iOS applicatie
- Backend (universele connector)

Tijdens het ontwikkelen ben ik erachter gekomen (zie ook hoofdstuk 6.1.1) dat de gekozen ontwikkelomgeving (Xamarin.Forms) geen ondersteuning heeft voor achtergrondprocessen. Hierdoor is het niet mogelijk geweest om volledig te voldoen aan de eisen van de opdrachtgever.

Bij een volgend project waarin de keuze voor de ontwikkelomgeving bij mij ligt, is het beter om eerst een kleine proof of concept te bouwen of een kort onderzoek te doen naar de mogelijkheden. Hierdoor wordt voorkomen dat tijdens de ontwikkeling functionaliteiten niet toegevoegd kunnen worden.

Naast dit probleem zijn de verschillende applicaties naar tevredenheid van mij, mijn opdrachtgever en technisch begeleider gebouwd. Door in de laatste sprint eenvoudig een systeem toe te kunnen voegen, heb ik aan kunnen tonen dat ik de juiste keuzes heb gemaakt en de onderhoudbaarheid van mijn applicatie hoog is gebleven.

7.2 Procesevaluatie

Tijdens de gehele afstudeerperiode zijn verschillende producten gemaakt, waarbij verschillende methode zijn gehanteerd om tot een eindproduct te komen. De onderstaande paragrafen geven inzicht in deze processen en tegen welke problemen ik aan ben gelopen.

7.2.1 Plannen

Tijdens de eerste weken van de afstudeerperiode zijn het onderzoek en de eisen van de opdrachtgever concreter geworden. Hierdoor heb ik in het plan van aanpak, zoals ook te lezen in hoofdstuk 4.1, een nieuwe planning gemaakt. Deze planning wijkt af van de planning in het afstudeerplan, maar is uiteindelijk wel overeengekomen met daadwerkelijke situatie. Door deze nieuwe planning heb ik geen vertraging opgelopen in het afstudeerproject en hebben een volledig onderzoek kunnen uitvoeren en alle eisen kunnen verwerken in de applicatie.

Ondanks dat de nieuwe planning overeenkomt, zijn er ook verbeterpunten voor een volgend project:

- Concreter opschrijven hoeveel tijd voor een bepaald product nodig is of opschrijven hoeveel tijd er maximaal aan besteed gaat worden. Hierdoor kunnen activiteiten beter gepland worden.
- Meenemen van overige activiteiten als een review of voortgangsgesprek, zodat deze uren niet ten kosten gaan van bijvoorbeeld het ontwikkelen.

7.2.2 Onderzoeken

Het opstarten van het onderzoek liep niet geheel vlekkeloos. Het was lastige om de goede onderzoeksvragen (hoofd- / deelvragen) te komen. Hierdoor heb ik, in overleg met mijn technisch begeleider, eerst een longlist van vragen opgesteld (zie ook hoofdstuk 5.1), welke met de opdrachtgever is besproken.

Een volgende keer dat ik een onderzoek moet uitvoeren voor een opdrachtgever, waarbij geen duidelijke hoofd en deelvragen aanwezig zijn, kan ik gelijk beginnen met het opstellen van een longlist. Hierdoor wordt veel tijd bespaart en is na het bespreken van de longlist ook duidelijk wat de opdrachtgever verwacht.

Het doen van het onderzoek en resultaten opschrijven in een rapport zijn volgens planning verlopen. Hierdoor is het onderzoek volledig gedaan en zijn mijn begeleider en ik tevreden over het verloop van dit proces.

7.2.3 Ontwikkelen

Het ontwikkelen van de applicaties is gedaan met de ontwikkelmethode 'Scrum'. Normaal wordt Scrum toegepast in projecten waarbij er een ontwikkelteam is, bestaande uit meer dan één persoon.

Doordat ik als enige aan het Butler project heb gewerkt, heb ik een aantal scrum onderdelen anders toegepast dan volgens de standaarden hoort. In hoofdstuk 4.3.1 is volledige lijst met aangepaste scrumonderdelen terug te vinden.

Doordat ik mee deed aan de stand-up meeting van een ander team, ben ik vooraf gaan nadenken over mijn werkzaamheden. Hierdoor kon ik mijn werkzaamheden beter plannen en heb ik de planning goed kunnen volgen.

7.3 Beroepstaken

In het afstudeerplan heb ik vier beroepstaken opgegeven, waarvan ik verwachtte ze te laten zien met dit project. In de onderstaande paragrafen worden deze beroepstaken besproken en leg ik uit hoe ik aan deze beroepstaken voldoe.

7.3.1 Beroepstaak 2.1 – Opstellen gegevensmodel voor database

Voor de beroepstaak 'Opstellen gegevensmodel voor database' heb ik een databasemodel opgesteld (zie sprint 3 en 4) dat voldoet aan verschillende eisen aan de universele connector. Hierbij zijn veel-op-veel relaties aangebracht tussen drie soorten gegeven (user, system en information). Deze relaties zijn in afbeelding 6-15 uitgewerkt door middel van koppeltabellen.

Naast de verschillende tabellen is in het ontwerp ook rekening gehouden met verschillende constraints zoals: een primary key (met 1 of 2 kolommen), kolommen die niet null mogen zijn en verschillende foreign keys.

Zoals vermeld in mijn afstudeerplan, zou ik een gegevensmodel maken waarbij verschillende multipliciteiten gebruikt worden en er verschillende soorten relaties aanwezig waren. Hoewel mijn huidige databasemodel niet aan deze voorwaarden voldoet, ben ik van mening dat ik aan de complexiteit 'Lastig' voldoe door de verschillende keuzes die gemaakt zijn in het eind ontwerp.

7.3.2 Beroepstaak 2.2 – Ontwerpen, bouwen en bevragen van een database

De beroepstaak 2.2 is in het afstudeerplan geschat op een complexiteit 'Lastig/Complex'. Tijdens het afstuderen heb ik minimaal het niveau 'lastig' bereikt door implementatie van onder andere de volgende functionaliteiten:

- Er is ondersteuning ingebouwd voor meerdere gebruikers;
  Een gebruiker moet op een mobiele device kunnen inloggen en zijn eigen profiel met koppelingen terug krijgen. Ook als de gebruiker op een ander device inlogt moet de lijst overeenkomen.

- Aan de hand van een gebruiker (userid) wordt verschillende data teruggegeven;
  Zoals hierboven ook staat wordt data terug gegeven afhankelijk van de gebruiker. Bij het aanroepen van de database (bijvoorbeeld voor een lijst met gekoppelde systemen) krijgt het systeem alleen informatie over de ingelogd gebruiker.

- Complexiteit queryies.
  Door de veel-op-veel relaties tussen user, system en information, zijn er queries geschreven die data ophalen uit een koppel tabel. Tegelijk met deze data moeten ook andere tabellen geraadpleegd worden voor meer informatie. Hierdoor zijn er verschillende queries geschreven die deze verschillende tabellen raadplegen.

7.3.3 Beroepstaak 3.2 – Ontwerpen systeemdeel

Voor het butler project zijn er 2 systemen ontworpen:

- De mobiele applicatie;
- De universele connector.

Bij het ontwerpen van de verschillende systemen is rekening gehouden met toekomstige functionaliteiten. Dit is gedaan door interfaces en abstracte classes te gebruiken indien er functionaliteiten aanwezig waren die bij meerdere classes overeenkomen (zie ook hoofdstuk 6 - Ontwikkeling). Daarnaast hebben alle gemaakte onderdelen (modules) relaties met elkaar, zoals te zien is in het architectuur overzicht (afbeelding 6-4).

Het ontwerpen van een systeemdeel heeft minimaal het complexiteitsniveau 'lastig' bereikt doordat er meerdere objectgeoriënteerde applicaties (universele connector en mobiele applicatie) zijn ontworpen. Hierbij is tijdens het ontwerpen en bouwen gelet op hergebruik en onderhoudbaarheid van de code, zodat toekomstige ontwikkelaars eenvoudig extra informatiebronnen kunnen toevoegen.

Voor alle use case, class, sequence en activity diagrammen die in dit afstudeerplan, requirements document, architectuur document, functioneel ontwerp en technisch ontwerp te vinden zijn, is het softwarepakket Enterprise Architect gebruikt.

7.3.4 Beroepstaak 3.3 – Bouwen applicatie

De beroepstaak 'Bouwen applicatie' is uitgevoerd op een Complex niveau doordat er verschillende applicaties gebouwd zijn welke voldoen aan de volgende eisen:

- Geen complexiteit in de code;
  Zoals in de verschillende ontwerpen te zien is, bevatten de classes alleen de benodigde methodes en variabelen.

- Hergebruik van code;
  De code is opgebouwd met behulp van abstracte classes, interfaces en static classes. Hierdoor kan code gedeeld worden tussen verschillende classes en methodes en hoeft dit niet onnodig gekopieerd te worden.

- Verschillende OTAP omgevingen toegepast;
  Zoals in hoofdstuk 4.3.3 te lezen is, zijn er drie omgevingen toegepast binnen het Butler project.

- Code wordt beheerd met versiebeheer;
  Met behulp van TFS (zie ook hoofdstuk 4.3.2) kunnen wijzigingen in de code bijgehouden worden.

- Gebruik van programmeerstandaarden.
  Tijdens het ontwikkelen is rekening gehouden met de C# programmeerstandaarden, zoals een interfacenaam dat met een I begint en een classnaam dat met een hoofdletter begint.

Literatuurlijst

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.*

GIT - Het wat en waarom van versiebeheer. (sd). Opgehaald van https://git-scm.com/book/nl/v1/Aan-de-slag-Het-wat-en-waarom-van-versiebeheer

IBM | Using Internet data in Android applications. (sd). Opgeroepen op Januari 4, 2016, van http://www.ibm.com/developerworks/library/x-dataAndroid/

InfoSupport | Geschiedenis Info Support. (sd). Opgehaald van http://www.infosupport.com/geschiedenis-info-support/

MSDN Microsoft . (sd). Opgehaald van https://msdn.microsoft.com/en-us/library/x2dbyw72(v=vs.71).aspx

Slack OAuth. (sd). Opgeroepen op December 14, 2015, van https://api.slack.com/docs/oauth

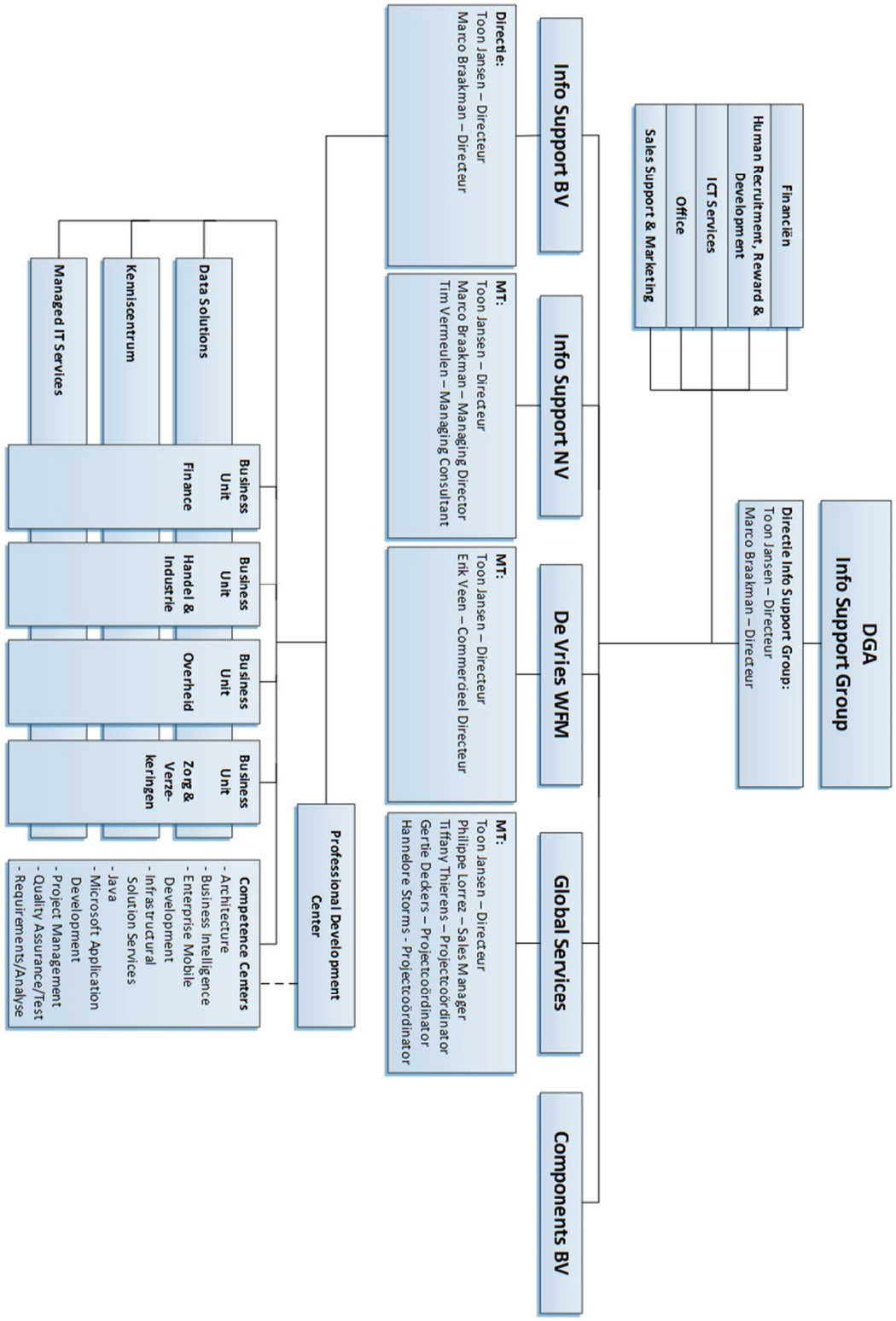
Xamarin Forms. (sd). Opgehaald van xamarin.com/forms

Begrippenlijst

Begrip	Omschrijving
API	Een API definieert de toegang tot functionaliteiten van de backend. De frontend hoeft hierdoor niet de implementatie van de verschillende functionaliteiten te weten, maar hoeft alleen via de API de gegevens op te vragen.
Azure	Azure is een cloud computing-platform van Microsoft, vergelijkbaar met de App Engine van Google en EC2 van Amazon. Met Azure biedt Microsoft verschillende (online) producten aan die los en gecombineerd met elkaar gebruikt kunnen worden.
Backend	De backend is het deel van de applicatie dat niet zichtbaar is voor de gebruiker. De frontend kan de backend aanspreken via API's.
Bluetooth Low Energie (ook BLE of Bluetooth Smart genoemd)	BLE is een protocol waarmee zenders gemaakt kunnen worden die weinig energie hebben en die specifieke informatie willen delen. (bijv. locatie-, fitness- of lichaamsgegevens)
Emulator	Een (software) emulator is een stuk software dat het mogelijk maakt software te draaien die voor andere hardware ontwikkeld is. Een voorbeeld hiervan is de Android emulator waarmee Android applicaties op Windows getest kunnen worden.
TFS (Team Foundation Server)	TFS is een online applicatie, ontwikkeld door Microsoft, waarmee de ontwikkeling van een applicatie ondersteund kan worden. Zo kan het onder andere de broncode van een applicatie beheren, automatische build starten en heeft het de mogelijkheid om als scrumboard te dienen.
Universele Connector	Een connector is 'een systeem' waar verschillende andere systemen, met verschillende protocollen, op aangesloten kunnen worden. De connector zet dit om naar één API of protocol. Hierdoor hoeft de frontend niet allerlei protocollen te ondersteunen en hoeft bij een wijziging aan een extern systeem niet de frontend geüpdatet te worden.
Versiebeheersysteem	Een versiebeheersysteem is een (computer)programma waarmee wijzigingen, die gemaakt worden in documenten of software, bijgehouden kunnen worden

Interne bijlages

Bijlage A - Organogram Info Support



Externe bijlages

Bijlage B – Plan van aanpak

Bijlage C – Onderzoeksplan

Bijlage D – Onderzoeksrapport

Bijlage E – Architectuur

Bijlage F – Requirements

Bijlage G – Functioneel ontwerp

Bijlage H - Technisch ontwerp