2020

Afstudeerverslag

Donation Marketing Journey editor

Naam:	Sjoerd van der Kraan
Studentnummer:	16091183
Opleiding:	HBO-ICT deeltijd
Hogeschool:	De Haagse Hogeschool (Den Haag)
Datum verdediging:	29-06-2020
Examencomissie:	Peter Becker
	Job Habraken

Voorwoord

Voor u ligt het afstudeerverslag "Donation Marketing Journey editor". In dit verslag wordt duidelijk gemaakt welke stappen er ondernomen zijn om de applicatie, de Donation Marketing Journey editor, te realiseren. Er is een kort onderzoek uitgevoerd naar methoden voor het testen van de applicatie. Naar aanleiding van de resultaten uit het onderzoek is een teststrategie ontwikkeld en geïmplementeerd. Dit verslag is geschreven in het kader van mijn afstuderen voor de opleiding Software Engineer (Deeltijd) aan de Haage Hogeschool Den Haag en in opdracht van Pixelzebra. De afstudeerperiode liep van 10 februari 2020 tot 5 juni 2020.

Bij dit afstudeertraject lag de nadruk op het bouwen van de applicatie en op het aantonen van beroepstaken. Ik heb een kleinschalig onderzoek uitgevoerd en geïmplementeerd in zes dagen. Dit was zeer complex om in een korte tijd te realiseren. Tijdens het realiseren van de applicatie stonden mijn werkplekbegeleiders, Robert van der kleij en Paddy van der Kaay, altijd voor mij klaar. Voor het opstellen van de documenten kon ik terecht bij mijn afstudeerbegeleider, Peter Becker. Zij hebben mijn vragen beantwoord waardoor ik verder kon bij onduidelijkheden.

Bij dezen wil ik graag mijn begeleiders bedanken voor de fijne begeleiding en hun ondersteuning tijdens dit traject. Zonder hun medewerking had ik de applicatie en het onderzoek niet kunnen voltooien. Tevens wil ik mijn collega's bij Pixelzebra graag bedanken. Ik heb vaak met hen op effectieve wijze kunnen sparren over het afstuderen. In het bijzonder wil ik Danny Nieuwmans bedanken voor de morele steun en de vele uren dat wij met elkaar hebben gespard. Ook wil ik mijn dank betuigen aan Svetlana van der Kraan, mijn vrouw. Zij heeft mij vier jaar lang op alle fronten ondersteund. Tot slot wil ik mijn ouders bedanken. Hun wijsheid en motiverende woorden hebben mij geholpen het afstuderen tot een goed einde te brengen.

Ik wens u veel leesplezier toe.

Sjoerd van der Kraan

Hazerswoude-Rijndijk, 30-05-2020

Samenvatting

Traditionele softwaresystemen voor donaties gaan veelal uit van eenmalige giften en maandelijkse facturatiebatches voor donaties, maar de wereld van goede doelen en donaties verandert snel. Consumenten geven nog steeds aan goede doelen, maar willen meer inzicht en controle in hun giften. Daarnaast neemt het doneren op structureel niveau af. Dit betekent dat er naast de structurele donaties meer mogelijkheden moeten komen voor individuele persoonlijke donaties en dat de betrokkenheid met donateurs moet worden vergroot. Om ANBI-instellingen de mogelijkheid te bieden in te spelen op het probleem, wilde Pixelzebra een platform voor hun klanten ontwikkelen. Op dit platform kunnen gepersonaliseerde marketing vervolgstappen geconfigureerd worden, ook wel een Donation Marketing Journey genoemd. Hiermee hopen zij donateurs meer te betrekken bij hun donaties, met als gevolgd dat zij vaker doneren of langer blijven doneren.

Het verslag is opgesteld in chronologische volgorde naar aanleiding van de uitvoering. Het Plan van Aanpak diende als leidraad van afstudeertraject, gevolgd door het opstellen van de requirements, indelen van de backlog en uitwerken van de architectuur. Om het development proces structureel aan te pakken zijn onderdelen van Scrum toegepast. De sprints hadden een tijdsduur van twee weken en werden afgesloten met een retrospective. Hierin werd samen met de Product Owner (Paddy van der Kaay) en de werkplekbegeleider (Robert van der Kleij), bepaald of de sprint succesvol was. Er werd beoordeeld of de gemaakte onderdelen voldeden aan de eisen van de User Stories. Hiernaast werd gekeken of de taken waren afgerond, of dat de opvolgende sprint moest worden aangepast. Om antwoord te kunnen geven op de hoofdvraag en deelvragen van het onderzoek, is er een Multivocal Literature Review (MLR) studie verricht. De resultaten van het onderzoek droegen bij aan de teststrategie, waarna een labonderzoek werd uitgevoerd.

Om de Donation Marketing Journey editor te realiseren is een front-end ontwikkeld en geïntegreerd met het Dynamics 365 CRM-softwarepakket. Hierbij is gebruik gemaakt van verschillende frameworks en libraries, zoals Vuejs, C#, JavaScript, HTML en CSS. Met behulp van Azure Functions kunnen berekeningen worden uitgevoerd en Spotler Campagnes worden aangeroepen vanuit het Spotler platform. Na de realisering van het Proof of Concept is er een onderzoek uitgevoerd en een passende teststrategie ontworpen. Hierbij is gekeken naar veelgebruikte testsoorten en testtools, mede op basis van de onderzoeksresultaten van de website, “The State Of JavaScript”. “Jest” is de testtools waarmee uiteindelijk de testen zijn uitgevoerd.

Tijdens het afstuderen lag de focus op het ontwerpen, ontwikkelen en implementeren van de Donatie Marketing Journey editor. Gebruikers van de editor zijn in staat om zelf de trigger, voorwaarden en vervolgacties te configureren. Door een goed geconfigureerde Donatie Marketing Journey krijgt de donateur meer inzicht in wat er gebeurt met zijn donatie en kunnen toekomstige acties worden voorgesteld bij donateurs, die hiervoor aangeven interesse in te hebben. Met behulp van een kleinschalig onderzoek is een teststrategie ontwikkeld en geïmplementeerd. Door de teststrategie te implementeren is tot de conclusie gekomen dat End-to-End testing een hogere meerwaarde kan bieden ten opzichte van Unit testing. Uiteindelijk is er een Proof of Concept neergezet dat bij verschillende klanten kan worden geïntegreerd in hun bestaande Dynamics 365 CRM-omgeving.

Inhoudsopgave

Voorwoord 1
Samenvatting 2
Inhoudsopgave 3
1 Organisatie 5
2 Opdrachtomschrijving 6
2.1 Probleemstelling 6
2.2 Doelstelling 7
2.3 Resultaat 7
2.4 Beroepstaken 7
3 Aanpak 8
3.1 Coronacrisis 8
3.2 Softwarecomponenten 9
4 Plan van Aanpak 12
5 Requirementsanalyse 14
5.1 Vergaren requirements 14
5.2 Waarborgen requirements 15
6 Componenten Architectuur 18
7 Development Donation Marketing Journey editor 20
7.1 Inleiding 20
7.2 Voorbereidingen 20
7.3 Sprint 1: De applicatie deployen als webresource in CRM 22
7.4 Sprint 2: Dynamisch opbouwen van de webpagina 23
7.5 Sprint 3: De Journey opslaan of bewerken 26
7.6 Sprint 4: Laatste hand kernfunctionaliteiten applicatie 28
8 Onderzoek ketentest en implementatie 33
8.1 Inleiding 33
8.2 Uitvoeren onderzoek 33
8.3 Maken teststrategie 34
8.4 Uitvoeren teststrategie 35
8.5 Retrospective onderzoek en implementatie 37
9 Handleiding en adviesrapport 38
10 Aantonen beroepstaken 39
10.1 A1: Analyseren probleemdomein & opstellen probleemstelling 39
10.2 Gc: Kritisch, onderzoekend en methodisch werken 40

10.3 Gf: Leren leren: voorbereiden op volgende studiefase en beroep 41
10.4 C6: Ontwerpen software 41
10.5 D14: Realiseren van software 42
10.6 D17: Configureren 42
10.7 D15: Testen 43
11 Evaluatie 44
11.1 Resultaat product 44
11.2 Algemeen 45
12 Literatuurlijst 47
13 Bijlagen 49

1 Organisatie

Pixelzebra is een snelgroeiend bedrijf met ongeveer 30 medewerkers waarvan het grootste deel bestaat uit consultants. Het pand is gevestigd in Nootdorp. Medewerkers Paddy van der Kaay (7 jaar in dienst), Mitchell Knaapen (3 jaar in dienst) en Sjoerd van der Kraan zijn tot op heden de software developers binnen het bedrijf in Nederland. Pixelzebra is een levendig en communicatief bedrijf. Het bedrijf richt zich voornamelijk op klanten die op zoek zijn naar mogelijkheden om de relatie met klanten, servicegebruikers, collega's, partners en leveranciers te versterken. Voorbeelden van klanten zijn o.a. FOX, 24Kitchen en Van Dale. Om deze bedrijven dichter bij hun klanten te brengen maken zij gebruik van Microsoft Dynamics 365 voor CRM-implementaties.

> *"Pixelzebra helpt u om uw klantrelaties inzichtelijk te maken en optimaal te beheren. Het hoofddoel van Pixelzebra is om het Microsoft platform 'Dynamics 365' in organisaties echt van de grond te krijgen en met impact te implementeren. Dat doen wij door Microsoft Dynamics 365 perfect aan te laten sluiten op uw bedrijfsprocessen. Bij Pixelzebra vinden we het namelijk belangrijk dat de aanpak van een project past bij het type organisatie en de omvang van een opdracht. Hiervoor hebben we de technische en bedrijfskundige kennis en ervaring in huis."* (Pixelzebra, 2019)

Dynamics 365 CRM biedt vele mogelijkheden voor configuratie. Het kan voorkomen dat de wens van een klant zo specifiek is, dat het niet mogelijk is om dit te configureren met de standaard functionaliteiten van Dynamics 365 CRM. In dit geval komen wij, de developers van Pixelzebra, tot de redding om het maatwerk met behulp van code te realiseren. Pixelzebra is een initiatief gestart genaamd Pixelzebra Social Matters. Dit is een tak die specifiek gericht is op goede doelen en cultuur. Op deze manier wil Pixelzebra iets teruggeven aan de maatschappij.

> *"Pixelzebra wil meer betekenen voor organisaties met een ANBI-status. Daarom hebben we een speciale oplossing ontwikkeld voor deze organisaties. Deze oplossing zal door gespecialiseerde consultants ingericht worden. Hiermee hopen we goede doelen en cultuur naar een hoger plan te helpen, waardoor ze voor nog meer impact kunnen zorgen bij hun doelgroep."* (Pixelzebra Social Matters, 2019)

2 Opdrachtomschrijving

2.1 Probleemstelling

Traditionele softwaresystemen voor donaties gaan veelal uit van eenmalige giften en maandelijkse facturatiebatches voor donaties, maar de wereld van goede doelen en donaties verandert snel. Consumenten geven nog steeds aan goede doelen, maar willen meer inzicht en controle in hun giften. Daarnaast neemt het doneren op structureel niveau af. Dit betekent dat er naast de 'lidmaatschappen' (structurele donaties) meer mogelijkheden moeten komen voor individuele persoonlijke donaties en dat de betrokkenheid met donateurs moet worden vergroot. Onderstaande trends zijn o.a. verzameld (door Pixelzebra) door interviews en requirementsanalyses af te leggen bij ABNI-instellingen.

Daarbij zien we een paar trends:

- Een gift dient geoormerkt. Bijvoorbeeld: "Ik wil alleen doneren voor kinderen uit mijn dorp die op de basisschool zitten."
- De betrokkenheid dient vergroot te worden door het relevant communiceren met de donateur op basis van zijn donatie. Wat is er gebeurd met het geld, wie is geholpen en wat is er beter geworden?
- Giften zijn meer actiegericht, eenmalig en komen voort vanuit een persoonlijke relatie of gebeurtenis. Dit vraagt om andere betaalmechanismes, zoals bijvoorbeeld "Tikkie".

> *"Het is een trend dat mensen meer betrokken willen zijn bij een doel en zelf betekenis willen geven."*
>
> *- Wilfried Nijkamp, fiscalist van ING* (ING, 2019)

2.2 Doelstelling

Om ANBI-instellingen de mogelijkheid te bieden in te spelen op de genoemde trends, wilde Pixelzebra een platform voor hun klanten ontwikkelen. Op dit platform kunnen gepersonaliseerde marketing vervolgstappen *(Donatie Marketing Journey ofwel DMJ)* geconfigureerd worden. Hiermee hopen zij donateurs meer te betrekken bij hun donaties, met als gevolgd dat zij vaker doneren of langer blijven doneren. De Donatie Marketing Journey kan geconfigureerd worden op actieniveau. Denk hierbij aan het bovenstaande voorbeeld: "Ik wil alleen doneren voor kinderen uit mijn dorp die op de basisschool zitten". De donateur zal dus alleen informatie krijgen over de acties waarin hij of zij geïnteresseerd is, of over de actie waarvoor gedoneerd is.

De afstudeeropdracht bestond uit het ontwerpen en ontwikkelen van een editor voor Donatie Marketing Journeys. Hiervoor is een grafische editor ontworpen en ontwikkeld die email campagnes aanstuurt vanuit het Spotler marketing platform. Spotler is een e-mailmarketingsysteem. Door de API van Spotler is het mogelijk de software te laten communiceren met Dynamics 365. De editor is generiek opgezet, wat betekent dat het eindproduct bij verschillende klanten geïntegreerd kan worden met hun Dynamics 365 CRM-systeem.

2.3 Resultaat

Tijdens het afstuderen heb ik mij gefocust op het ontwerpen, ontwikkelen en implementeren van de Donatie Marketing Journey editor. Gebruikers van de editor zijn in staat om zelf de trigger, voorwaarden en vervolgacties te configureren. Door een goed geconfigureerde Donatie Marketing Journey krijgt de donateur meer inzicht in wat er gebeurt met zijn donatie en kunnen toekomstige acties worden voorgesteld bij donateurs die hier interesse in hebben.

2.4 Beroepstaken

Naast de opdracht is het ook van belang dat vooraf gedefinieerde competenties worden aangetoond. In het afstudeerplan heb ik zeven beroepstaken beschreven die ik door middel van dit verslag aantoon. In hoofdstuk 10 staat de onderbouwing van de beroepstaken.

3 Aanpak

In dit hoofdstuk ga ik in de op de aanpak die ik heb gebruikt voor dit project. Hierbij onderbouw ik waarom ik onderdelen van Scrum heb toegepast, welke softwarecomponenten ik heb gebruikt en waarom en hoe ik mijn werkwijze heb moeten aanpassen door Corona.

Tijdens het maken van het Plan van Aanpak kwam ik erachter dat het toepassen van de Scrum-methodiek beter past dit project dan Kanban. Kanban wordt veelal gebruikt bij continuous development, waarbij Scrum voornamelijk wordt gebruikt voor het developen van tijdsgebonden projecten (Ismail K, 2018). Ik heb gebruik gemaakt van sprints met een tijdsduur van twee weken. De tijdsduur is bepaald op basis van het Plan van Aanpak en een stukje ervaring. In het PvA heb ik de werkzaamheden ingepland per twee weken, omdat een enkele weekinschatting te lastig was door het gebrek aan requirements. Het inschatten per drie weken daarentegen was niet concreet genoeg. Hiernaast ben ik bij Pixelzebra gewend om in sprints van twee weken te werken. Ik heb gemerkt dat de werkzaamheden hierdoor overzichtelijk blijven. Iedere twee weken keek ik samen met de Product Owner (Paddy van der Kaay), en wanneer mogelijk mijn begeleider (Robert van der Kleij), of de sprint succesvol was afgerond. Hierbij kijken we naar de taken die waren afgerond, de kwaliteit van de code en hoe de volgende sprint eruit ging zien. Dit wordt bij Scrum een retrospective genoemd. De sprints waren voor mij duidelijke deadlines en hielpen bij het tijdig realiseren van het project.

> *"Kanban isn't necessarily focused on cross-functional teams and it doesn't use sprints. We find the time-framing of sprints to be an excellent driver of increased speed, which is important in digital transformation."*
>
> *- Nicholas Carrier* (Ismail K, 2018)

Er werd geen gebruik gemaakt van de daily-standup standaard, omdat er op zaterdag en zondag geen daily-standup kon plaatsvinden. Het team is ook kleiner dan een gebruikelijk Scrum-team. Paddy van der Kaay is in dit geval de Product Owner en ik nam de rol op me van Scrum Master en Team Member. In verband met de Coronacrisis zijn we vanaf week acht de vergaderingen wekelijks gaan doen. Vóór de crisis zat ik met de Product Owner in dezelfde kamer en hadden we dagelijks contact. Dit was helaas niet meer mogelijk en hierdoor konden wij elkaar minder goed op de hoogte houden. Robert van der Kleij en ik waren het met elkaar eens, dat we elkaar meer op de hoogte wilde houden over de voortgang. Om deze reden zijn we elke vrijdag gaan zitten om de week te bespreken en de prioriteiten op orde te stellen. De sprints behielden wel de tijdsduur van twee weken.

Het verslag is opgesteld in chronologische volgorde naar aanleiding van de uitvoering. Ik ben begonnen met het Plan van Aanpak, gevolgd door het opstellen van de requirements. Na het indelen van de product backlog op het Scrumboard kon ik de architectuur uitwerken. Het developmentproces is onderverdeeld in sprints. Hierdoor is duidelijk te zien welke onderdelen ik heb opgeleverd in de desbetreffende sprint en hoe ik aan de hand van het resultaat heb ingespeeld op de opvolgende sprints.

3.1 Coronacrisis

De Coronacrisis maakte de communicatie een stuk lastiger. Zo was het bijvoorbeeld niet mogelijk om voor een klein vraagje het kantoor binnen te stappen van Robert, Paddy of andere collega's. De vraag mailen, of een vraag stellen via de Teams chat, zorgde veelal niet direct voor antwoord. Hierdoor moest er soms een meeting worden ingepland via Teams. Dit was een vrij lastige opgave aangezien de

agenda's van mijn collega's veelal waren volgeboekt. Dit was bijvoorbeeld een vervelend probleem toen bleek dat mijn Dynamics 365 CRM-trial was verlopen. Eerst moest ik uitzoeken bij wie ik moest zijn om hier een oplossing voor te verzinnen. Toen ik de juiste persoon had gevonden, en bleek dat dit een uitzonderlijke situatie was, moest contact worden opgenomen met Microsoft. De omgeving kon met wat moeite worden verlengd, maar de verlenging was maar met een maand. Hiernaast had Pixelzebra geen ruimte om de trial om te zetten naar een productieomgeving, dus moesten we samen kijken naar mogelijkheden om een bestaande omgeving te resetten. Gelukkig waren er een aantal collega's die ruimte hadden om mij te helpen. Uiteindelijk bleek dat ik al snel wat uren was verloren, voordat er een oplossing was gevonden.

3.2 Softwarecomponenten

3.2.1 Inleiding

De rol van dit hoofdstuk is om aan te geven van welke softwarecomponenten ik gebruik heb gemaakt, of gebruik van wilde maken. Sommige componenten zijn later in het proces achterwege gelaten, maar zijn wel van belang geweest om bepaalde vraagstukken helder te krijgen.

3.2.2 Vue.js

Pixelzebra is gericht op CRM-implementaties en heeft om die reden weinig projecten met een zelfgemaakt front-end. Omdat we nog zoekende zijn naar een passend front-end framework, heb ik voorgesteld om Vue.js te gebruiken als front-end framework. Op deze manier kunnen we proeven van een framework dat we nog niet hebben gebruikt. Tot op heden zijn er projecten gemaakt met Angular JS en React. De reden dat ik niet heb gekozen voor Blazor, een Microsoft front-end product, heeft te maken met mijn kennis m.b.t. Blazor en de kennis van mensen om mij heen. Blazor heb ik zelf nog nooit gebruikt en er is bij Pixelzebra niemand bekend met het framework. Daarentegen heb ik met Vue wel wat kennis opgebouwd. In het verleden heb ik een verschillende front-end frameworks onderzocht. Hier kwam uit dat Vue.js een gebruiksvriendelijk framework is, waarmee makkelijk en snel een project kan worden opgezet. Hiernaast heb ik de mogelijkheid om vragen te stellen aan Danny Nieuwmans als ik eventueel vastloop op een onderdeel. Ik werk op school als vier jaar samen met Danny en hij heeft veel ervaring op het gebied van front-end frameworks. Het belangrijkste doel bij de keuze om Vue te gebruiken, is om helder te krijgen of dit framework past bij Pixelzebra. Tijdens het developen is gebleken dat, door mijn gebrek aan front-end kennis, het oppakken van dit framework voor veel problemen zorgde. Om deze reden had ik ervoor gekozen om de front-end te realiseren met "vanilla" JavaScript en een aantal plugins. Hoe het eindoordeel van deze keuze precies tot stand is gekomen, is te lezen in hoofdstuk 7.2.

3.2.3 Azure

Om dit project te realiseren heb ik gebruik gemaakt van verschillende componenten van Azure. Pixelzebra is Microsoft Gold Partner. Dit betekent dat ze korting krijgen en sommige producten gratis mogen gebruiken. Hierdoor kunnen ze services goedkoper aanbieden bij klanten. Hiernaast vind ik het belangrijk dat de architectuur die ik ontwerp en de software die ik bouw, aansluit bij de huidige architectuur van Pixelzebra. Door gebruik te maken van Azure is de code onderhoudbaar en is het mogelijk voor collega's om het project in de toekomst gemakkelijk op te pakken. Voor het uitvoeren

van berekeningen en voor het benaderen van de Dynamics 365 CRM-database, maak ik gebruik van Azure Functions. Azure DevOps werd gebruikt als repository en om de backlog en sprints mee te definiëren. Ik maak voor mijn werkzaamheden dagelijks gebruik van Azure componenten. Om deze reden lag de uitdaging voornamelijk bij de front-end.

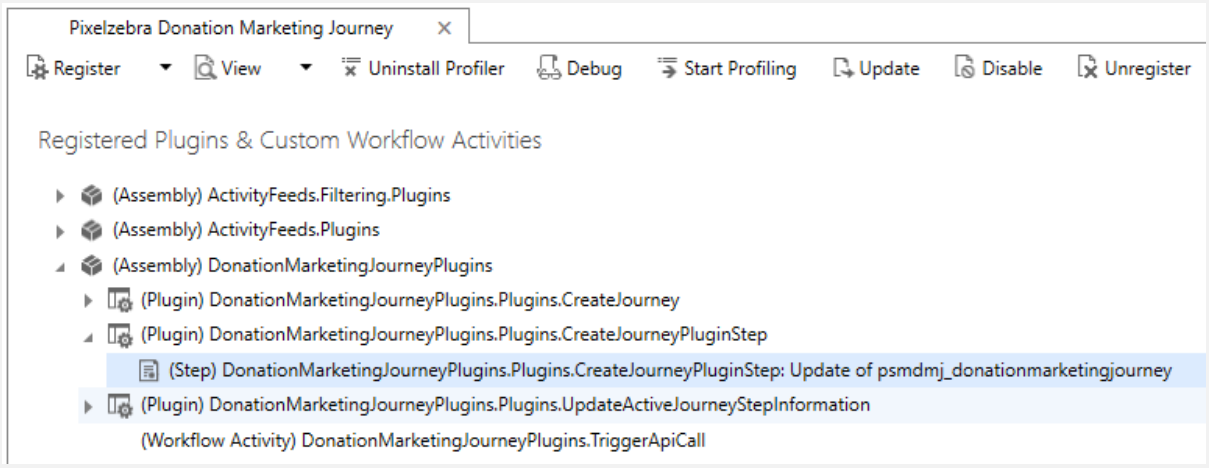
3.2.4 Dynamics 365 CRM

Dynamics 365 CRM (Customer Relationship Management) wordt gebruikt om dezelfde redenen als onderbouwd in hoofdstuk 3.2.3. Het pakket fungeert als databasemodel en configuratieplatform. Via de CRM-SDK (Software Development Kit) is het mogelijk om te communiceren met de database van CRM. De front-end van CRM maakt het mogelijk om data overzichtelijk en configureerbaar te maken. Hier kunnen zowel developers als gebruikers gemakkelijk data manipuleren.

Xrm.Sdk Om op een veilige manier CRM-data te bewerken in C#, maken we gebruik van de Microsoft Dynamics 365 SDK (Software Development Kit). Met behulp van de Xrm.Sdk kan een connectie worden gelegd met de CRM-omgeving via een Application User (te configureren via Azure). De connectie kan gebruikt worden om CRUD-acties mee uit te voeren en om informatie te loggen in CRM.

Xrm.WebApi De Xrm.WebApi wordt gebruikt om CRUD-acties mee uit te voeren vanuit de front-end. Om hier gebruik van te kunnen maken moet op een pagina de "ClientGlobalContext.js.aspx" toegevoegd worden script. Via de WebApi worden bijvoorbeeld API's aangeroepen om data te verkrijgen of business logica uit te voeren.

Om op een veilige manier CRM-data te bewerken in C#, maken we gebruik van de Microsoft Dynamics 365 SDK (Software Development Kit). Om dit te kunnen benaderen roepen we in CRM een CWA (Custom Workflow Activity) aan. De CWA voert C# code uit en geeft eventueel een bestand terug. Het bestand is voor dit project over het algemeen een Json. Een plugin fungeert over het algemeen hetzelfde, maar wordt op een andere manier aangeroepen binnen CRM. Een CWA wordt aangeroepen via een Workflow en een Plugin wordt aangeroepen via een trigger. Deze trigger kan bijvoorbeeld geconfigureerd worden met behulp van de Plugin Registration Tool (afbeelding 1).



Afbeelding 1: Plugin Registration Tool

3.2.5 Spotler

Voor het versturen van mailcampagnes had ik drie opties. Mailchimp, Hubspot en Spotler zijn programma's die onze klanten voornamelijk gebruiken. Een koppeling tussen CRM en Spotler wordt het meeste verkocht. Om deze reden willen we Spotler inzetten voor dit project, zodat we de grootste doelgroep van Pixelzebra aanspreken.

> *"Met Spotler maak je in een splitsecond onder andere nieuwsbrieven, landingspagina's, enquêtes, automatische multichannel campagnes en webformulieren. De software is gericht op niet-technische marketeers die te gekke resultaten willen behalen. Spotler is de enige e-mailmarketingserviceprovider die 4 jaar achter elkaar nummer 1 is geworden in de Emerce 100."* (Wegink, 2019)

3.2.6 Draw.io

In de afgelopen drie jaar heb ik veelvoudig gebruik gemaakt van "draw.io". De interface is intuïtief, beschikt over een groot assortiment aan modelleermodellen en het is gratis. Tijdens mijn studie heb ik verschillende softwarepakketten geprobeerd, waaronder Archimate. Dit product is ook gratis, maar moet lokaal worden uitgevoerd en voelt naar mijn mening minder intuïtief. Werken in draw.io wordt gedaan in de onlineomgeving en kan op verschillende manieren worden opgeslagen, zoals in de Cloud of lokaal. Op draw.io heb ik het architectuurdiagram gemodelleerd volgens de Archimate 3.0.1 notatiewijze en heb ik het process flow diagram uitgewerkt.

> *"Draw.io is een volledig gratis online diagrameditor waarmee u stroomdiagrammen, UML, entiteitsrelaties, netwerkdiagrammen, mockups en meer kunt maken."* (Draw.io Diagrams, 2020)

4 Plan van Aanpak

Om de opdracht goed helder te krijgen ben ik met Robert van der Kleij (begeleider) gaan zitten om de specificaties van de applicatie te bespreken. In het gesprek is naar voren gekomen hoe Robert de functionaliteiten van de applicatie voor zich ziet. Hierbij moest ik in het achterhoofd houden dat dit een aanname is en dat ik zelf een interpretatie moet geven over de werking. Door het duidelijk uitgewerkte afstudeervoorstel kwamen er geen verrassingen naar boven. Toen de opzet eenmaal helder was ben ik het PvA (Plan van Aanpak) gaan opstellen (zie bijlage 1). Hiervoor heb ik het template gebruikt van Scribbr (Benders, 2020). Het PvA heb ik opgestuurd naar mijn afstudeerbegeleider Peter Becker.

Om er zeker van te zijn dat ik het initiële idee goed heb verwoord in het PvA, ben ik dit z.s.m. gaan afstemmen met Robert van der Kleij (begeleider) en Peter Becker. Ik heb een vergadering ingepland zodat zij met elkaar kennis konden maken en een kritisch oog konden leggen op het PvA. Hierin kwam als feedback naar voren dat het onderzoek niet heel duidelijk was omschreven. Na kort overleg hebben we besloten om dit onderzoek klein te houden, omdat de focus voornamelijk ligt op het bouwen van de applicatie. Het onderzoek is afgebakend tot het geautomatiseerd testen van de front-end.

Aantekeningen naar aanleiding van de vergadering	• Onderzoek naar geautomatiseerde testen • Geen klanten geïnterviewd omdat we vooruitlopen op de denkwijze. Pixelzebra gelooft in zijn eigen visie. Ze toetsen deze visie bij klanten meerdere keren per jaar in workshops en sessies. Het vertalen van de wensen van klanten naar een product doet Pixelzebra zelf, om te voorkomen dat er vanuit huidige beperkingen van bestaande klanten wordt nagedacht. • Methodieken uitleggen en onderbouwen. • Hoe waarborgen we de requirements. • Tijdige reflectie, eventueel schakelen waar nodig • Veel in de ik vorm schrijven • Concreet en SMART bouwfases formuleren • Afgesproken met Peter dat er een Afstudeerverslag template ter beschikking komt.

Het verslag en het PvA moet in de ik-vorm worden geschreven, zodat duidelijk is welke onderdelen ik heb gemaakt. Dit is weer even omschakelen, aangezien we vorig semester een paper moesten schrijven en hier de ik- en wij-vorm moesten ontwijken. Hiernaast heb ik de planning met meer diepgang omschreven en is er op verschillende plekken duidelijker en uitgebreider geformuleerd.

In week 4 begon ik gevoelsmatig achter te lopen. Een aantal onderdelen had ik nog niet volledig afgerond en in week 6 moest ik, volgens het PvA, al beginnen met bouwen. Dit betekent dat ik de requirements en architectuur goed helder moest hebben. Om druk achter de ketel te zetten ben ik het anders gaan aanpakken. Ik ben de werkzaamheden gaan verdelen over de volledige week. De werkzaamheden van het weekend zijn verdeeld over de avonden van de werkdagen. Zo kon ik elke avond anderhalf uur aan het afstuderen zitten om constant progressie te maken en had ik in het weekend meer speling voor achterlopende onderdelen. De volgende weekplanning was opgesteld:

Planning (vooraf)

Maandag	2 uur	5 User Stories
Dinsdag	2 uur	Afstudeerdossier uitleg op school
Woensdag	2 uur	Afstudeerdossier opzetten en logboek week 1 t/m 3 erin verwerken (steekwoorden)
Donderdag	8 uur	Architectuur opzetten en uitwerken met behulp van Draw.io
Vrijdag	8 uur	Requirements afronden en Architectuur uitwerken
Zaterdag	6 uur	Azure omgeving en CRM-omgeving opzetten
Zondag	6 uur	Azure API-call doen naar CRM en gegevens tonen in Vue app. Architectuur verder uitwerken.

Vanaf dit moment ben ik elke zondag een planning gaan maken voor de opvolgende week. Dit heeft mij rust en overzicht gegeven, omdat ik helder had welke stappen er gezet moesten worden. Hiernaast werkte het opbreken van de onderdelen heel motiverend, omdat doelen kleiner werden en makkelijker te behalen of te verschuiven waren. Zo zijn er momenten geweest dat ik de afstudeerdagen heb verschoven naar de maandag en dinsdag, in plaats van de donderdag en vrijdag. Dit was nodig, omdat ik werkzaamheden van werk wilde synchroniseren met de werkzaamheden van mijn collega's. Hierdoor kon ik gemakkelijk de taken van die dagen omruilen en was niet mijn hele planning overhoop.

5 Requirementsanalyse

5.1 Vergaren requirements

Na de vergadering van week 2 over het PvA stond de opdracht helder op mijn netvlies. Op dezelfde dag had ik een vergadering ingepland voor het ophalen van de requirements. Hiervoor had ik Robert van der Kleij (begeleider), Paddy van de Kaay (senior developer), Ronald Hubert (projectmanager) en Jim van Zijl (consultant) uitgenodigd. Robert was mijn begeleider en ook opdrachtgever. Het was voor mij van groot belang dat Robert altijd op de hoogte is over de voortgang. Paddy is de developer die het langst werkzaam is bij Pixelzebra en de development-architectuur heeft opgezet. Ronald is veel betrokken bij het hoofdproduct van de applicatie die ik ga maken. Helaas was het voor Ronald niet mogelijk om aanwezig te zijn. Jim een consultant bij Pixelzebra op het gebied van CRM. Net als Robert, is Jim veel betrokken bij het hoofdproduct waar deze applicatie op aan gaat sluiten.

Ik heb geprobeerd de vergadering zoveel mogelijk voor te bereiden door mij aan tips te houden van verschillende websites (Schriel, 2009) (Bos, 2016). Zo heb ik bijvoorbeeld rekening gehouden met het tijdig inplannen van de vergadering. Om gestructureerd de vergadering in te gaan, heb ik opgeschreven welke onderdelen ik wil bespreken en helder wil hebben aan het einde van de vergadering. Aan het begin van de vergadering heb ik de planning en doelen voorgelegd bij de deelnemers. Ik heb ervoor gekozen om zelf te notuleren. Deze keuze heb ik gemaakt tijdens de vergadering, omdat dit de beste keuze was voor het moment. Er ontstonden discussies die mij hielpen bij het beantwoorden van mijn vraagstukken.

Voorbereiding vergadering requirements
 	
Deel 1
	• Welke onderdelen/modules willen we in de DMJ
	• Voorbeelden:
	  ○ Als een donateur heeft gedoneerd voor het WNF met een interesse voor bomen, dan emailcampagnes versturen naar de donateur die betrekking hebben tot acties van het WNF of bomen?
	  ○ Er komt een donatie binnen met persoonsgegevens en er is duidelijk waarvoor de donateur heeft gedoneerd. (Waar komt dit binnen?) Deze informatie wordt dan gebruikt om de juiste DMJ te kiezen voor de donateur? Wat is de trigger, of vanuit welke solution?
	• Andere voorbeelden?
	• Ideeën m.b.t. de architectuur
	• Wat wordt de scope/MVP
	• Wat moet het product minimaal kunnen bij de oplevering?

Tijdens de vergadering werd duidelijk uit welke onderdelen de applicatie moet bestaan en welke onderdelen met elkaar moeten communiceren. In het requirementsanalyse rapport heb ik genoteerd waar het MVP (Minimum Viable Product) aan moet voldoen en aan welke standaarden ik mij houd. Ook heb ik User Stories en Use Cases gedefinieerd.

Bij onduidelijkheden kon ik terecht bij Paddy van der Kaay. Paddy is tijdens de requirementsanalyse vergadering toegewezen als Product Owner. Hij was voor dit project mijn aanspreekpunt en kon eventuele knopen doorhakken. De vergadering verliep naar mijn mening goed. Ik had mij voorbereid met de juiste vragen om helder te krijgen waar ik de aankomende tijd naartoe moest werken. Hiernaast durfde ik de leiding te nemen door het gesprek te openen en bij te sturen waar nodig. De discussies over de werking zorgde ervoor dat veel ideeën op tafel kwamen. Hierdoor kon ik op mijn beurt weer concrete vragen stellen bij onduidelijkheden. Nadat ik een opzet had gemaakt van het requirementsanalyse document heb ik dit voorgelegd bij Paddy ter goedkeuring.

5.2 Waarborgen requirements

Om ervoor te zorgen dat ik voldoende vooruitgang boekte en de gemaakte onderdelen voldeden aan de verwachtingen van de PO, had ik elke twee weken een vergadering ingepland. Hierin werd besproken wat ik de afgelopen tijd had behandeld, wat de resultaten hiervan waren en welke onderdelen ik moest verbeteren. Zo had ik bijvoorbeeld een vergadering ingepland op 11 maart. Hierin had ik de requirementsanalyse en de architectuur op tafel gelegd. Hiernaast wilde ik bespreken of de aanpak die ik ging hanteren, met betrekking tot het developen, de juiste is. “Klopt alles wat ik nu heb genoteerd en is dit volgens de ideeën van de PO en begeleider?”, om antwoord te kunnen geven op deze vraag ben ik de vergadering gaan voorbereiden.

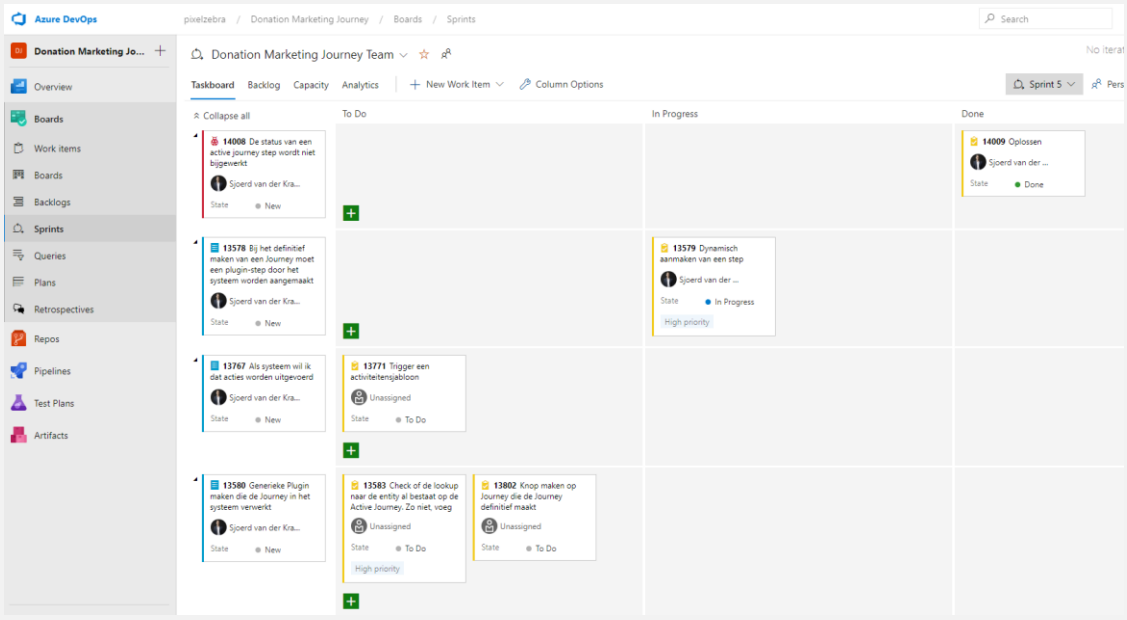
Voorbereiding vergadering requirements **Deel 2**	• Voldoet de opzet die ik heb met betrekking tot de architectuur en de User Stories aan de ideeën van de opdrachtgever? • Voldoet het geheel aan de richtlijnen van Pixelzebra. Denk hierbij aan de architectuur en development technieken. • Klopt de architectuur die ik heb opgezet? • Zijn de User Stories volledig en kunnen we hiermee de backlog opzetten? • Zijn er eventueel nog aanpassingen of uitbreidingen nodig op bepaalde vlakken? • Is de opdracht haalbaar zoals ik het nu heb genoteerd? • Is het mogelijk om hulp van Jim van Zijl te krijgen bij het opzetten van het project in JavaScript?

Tijdens de vergadering kwamen we erachter dat er over bepaalde zaken nog niet goed was nagedacht. Zo was ik bijvoorbeeld begonnen met het uitwerken van de Use Cases in plaats van de User Stories. Hierdoor had ik bepaalde requirements nog niet goed helder, zoals logging, security en het bijhouden van de status van een Journey. De resultaten van deze vergadering heb ik verwerkt in de hoofdstukken die daarop van toepassing zijn. Zo kon ik bijvoorbeeld een verbeterslag doen op de architectuur: “Hoe ga je loggen als er iets misgaat?” en “Hoe ga je om met securityonderdelen?”. Het was voor mij vrij duidelijk dat deze vergaderingen van groot belang zijn. Niet alleen bracht ik de PO en begeleider op de hoogte van mijn voortgang, maar kreeg ik ook de feedback waar ik op hoopte om de kwaliteit van mijn afstuderen te verbeteren. Als alle requirements nog niet zijn uitgeschreven en goedgekeurd, ben je aan het prototypen. Dit is uiteraard niet de bedoeling en heb daarom druk achter de ketel gezet om de requirementsanalyse af te ronden.

Ik probeerde de requirementsanalyse gestructureerd aan te pakken door een stap terug te doen. Eerst moesten de requirements goed helder zijn, voordat ik aan de slag kon met het opzetten van de User Stories. Aan de hand van de bevindingen van de vergadering, ben ik de requirements gaan formuleren. Hierbij heb ik geprobeerd de requirements SMART te houden, door mij constant af te vragen: "Is dit concreet genoeg omschreven?", "Wat bereiken we hiermee?", "Hoeveel tijd gaat dit ongeveer kosten?" en "Is dit een realistisch doel?". Na het opstellen van de requirements was helder aan welke eisen de applicatie moest voldoen. Hierna kon ik de Epics opstellen om een beter beeld te creëren van de werking van de applicatie. Zo heb ik als eerste Epic de kern van de applicatie geformuleerd: "Als gebruiker wil ik een Journey maken, zodat ik de donateur gepersonaliseerd kan benaderen". De Epic kon worden opgedeeld in verschillende User Stories. Namelijk, het definiëren van de Trigger dat bestaat uit drie Use Cases, het selecteren van een voorwaarde dat bestaat uit twee Use Cases en het bepalen van de uit te voeren actie als laatste Use Case. Zodoende kon ik voor de overige User Stories de Use Cases uitwerken en refereren naar bijbehorende functionele requirements. Het requirementsanalyse document heb ik nogmaals laten beoordelen door de Product Owner. Toen dit was goedgekeurd kon de backlog worden opgesteld.

De backlog

Voor het opstellen van de backlog heb ik gebruik gemaakt van Microsoft Azure DevOps. Het is mogelijk om verschillende onderdelen van een applicatie in één project te hebben. In Repos staat de code van applicatie en in Boards (afbeelding 2) staan de backlog items en sprints. Hierdoor blijft alles van een project op één plek en behoud je overzicht. Hiernaast kan bijvoorbeeld bij een "code commit" de taak aan de commit worden gekoppeld. Dit is een zeer praktische functie en geeft een duidelijk inzicht van wat gecommit is en wanneer. Ik heb de User Stories verdeeld over de 4 sprints en ben deze structureel gaan afwerken. Elke week tijdens de weekly standup heb ik met Robert van der Kleij en Paddy van der Kaay de backlog doorgenomen. Hierbij keken we naar de taken die waren afgerond, of niet afgerond konden worden. Als bleek dat een taak niet kon worden afgerond, heb ik onderbouwd waarom dit niet mogelijk was en waar ik tegenaan ben gelopen. De planning werd vervolgens aangepast voor de opvolgende week.



Afbeelding 2, Azure DevOps: Donation Marketing Journey project

In het afgelopen jaar heb ik verschillende evaluatiegesprekken gehad met mijn werkgever. Hieruit bleek dat mijn werktempo verhoogd kan worden als ik niet afwijk van de taak die ik op dat moment behandel. Om dit te stimuleren ben ik bij afwijkingen direct taken gaan aanmaken en gaan oppakken volgens de planning. In de code refereer ik naar de taak als een "TODO" (afbeelding 3), zodat deze gemakkelijk kan worden opgepakt.

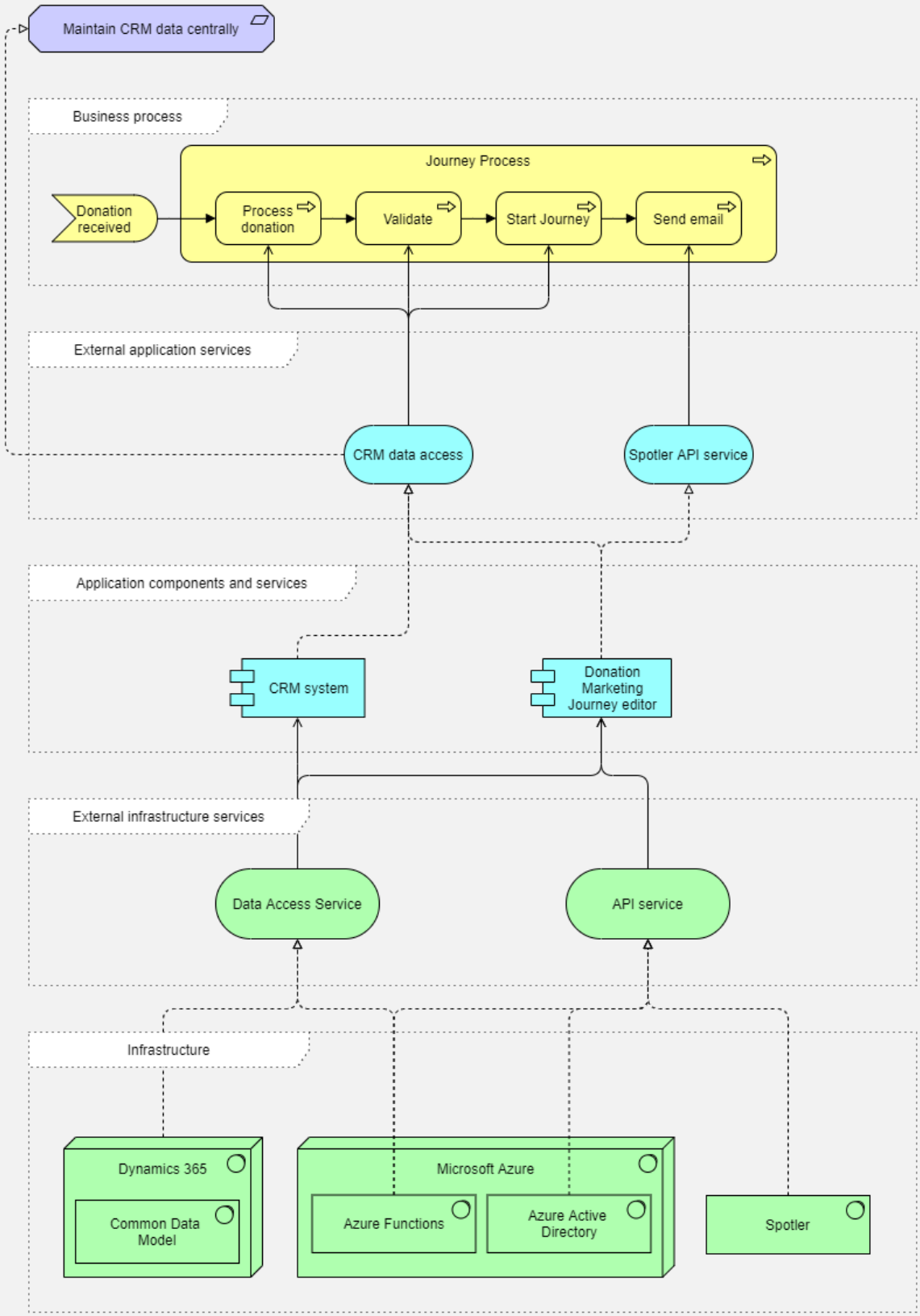
```
psmdmj_activejourney activeJourney = new psmdmj_activejourney
{
    psmdmj_JourneyId = journey.ToEntityReference(),
    [$"psmdmj_{triggerEntity.LogicalName}id"] = triggerEntity.ToEntityReference(),
    psmdmj_name = journey.psmdmj_name,
    psmdmj_emailaddress = (string)triggerEntity["emailaddress1"], // TODO: #13566
    psmdmj_JourneyJson = journey.psmdmj_JourneyJson
};
```

Afbeelding 3, Referentie naar taak in code

6 Componenten Architectuur

Om de applicatie te realiseren maak ik gebruik van verschillende bestaande softwarepakketten. In het componenten architectuurdiagram (afbeelding 4) is te zien dat er gebruik wordt gemaakt van Microsoft Dynamics 365, Spotler en functionaliteiten van Azure. In hoofdstuk 3.2 heb ik omschreven van welke softwareonderdelen en componenten ik gebruikt heb en waarom.

Mijn product, de Donation Marketing Journey editor, beschikt over een front-end interface. De interface communiceert met Azure Functions om berekeningen uit te voeren en data door te sturen. Vanuit de Azure Functions wordt ook Spotler aangeroepen om mailcampagnes mee af te vuren. Via het Dynamics 365 Common Data Model wordt data opgeslagen en kan dit worden gepresenteerd in CRM. Om de werking en communicatie tussen componenten duidelijk in beeld te brengen heb ik een architectuurdiagram gemaakt (afbeelding 4). Hierbij heb ik gebruik gemaakt van de “ArchiMate 3.0.1.” notatiewijze. Ik heb uiteindelijk gekozen om het Design Pattern aan te houden van de officiële website “visual-paradigm.com” (visual-paradigm, 2020). De reden voor deze keuze is, omdat ArchiMate deels is gebaseerd op de IEEE 1471 standaard. Dit betekent dat de modeleringtaal voldoet de officiële software architectuur richtlijnen. Hiernaast bestaat de taal al meer dan 15 jaar en zijn er tot en met 2017 nieuwe versies uitgebracht.



Afbeelding 4: Componenten architectuur

7 Development Donation Marketing Journey editor

7.1 Inleiding

In dit hoofdstuk wordt uitgelegd welke stappen ik heb genomen en welke keuzes ik heb gemaakt bij het developen van de Donation Marketing Journey editor. Het development proces is opgedeeld in vier sprints en elke sprint bestaat uit een timebox van twee weken.

7.2 Voorbereidingen

Om even afstand te nemen van het documenteren ben ik in week 3 een Vue project gaan opzetten. Ik wilde alvast wat voorbereidingen treffen voor week 5 en 6, de weken waarin ik een aantal functionaliteiten wil realiseren. Hiernaast had ik nog niet voor de volle honderd procent helder hoe ik de architectuur moest vormgeven. Ik wilde het opzetten van dit project gebruiken om de vraagstukken mee te beantwoorden. Door slim van start te gaan heb ik om tips gevraagd bij een professional op het gebied van Vue. Hij gooide een aantal termen op die ik moest gaan uitzoeken en eventueel toepassen waar nodig. Dit waren: Yarn, Vue Command Line Interface (CLI) en Axios. Initieel had ik meer informatie willen noteren van de onderwerpen, maar in verband met de planning (PvA), en het feit dat ik van Vue.js ben afgestapt, heb ik ervoor gekozen om dit gedeelte beknopt te houden. In hoofdstuk 7.4 leg ik uit waarom ik ben afgestapt van Vue.js.

7.2.1 Yarn

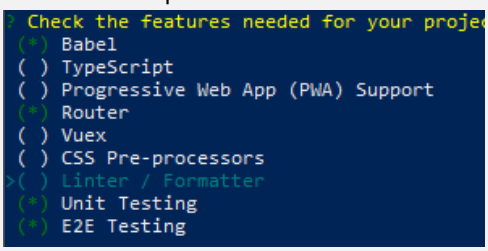
Yarn is een package manager. Front-end programmeurs zijn veelal bekend met de term: "NPM", een software register met meer dan 800.000 code packages. Yarn heeft hierop voortgebouwd en beweert sneller te functioneren en veiliger te zijn dan NPM (McKenzie, 2018).

7.2.2 Axios

Axios maakt het mogelijk om "XMLHttpRequests" uit te voeren vanuit de browser en HTTP-requests vanuit "Node.js". Achteraf is gebleken dat we hier weinig gebruik van maken, omdat de Vue applicatie is gedeployed in de webresources van CRM. Dit betekent dat we geen HTTP-requests hoeven te doen, maar direct een WebApi van CRM kunnen aanroepen.

7.2.3 Vue CLI

"Vue.js CLI" is een tooling library voor Vue.js projecten. De library kan als globaal pakket worden geïnstalleerd, zodat het overal via de terminal te benaderen is. Hierdoor is het mogelijk om snel een nieuw project te maken via "vue create". "Vue serve" start een clientside webpack server met hotmodule replacement, waardoor een wijziging direct worden toegepast bij een wijziging. Projecten kunnen beheerd worden met behulp van een grafische interface via Vue UI (afbeelding 5). Bij het aanmaken een project via de interface, kan gekozen worden uit componenten. De componenten heb ik op de volgende pagina in het kort omschreven.



Afbeelding 5, Vue.js CLI componenten

Babel	Babel is een javascript transpiler. Dit wordt gebruikt om ECMAScript (ES) te converteren naar vanilla JavaScript. Op deze manier kunnen alle browsers de JavaScript uitvoeren. Hierdoor is het bijvoorbeeld mogelijk om een asynchrone call naar Dynamics wel synchroon te maken.
TypeScript	TypeScript is een superset bovenop JavaScript. Hierdoor is het mogelijk om bijvoorbeeld datatypes te gebruiken. Dit is niet mogelijk is vanilla JavaScript.
Prograssive Web App (PWA)	De Progressive Web App (PWA) is een van de nieuwste ontwikkelingen op het gebied van apps. Een PWA verbindt de eigenschappen van een website met veel kenmerken van native mobile apps. Daardoor functioneert de applicatie onafhankelijk van het besturingssysteem en tegelijkertijd als website en app.
Router	De Router feature zorgt voor clientside routing. Op deze manier worden componenten ingeladen in plaats van een harde pageload.
Vuex	Vuex fungeert als statemanager, waardoor je data globaal beschikbaar kan maken. Normaliter wordt data gewijzigd door properties aan te passen van elementen. Vuex maakt hier een schil omheen zodat je via de globale variabel de data kan aanpassen.
CSS Pre-processors	Met de CCS Pre-processor feature is het mogelijk om gebruik te maken van bijvoorbeeld SCSS/SASS of LESS. Dit zijn componenten die het toepassen van CSS gebruiksvriendelijker maken. In het verleden heb ik met SASS gewerkt en hierdoor was het mogelijk om class-based CSS te schrijven en variabelen te definiëren.
Linter/Formatter	Linter maakt het mogelijk om via een configuratiebestand aan te geven volgens welke codestijl je te werk moet gaan. Denk hierbij aan het correct afsluiten van functies, het verwijderen van “console.log()” code en dergelijke regels.
Unit Testing	Met behulp van Unit Testing kunnen we geïsoleerde onderdelen testen. Meestal zijn dit functies of losse components.
End-to-End (E2E) testing	En als voortbouwing op Unit Testing kunnen we met behulp van E2E testing, verschillende scenario’s testen vanuit de optiek van een gebruiker. Denk hierbij bijvoorbeeld aan het testen van de inlogprocedure.

Nadat ik had uitgezocht welke onderdelen meerwaarde konden bieden, ben ik het project gaan opzetten. Dit was een proces van trial-and-error. Er kwamen verschillende foutmeldingen naar voren, waarvan voornamelijk versieconflicten. Gelukkig kom je met foutmeldingen Googelen een heel eind tegenwoordig. Toen het project succesvol was gebouwd, en ik een testpagina kon builden, ben ik een API-call gaan maken en testen met mockup data.

7.3 Sprint 1: De applicatie deployen als webresource in CRM

7.3.1 Doel

Het doel van de eerste sprint was om te beginnen met het bouwen van de front-end modules. De modules zijn onderdelen die de gebruiker kan configureren om de Donatie Marketing Journey vorm te geven.

7.3.2 Uitvoering

In week 4 was het gelukt om een API-call te doen met mock-data. Daardoor kon ik in week 5 een endpoint bouwen en de mock-data te vervangen voor echte data. Ik ben de endpoint gaan opzetten met behulp van een Azure Function op .Net framework versie 4.7.1, dit is de meest recente versie voor CRM. Het endpoint heb ik getest met Postman, een softwarepakket waarmee je gemakkelijk API-calls kan sturen naar endpoints. Dit verliep voorspoedig totdat ik de API-call ging doen vanuit de website. Hier kreeg ik te maken met CORS-policy foutmeldingen. Door mijn gebrek aan front-end kennis heb ik flink moeten worstelen om erachter te komen wat het probleem veroorzaakt en hoe ik dit op kan lossen. Ik heb best-practices opgezocht (Hobbs, 2019) en gekeken naar oplossingen aan de front-end kant. Tevergeefs, zonder succes. Hierna ben ik gaan kijken of het probleem wordt veroorzaakt door de backend. Hierover waren ook een aantal artikelen te vinden, zoals het omgaan met CORS op Azure (Gowtham, 2019).

Na een aantal artikelen te hebben gelezen, en ik weinig vooruitgang had geboekt, ben ik in gesprek gegaan met Paddy van der Kaay (Product Owner). Hij gaf aan dat het niet nodig is om het project op deze manier op te zetten, omdat de front-end gedeployed wordt in CRM als Webresource. Door mijn gebrek aan kennis van CRM, was dit concept bij mij nog niet bekend. Ik was in de veronderstelling dat ik een aparte website moest maken als prototype. Dit was blijkbaar niet de bedoeling. De bedoeling was om de Vue website te projecteren in CRM. Dit is te realiseren door de website te builden en te deployen met behulp van Webresources. Dit veranderde voor mij het één en ander. Niet alleen moest ik de architectuur anders opdelen, maar ook kennis gaan opdoen over deze manier van implementatie. Kortom, tijd voor een moment om de werkzaamheden te herzien.

Gelukkig ben ik niet de eerste persoon die een Vue project wil deployen in CRM. Door te zoeken op Google naar “vuejs webresources” heb ik een tutorial (MK, 2018) gevonden die mij handvatten gaf om het project draaiend te krijgen via Webresources in CRM. Met behulp van deze tutorial is het gelukt om de Vue applicatie te projecteren in CRM met mock-data. Helaas liep ik al snel tegen het probleem aan dat ik niet bij de data van CRM komen. De library die ik moest importeren om het werkend te krijgen, kreeg ik na veel pogingen niet ingeladen. Dit is het moment waarop ik erachter kwam dat mijn kennis, op het gebied van front-end frameworks, te laag is om Vue.js te gebruiken. De tijd begon aardig te dringen. Ik moest een beslissing maken tussen, veel tijd steken in het leren van Vue.js of het project opzetten in “vanilla JavaScript”.

7.3.3 Retrospective Sprint 1

Het grootste obstakel van deze sprint, was het feit dat ik de website moest deployen als Webresource. Naast het feit dat ik nog niet bekend was met Webresources, moest ik ook de architectuur en mijn denklogica gaan aanpassen. Door niet op tijd de architectuur te laten toetsen, moest ik nu op de blaren zitten. Ik was te veel gefocust op de planning. Hierdoor had ik bepaalde onderdelen, zoals de architectuur, niet voldoende aandacht gegeven. Voor Paddy en Robert was het een gegeven om de architectuur zo op te zetten, omdat zij dit concept kennen en het vaker hebben toegepast. Ik daarentegen weet een heel stuk minder van CRM en kende deze functionaliteit niet. Hierdoor had ik een bepaalde architectuur in mijn hoofd en heb ik te weinig vragen gesteld over de opzet van de architectuur. In de toekomst zal ik hier meer op moeten letten. Ik moet minder uitgaan van de kennis die ik heb en blijven doorvragen in plaats van aannames maken. Het gebruiken van Webresources bracht ook voordelen met zich mee. Zo hoefde ik bijvoorbeeld geen login te maken met behulp van de Azure Active Directory, want de gebruiker is al ingelogd en ziet alleen de pagina die hij mag zien. Dit verhielp ook het probleem met de CORS, omdat we gebruik maken van de context van Dynamics 365 CRM.

7.4 Sprint 2: Dynamisch opbouwen van de webpagina

7.4.1 Doel

Het doel van de tweede sprint was om het bouwen van de modules af te ronden. De Trigger module zou gerealiseerd moeten zijn in sprint 1. Dit was door omstandigheden helaas niet gelukt en is om die reden meegenomen naar sprint 2. Hiernaast moesten de Voorwaarde module en de Actie module gebouwd worden.

7.4.2 Uitvoering

Na een korte overweging ben ik het project opnieuw gaan opzetten in vanilla JavaScript. Hierdoor begon ik een week achter te lopen, maar kreeg ik meer controle over de situatie en kon ik werken met de kennis die ik had. Doordat ik alles weer overzichtelijk had, is het gelukt om data op te halen van Dynamics via de "Xrm.WebApi" (hoofdstuk 3.2.4) client. Dit was voor mij een hele opluchting, omdat dit betekende dat ik de basis klaar had en aan de slag kon gaan. Het proces van de Api call gaat als volgt. Met behulp van JavaScript maak ik een request dat een "Action" aanroept van Dynamics. Via de Dynamics Action entiteit kunnen er vervolgstappen worden geconfigureerd, zoals het aanroepen van een Custom Workflow Activity (CWA) (hoofdstuk 3.2.4). Door de request uit te voeren via de Xrm.WebApi hoeven we ons geen zorgen te maken over security issues. De Api is alleen beschikbaar op de pagina waarin de Webresource is gedeployed. Hiernaast moet er via de Azure Active Directory zijn ingelogd, zodat Dynamics weet dat de ingelogde persoon de pagina mag bezoeken. De Dynamics Action is geconfigureerd om variabelen te ontvangen en door te sturen naar een CWA. Door de meegestuurde variabelen, weet de generieke CWA om welke entiteit het gaat en welke Azure Function aangeroepen moet worden met behulp van een route. Voor overige inputdata is er een Json variabel beschikbaar. Als de Azure Function een "Get request" ontvangt, wordt de data geconverteerd naar een Json en gaat het proces via dezelfde weg terug naar de front-end. Hier komt de data terecht in de succes functie van de JavaScript methode en kan dit uiteindelijk worden weergegeven op de pagina.

(A)synchroon Tijdens het uitvoeren van de Api call liep ik tegen het probleem aan dat ik geen synchroon calls kon doen met de Xrm.WebApi. Dit betekent dat de code in JavaScript niet wacht op het resultaat van de Api call en gewoon doorgaat, met ongedefinieerde data als gevolg. Dit probleem ben ik gaan uitzoeken en al snel bleek dat het niet mogelijk is om de Xrm.WebApi call asynchroon uit te voeren. Er zijn een aantal "workarounds", maar geen oplossingen waar je als developer blij van wordt. Zo is het bijvoorbeeld mogelijk om functies te "chainen" (Wang, 2018). Als een Api call wordt uitgevoerd, wachten we het resultaat af en gaan we in de result functie pas de volgende stap uitvoeren. Dit kan handig zijn als je een kleine applicatie hebt die maar een aantal stappen uitvoert. In mijn geval wil ik data verzamelen van verschillende bronnen en hier bewerkingen op uitvoeren. Hierdoor is het chainen van functies niet gewenst en zal het de code ook zeer onleesbaar maken.

Mijn developer collega's Paddy en Mitchell ondervinden dit probleem ook regelmatig. Het leek mij dus een goed idee om hier een passende oplossing voor te verzinnen. Aangezien Google alleen een workaround kon bieden, ben ik in gesprek gegaan met Danny Nieuwmans (school collega) om te achterhalen hoe ik dit probleem kan oplossen. Na een verhelderend gesprek kwam ik erachter dat het Babel component (hoofdstuk 7.2.3) gebruikt kan worden voor dit probleem. Babel zorgt ervoor dat ik gebruik kan maken van ECMAScript, een nieuwe versie van JavaScript. In ECMAScript zit, met behulp van de asynchroon plugin (Babel, 2020), een asynchroon methode waarop ik een "await" kan uitvoeren. De "await" wacht op het resultaat van de asynchroon call en zal daarna pas verder gaan met het uitvoeren van de JavaScript. Hierdoor hoef ik niet meer te chainen (nesten), wordt de code overzichtelijker en krijgen we geen "spaghetti-code". Nog een groot voordeel is het kunnen definiëren van classes (Babel, 2020) met behulp van ECMAScript. Dit is vergelijkbaar met hoe wij bij Pixelzebra normaal gesproken programmeren in C#. Dit zal mij, en de developers van Pixelzebra, helpen om de code begrijpelijk en schoon te houden.

Helaas bracht het implementeren van Babel wel een probleem met zich mee. Als het project gebuild wordt door Babel, worden er verschillende bestanden gegenereerd. Dit is in verband met onderhoudbaarheid niet wenselijk. Als er een code update is, moeten de Dynamics Webresources stuk voor stuk worden aangepast. Dit is tijdrovend en zeer foutgevoelig. Om dit probleem te tackelen heb ik gekozen voor Webpack. In het artikel m.b.t. vragen over Webpack (Levi, 2018) wordt duidelijk gemaakt wat de voordelen zijn ten opzichte van andere build/bundle-tools, zoals Gulp of Grunt. De doorslaggevende factor was voor mij de gebruiksvriendelijkheid van de configuratie, de duidelijke documentatie en hoe gemakkelijk er plugins en extra functionaliteiten toegevoegd kunnen worden. Nadat Webpack met succes was geïmplementeerd, en er maar één JavaScript bestand wordt gegenereerd na een build, kon ik beginnen aan het opzetten van de componenten. Aangezien ik een week achterliep heb ik mijn werkrooster aangepast. Om de achterstand te beperken ben ik elke dag gaan werken van 9:00 tot 21:00 en in het weekend van 10:00 tot 18:00.

Component Builder

Veel velden die geconfigureerd kunnen worden in de Journey editor zijn afhankelijk van elkaar. Dit betekent dat ik twee opties had. Het was mogelijk om de volledige HTML-pagina van tevoren te bouwen en om daarna alle waardes in de velden te zetten. Dit maakt het proces erg ingewikkeld, omdat je heel goed na moet denken over wanneer iets zichtbaar mag zijn en gevuld moet worden. De tweede manier was om componenten te bouwen die dynamisch HTML genereren met de juiste waardes en events. Het was voor mij duidelijk dat de tweede optie meer werk zal kosten in het begin, maar uiteindelijk de beste oplossing is voor het genereren van de pagina. Hiernaast vind ik het belangrijk om dit proces goed te snappen. De reden dat ik niet met Vue.js ben doorgegaan, was door mijn gebrek aan front-end kennis. Met het maken van een component-builder hoopte ik mijn kennis te verbreden, zodat ik in de toekomst wel een front-end framework kan oppakken.

Het voordeel van werken in objecten is, dat het makkelijker wordt om de pagina om te zetten naar een Json. Dit is nodig voor het opslaan en wijzigen van een Journey. Als de persoon klaar is met het configureren van een Journey, wordt deze opgeslagen op de Journey entiteit. De Journey is dan een statische versie die gewijzigd of definitief gemaakt kan worden. Voor het opslaan van de waardes naar een Json maak ik gebruik van een Store. Dit is een onderdeel van het zogeheten Flux Pattern (Flux, 2019).

> *"Stores contain the application state and logic. Their role is somewhat similar to a model in a traditional MVC, but they manage the state of many objects — they do not represent a single record of data like ORM models do. Nor are they the same as Backbone's collections. More than simply managing a collection of ORM-style objects, stores manage the application state for a particular domain within the application."*

Dit betekent dat een gekozen waarde, ofwel de "state" van een DOM-element (Document Object Model), wordt opgeslagen in een apart JavaScript bestand. Als de waarde verandert van het DOM-element dan wordt de waarde aangepast van het object in de Store (afbeelding 6). Op deze manier hebben we een verzameling (object) van alle gekozen waardes en kunnen we het element deserialiseren naar een Json tekst.

```
addCondition(key) {
    const newCondition = {
        index: null,
        leftValue: null,
        comparisonOperator: null,
        rightValueType: null,
        rightValue: null,
        rightValueInput: null,
    }

    this.conditions[key] = {...newCondition};

    return this.conditions[key];
}
```

Afbeelding 6: Store object

7.4.3 Retrospective Sprint 2

In sprint 2 heb ik er veel aan gedaan om de "verloren" week uit sprint 1 in te halen. Ik ben lange dagen gaan draaien om zoveel mogelijk taken te realiseren die ik nog open had staan. Hierdoor kwam ik meer op schema en kon ik de planning uit het Plan van Aanpak aanhouden. Het voelde goed om de front-end werkend te krijgen, maar ik merk toch een grote kloof tussen het developen van front-end en backend. Front-end, zoals ik het heb opgezet, voelt losjes en geeft weinig feedback tijdens het developen. Pas nadat alles is gebuild en wordt uitgevoerd (runtime), krijg je pas te horen wat er fout gaat. Tijdens het developen in de backend met C#, worden veel foutmeldingen al aangegeven tijdens het typen. Dit komt omdat C# een "strict" taal is. JavaScript daarentegen is niet strict en laat veel toe, totdat er runtime tegen een probleem wordt aangelopen. In de toekomst zal ik ervoor kiezen om gebruik te maken van TypeScript. Dit is een laag bovenop JavaScript om de taal strict te maken en

eventuele runtime fouten preventief aan te geven. Hiernaast voegt het ook een object georiënteerd aspect toe, zoals het werken in classes volgens ECMAScript 2015 (Bright, 2012).

Ik merkte op dat ik nu op de helft zou moeten zitten van het developen, maar was nog steeds volop met de font-end bezig. De componenten werkte al aardig, maar het was nog niet mogelijk om de Journey op te slaan of in te laden. Dit had ik in sprint 2 af willen hebben, maar is door de omstandigheden niet gelukt. Hierdoor heb ik (figuurlijk) met Paddy aan tafel gezeten, om te bespreken hoe we het beste verder kunnen. We gingen akkoord met het feit dat de MVP bijgewerkt moest worden. Het belangrijkste is dat er een Journey geconfigureerd kan worden en dat er een Spotler campagne aangeroepen moet worden op het moment dat de Donateur aan de voorwaarde(n) voldoet. Hierdoor was mijn doel voor de aankomende twee sprints weer duidelijk en ben ik dit gaan verwerken in de backlog.

7.5 Sprint 3: De Journey opslaan of bewerken

7.5.1 Doel

Het doel van de derde sprint was om de applicatie zoveel mogelijk af te ronden, zodat deze beoordeeld kon worden door Paddy van der Kaay en Robert van der Kleij.

7.5.2 Uitvoering

Sprint 3 begon met wat problemen. De Donation Marketing Journey (Dynamics 365 CRM) omgeving is een trial en deze was verlopen. Normaal gesproken krijg je hier een waarschuwing van, maar dat is deze keer niet gebeurd en zodoende is de omgeving op non-actief gezet. Hierdoor heb ik tijd moeten besteden aan het heractiveren van de omgeving, terwijl dit niet in de planning zat. Allereerst ging ik uitzoeken welke persoon ik hiervoor moest benaderen, want de omgeving staat in de tenant van Pixelzebra. Na een aantal collega's te hebben gesproken, kwam naar voren dat alleen Microsoft ons hierbij kon helpen. Mijn collega Alexander (CRM-specialist) is zo vrij geweest om zich hierover te ontfermen en heeft het voor elkaar gekregen om een dag later de omgeving te heractiveren. In de tijd dat ik niet heb kunnen developen had ik gelukkig nog genoeg te documenteren.

Toen alles weer up-and-running was, kon ik aan de slag met het exporteren en importeren van de front-end data. Voordat ik het project had opgezet, heb ik nagedacht over het gebruiken van een Json voor de save en load van de Journey. Hierdoor heb ik tijdens het bouwen zoveel mogelijk rekening gehouden met een object gebaseerde opzet. Toch was het niet makkelijk om de geëxporteerde Json weer in te laden in de Journey editor. De volgorde van hoe de pagina wordt opgebouwd was onderhand bijzonder complex geworden en om deze reden liep ik vaak tegen runtime errors aan. Het meest lastige aspect was het inladen van de Json data naar een statische class, genaamd de Store. De Store die ik vanaf het begin had opgezet, functioneerde naar behoren. Totdat ik begon met het bouwen van de import. Als ik de Json in deze class ging verwerken, stond de data op een bepaalde plek in het geheugen waar ik niet bij kon vanuit elke class.

Na dit probleem te hebben besproken met Danny Nieuwmans, heb ik ervoor gekozen om een class instantie te maken van de Store en deze te koppelen aan "window". Het window object is de basis van elke webpagina en beschikt over verschillende eigenschappen (properties) en methodes (methods). Het is ook mogelijk om hier een referentie aan toe te voegen, zoals een referentie naar een class. Om deze reden heb ik de store toegevoegd aan de window, zodat het mogelijk was om "window.$store" aan te roepen vanuit elke class. Door deze aanpassing heb ik flink wat code moeten aanpassen, maar daardoor

was het wel mogelijk om de Json data op de juiste manier in te laden. Het is belangrijk om in het achterhoofd te houden dat andere mensen ook met de code moeten werken. Vanuit dit oogpunt, en voor de onderhoudbaarheid, wilde ik de code zo overzichtelijk mogelijk houden. Dit is naar mijn mening goed gelukt door de statische Store om te zetten naar een Store instantie en deze te koppelen aan het window object.

Her-evalueren requirements

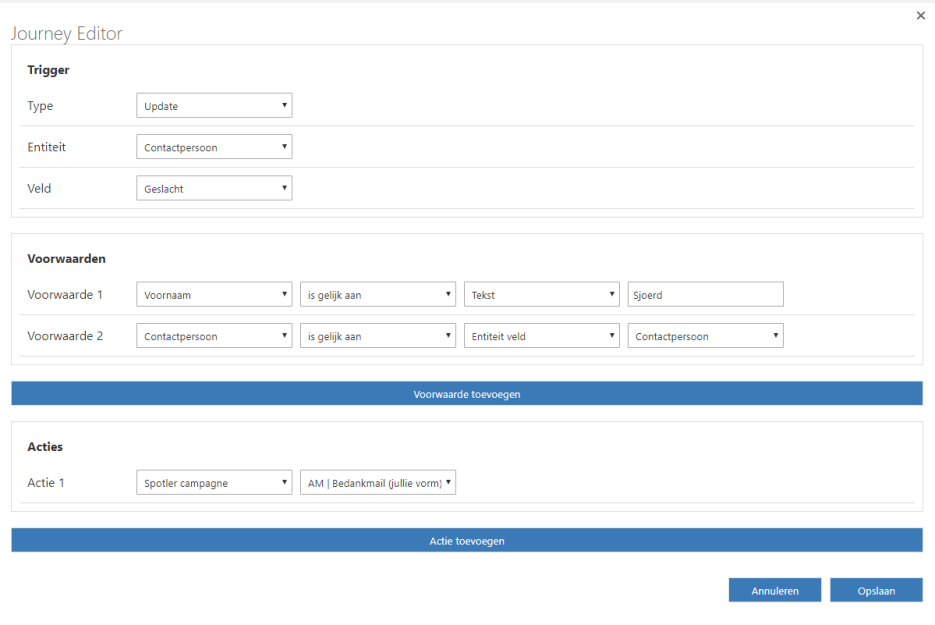
Nu de front-end voldeed aan de requirements kon ik (bijna) beginnen aan de backend. Er waren door bevindingen wat requirements bijgekomen en weggevallen. Om deze reden wilde ik de openstaande taken evalueren en afstemmen met Paddy van der Kaay. Ik ben de backlog gaan bijwerken n.a.v. de wijzigingen en bevindingen en heb duidelijk uitgewerkt welke stappen er ondernomen moeten worden. In de requirements had ik als User Story, dat een gebruiker de progressie van een actieve Journey moet kunnen volgen (bijlage 2, hoofdstuk 3.4). Hiervoor is het van belang dat de actieve Journey "child" entiteiten krijgt. Op deze entiteiten wordt bijgehouden welke actie er uitgevoerd moet worden, wanneer deze uitgevoerd moet worden en wat de status is van de uitvoering. Hierdoor kan de gebruiker in één oogopslag zien welke stappen er gelukt en mislukt zijn en wat de status is.

7.5.3 Retrospective Sprint 3

De sprint begon wat lastig omdat ik tijd verloren had aan het oplossen van een probleem, waar ik niet goed over had nagedacht. Ik was me bewust dat ik te maken had met een trial versie van Dynamics 365 CRM, maar ik heb dit niet goed in de gaten gehouden. Hierdoor had ik ongeveer vier uur verloren en heb ik tijd van collega's moeten vragen. Het is dus belangrijk om dit soort aspecten vroegtijdig te zien aankomen en hier taken voor aan te maken. Als ik dit als taak had toegevoegd op sprint 2, had ik dit eventueel kunnen voorkomen. Ik merkte dat de Coronacrisis invloed had op het benaderen van de juiste personen, omdat de agenda van iedereen was volgepland met meetings. Dit maakte het vrij lastig om snel een oplossing te vinden voor het probleem. Communicatie was een lastig onderdeel geworden door de Coronacrisis. De agenda's van collega's waren volgepland en er was weinig speling voor escalaties. Ik moest snel aan de bel trekken bij onduidelijkheden, anders kon dit mijn voortgang belemmeren. Vragen kwamen vaak op een ongelegen moment en konden niet direct worden opgelost. Ik probeerde hier zoveel mogelijk rekening mee te houden, door direct aan de bel te trekken bij vragen of onduidelijkheden.

Bij elkaar ben ik meer dan een week bezig geweest om de Json te exporteren en importeren. Het was een stuk lastiger dan ik had verwacht, omdat ik tegen veel problemen aan liep waar ik geen ervaring mee had. Hierdoor heb ik veel moeten ombouwen om het werkend te krijgen. Ik denk dat dit voornamelijk kwam door mijn gebrek aan front-end kennis. Normaal kan ik goed overweg met statische classes als het gaat om programmeren met C#. Front-end werkt toch anders en in dit soort situaties merk je de verschillen goed.

Ik kwam er tijdens deze sprint achter dat er vaak kleine taken tussendoor komen, zoals bijvoorbeeld het aanpassen van de stijl. Toen ik Robert van der Kleij de front-end liet zien was hij tevreden en had hij een interessante verbetering m.b.t. het uiterlijk. Robert liet mij een onderdeel zien in Dynamics 365 CRM waar een gelijkwaardige configuratie plaatsvindt. We waren het er samen over eens dat onze applicatie hierop kan lijken met een paar aanpassingen. Hierdoor behouden we continuïteit in de stijl van Dynamics, zodat de gebruiker vertrouwd blijft met de User Interface. Deze werkzaamheden wilde ik er toch nog even "tussen proppen" en heeft gezorgd voor een duidelijkere interface (afbeelding 7). Normaliter wil ik mij zoveel mogelijk aan de planning houden, maar soms hebben onvoorziene taken tussendoor veel impact op het uiteindelijke resultaat.



Afbeelding 7: uiteindelijke ontwerp

7.6 Sprint 4: Laatste hand kernfunctionaliteiten applicatie

7.6.1 Doel

Het doel van sprint 4 was om eventuele feedback van sprint 3 te verwerken. Aangezien er nog onderdelen afgerond moesten worden, had ik nog geen feedback ontvangen van een afgemaakt product. De backend moest grotendeels nog worden gemaakt, dus heb ik hier al de sprint aan toegewijd.

7.6.2 Uitvoering

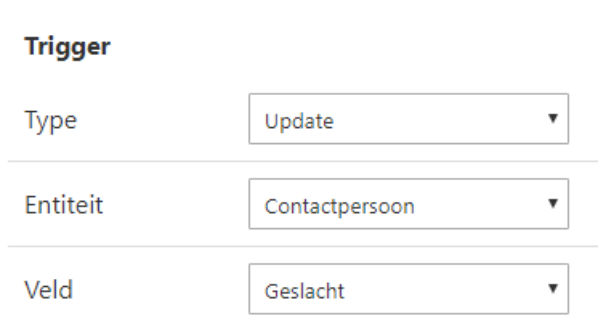
Met heldere requirements, en een goede opzet in CRM, ging ik verder met de backend. Tijdens het maken van de front-end heb ik al aardig wat backend werkzaamheden moeten verrichten, zoals het opzetten van de solution architectuur en het maken van Azure Functions. Om de trigger af te laten gaan moest ik gebruik maken van een Plugin uit de Xrm.Sdk (hoofdstuk 3.2.4). De Journey Trigger is te configureren op de Journey entiteit en kan de Plugin bijvoorbeeld laten afgaan op: "het wijzigen van het emailadres van een contactpersoon". Het nadeel is dat we een Plugin lastiger kunnen debuggen dan een Azure Function. Hiernaast veroorzaakt dit business logica in twee verschillende projecten binnen de solution. Om deze reden heb ik ervoor gekozen om in de plugin een HTTP-request te doen naar een Azure Function. In de content van de "HttpRequestMessage" zet ik de benodigde variabelen. Voor het maken van een Active Journey zijn dit bijvoorbeeld: de entiteit die de Plugin aanroept, de ID van de entiteit en de Journey waarvan een Active Journey gemaakt moet worden. De Azure Function voert vervolgens de business logica uit.

Van trial naar productie

In sprint 3 heb ik aangegeven dat de CRM-omgeving is aangemaakt als een trial. Dit zorgde voor wat problemen omdat deze omgeving was verlopen. Gelukkig kon de omgeving met wat moeite worden verlengd met een maand. Dit was uiteraard niet het einde van dit proces en zodoende moest de solution nog worden overgezet naar een productieomgeving van Pixelzebra. Door gebrek aan ruimte hebben we eerst na moeten gaan welke omgeving we konden resetten, om deze vervolgens te laten functioneren als de Donation Marketing Journey omgeving. Na het één en ander geregeld te hebben was er een omgeving gereset en kon ik alles overzetten. Dit had ik kunnen laten doen door een Dynamics 365 specialist, maar heb ervoor gekozen om dit zelf op te pakken als leerproces. Het is altijd weer een verrassing wat er gaat gebeuren bij het overzetten van een solution. Zo kunnen er bijvoorbeeld problemen met afhankelijkheden, ofwel "dependency issues", optreden. Door het correct opzetten van de omgeving en de solutions, heb ik gelukkig geen problemen gehad met het overzetten. Hier merkte ik dat mijn goede voorbereiding ervoor heeft gezorgd dat dit proces niet ongewenst veel tijd ging kosten. Desondanks heeft het mij bij elkaar toch weer bijna een dag gekost om alles goed over te zetten. Een aantal onderdelen konden niet één op één worden overgezet en moesten opnieuw worden gemaakt en geconfigureerd.

Toen alles weer naar behoren functioneerde, was het tijd om de focus te leggen op kernonderdelen van de applicatie. Ik wilde een product opleveren dat beheerst over de afgesproken basisfunctionaliteiten. "De gebruiker moet een Journey kunnen configureren", dit was werkend op het gebied van front-end maar nog niet in de backend. Het configureren van de Journey bestaat uit drie onderdelen: het definiëren van de Trigger (afbeelding 8), het toevoegen van eventuele voorwaarden en het bepalen van de uit te voeren acties. Om dit te realiseren ben ik o.a. gebruik gaan maken van de entiteiten die ik heb aangemaakt in sprint 3. Als de gebruiker de Journey op "Definitief" zet, voer ik een Plugin uit die een Plugin-stap toevoegt met behulp van de Microsoft CRM-SDK. Initieel zou ik in deze stap ook een lookup (entiteit referentie) aanmaken op de Active Journey naar een nieuwe lookup, mits deze nog niet bestaat.

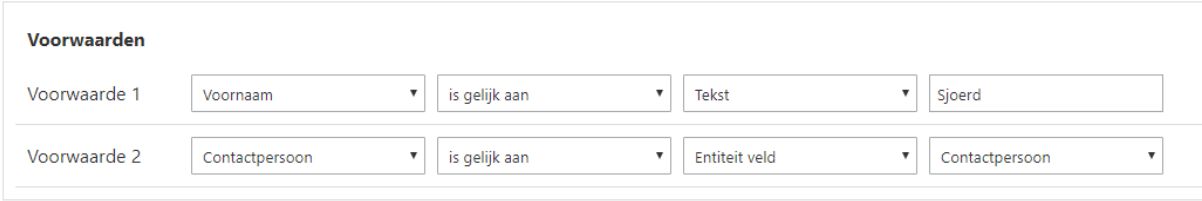
Om de Proof Of Concept (POC) binnen de scope te houden hebben we ervoor gekozen om dit in een latere fase toe te voegen en ons te focussen op de standaard klant-entiteiten, namelijk Account, Contact en Lead. Als de Plugin-stap is aangemaakt, betekent dit dat de Plugin-trigger zal afgaan aan de hand van de Journey configuratie.



Afbeelding 8: Trigger definitie

Hierna ben ik mij gaan focussen op het tweede gedeelte, namelijk het toevoegen van eventuele voorwaarden (afbeelding 9). Dit betekent dat een donateur pas wordt toegewezen aan een Journey als hij voldoet aan de voorwaarden. Ik kwam er tijdens het developen achter dat dit onderdeel een stuk lastiger was dan ik had verwacht. Je hebt te maken met verschillende datatypes en kan hierdoor vaak niet dezelfde vergelijkingstechniek gebruiken. Zo wil je bijvoorbeeld bij het vergelijken van "string" waardes hoofdletters negeren, datums vergelijken van verschillende formats, of de Guid vergelijken van

een Entity Reference (lookup). Om deze reden heb ik ervoor gekozen om per datatype te kijken welke vergelijking ik moet toepassen.



Afbeelding 9: Voorwaarde(n) definitie

Als blijkt dat de donateur voldoet aan de voorwaarden wordt er een Actieve Journey aangemaakt met bijbehorende Actieve Journey Stappen. De Actieve Journey is een statische versie van de originele Journey. Hierdoor kunnen de gedefinieerde acties uitgevoerd worden, zelfs als deze veranderen in de originele versie. Aan de Actieve Journey koppelen we Actieve Journey Stappen (acties), zodat duidelijk is welke handeling er uitgevoerd moet worden. Voor het POC hebben we ervoor gekozen om alleen Spotler Campagnes te triggeren. Uiteraard heb ik tijdens het developen wel rekening gehouden met het uitvoeren van een Activiteitesjabloon, zoals te zien in afbeelding 10. Zo heb ik op verschillende plekken todo's neergezet die leiden naar een taak die is aangemaakt in Azure DevOps. Hierdoor kunnen we in de volgende bouwfase de taken makkelijker oppakken.

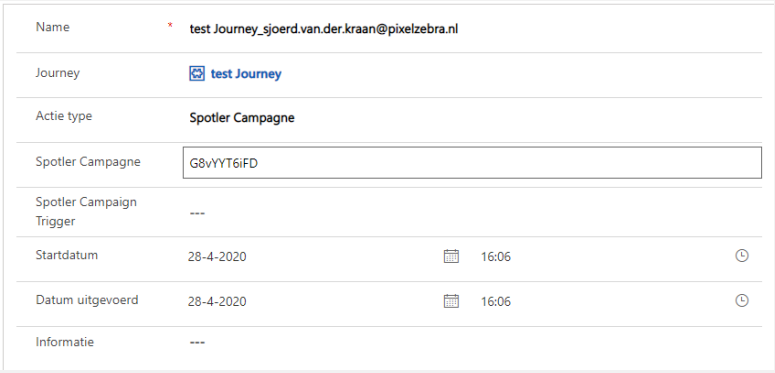
```
public void ExecuteStep(psmdmj_activejourney activeJourney, psmdmj_activejourneystep activeJourneyStep)
{
    if (activeJourneyStep.psmdmj_journeyactiontypeEnum == psmdmj_journeyactiontype.SpotlerCampagne)
    {
        UpdateActiveJourneyStepStatusExecuted(activeJourneyStep.Id);

        SpotlerAssistant spotlerAssistant = new SpotlerAssistant(_service);
        pix_mailpluscampaigntrigger spotlerCampaignTrigger = spotlerAssistant.TriggerSpotlerCampaign(activeJourney, activeJourneyStep);

        activeJourneyStep.Attributes["spotlercampaigntrigger"] = spotlerCampaignTrigger.Id; // todo: use early bounds
        _utils.Request(activeJourneyStep, RequestType.Update, _service, out string error);
    }
    else if (activeJourneyStep.psmdmj_journeyactiontypeEnum == psmdmj_journeyactiontype.Activiteitensjabloon)
    {
        // todo execute Activity template #13771
    }
}
```

Afbeelding 10: Referentie naar taak in DevOps

Voor dit POC zullen we de acties direct uitvoeren, omdat het nog niet mogelijk is om de acties op een later tijdstip te verwerken. Ik heb er wel voor gekozen om een Timed Azure Function te maken. De Timed Function kijkt iedere 5 minuten naar de "Startdatum" van acties die in het verleden liggen en nog niet zijn uitgevoerd. Als blijkt dat dit het geval is, dan wordt de actie uitgevoerd voor de desbetreffende stap en komt er een datum te staan bij "Datum uitgevoerd". De status wordt veranderd naar "Wordt uitgevoerd" en wordt later bijgewerkt wanneer bekend is of de actie succesvol is uitgevoerd. Ook wordt het informatieveld gevuld met gedetailleerde informatie over de status van de uitgevoerde actie (afbeelding 11).



Afbeelding 11, Actieve Journey Stap informatie

Als actie kan er een Spotler Campagne worden getriggerd. Hiervoor heb ik de Spotler koppeling van Pixelzebra gebruik en geconfigureerd. Als de Spotler Campagne is getriggerd, wordt op de entiteit weggeschreven of de trigger succesvol is uitgevoerd. Om dit op te vangen heb ik een Plugin gemaakt die afgaat op het wijzigen van de Spotler Campagne Trigger status. De status die daar binnenkomt, schrijf ik weg op de Actieve Journey Stap. Op deze manier weet de gebruiker of de Spotler Campagne met succes is verstuurd naar de donateur, of waarom het niet is gelukt om de campagne te versturen. Toen de kerncomponenten van de applicatie waren gebouwd, was het tijd om te testen. Samen met de Product Owner ben ik scenario's gaan testen. Hierdoor konden wij constateren dat de applicatie in zijn geheel naar behoren werkt. We hebben gekeken naar de backlog en zijn we per taak de functionaliteiten gaan doorlopen. In verband met de tijd, hebben wij de focus gelegd op de happy flow van de componenten. Bij een succesvolle manual test, kon het backlog item van "Review" worden gehaald en op "Done" worden gezet.

Proof Of Concept

Om het geheel samen te vatten kan er via een front-end pagina worden aangegeven wanneer een Journey moet starten door middel van het configureren van de Trigger. Hierna kunnen er eventueel voorwaarden worden toegevoegd waaraan de donateur moet voldoen. Uiteindelijk bepaalt de actie welke Spotler Campagne er uitgevoerd moet worden. In een latere fase kan er ook een Activiteitensjabloon worden toegevoegd als actie. Als de Journey is geconfigureerd, zet de gebruiker de Journey op "Definitief" en wordt dynamisch een stap toegevoegd aan de Plugin. Dit zorgt ervoor dat de Trigger daadwerkelijk afgaat. Als de Trigger afgaat, wordt gekeken of de donateur voldoet aan de voorwaarden. Bij een succesvolle check wordt er een Actieve Journey record aangemaakt met Actieve Journey Stappen. Op de Active Journey staat de Journey als een statische versie, zodat deze niet verandert voor de donateur als de originele Journey wijzigt. Ook wordt op de Actieve Journey bijgehouden aan welke donateur de Journey is gekoppeld. Op de zogeheten Actieve Journey Stappen staat aangegeven wanneer en welke actie er is uitgevoerd, of nog uitgevoerd moet worden. De status van de uitvoering wordt hier ook weggeschreven, zodat de gebruiker gemakkelijk kan achterhalen waarom de actie wel of niet is uitgevoerd.

Het toevoegen van de trigger stap moet op dit moment handmatig gedaan worden met behulp van de Plugin Registration Tool. Het is niet gelukt om dit proces volledig te automatiseren, omdat ik te maken kreeg met onbekende aspecten zoals CRM-metadata. Hierdoor had ik niet voldoende tijd om dit onderdeel van de applicatie af te maken.

7.6.3 Retrospective Sprint 4

Sprint 4 was een intensieve sprint omdat ik nog veel moest doen aan de backend. Niet alleen moest ik de basis van de applicatie werkend krijgen, maar moest de gehele omgeving worden overgezet. Ik ben er uiteindelijk toch in geslaagd om de applicatie voor 95% af te ronden. De laatste 5% is niet gelukt, omdat ik het toevoegen van de Plugin stap niet kon automatiseren. Ik ben trots met het resultaat dat ik in een korte tijd heb weten op te zetten. Ik had, zo goed als, geen ervaring op het gebied van front-end en heb dit toch weten te combineren met de interface van Dynamics 365 CRM. De front-end kostte om deze reden ook meer tijd dan ik had ingeschat en zorgde voor een overvolle sprint 4. Zoals ik in sprint 3 heb vermeld, lopen de uren al snel hoog op met onvoorziene taken. Het blijft altijd weer een kunst om hierop in te spelen en je toch aan de planning te blijven houden. Het bijhouden van een duidelijke backlog en het structureel afwerken van taken hebben mij flink geholpen. Tijdens de vergaderingen en

het developen kwamen er constant ideeën bij, of veranderde bestaande ideeën. Deze heb ik direct toegevoegd aan de backlog, zodat ze niet verloren gaan en in de toekomst kunnen worden opgepakt. Ik vond het lastig om de applicatie niet voor de volle honderd procent te kunnen aanleveren. De volledige flow van de applicatie werkt op dit moment alleen als handmatig de Trigger wordt toegevoegd aan de Plugin Registration Tool. Dit had ik graag nog toe willen voegen voor de oplevering, maar staat nu gepland voor fase twee van het development proces.

8 Onderzoek ketentest en implementatie

8.1 Inleiding

In dit hoofdstuk ga ik in op de literatuurstudie van het onderzoek en de implementatie van de teststrategie. Voor dit onderzoek, en de uitvoering, is een tijdsbestek gereserveerd van twee weken. Van deze twee weken kan ik drie dagen per week besteden aan het onderzoek. Ik wilde mijzelf uitdagen door in zes dagen een onderzoek en teststrategie te maken én om deze te implementeren waar mogelijk.

8.2 Uitvoeren onderzoek

Om een gericht onderzoek uit te voeren heb ik de aanleiding, doelstelling en methode gedefinieerd. Hiervoor heb ik gekeken naar het probleem waar Pixelzebra tegenaan loopt. In de afgelopen maanden heb ik bij Pixelzebra een nieuwe applicatie ontwikkeld, namelijk de Donation Marketing Journey editor. Het is voor Pixelzebra van groot belang dat de applicatie naar behoren blijft werken, ook tijdens het developen. Op dit moment zijn er 4 developers die het maatwerk voor Dynamics 365 CRM ontwikkelen. De werkdruk ligt hoog en de tijd voor manueel testen is beperkt. Om deze reden is het van groot belang dat er geautomatiseerde testen worden uitgevoerd om tijd te besparen en de kans op fouten te minimaliseren.

Het was helder waarom er belang is bij het automatiseren van testen. Om een duidelijke scope te creeëren voor het onderzoek ben ik een hoofdvraag en deelvragen gaan opstellen. Met behulp van dit onderzoek wil ik achterhalen hoe we door middel van geautomatiseerde testen de kwaliteit kunnen waarborgen van de Donation Marketing Journey editor. Om de hoofdvraag te beantwoorden heb ik vier deelvragen opgesteld. Het is allereerst belangrijk om te weten aan welke eisen de testen en testtools moeten voldoen. Veelal maakt criteria het moeilijker om bepaalde keuzes te maken. Denk hierbij aan de hoeveelheid tijd die beschikbaar is en welke elementen er minimaal getest moeten worden. Deze criteria heb ik opgesteld in de scope definitie van het theoretisch kader (zie bijlage 3, hoofdstuk 2.1).

Ik ben gaan uitzoeken wat de meestgebruikte testsoorten zijn. Hiervoor het ik gekozen om gebruik te maken van de testpiramide, omdat dit model duidelijk aangeeft wat de impact is van de testsoort. Het schrijven van Unit testen gaat sneller dan het schrijven van End-to-End testen en het kost minder geld (Fowler, 2012). Aan de andere kant kunnen er meer scenario's worden gestest met End-to-End testing, omdat de volledige flow van de applicatie getest kan worden (Fowler, 2012).

Voor het kiezen van de juiste tools heb ik gekozen voor de top 5 van de website "The State of JavaScript". Deze website houdt al jaren de trends bij op het gebied van testtools. Hierbij meten ze of de gebruikers bekend zijn met de tool, of ze de tool hebben gebruikt én of de gebruiker de tool nogmaals zal gebruiken. Om doelgericht te zoeken heb ik twee criteria aangehouden waar de tool aan moet voldoen. De implementatie moet haalbaar zijn in twee dagen, dus de complexiteit van de tool moet zo laag mogelijk liggen. Hiernaast moet de tool Unit testen kunnen uitvoeren, omdat deze testsoort het snelste is om te implementeren (Valori, 2018). Tijdens het doorspitten van de documentaties heb ik de keuze gemaakt om de testtool Storybook niet verder te onderzoeken. Deze tool is bedoeld voor het testen van UI componenten in React, Vue en Angular en viel buiten de scope van het onderzoek.

Best Practices

Om de teststrategie goed vorm te geven, heb ik gebruik gemaakt van de goede en slechte ervaringen van professionals op het gebied van software testing. Door gebruik te maken van Best Practices kan er sneller en efficiënter een teststrategie worden opgezet en vergroot dit hiermee de kans op een succesvolle implementatie. Internet staat vol met Best Practices, dus heb ik er een zestal tussenuit gevist. De Best Practices (bijlage 3, hoofdstuk 4.6) zijn naar mijn mening het beste van toepassing voor het testen van de applicatie en zijn opgesplitst in twee categorieën. De eerste categorie is specifiek gericht op het testen van software. De tweede categorie is gericht op programmeren in het algemeen en helpt bij het efficiënt schrijven van testen.

8.3 Maken teststrategie

Allereerst ben ik de benadering en het doelstelling gaan definiëren, zodat helder werd wat het doel is van de teststrategie en aan welke criteria deze moet voldoen. Het doel van de teststrategie is om vast te stellen welke onderdelen er getest moeten worden, welke tools hiervoor worden gebruikt en welke implementatiemethode gehanteerd wordt. Om Scrum aan te houden heb ik gebruik gemaakt van timeboxing. Scrum gebruikt veel timeboxing en iteratieve ontwikkeling (Coplien, 2010). Bij Scrum wordt uitsluitend in sprints gewerkt (Cohn, 2010). De typische lengte van een sprint is 30 dagen (Leffingwell, 2011) Sprint planning, sprint retrospective en sprint reviews zitten allemaal in een timebox (Schwaber, 2000).

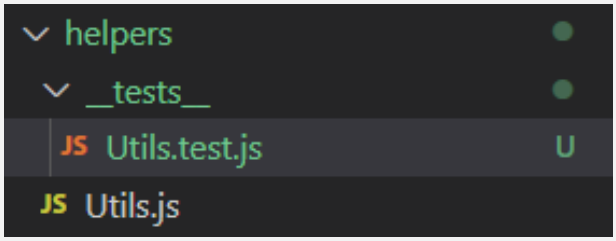
Als tweede heb ik vastgesteld voor welke onderdelen testen moeten worden geschreven. Hiervoor heb ik een tabeloverzicht gemaakt van de onderdelen, met omschrijving, en de hoeveel tijd die eraan besteed mag worden (zie bijlage 3, hoofdstuk 5.3). Als laatste is beschreven welke testsoorten en testtools gebruikt worden om het gewenste resultaat te behalen. Hierin geef ik aan dat er een timebox van ongeveer 3 uur is gerekend voor het eerste onderdeel. De eerste timebox is cruciaal, omdat het kiezen van de juiste tool bepalend is voor de resterende implementatie. Als het niet is gelukt om 50% van de implementatie te behalen binnen de timebox tijdslimiet, wordt overgestapt op Cypress om verdere uitloop te voorkomen.

Onderdeel	Omschrijving	Timebox
Jest integratie	De integratie moet plaatsvinden met de huidige front-end applicatie	*2,5 uur*
Sanity test	Met behulp van een sanity test wordt gekeken of de integratie tussen de front-end en Jest is gelukt. Hierbij wordt getest of de testen überhaupt kunnen worden uitgevoerd.	*0,5 uur*
Utils functies	De Utils class helpt de Journey editor bij het aanpassen van de DOM-elementen en beschikt over generieke functionaliteiten.	*5 uur*
Store	De Store variabelen zijn van groot belang en bevatten zowel de constanten als de dynamische variabelen van de applicatie. Deze waardes worden geïmporteerd of geëxporteerd, zodat de gebruiker de Journey kan wijzigen of kan toevoegen aan het systeem.	*4 uur*
API-calls	Jest maakt het mogelijk om asynchrone calls uit te voeren en te wachten op het resultaat (jestjs.io, 2020).	*4 uur*

Afbeelding 12, Tabel overzicht werkzaamheden toepassen teststrategie

8.4 Uitvoeren teststrategie

Om gestructureerd aan de slag te gaan heb ik een planning gemaakt (Afbeelding 12). Allereerst moest worden uitgezocht hoe ik Jest kon integreren met de huidige front-end applicatie. In de documentatie van de tool staat aangegeven dat dit te realiseren is door "yarn add jest --dev" in de terminal uit te voeren. Na een succesvolle installatie is gekeken naar de folder architectuur, om de juiste folders en bestanden aan te maken (afbeelding 13) met de correcte benaming. Om te achterhalen of de opzet succesvol is opgezet heb ik een sanity test geschreven (afbeelding 14). Een sanity test is bedoeld om vast te stellen of de basis opzet voor testen naar behoren werkt en het mogelijk is om de testen uit te voeren (Sammi, 2011).



Afbeelding 13, folder architectuur voor testbestanden

```
✓ test('Sanity test', () => {
    expect(true).toBe(true);
});
```

Afbeelding 14, Sanity test

Het doel van deze teststrategie implementatie, draait niet om kwantiteit maar kwaliteit. Het is niet de bedoeling om zoveel mogelijk testen te schrijven, maar om de moeilijkheidsgraad te verhogen na iedere geslaagde test. Hiermee kan de complexiteit en de toevoeging van de implementatie worden bepaald. Ik ben begonnen met het testen van de Utils class. De class helpt de Journey editor bijvoorbeeld bij het aanpassen van het DOM (Document Object Model) en bij het verkrijgen van waardes uit de URL. Hierbij liep ik tegen het probleem aan dat ik testen moest uitvoeren op DOM-elementen. Deze kunnen worden toegevoegd aan de "innerHTML" van het DOM (afbeelding 15). Dit was in het begin verwarrend, omdat je niet ziet dat de applicatie wordt gebouwd en beschikt over een "document". Door de testen uit te voeren met behulp van het commando "Jest test", wordt onderliggend een DOM gegenereerd waar de testen op uitgevoerd kunnen worden.

```
describe('Utils', () => {
    ✓ test('hasContent', () => {
        document.body.innerHTML = '<div id="key">HasContent</div>';
        expect(hasContent('key')).toBe(true);
    });
```

Afbeelding 15, Gebruik maken van het DOM

De voorgaande testen verliepen redelijk voorspoedig. Het verkrijgen van de URL-parameters, ook wel "query string" genoemd, bracht daarentegen een probleem met zich mee. Bij het uitvoeren van een test is er niet standaard een URL gedefinieerd. Na nader onderzoek was ik niet de eerste die tegen dit probleem aan liep. De oplossingen die ik kon vinden op Google waren helaas "outdated", omdat de "Object.defineProperty" (afbeelding 16) niet werkt met de huidige versie van Jest. Ryan Doll heeft dit probleem weten te tackelen (Doll, 2018) door een test url toe te voegen aan de configuratie (afbeelding 17). Hierdoor was het mogelijk om een "stub" URL te genereren. Aan de stub URL konden parameters worden toegevoegd, om deze er vervolgens weer uit te halen met behulp van de Utils functie.

```
Object.defineProperty(window.location, 'href', {
  writable: true,
  value: 'https://www.somthing.com/test.html?query=true'
});
```

Afbeelding 16, "Object.defineProperty" methode mock URL

```
module.exports = {
    verbose: true,
    "testURL": "https://www.test.com/test.html"
};
```

Afbeelding 17, Test url toevoegen aan Jest configuratie

Ik heb gekozen om voor de Jest configuratie een apart configuratiebestand te maken. Dit zorgt ervoor dat de Webpack configuratie overzichtelijk blijft en dat de verantwoordelijkheid wordt neergelegd bij het juiste bestand. Na het schrijven van testen voor de Utils class, ben ik me gaan focussen op de Store. De Store variabelen zijn van groot belang en bevatten zowel de constanten als de dynamische variabelen van de applicatie. Deze waardes worden geïmporteerd of geëxporteerd, zodat de gebruiker de Journey kan wijzigen of kan toevoegen aan het systeem. Om de Store te kunnen testen heb ik een export gemaakt van de Json en deze gebruikt als stub.

Als laatste stond het testen van API-calls op de planning. Jest maakt het mogelijk om asynchrone calls uit te voeren en te wachten op het resultaat. Ik had over het hoofd gezien dat het bij Unit testen niet de bedoeling is om een andere service aan te roepen. Het aanroepen van andere services wordt gedaan bij End-to-End testing. Desondanks wilde ik wel uitzoeken hoe het testen van asynchroon functies in zijn werk gaat. Het aanroepen van de functie werkt op dezelfde manier als bij niet-asynchroon testen, maar met de gebruikelijke "await" voor het aanroepen van de methode. Hiernaast moet niet de andere service worden aanroepen, maar stub data worden teruggeven. Om deze reden moet er gekeken worden naar de staat van het environment (afbeelding 18). Als het environment op "test" staat wordt stub data teruggeven, anders de data van de gebruikelijke call. Na het aanpassen van de bestaande functie kon de test succesvol en asynchroon worden uitgevoerd.

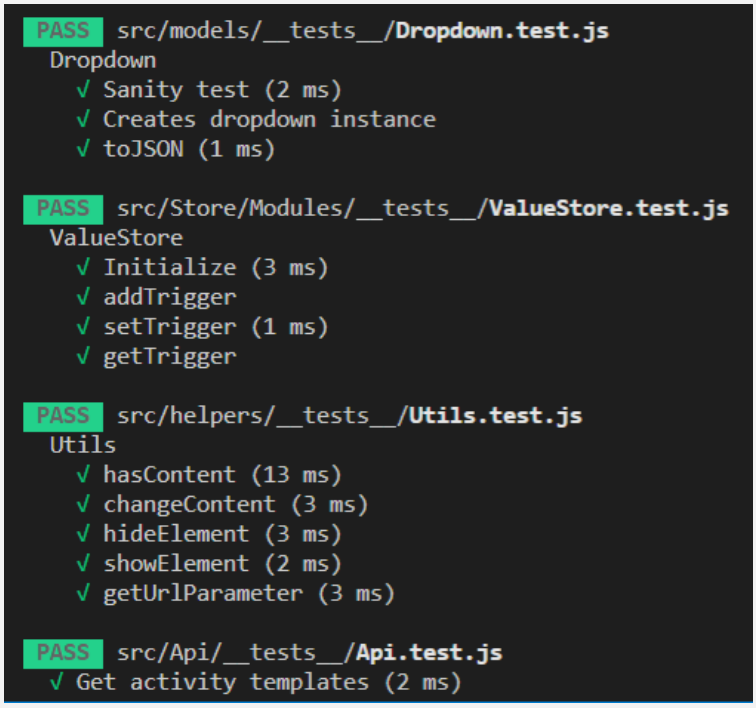
```
static async doRequest(req) {
    if (process.env.NODE_ENV === 'test') {
        return Promise.resolve(req.Route || req.logicalName);
    }

    const response = await Xrm.WebApi.online.execute(req);
    const data = await response.json();

    if (data && data.File) {
        return JSON.parse(atob(data.File));
    }       You, 2 months ago • Initial commit
}
```

Afbeelding 18, Environment check

Om het geheel overzichtelijk te maken bij het uitdraaien van de testen, heb ik de testen toegevoegd aan "describe" classes (afbeelding 15). Door de testen in een "describe" class te zetten, wordt een verzameling gevormd tijdens de test run (afbeelding 19).



Afbeelding 19, Resultaat van testen in "describe" classes.

8.5 Retrospective onderzoek en implementatie

In het Plan van Aanpak had ik voor dit onderzoek twee weken gereserveerd. Ik had zes afstudeerdagen voor zowel het uitvoeren van de literatuurstudie als voor de implementatie van de teststrategie. Dit was achteraf gezien aan de krappe kant en heb hierdoor bepaalde aspecten niet kunnen behandelen. Ik had mij bijvoorbeeld graag nog willen focussen op tools voor het Unit testen van de backend. Hiernaast is het lastig om in zo'n korte tijd goede literatuur te vinden. Het gebruiken van "grijze" literatuur is niet wetenschappelijk onderbouwd en soms onbetrouwbaar. Daarentegen heb ik in een zeer korte tijd veel geleerd over het implementeren van Unit testen voor een front-end applicatie. Het was vrij eenvoudig om Unit testen op te zetten met Jest. De testtool werkt intuïtief en ik had hierdoor binnen de een aantal uur een sanity test gedraaid. In de toekomst wil ik meer gebruik maken van deze tool en mijn kennis eventueel uitbreiden met de testtool Cypress. Het testen van software is een onderwerp waar ik vrijwel geen ervaring mee had en veel van heb mogen leren.

9 Handleiding en adviesrapport

Om een duidelijk beeld te scheppen van de Donation Marketing Journey editor, heb ik een handleiding en adviesrapport opgesteld. In de handleiding (bijlage 4) wordt duidelijk gemaakt wat het doel is van de applicatie. Het is van belang dat de lezer weet wat de aanleiding is geweest voor de ontwikkeling van de Donation Marketing Journey editor. Hierdoor kan de lezer zich inleven in het probleem en bepaalde functionaliteiten van de applicatie beter plaatsen. Toen dit was uitgewerkt heb ik de aandacht gelegd op de functionaliteiten van de applicatie. Hierin staat duidelijk omgeschreven hoe de applicatie werkt en welke onderdelen geconfigureerd kunnen worden door de gebruiker. Om ervoor te zorgen dat het product kan worden geïntegreerd met Dynamics 365 CRM-omgevingen van klanten, heb ik de implementatie beschreven. Hier wordt uitgelegd welke onderdelen configuratie vereisen en welke solutions al bij de klant in het CRM-systeem moeten zitten. Hiernaast staat genoteerd waar alle elementen van de applicatie zijn opgeslagen. De elementen zijn bijvoorbeeld de URL van de omgeving, de repository en de backlog. Als laatste heb ik de aandacht gelegd op specificaties waar developers rekening mee moeten houden. Denk hierbij aan het installeren van Nodejs, yarn en Jest. Dit zijn componenten die noodzakelijk om de applicatie mee te testen of aan te passen. De backend is opgesteld op basis van de Pixelzebra standaard. Om deze reden ben ik voor de documentatie niet dieper ingegaan op de backend.

In het adviesrapport (bijlage 4, hoofdstuk 5) wordt ingegaan op het onderzoek. Hierin wordt gekeken naar manieren om geautomatiseerde testen toe te passen voor de DMJ-editor, zodat de kwaliteit kan worden gewaarborgd. Om het onderzoek helder neer te zetten ben ik begonnen met het formuleren van de aanleiding, gevolgd door de teststrategie en de implementatie hiervan. Naar aanleiding van de implementatieresultaten heb ik het volgende voorstel gedaan: *"Het is belangrijk dat de volledige flow van de applicatie wordt getest. Het schrijven van Unit testen help om de juiste werking van functies helder te krijgen, maar hierdoor wordt er niet gewerkt met real-time data. Hiernaast moeten bestaande functies worden aangepast om de stub data te retourneren. Om deze reden wordt End-to-End testing aangeraden, omdat dit een hogere meerwaarde biedt voor het testen van de front-end applicatie ten opzichte van Unit testing."*

Het document is afgesloten met aandachtspunten. De Donation Marketing Journey editor applicatie is op dit moment bijna volledig geautomatiseerd. De laatste verbeterslag die moet worden uitgevoerd is "Het dynamisch aanmaken van een plugin step" (taak #13579). Hiernaast is een backlog opgesteld, zodat er een duidelijk overzicht is van de aandachtspunten. De taken met betrekking tot het development staan genoteerd in de code (afbeelding 20).

```
psmdmj_JourneyId = journey.ToEntityReference(),
[$"psmdmj_{triggerEntity.LogicalName}id"] = triggerEntity.ToEntityReference(),
psmdmj_name = journey.psmdmj_name,
psmdmj_emailaddress = (string)triggerEntity["emailaddress1"], // TODO: #13566
psmdmj_JourneyJson = journey.psmdmj_JourneyJson
```

Afbeelding 20, TODO in code

10 Aantonen beroepstaken

10.1 A1: Analyseren probleemdomein & opstellen probleemstelling.

Requirementsanalyse

Voor het opstellen van het afstudeerplan ben ik met Robert van der Kleij om tafel gaan zitten voor een inventarisatie van het probleem. In het eerste interview werd het probleem voorgelegd met een opzet voor een oplossing. Dit ben ik verder gaan uitwerken om helder te krijgen wat precies het probleem en de doelstelling was. In het tweede interview heb ik dit voorstel voorgelegd aan Robert, Paddy van der Kaay en Jim van Zijl. Samen zijn we het eens geworden over het op te leveren eindproduct (hoofdstuk 5.1). *Hiermee geef ik aan dat ik een intake heb uitgevoerd bij de opdrachtgever en betrokken partijen*. Naar aanleiding van de interviews ben ik de requirementsanalyse op gaan zetten (bijlage 2). In dit document staat duidelijk geformuleerd waar het product voor dient en wat de scope is van de applicatie (bijlage 2, hoofdstuk 1). Voor de afbakening heb ik rekening gehouden met de beschikbare tijd en ben ik mij gaan richten op de kernfunctionaliteiten van de applicatie. Hiernaast moest ik rekening houden met verschillenden standaarden en richtlijnen van Pixelzebra. Denk hierbij aan keuzes voor het gebruik van software, versies van tooling en securitymaatregelen (bijlage 2, hoofdstuk 2.1 t/m 2.4). Na het formuleren van de requirements zijn deze in overleg met Paddy van der Kaay (Product Owner) geprioriteerd (bijlage 2, hoofdstuk 2.5 t/m 2.7).

Onderzoeksrapport

Het was een uitdaging om in een timebox van twee weken een onderzoek te doen. Ik ben gestructureerd het onderzoeksrapport op gaan zetten door een aanleiding en doelstelling te formuleren (bijlage 3, hoofdstuk 1). Het doel van het onderzoek was om te achterhalen welke vorm van geautomatiseerd testen het beste past bij de Donation Marketing Journey. Een belangrijk aspect dat meespeelde was de korte tijd waarin het gerealiseerd moest worden. Voorafgaand aan de literatuurstudie heb ik geformuleerd welke methoden ik ging gebruiken (bijlage 3, hoofdstuk 3). Hierin heb ik aangegeven dat er gebruik werd gemaakt van een Multivocal Literature Review (MLR) studie. Dit houdt in dat er naast gepubliceerde papers ook gebruik is gemaakt van "grijze" literatuur, zoals blogs en white papers. Hiernaast heb ik aangegeven dat er een labonderzoek wordt uitgevoerd dat betrekking heeft op het uitvoeren van de teststrategie. *Hiermee geef ik aan dat ik een onderzoeksvoorstel inclusief vraagstelling heb opgesteld.* De resultaten van de literatuurstudie zijn vervolgens genoteerd (bijlage 3, hoofdstuk 4). Naar aanleiding van de literatuurstudieresultaten heb ik een teststrategie ontworpen (bijlage 3, hoofdstuk 5) en toegepast op de front-end van de Donation Marketing Journey editor. Aan de hand van de resultaten van de teststrategie (bijlage 3, hoofdstuk 6) is een conclusie getrokken (bijlage 3, hoofdstuk 7). De conclusie had ik graag met meer feiten willen onderbouwen, maar dat is achteraf gezien lastig met een implementatietijd van twee dagen. Het onderzoeksrapport is afgesloten met een discussie waarin ik aanraad om End-to-End testing toe te passen (bijlage 3, hoofdstuk 8).

10.2 Gc: Kritisch, onderzoekend en methodisch werken

Best Practices en Scrum

Zoals aangegeven in hoofdstuk 10.1, heb ik verschillende vergaderingen georganiseerd en geleid om de requirements te vergaren en te bewaken. Om de vergaderingen gestructureerd te houden, heb ik mij goed voorbereid met vragen en geformuleerd wat ik wilde behalen met de vergadering (hoofdstuk 5.1). Hierbij heb ik mij gehouden aan best practices met betrekking tot het leiden van vergaderingen (hoofdstuk 5.1). Tijdens de vergaderingen heb ik veel genotuleerd, doorgevraagd bij onduidelijkheden en ideeën op tafel gelegd waar mogelijk. De uitwerking van de notulen zijn naderhand voorgelegd bij de betrokkenen ter goedkeuring. *Hiermee toon ik aan dat ik vraag om feedback en deze beoordeel*. Om het development proces in goede banen te laten leiden, heb ik onderdelen van de Scrum methodiek toegepast (hoofdstuk 3). Het proces is opgedeeld in sprints van twee weken. Iedere sprint eindigt met een retrospective om te achterhalen wat wel of niet goed ging. De planning van de volgende sprint werd vervolgens aangepast om op schema te blijven. *Hiermee toon ik aan dat ik werk volgens een gevalideerd proces*. Hiernaast heb ik met Robert van der Kleij en Paddy van der Kaay afgesproken om hen iedere week een update te geven. Op deze manier betrok ik ze zoveel mogelijk bij het proces, zodat we op één lijn zaten. *Hiermee toon ik aan mijn collega's te overtuigen op basis van argumenten.*

Onderzoeksrapport

Zoals aangegeven in hoofdstuk 10.1, maak ik voor het onderzoek gebruik van een Multivocal Literature Review (MLR) studie. *Hiermee toon ik aan dat ik werk volgens een gevalideerd proces.* De voornaamste reden voor deze keuze ligt bij het gebrek aan tijd. Het zoeken naar gepubliceerde artikelen met betrekking tot testen is lastig in een korte tijd. Hiernaast gaat de ontwikkeling op het gebied van testen hard, wat betekent dat gepubliceerde artikelen al snel achterlopen op de feiten. Het nadeel van grijze literatuur is dat het niet altijd even betrouwbaar is. Om deze reden heb ik vaak gebruik gemaakt van verschillende bronnen, zodat ik mijn uitspraak standvastiger kon onderbouwen (bijlage 3, hoofdstuk 4.4). *Hiermee toon ik aan dat ik kritisch lees en denk.* Om uit te zoeken welke verschillende testsoorten er zijn heb ik gebruikt gemaakt van de testpiramide (Valori, 2018). In dit model is duidelijk aangegeven hoeveel tijd het kost om de testsoort te implementeren. In verband met de korte tijdslimiet voor het onderzoek, heb ik ervoor gekozen om Unit testen te implementeren.

Om te achterhalen wat de meest gebruikte testtools zijn op dit moment, heb ik gebruik gemaakt van de website "The State Of JavaScript" (The State of JavaScript, 2019). Deze website heeft meerdere jaren een grootschalig onderzoek gedaan. Tijdens dit onderzoek hebben zij gekeken naar wat developers vinden van hedendaagse testtools en welke tools de voorkeur hebben, of juist niet. Ook heb ik per testsoort gekeken naar de officiële documentatie, zodat ik kon bepalen welke testsoort het beste past binnen de teststrategie. *Hiermee toon ik aan dat ik onderscheid maak tussen feiten en meningen.* Naar aanleiding van de literatuurstudie is de teststrategie ontworpen en toegepast. De resultaten van de teststrategie implementatie zijn opgenomen in het onderzoeksrapport en hebben bijgedragen aan de uiteindelijke conclusie. Ik ben zeer tevreden met wat ik in twee weken op heb kunnen zetten qua onderzoek en implementatie.

10.3 Gf: Leren leren: voorbereiden op volgende studiefase en beroep

In het afstudeerplan heb ik duidelijk geformuleerd aan welke beroepstaken ik wil voldoen voor het einde van het afstuderen. *Hiermee toon ik aan dat ik duidelijke en afgebakende leerdoelen opstel*. In het Plan van Aanpak heb ik een overzichtelijke planning gemaakt per twee weken (bijlage 1, hoofdstuk 7). Eerst ben ik de werkzaamheden gedetailleerd gaan formuleren en heb deze vervolgens samengevoegd in een overzichtelijke tabel (bijlage 1, hoofdstuk 9). Op deze manier had ik altijd een duidelijke planning bij de hand waar ik mij aan kon houden. *Hiermee toon ik aan dat ik het leerproces goed heb gepland*. Voor het developen heb ik gebruik gemaakt van onderdelen uit de Scrum methodiek, zoals sprints van twee weken en retrospectives. *Dit heeft mij geholpen bij het waarborgen van mijn leerproces*. Door over te schakelen van Vue.js naar vanilla JavaScript, heb ik eventuele achterstand kunnen voorkomen. Ik heb niet te lang geprobeerd om het werkend te krijgen, maar ben op zoek gegaan naar alternatieven. *Hiermee toon ik aan dat ik weet waar de prioriteit ligt en wat mijn leerstijl is*.

Ik wilde in een tijdspan van twee weken een onderzoek doen en een teststrategie maken en implementeren. *Hiermee toon ik aan dat ik duidelijke en afgebakende leerdoelen opstel*. Op het gebied van front-end had ik weinig tot geen kennis. Hierdoor heb ik goed moeten onderzoeken welke testtool ik het beste kon gebruiken voor Unit testing. *Hiermee toon ik aan dat ik bijblijf in het vakgebied*. Bij grijze literatuur heb ik gekeken naar meerdere bronnen om een zo betrouwbaar mogelijk beeld te krijgen. *Hiermee toon ik aan dat ik relevant studiemateriaal zoek en beoordeel*.

10.4 C6: Ontwerpen software

Naar aanleiding van de vergaderingen voor het ophalen van de requirements, ben ik deze gaan uitwerken in een requirementsanalyse document (bijlage 2). Vanuit de functionele requirements heb ik Epics opgesteld om een duidelijk beeld te krijgen van de samenwerking tussen functionaliteiten (bijlage 2, hoofdstuk 3). De Epics zijn opgedeeld in User Stories om een duidelijker beeld te scheppen van de componenten die gemaakt moeten worden (bijlage 2, hoofdstuk 4). Uiteindelijk zijn Use Cases ontworpen om precies aan te geven welke handelingen de gebruiker en het systeem uitvoeren (bijlage 2, hoofdstuk 5). *Hiermee geef ik aan dat ik op een gestructureerde manier de software architectuur ontwerp*.

Om de werking en communicatie tussen componenten duidelijk in beeld te brengen heb ik een componenten architectuur diagram gemaakt. Hierbij heb ik gebruik gemaakt van de "ArchiMate 3.0.1." notatiewijze. Tijdens het tussentijds assessment kwam naar voren dat ik niet de correcte notatiewijze had aangehouden. In de afgelopen vier jaar hebben wij op school weinig tijd besteed aan het opstellen van architectuurmodellen. Hierdoor heb ik veel moeite gehad met het vinden van en toepassen van de juiste notatiewijze. Ik heb uiteindelijk gekozen om het Design Pattern aan te houden van "visual-paradigm.com" (visual-paradigm, 2020). De reden voor deze keuze is, omdat ArchiMate deels is gebaseerd op de IEEE 1471 standaard. Dit betekent dat de modeleringtaal voldoet de officiële software architectuur richtlijnen. Hiernaast bestaat de taal al meer dan 15 jaar en zijn er tot en met 2017 nieuwe versies uitgebracht. *Hiermee toon ik aan dat ik Design Patterns hanteer*.

Voor de database maak ik gebruik van het Dynamics 365 Common Data Model. Dit datamodel maakt het mogelijk om vanuit verschillende Microsoft-applicaties gebruik te maken van de database (Simens, 2017). Met behulp van "Entities" kan data worden gelezen, aangemaakt, bijgewerkt of verwijderd. Voor dit project heb ik gebruik gemaakt van de Dynamics 365 SDK (hoofdstuk 3.2.4). De data is zichtbaar in Dynamics 365 CRM en kan ook op dit platform worden gewijzigd. *Hiermee toon ik aan dat ik de applicatie heb ontworpen met behulp van een Object Oriented programmeermodel*.

10.5 D14: Realiseren van software

Naar aanleiding van het requirementsanalyse document ben ik de backlog gaan indelen. De werkzaamheden zijn verdeeld over vier sprints van twee weken per sprint. Hierdoor heb ik structuur en overzicht gecreëerd voor de onderdelen die gerealiseerd moeten worden. Op basis van de componentenarchitectuur zijn de verschillende onderdelen aan elkaar gekoppeld. *Hiermee toon ik aan dat ik de applicatie ontwikkel op basis van een ontwerp*. Voor het realiseren van de software heb ik goed gekeken naar de huidige architectuur van Pixelzebra. Naar aanleiding van de eisen die Microsoft stelt aan haar producten, en de huidige architectuur van Pixelzebra, heb ik keuzes gemaakt met betrekking tot de gebruikte softwaretalen en tools (hoofdstuk 3.2). *Hiermee toon ik aan dat ik gebruik maak van verschillende frameworks*.

De front-end applicatie is geïntegreerd in Dynamics 365 CRM. De applicatie wordt ingeladen in een IFrame op de entiteitpagina en haalt eventueel benodigde informatie uit de database. De componenten worden dynamisch opgebouwd en gevuld met de opgehaalde data. Met behulp van de Xrm.WebApi (hoofdstuk 3.2.4) kan via de front-end applicatie een CWA (hoofdstuk 3.2.4) worden aangeroepen. De CWA voert een API-call uit om een opdracht uit te voeren of data te verkrijgen. De verkregen data wordt verwerkt in de front-end applicatie en componenten worden bijgewerkt waar nodig. *Hiermee toon ik aan dat ik een dynamische website heb ontwikkeld*. Via de backend worden verschillende acties en berekeningen uitgevoerd. Van het ophalen van data uit de database, tot het aanroepen van Spotler Campagnes vanuit het Spotler platform. *Hiermee toon ik aan dat ik (op z'n minst) kan programmeren met een for-loop*. Om het geheel te realiseren heb ik gebruik gemaakt van de programmeer-, opmaak- en scripttalen: C#, JavaScript, HTML en CSS.

10.6 D17: Configureren

Om gebruik te kunnen maken van het Dynamics 365 Common Data Model, moest er een Dynamics 365 CRM-solution worden opgezet en geconfigureerd. Ik heb verschillende entiteiten aangemaakt, deze kunnen worden vergeleken met tabellen uit een SQL-database. Een entiteit heeft een paar belangrijke eigenschappen zoals een "name", "logicalName" en "Id". De entiteit bestaat uit verschillende velden waarin data kan worden opgeslagen, ook wel te vergelijken met rijen uit een tabel. Binnen Dynamics 365 CRM zijn er een groot aantal datatypes en opties beschikbaar die per veld moeten worden geconfigureerd. Veelal kunnen gegevens achteraf niet worden aangepast, dus het was belangrijk om secuur te werk te gaan en mij te houden aan de best practices met betrekking tot Dynamics 365 CRM. *Hiermee toon ik aan dat ik een Customer Relationship Management systeem kan configureren*. Toen de entiteiten en velden eenmaal waren geconfigureerd, moest de front-end applicatie worden getoond op de Journey entiteit. Dit heb ik gerealiseerd door gebruik te maken van een Dynamics 365 CRM Webresource (hoofdstuk 7.4) in combinatie met de AlertJS library. Met behulp van de XrmToolbox heb ik op de Journey entiteit een knop geconfigureerd die een script aanroept. In het script maak ik gebruik van AlertJS om een venster te creëren waar een Webresource in zit. De Webresource is in dit geval de front-end applicatie. *Hiermee toon ik aan dat ik met behulp van configuratie en scripting een applicatie kan integreren met een bestaand softwarepakket*.

Naast Dynamics 365 CRM vond er ook configuratie plaats op het e-mail marketing platform Spotler. Om de integratie mogelijk te maken moest er een API key en secret worden gegenereerd. Deze waardes worden gebruikt om een veilige connectie tot stand te brengen tussen Dynamics 365 CRM en Spotler. De API key en secret heb ik opgeslagen op de configuratie entiteit, zodat deze kunnen worden opgehaald via de CRM SDK. Om te kunnen werken met testdata heb ik contacten aangemaakt, e-mail templates ontworpen en Spotler Campagnes geconfigureerd. *Hiermee toon ik aan dat ik e-mail marketing automation software kan configureren en kan integreren met bestaande softwarepakketten*.

10.7 D15: Testen

Onderzoeksrapport

In een timebox van twee weken heb ik een onderzoek uitgevoerd. In het onderzoek heb ik gekeken naar verschillende testsoorten en testtools, zodat ik teststrategie kon ontwerpen en implementeren. Ik ben gestructureerd het onderzoek op gaan stellen door een aanleiding en doelstelling te formuleren (bijlage 3, hoofdstuk 1). Het doel van het onderzoek was om te achterhalen welke vorm van geautomatiseerd testen het beste past bij de Donation Marketing Journey. Een belangrijk aspect dat meespeelde was de korte tijd waarin het gerealiseerd moest worden. De resultaten van de literatuurstudie zijn vervolgens genoteerd (bijlage 3, hoofdstuk 4) en er is bepaald welke testsoort en testtool het beste geschikt is voor de teststrategie. Ook heb ik een overzicht gemaakt van een aantal belangrijke best practices, zoals: "Maak duidelijk wat je doel is met testen", "Houd rekening met 'false positives' en 'false negatives'" en "vermijd 'harde pauzes' bij het schrijven van softwaretesten". *Hiermee toon ik aan dat ik planmatig en volgens best practices heb gewerkt*. Naar aanleiding van de literatuurstudieresultaten heb ik een teststrategie ontworpen (bijlage 3, hoofdstuk 5). In de teststrategie maak ik duidelijk wat het doel is, welke omgeving en onderdelen er getest moeten worden en welke testsoort en testtool hiervoor gebruikt wordt. *Hiermee toon ik aan dat ik een testplan heb opgesteld*. De teststrategie is toegepast op de front-end van de Donation Marketing Journey editor. Aan de hand van de resultaten van de teststrategie (bijlage 3, hoofdstuk 6) is een conclusie getrokken (bijlage 3, hoofdstuk 7). Het onderzoeksrapport is afgesloten met een discussie waarin ik aanraad om End-to-End testing toe te passen (bijlage 3, hoofdstuk 8). *Hiermee toon ik aan dat ik een testplan heb geïmplementeerd*.

Het testproces

Het is in twee dagen natuurlijk onmogelijk om een hoge test-coverage te behalen, maar dit was ook niet het doel van de implementatie. Om mij wel te houden aan de best practices van test-coverage, ben ik Unit testen gaan schrijven voor de kernonderdelen van de front-end applicatie. *Hiermee toon ik aan dat ik rekening houd met test-coverage*. Met behulp van Jest was het mogelijk om in een korte tijd voor verschillende onderdelen van de applicatie Unit testen te schrijven. Door te beginnen met een "sanity test" kon ik achterhalen of de integratie met Jest succesvol was geslaagd. Hierna ben ik mij gaan focussen op de Utils class en ben ik de moeilijkheidsgraad steeds gaan verhogen. Het is uiteindelijk gelukt om stub data te retourneren bij asynchrone functies. Dit was het laatste onderdeel van de planning en toont aan dat de planning realistisch was opgesteld. *Hiermee toon ik aan dat ik Unit testen heb geprogrammeerd*.

11 Evaluatie

11.1 Resultaat product

Met trots kan ik melden dat het product voor 95% voldoet aan de verwachtingen. Het enige aspect dat nog niet volledig is geautomatiseerd, is het aanmaken van de "stap". De Product Owner heeft aangegeven dat dit geen probleem is en in een korte tijd kan worden gerealiseerd. Er zijn nog een groot aantal werkzaamheden die ik zelf wil verrichten om het product te verbeteren. Zo is het bijvoorbeeld door interne omstandigheden niet gelukt om het Activiteitensjabloon te implementeren. Dit is een onderdeel van de applicatie die de toevoeging en verkoopwaarde flink kan vergroten. Hiernaast zijn er aspecten zoals, het gebruiksvriendelijker maken van de interface met behulp van knoppen, het kunnen configureren of Acties eenmalig of meerdere keren worden verstuurd, het kunnen annuleren van een Journey, etc. Alle verbeteringen die Paddy, Robert en ik konden bedenken zijn toegevoegd aan de backlog. Deze taken pak ik graag op na het afstuderen om het product klaar te maken voor de verkoop.

Om het voor mij en andere developers zo overzichtelijk mogelijk te houden, heb ik mij gehouden aan verschillende richtlijnen. Voor versiebeheer heb ik gebruik gemaakt van Azure DevOps. Dit is de gebruikelijke werkwijze van Pixelzebra en zorgt ervoor dat alles herleidbaar is. Hiernaast heb ik de solution opgezet via de gewenste indeling van projecten en classes en maak ik gebruik van de juiste Nuget packages. De keuzes hiervoor zijn tot stand gekomen in overleg met Paddy van der Kaay. Ook tijdens het developen heb ik veel nagedacht over onderhoudbaarheid en leesbaarheid van de code. Het is niet altijd van belang om de code in zo min mogelijk regels te "proppen". Om deze reden maak ik bij voorkeur gebruik van een normale "foreach" (afbeelding 21) ten opzichte van een "Linq expression" (afbeelding 22). In het laatste voorbeeld is de Linq expression nog vrij duidelijk, maar dit wordt al snel onleesbaar als er meer "chaining" plaatsvindt. Chaining wordt gebruikt om meerdere functies achter elkaar aan te roepen in plaats van instanties aan te maken, bijvoorbeeld "campaigns.Where().Select().ToDictionary()".

```
public Dictionary<string, string> SpotlerCampaignsToDictionary(List<Campaign> campaigns)
{
    Dictionary<string, string> result = new Dictionary<string, string>();

    foreach (Campaign campaign in campaigns)
    {
        if (IsValidCampaign(campaign))
            result.Add(campaign.Triggers.Trigger.First().EncryptedId, campaign.Name);
    }

    return result;
}
```

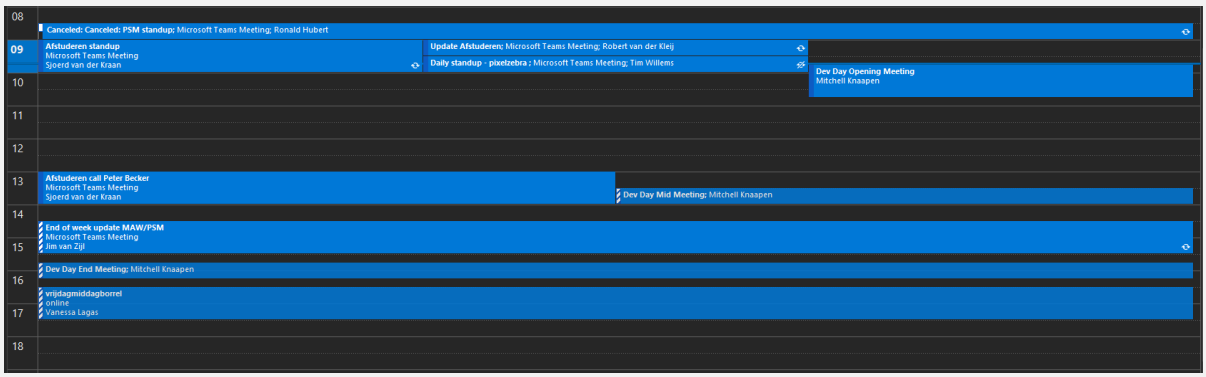
Afbeelding 21, Schrijven van code m.b.v. een "foreach"

```
public Dictionary<string, string> SpotlerCampaignsToDictionary(List<Campaign> campaigns)
{
    return campaigns
        .Where(IsValidCampaign) // IEnumerable<Campaign>
        .ToDictionary(campaign => campaign.Triggers.Trigger.First().EncryptedId, campaign => campaign.Name);
}
```

Afbeelding 22, Schrijven van code m.b.v. "Linq expression"

11.2 Algemeen

Ik vond het combineren van werk en afstuderen lastiger dan verwacht. Pixelzebra had mij de vrijdag gegeven om aan het afstuderen te zitten. Dit kon ik combineren met de donderdag, de dag dat ik officieel vrij ben, en de zaterdag en zondag. Achteraf bleek dat ik op de vrijdag vaak werkzaamheden opgedragen kreeg en soms zelfs op de donderdag (afbeelding 23). Dit werd grotendeels veroorzaakt door de Coronacrisis. Door de nieuwe werkwijze die gehanteerd moest worden, was het lastiger om te zien welke werkzaamheden er door collega's werden uitgevoerd. De "daily standup" zorgde ervoor dat hier meer inzicht in kwam, maar ik merkte alsnog dat mijn collega's en ik moeite hadden om een goed overzicht te creëren.



Afbeelding 23, Agenda overzicht van een afstudeerdag

Toen het vaker voorkwam dat collega's mij op de donderdag en vrijdag werkzaamheden opdroegen, ben ik strenger voor mijzelf en voor mijn collega's geworden. Ik heb aangegeven dat ik de tijd hard nodig heb en ben zodoende mijn Teams status op "Niet storen" gaan zetten. Op deze manier kreeg ik geen berichten binnen en kon ik mij beter focussen. Uiteraard heb ik wel aangegeven dat ik te bereiken was bij escalaties of belangrijke vragen. Mails daarentegen vond ik wel lastig om naast mij neer te leggen. Als er een mail binnenkomt wil ik altijd weten of deze belangrijk is, dus lees ik de mail vluchtig door. Dit kost over het algemeen niet veel tijd, maar haalt me toch uit mijn concentratie en geeft soms de behoefte om de mail te beantwoorden. Het is makkelijk om afgeleid te worden en af te dwalen. Dit was wel een leerproces waar ik veel uit heb gehaald en in de toekomst strenger in ga zijn.

Het maken van een backlog heeft mij geholpen bij het inzichtelijk krijgen en plannen van de werkzaamheden. Sprint 1 zorgde voor wat vertraging, omdat ik een belangrijk onderdeel niet werkend kreeg. Om deze reden ben ik in sprint 2 van het development proces dagen van 9 tot 9 gaan werken om de "verloren" tijd in te halen. Dit leek in het begin een goede oplossing, maar ik merkte al snel dat ik dit niet lang vol kon houden. Na anderhalve week raakte ik te vermoeid en kon ik mij slechter concentreren. Hierdoor presteerde ik minder goed op werk en boekte ik steeds minder vooruitgang met betrekking tot het afstuderen. In de tijd dat ik dit had volgehouden was ik gelukkig weer aardig op schema geraakt. Vanaf sprint 3 ben ik weer normale werktijden gaan aanhouden.

Het grootste probleem tijdens het development proces was mijn gebrek aan front-end kennis. Tot op heden zijn er bij Pixelzebra projecten gerealiseerd met behulp van React en Angular. Ik wilde graag gebruik maken van Vue.js, zodat we kunnen bepalen welk front-end framework het beste past bij Pixelzebra. Helaas heb ik van Vue.js af moeten stappen, omdat ik het niet voor elkaar kreeg om Vue.js te implementeren in Dynamics 365 CRM. Ik ben blij met de keuze die ik heb gemaakt aan het begin van sprint 2 om van Vue.js af te stappen en door te gaan met "vanilla" JavaScript. Het is lastig om in te

schatten of het toepassen van Vue.js verder in het project tijd zou besparen, als de implementatie uiteindelijk wel was gelukt. Als ik ben afgestudeerd ga ik een project starten om te oefenen met Vue.js.

Zoals ik in hoofdstuk 3.1 heb aangegeven heeft de Coronacrisis een impact gehad op het afstuderen. Zo was het bijvoorbeeld niet mogelijk om voor een klein vraagje even het kantoor binnen te stappen van Robert, Paddy of andere collega's. De vraag mailen of vragen via de Teams chat, zorgde vaak niet direct voor antwoord. Hierdoor moest er soms een meeting worden ingepland via Teams. Dit was een vrij lastige opgave aangezien de agenda's van mijn collega's veelal waren volgeboekt. Afgezien van bereikbaarheid heeft de Coronacrisis verder weinig impact gehad op het proces.

12 Literatuurlijst

1 Babel (2020). *@babel/plugin-proposal-class-properties · Babel*. Geraadpleegd op 19 maart 2020, van https://babeljs.io/docs/en/babel-plugin-proposal-class-properties

2 Babel (2020). *@babel/plugin-transform-async-to-generator · Babel*. Geraadpleegd op 19 maart 2020, van https://babeljs.io/docs/en/babel-plugin-transform-async-to-generator

3 Benders, L. (2020, 14 januari). *Je Plan van Aanpak (PvA) in een keer goedgekeurd*. Geraadpleegd op 20 februari 2020, van https://www.scribbr.nl/scriptie-structuur/plan-van-aanpak/

4 Bos, C. (2016, 7 juli). *7 Tips voor voorzitters | Vergaderen doe je zo*. Geraadpleegd op 22 februari 2020, van http://www.vergaderendoejezo.nl/7-tips-voor-voorzitters

5 Bright, P. (2012, 3 oktober). Geraadpleegd op 13 april 2020, van Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem?

6 Cohn, M. (2010), *Succeeding with Agile: Software Development using Scrum*. Addison-Wesley, Upper Saddle River, p. 257–284. ISBN 978-0-321-57936-2.

7 Coplien, J. (2010), *Lean Architecture for Agile Software Development*. Wiley, Chichester Hoboken, N.J, p. 25. ISBN 978-0-470-68420-7.

8 Doll, R. (2018, 30 maart). *Jest and URL Mocking*. Geraadpleegd op 17 mei 2020, van https://www.ryandoll.com/post/2018/3/29/jest-and-url-mocking

9 Draw.io. *Diagrams - G Suite Marketplace*. (z.d.). Geraadpleegd op 9 maart 2020, van https://gsuite.google.com/marketplace/app/drawio_diagrams/671128082532

10 Flux. (2019, 5 juli). Geraadpleegd op 7 april 2020, van https://facebook.github.io/flux/docs/in-depth-overview/

11 Fowler, M. (2012, 1 mei). *Bliki: TestPyramid*. Geraadpleegd op 15 mei 2020, van https://martinfowler.com/bliki/TestPyramid.html

12 Gowtham, K. (2019, 20 januari). *Handling CORS Policy In Azure Functions*. Geraadpleegd op 3 maart 2020, van https://www.c-sharpcorner.com/article/handling-cors-in-azure-function/

13 Hobbs, S. (2019, 16 april). *CORS Tutorial: A Guide to Cross-Origin Resource Sharing*. (2019, 16 april). Geraadpleegd op 03 maart 2020, van https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/

14 ING. *Wat brengt 2019 voor goede doelen en gevers?* (2019). Geraadpleegd van http://tiny.cc/goededoelen2019

15 Ismail, K. (2018, 6 juli). *Agile vs Scrum vs Kanban Weighing the Differences*. Geraadpleegd op 28 februari 2020, van https://www.cmswire.com/information-management/agile-vs-scrum-vs-kanban-weighing-the-differences/

16 Leffingwell, D. (2011), *Agile Software Requirements: Lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley, Upper Saddle River, p. 15. ISBN 978-0-321-63584-6.

17 Levi, S. (2018, 26 mei). *30 Answers from Stack Overflow to the most popular webpack questions*. Geraadpleegd op 3 maart 2020, van https://medium.com/wizardnet972/30-answers-from-stack-overflow-to-the-most-popular-webpack-questions-49980770d5dc

18 McKenzie, S. C. N. (2018, 26 juni). *Yarn: A new package manager for JavaScript*. Geraadpleegd op 27 februari 2020, van https://engineering.fb.com/web/yarn-a-new-package-manager-for-javascript/

19 MK (2018, 24 december) *Building Dynamics 365 Web Resources with Vue JS and NPM*. (2018, 24 december). Geraadpleegd op 4 maart 2020, van https://www.dynamicsforcrm.com/building-dynamics-365-web-resources-with-vue-js-and-npm/

20 Pixelzebra. *Wie zijn wij*. (2019, 24 april). Geraadpleegd van https://www.pixelzebra.nl/wie-zijn-wij/

21 Pixezelbra Social Matters. *Social Matters, gericht is op goede doelen en cultuur*. (2019, 19 augustus). Geraadpleegd van https://www.pixelzebra.nl/social-matters/

22 Sammi, R.; Masood, I; Jabeen, S. (2011). *A Framework to Assure the Quality of Sanity Check Process*. Software Engineering and Computer Systems. Communications in Computer and Information Science. Berlin, Heidelberg: Springer. 181: 143–150.

23 Schriel, P. (2009, 25 juni). *10 stappen voor een succesvolle vergadering*. (2017, 22 juli). Geraadpleegd op 22 februari 2020, van https://patrickschriel.nl/2009/06/25/10-stappen-voor-een-succesvolle-vergadering/

24 Schwaber, K. (2009), *Agile Project Management with Scrum*. O'Reilly Media, Inc. ISBN 978-0-7356-3790-0.

25 Simens, J. (2017, 28 februari). *What is the Dynamics 365 Common Data Model?* Geraadpleegd op 28 mei 2020, van https://us.hitachi-solutions.com/blog/what-is-the-dynamics-365-common-data-model/

26 The State of JavaScript 2019: *Testing*. (2019, 1 januari). Geraadpleegd op 7 mei 2020, van https://2019.stateofjs.com/testing/

27 Valori. (2018, 1 maart). *Blogserie geautomatiseerd testen deel 5: De testpiramide*. Geraadpleegd op 9 mei 2020, van https://www.valori.nl/blogs/blogserie-geautomatiseerd-testen-deel-5-de-testpiramide

28 visual-paradigm. (2020, 12 februari). *How to Draw ArchiMate 3.0.1 Diagrams?* Geraadpleegd op 28 mei 2020, van https://www.visual-paradigm.com/support/documents/vpuserguide/4455/4409/86421_howtodrawarc.html

29 Wang, Z. (2018, 9 juli). *How to make Xrm.WebApi call Synchronous in v9.0?* - Dynamics 365 Customer Service Forum Community Forum. Geraadpleegd op 18 maart 2020, van https://community.dynamics.com/365/customerservice/f/dynamics-365-for-customer-service-forum/288153/how-to-make-xrm-webapi-call-synchronous-in-v9-0

30 Wegink, B. (2019, 1 november). *E-mailmarketing: Spotler*. Geraadpleegd op 9 maart 2020, van https://www.happyidiots.nl/specialismes/email-marketing/spotler

13 Bijlagen

1 Plan van aanpak - Sjoerd van der Kraan

2 Requirements analyse - Sjoerd van der Kraan

3 Onderzoeksrapport - Sjoerd van der Kraan

4 Handleiding en adviesrapport - Sjoerd van der Kraan