Bitman

Afstudeerscriptie

Het ontwikkelen van een universele troubleshooter

Quang Hang

Hoofddorp – 4 juni 2021

Versiebeheer

Datum	Versie	Opmerking
8-2-2021	0.0.1	Initiële versie
10-2-2021	0.0.2	Bedrijfsomschrijving Bitman en bedrijfsoriëntatie toegevoegd.
25-2-2021	0.0.3	Probleemstelling en aanpak toegevoegd.
10-3-2021	0.0.4	Ontwikkelmethode en algemene verantwoording toegevoegd.
28-4-2021	0.0.5	Concept feedback verwerkt
6-5-2021	0.1	Oplevering voor het tussentijds assessment.
4-6-2021	1.0	Feedback tussentijds assessment verwerkt en ontbrekende onderdelen beschreven en toegevoegd.

Contactgegevens

Anton Waslander

Bedrijfsmentor

✉

Arnold Gijzen

Technisch begeleider

✉

Anneke Wieman

Begeleidend examinator

✉

Victor Huisman

Expert examinator

✉

Quang Hang

Afstudeerder

✉

Voorwoord

Deze scriptie is opgesteld door Quang Hang, een vierdejaars HBO-ICT student aan De Haagse Hogeschool. Het afstudeertraject is het in het kader van Software Engineering uitgevoerd bij Bitman BV te Haarlem. Dit document zal inzicht geven in hoe het eindproduct tot stand is gekomen op basis van analyse, ontwerp en realisatie.

Het afstudeertraject is afgelegd tijdens de periode van 8 februari tot en met 4 juni 2021. De werkzaamheden werden bij Bitman te Haarlem afgelegd van 1 februari tot en met 28 mei 2021. Ik was een week eerder begonnen bij Bitman dan de uiterlijke startdatum voor het afstudeertraject. Dit zorgde ervoor dat ik aan het eind iets meer speling had voor het werken aan het afstudeerdossier. Dit was een bewuste keuze en vooral met de bijzondere situatie tijdens de pandemie leek het mij handig om met allerlei soorten scenario's rekening te houden. Zo kon het zijn dat er vanwege eventuele ziekte en quarantaine er belemmeringen zouden optreden.

Ik heb mijzelf verder kunnen ontwikkelen in de sector van web development. Gedurende het traject heb ik een proof of concept kunnen opleveren en de daarbij komende deelproducten. Het was een leerzame en uitdagende opdracht en het thuiswerken maakte dat voor mij net wat lastiger. Desondanks was ik enthousiast om een oplossing voor Bitman te kunnen bieden die zij zouden kunnen gebruiken om in de toekomst op verder te bouwen.

Ik wil de bedrijfsmentor, de technische begeleider, alle collega's en mijn begeleiders vanuit de opleiding bedanken voor de begeleiding en ondersteuning die ze mij geboden hebben. Het zijn de volgende personen:

Anton Waslander: De eigenaar van Bitman BV, mijn bedrijfsmentor en opdrachtgever. Hij is verantwoordelijk geweest voor het aannemen van mij als afstudeerder. Bovendien had hij de afstudeeropdracht opgesteld en heb ik samen met hem alle zaken geregeld met betrekking tot de afstudeerovereenkomst.

Arnold Gijzen: Software ontwikkelaar bij Bitman. Hij was mijn technische begeleider en heeft mij bij technische vraagstukken hulp gegeven. Hij heeft mijn functioneel en technisch ontwerp bekeken en daar feedback op gegeven. Ook voor inhoudelijke vragen over de troubleshooter kon ik bij Arnold terecht.

Stephano Hondeveld: Software ontwikkelaar bij Bitman. Ik ben via Stephano bij Bitman terechtgekomen. Ik ben hem zeer dankbaar voor het introduceren van mij aan Anton. Tevens was hij beschikbaar voor technische vraagstukken waardoor hij gezien kon worden als mijn tweede technische begeleider.

Anneke Wieman: De begeleidende examinator vanuit De Haagse Hogeschool. Zij was heeft mij goede feedback gegeven op het concept van het afstudeerdossier. Zij komt meer vanuit de communicatie wereld waarbij zij goed verslagen kan beoordelen. Deze feedback kon ik verwerken ten behoeve voor het tussentijds assessment. Bovendien was zij aanwezig voor het bedrijfsbezoek samen met Anton en mij. Vanwege de coronapandemie vond het bedrijfsbezoek online plaats via Microsoft Teams.

Victor Huisman: De expert begeleider vanuit De Haagse Hogeschool. Hij heeft mij bewust gemaakt van eventuele vragen die door de extern gecommiteerde gesteld zouden kunnen worden tijdens de examenzitting. Daarnaast had hij een meer technische blik op het afstudeerdossier en daar gaf hij feedback over bij het tussentijds assessment.

Inhoudsopgave

1. Inleiding

Het afstudeertraject wordt bij Bitman BV te Haarlem afgelegd. Wat voor bedrijf dit is zal te lezen zijn in hoofdstuk 2: Bitman. Uit het introductiegesprek met Anton Waslander is gebleken dat ze een nieuwe helpdesktool willen bouwen. De afstudeeropdracht zal betrekking hebben op deze tool die momenteel door de klantenservice van T-mobile wordt gebruikt. Het huidige systeem is gebouwd in een oud framework. Het probleem hierbij is dat er geen updates meer voor worden uitgebracht wat ertoe leidt dat het uitbreiden van het systeem lastiger wordt.

De kern van de afstudeeropdracht is dat er een deel van het systeem helemaal vanaf het begin gebouwd gaat worden in een modern framework. De focus ligt hierbij op de troubleshooter met de decision tree[1]. Deze is onderdeel van de troubleshooter waar callcentermedewerkers mee werken. Het begeleidt de medewerker door een stappenplan heen om zo de klant van dienst te kunnen zijn. Het is de bedoeling dat het systeem door verschillende organisaties gebruikt kan gaan worden, zodat Bitman het aan toekomstige klanten kan gaan verkopen. Het gaat hierbij om organisaties die gebruikmaken van een klantenservice, denk hierbij aan zorgverzekeraars, online shops, banken, IT-diensten en onderwijs.

Er zal tijdens het afstuderen onderzoek gedaan worden naar de werking van de huidige troubleshooter, of er bestaande decision tree oplossing software bestaat en of er componenten van het huidige systeem als basis gebruikt kunnen worden. De onderzoeksmethodieken die toegepast zullen worden zijn deskresearch, inventarisatieonderzoek en toegepast onderzoek. Het gehele onderzoek betreft een kwalitatief onderzoek. Er zullen een hoofdvraag en deelvragen worden opgesteld. De antwoorden op deze vragen zullen samen de conclusie vormen.

Alles wat in de scriptie beschreven wordt is op basis van de bijlagen en rapporten die tijdens het afstudeertraject gemaakt zijn. De belangrijkste aandachtspunten hierbij zijn:

- Aantoning van beroepstaken
- Verantwoordelijkheden binnen het project
- De gemaakte keuzes en genomen beslissingen
- Evaluatie en reflectie

1.1 Aanleiding

Bitman heeft in 2012 voor KPN een “knowledge base” systeem[2] ontwikkeld voor de toenmalige glasvezelafdeling. Dit systeem werd gebruikt om consumenten support te kunnen leveren bij aanschaf en problemen met hun glasvezelaansluiting. Het is opgezet om diverse glasvezelaanbieders te kunnen ondersteunen. Een van de merken die KPN ondersteunde was Vodafone. Na de breuk met KPN in 2012 heeft Vodafone het systeem overgenomen. Dit is wederom gebeurd in 2018 na de overname van Vodafone Vast door T-Mobile. Op dit moment werken er diverse callcenters van T-mobile met dit systeem. Elke supportmedewerker van T-Mobile Vast werkt met deze knowledge base.

Het huidige systeem is opgezet met het PHP Fusebox framework[3], een oud framework dat niet meer geüpdatet wordt en vervanging is dus nodig. De kracht van het systeem is de ‘decision tree’ structuur gecombineerd met een op Elastic[4] ontwikkelde zoekfunctionaliteit. Het doel bij het opzetten van het systeem was het kunnen behouden en uitbreiden van kennis binnen de supportafdeling van de verschillende internet serviceproviders. Vaak zit er veel kennis bij bepaalde

[1] Guided Decisions for Customer Service Management
[2] Een kennissysteem waarbij de kennis van ervaren supportmedewerkers gedeeld kan worden met minder ervaren supportmedewerkers.
[3] PHP Fusebox
[4] What is Elasticsearch?

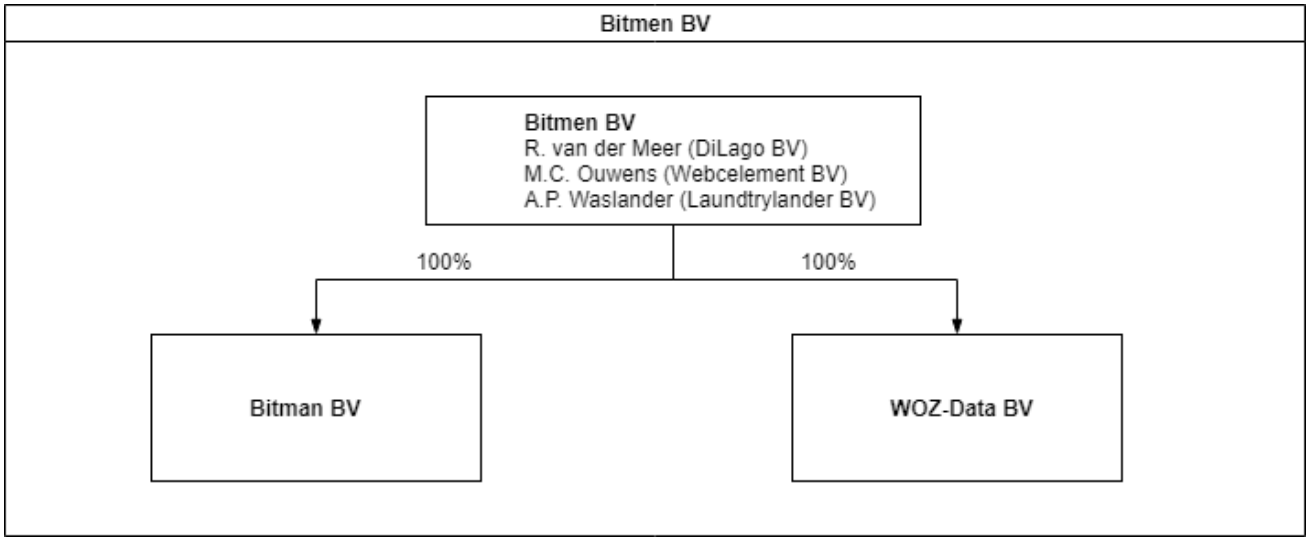
werknemers, maar is het verloop ook hoog. Het vasthouden van deze kennis is de kern van het systeem. Dit maakt het dan ook geschikt voor andere organisaties die afhankelijk zijn van support. Denk hierbij aan webwinkels, energiebedrijven, verzekeringsmaatschappijen, medische organisaties etc. Het systeem kan zowel kleine als grote organisaties helpen de kennis te behouden en op een makkelijke manier te ontsluiten.

2. Bitman

Bitman is een webbureau dat in 2003 is opgericht en is gevestigd aan de Houtmarkt te Haarlem. Waar ze gespecialiseerd in zijn is het maken van maatwerk en technisch complexe webapplicaties. Een aantal kenmerken waar Bitman zich mee onderscheidt is ElasticSearch[5], datavisualisatie, maatwerk en TrafficTower CMS. Het laatste, TrafficTower CMS, is een contentmanagementsysteem[6] dat door Bitman[7] zelf is ontwikkeld. De klanten variëren van telecombedrijven zoals KPN, Vodafone & T-mobile tot gemeenten. Momenteel zijn er negen medewerkers werkzaam inclusief het management.

2.1 Organisatiestructuur

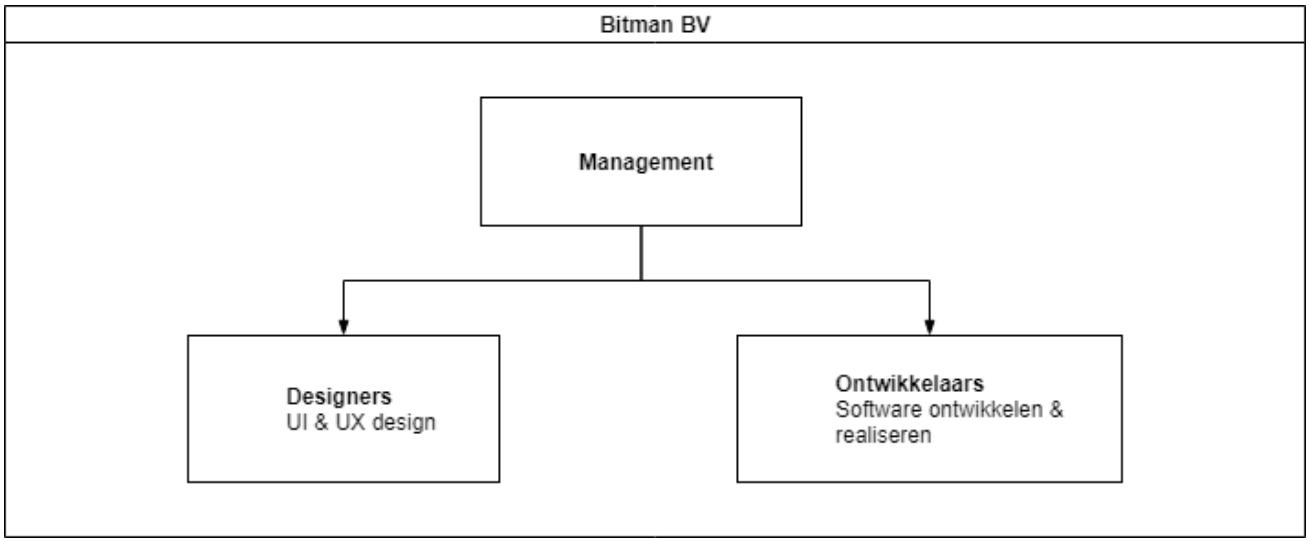
Bitman maakt onderdeel uit van Bitmen BV en de eigendomsverhoudingen hiervan zijn verdeeld over Maarten, Rob en Anton (zie onderstaande organogram). Ieder van hen heeft een eigendom van 33,3 procent van Bitmen. Bitmen BV bezit 100 procent van zowel Bitman BV als van WOZ-Data BV.



Figuur 1: Organogram Bitmen BV

Binnen Bitman zelf wordt gebruik gemaakt van de platte organisatiestructuur waarbij er een informele sfeer heerst. Er is dus geen sprake van verschillende lagen binnen het bedrijf. Het management, bestaande uit Anton, Maarten en Rob, heeft direct contact met de rest van de medewerkers. Dit is in onderstaande organogram weergegeven.

[5] What is Elasticsearch?
[6] Content Management System
[7] Over Bitman



Figuur 2: Organogram Bitman BV

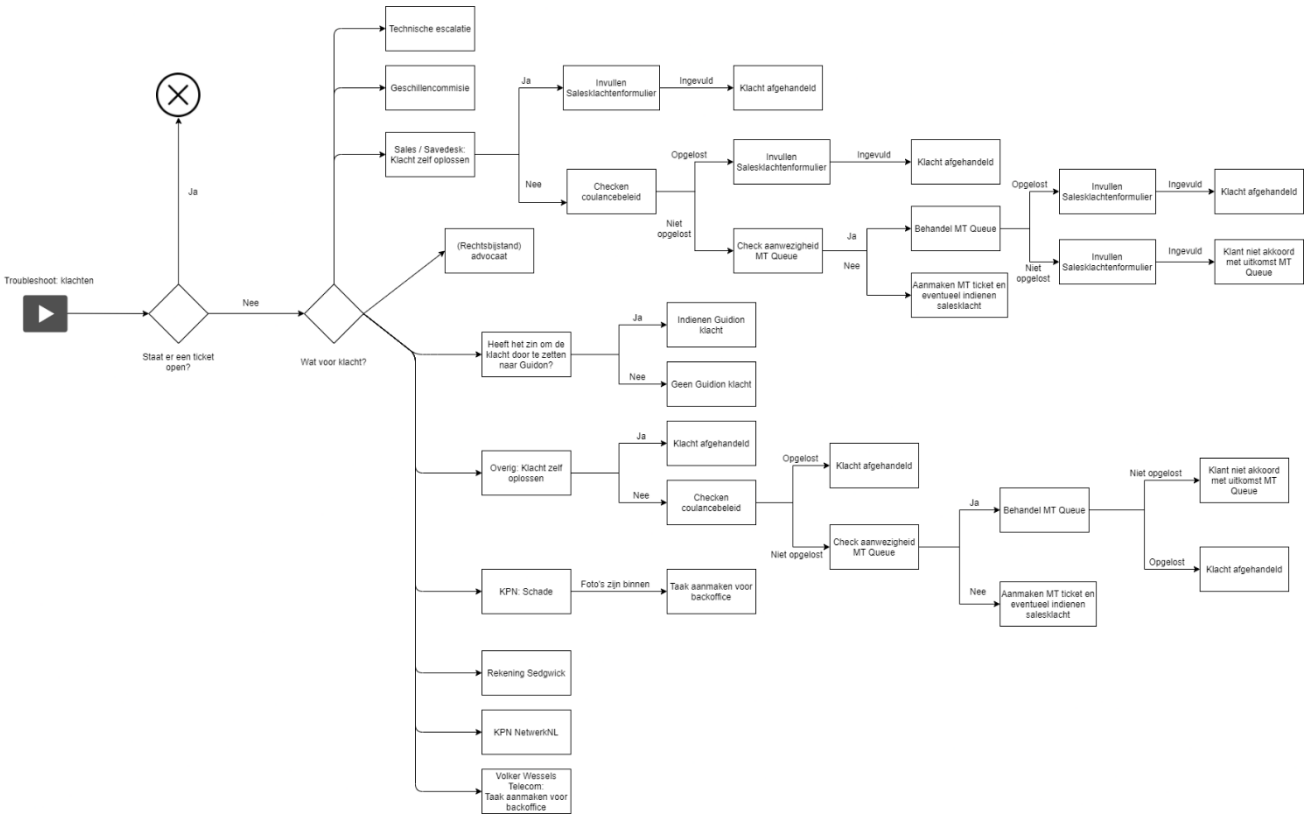
3. Afstudeeropdracht

3.1 Probleemstelling

Bitman heeft in 2012 een knowledge based system ontwikkeld wat momenteel door T-Mobile gebruikt wordt. De decision tree opzet is de kern van het systeem. Het huidige systeem is opgezet in PHP Fusebox, een verouderd framework waar geen updates meer voor worden uitgebracht. Dit heeft als gevolg dat het lastig is om het systeem uit te breiden.

Naast het knowledge based system is er ook een Business Intelligence (BI) component ontwikkeld. Hierin wordt bijgehouden welke problemen op welk punt in de decision tree worden opgelost of niet. Op basis van deze data kan men daarop gaan sturen. Het BI-systeem is in CodeIgniter 3[8] ontwikkeld.

Beide applicaties zullen vervangen moeten worden door één systeem. Voor de afstudeeropdracht valt het BI-systeem buiten de scope. De kern van de opdracht is om een klein deel van het systeem te bouwen en het is fullstack, wat wilt zeggen dat de database, front-end en backend aan bod komen. De decision tree is de belangrijkste functionaliteit die in het nieuwe systeem moet zitten. Het is een datastructuur dat bestaat uit een reeks mogelijkheden. Bij elke stap kan de medewerker bepaalde keuzes maken op basis van het gesprek met de klant. Elke keuzemogelijkheid kan tot nieuwe vertakkingen leiden binnen de beslissingsboom en het eindigt wanneer het probleem opgelost is. Onderstaande figuur is een visualisatie van een decision tree.



Figuur 3: Visualisatie van de decision tree

De comptabiliteit met het huidige systeem is van groot belang. In eerste instantie zal het oude systeem vervangen worden door het nieuwe systeem. Het is hierbij van belang dat de data van de huidige decision trees en informatie een op een beschikbaar moeten zijn. De interface mag gemoderniseerd worden, maar de data dient geheel intact te blijven. De gewenste situatie is dus dat er een nieuw systeem wordt gebouwd dat aan een aantal eisen moet voldoen. De troubleshooter

[8] Code Igniter

zal uiteindelijk door organisaties gebruikt worden die een klantenservice hebben en met name de supportmedewerkers zullen er gebruik van maken.

Om dit in zo goed mogelijke banen te leiden zal ik onderzoek doen, requirements vergaren door geschikte elicitatietechnieken toe te passen, een functioneel en technisch ontwerp maken en kritisch zijn op de keuze uit programmeertalen en frameworks.

3.2 Doelstelling

Het doel wat Bitman voor ogen heeft is om dit systeem te kunnen verkopen aan toekomstige klanten die gebruikmaken van een klantenservice. Denk hierbij aan IT-diensten, verzekeraars, de zorg, online shops, banken en het onderwijs. Bij het ontwikkelen van het systeem moet dus rekening gehouden worden met hoe het gebruikt kan gaan worden door meerdere organisaties. Wat mogelijk zou kunnen zijn is om één backend te bouwen voor meerdere klanten of per klant een aparte backend. Gezien de tijd van het afstudeertraject en de scope zal één backend voor de hand liggen.

4. Aanpak

In dit hoofdstuk zal de aanpak ten behoeve van het uitvoeren van het afstudeertraject besproken worden. Het betreft de planning over de gehele afstudeerperiode en deze is van tevoren opgesteld. Het doel van deze planning was om een inzicht te creëren voor het gehele traject om zo gericht mogelijk te kunnen werken. Aangezien de planning van tevoren is opgesteld is het goed mogelijk dat er wijzigingen en/of afwijkingen optreden tijdens het uitvoeren van het afstudeertraject. Wanneer deze wijzingen en afwijkingen optreden zal ik deze beschrijven en beargumenteren wat de reden hiervan is.

4.1 Fases

Het afstudeertraject heb ik ingericht in een aantal fases. Door dit te doen heb ik een goed overzicht over wat er allemaal te doen staat binnen het afstudeertraject.

Voorbereidings-en onderzoeksfase (+/- 20 dagen)

Als eerst zal er een *plan van aanpak* gemaakt worden, hierin wordt beschreven welke stappen er gemaakt zullen worden, hoelang deze duren en wat de globale planning van het traject gaat zijn. Zo weten beide partijen (de afstudeerder en opdrachtgever) wat er globaal te verwachten is tijdens het afstudeertraject. Na het maken van het plan van aanpak komt het oriëntatiegedeelte. Deze bestaat uit de verkenning van het huidige systeem, het opzetten van de ontwikkelomgeving, het onderzoek en het achterhalen van de requirements.

Na de verkenning van het huidige systeem komt het onderzoek waarin gekeken wordt naar het bestaande systeem en hoe deze werkt, bestaande decision tree software, wat de requirements zijn en of er componenten als basis gebruikt kunnen worden voor de te bouwen applicatie. Alle details hiervan zijn terug te vinden in het onderzoeksrapport. Vervolgens wordt het *functioneel ontwerp* gemaakt waarin onder andere de requirements, use cases en een stakeholderanalyse staan. Dit ontwerp zal tevens voor Bitman een deelproduct zijn. Voor de prioritering van de requirements wordt de MoSCoW[9]-methodiek toegepast.

Ontwerpfase (+/- 25 dagen)

Op basis van de verkregen requirements en stakeholderanalyse kunnen de use cases en user stories opgesteld worden. Deze zijn zoals eerder genoemd terug te vinden in het functioneel ontwerp. Het tweede deelproduct wat opgeleverd gaat worden is het *technisch ontwerp* waarin zaken staan zoals de infrastructuur van de software, de database en de gebruikersinterface. Er zijn door de vorige ontwikkelaars geen ontwerpen gemaakt van het huidige systeem. Er zal een klein deel gedocumenteerd worden en voor het grote deel worden er ontwerpen door mij gemaakt. Hoe dit zich tot elkaar verhoudt zal besproken worden in ‘**Ontwerpfase’**. Goede documentatie en ontwerpen van de software leidt immers tot betere onderhoudbaarheid waar toekomstige ontwikkelaars ook baat bij zullen hebben.

Ontwikkelfase (+/- 30 dagen)

Tijdens deze fase wordt de applicatie ontwikkeld waarbij de gemaakte ontwerpen in de praktijk toegepast worden. Hierbij ligt de focus op de functionaliteit van de troubleshooter met de decision trees. Bij de ontwikkelmethode zullen SCRUM[10]-elementen toegepast worden, waarbij Asana[11] de tool is waar alles in vastgelegd zal worden. Bij het programmeren zal gebruikt gemaakt worden van het Model-View-Controller Pattern[12] (later afgekort als MVC) in combinatie met het PHP Laravel

[9] Swart, N. (2017). Handboek Requirements. Reed Business Education. blz. 227
[10] Randen, H. J. (2015). Aan de slag met Scrum (1ste ed.). Academic Service, blz. 13
[11] Asana
[12] MVC pattern

framework[13]. Daarnaast zal er ontwikkeld worden met VUEjs[14], HTML/CSS[15], MySQL[16] en Vagrant[17]. Voor het versiebeheer zullen Git Bash[18] en SourceTree[19] in combinatie met Bitbucket[20] gebruikt worden.

Afrondingsfase (+/- 10 dagen)

Deze fase zal plaatsvinden na het tussentijdse assessment waar ik feedback van mijn examinatoren krijs op het gehele afstudeerdossier. Hierbij zullen de werkzaamheden ten behoeve van Bitman afgeschaald worden waarbij de prioritering naar het afstudeerdossier zal verschuiven.

4.2 Ontwikkelmethode

Tijdens het uitvoeren van het afstudeertraject zal er met twee methoden gewerkt gaan worden. Het betreft software-ontwikkelmethoden zoals waterval en SCRUM[21]. Tijdens de eerste twee fases (voorbereiding en ontwerp) zoals ik heb gedefinieerd in hoofdstuk 4.1, maak ik gebruik van de watervalmethode. Voor de implementatiefase maak ik gebruik van SCRUM-elementen, hierbij gebruik ik de Asana tool. Binnen deze tool worden categorieën aangemaakt, denk hierbij aan Backlog, Upcoming, In Progress, Review en Complete.

Bij het plannen, designen en achterhalen van de requirements wordt het principe van de watervalmethode[22] gebruikt. Waterval maakt gebruik van een aantal fasen[23], zoals analyse, basisontwerp, technisch ontwerp, bouwen en testen. Een aantal fasen die ik gedefinieerd heb maken onderdeel uit van de watervalfasen. Het betreft de voorbereidings-en ontwerpfases, want binnen deze fasen vinden werkzaamheden ten behoeve van de analyse, het onderzoek, het functioneel en technisch ontwerp plaats.

Bij de ontwikkelfase zal er deels volgens de SCRUM-methodiek gewerkt gaan worden. De ontwikkeling zal opgedeeld worden in twee wekelijkse sprints waarbij er na afloop een review zal plaatsvinden. Vervolgens vindt de retrospective plaats waarin gekeken wordt welke verbeteringen er mogelijk meegenomen kunnen worden voor de volgende sprint. Echter worden er geen daily standups gehouden, maar in plaats daarvan wordt er wekelijks besproken wat de voortgang is. In Asana worden taken in een bepaalde categorie geplaatst om een overzicht te houden over alle taken. In feite wordt tijdens het gehele afstudeertraject een deel volgens waterval en het andere deel met SCRUM gewerkt.

4.2.1 Verantwoording

Er is specifiek gekozen om met zowel waterval als SCRUM te werken. Ten eerste is er door eerdere ontwikkelaars van de huidige applicatie niets gedocumenteerd, waardoor het lastiger is voor toekomstige ontwikkelaars om voort te bouwen op het huidige systeem of in dit geval om een nieuw systeem vanaf begin af aan te bouwen. Dankzij het gebrek aan documentatie en ontwerpen is het duidelijk dat er een functioneel en technisch ontwerp gemaakt moeten worden. Het is namelijk ontzettend belangrijk om een goed ontwerp te maken van de te bouwen applicatie, zodat er een goede basis is om op te gaan ontwikkelen. De basisstructuur van de documentatie staat

[13] Laravel Framework
[14] VueJS
[15] HTML CSS
[16] MySQL
[17] The Vagrant Box
[18] Git Bash
[19] SourceTree
[20] Bitbucket
[21] Randen, H. J. (2015). Aan de slag met Scrum (1ste ed.). Academic Service. blz 13
[22] Software-ontwikkeling: welke methoden zijn er en waarin verschillen ze?
[23] Waterfall vs. Agile Methodology

grotendeels vast waar de watervalmethode een goed middel voor is. Tevens is het van belang dat elke fase afgerond is voordat er aan de volgende begonnen wordt.

Ten tweede is het niet alleen van belang dat het systeem in een modern framework gebouwd gaat worden, maar ook dat deze uiteindelijk door verschillende soorten organisaties te gebruiken is. Het huidige systeem is sterk gericht op de werkwijze bij T-Mobile wat ervoor zorgt dat het niet uitbreidbaar is naar andere klanten toe van Bitman. Er zal dus een basis applicatie gebouwd moeten gaan worden waarbij de kern zo in elkaar zit dat deze gemakkelijk uit te breiden is voor toekomstige Bitman-klanten. Hierbij zullen de requirements aan de applicatie enigszins afwijken van de bestaande applicatie, waardoor het werken met elementen van SCRUM een goede keuze is. De applicatie wordt dan stap voor stap opgebouwd. Bovendien vind ik het belangrijk om met tweewekelijkse sprints in de ontwikkelfase te werken, zodat ik met deadlines werk, waardoor ik de tijd niet uit het oog verlies.

Al met al zal er volgens de watervalmethode gewerkt worden tijdens de eerste twee fases zoals ik die benoemd heb in paragraaf 4.1, de voorbereidings-en onderzoekfase en de ontwerpfase, en met SCRUM-elementen in de ontwikkelfase. De analyse, basisontwerp (functioneel) en technisch ontwerp zijn feitelijk de fasen van waterval die ik gebruik en voor het bouwen van de applicatie maak ik gebruik van SCRUM.

4.3 Programmeertalen & frameworks

In dit gedeelte zal het gaan over waarom er gekozen is om de afstudeeropdracht uit te voeren met hulp van het Laravel framework. Er zullen meerdere programmeertalen onderzocht worden met de voor-en nadelen van elke taal. Op basis van deze voor-en nadelen, met inachtneming van de bedrijfssituatie bij Bitman, zal een weloverwogen afweging gemaakt worden. Voor een overzicht van de voorwaarden die ik opgesteld heb waar deze talen aan moeten voldoen verwijs ik naar paragraaf 4.4 ‘**Overzicht keuze programmeertaal**’.

Voor het bouwen van de applicatie is er een verscheidenheid aan programmeertalen die gebruikt zouden kunnen worden. Een aantal talen waaraan gedacht zou kunnen worden zijn C#, C++, Java, Python en PHP. Uit de verkenning tijdens de voorbereidings-en onderzoeksfase blijkt dat de helpdesktool een webapplicatie betreft, hier zal dus rekening mee gehouden worden bij de keuze voor een programmeertaal.

4.3.1 C#

C# is een moderne programmeertaal die object georiënteerd en type-safe is[24]. Dankzij het .NET ecosysteem kunnen ontwikkelaars veel soorten veilige en robuuste applicaties ontwikkelen die hierop draaien. Daarnaast is .NET, samen met ASP.NET open-source waar een grote community achter zit. De oorsprong van C# zit in de C-familie van programmeertalen en zal voor ontwikkelaars die bekend zijn met C, C++, Java en JavaScript snel op te pakken zijn dankzij de gelijkenissen.

Bij C# wordt het ASP.net framework toegepast bij het ontwikkelen van webapplicaties. Ten eerste, een van de grootste voordelen van ASP.net is dat het flexibel is en object-georiënteerde kenmerken heeft. Tegenwoordig ondersteunt het framework een naadloze integratie met verschillende programmeertalen zoals Visual Basic .NET en C#. Het houdt in dat er tijdens een project van programmeertaal veranderd kan worden zonder dat alles herschreven hoeft te worden. Tevens kunnen meerdere ontwikkelaars aan hetzelfde project werken en onderling andere talen gebruiken, zoals Visual Basic.NET en C#.

Ten tweede is dat de library van ASP.NET gebaseerd is op taken. Ontwikkelaars kunnen tijd besparen op veelvoorkomende development taken, doordat de library zo georganiseerd is dat een

[24] A tour of the C# language

bepaalde taak overervende classes bevat. Bovendien bevat de bijbehorende IDE[25] (Visual Studio) IntelliSense, een automatische aanvullingsfunctie dat ervoor zorgt dat ontwikkelaars methodenamen en variabelen niet uit hun hoofd hoeven te kennen.

Een van de grootste nadelen van ASP.NET en Visual Studio is dat ze duur zijn en het onderhoud veel resources kost. Daarnaast maakt ASP.NET veel meer gebruik van webserver resources dan nadere programmeertalen. Dit zorgt ervoor dat er meer geïnvesteerd moet worden in betere servers of dat er meer servers aangeschaft moeten worden.

Bovendien is C# onderdeel van het .NET framework, waardoor de server waar de applicatie op draait Windows als besturingssysteem moet hebben. Kortom het houdt in dat een .NET applicatie een Windows platform nodig heeft om te kunnen draaien. Veel bedrijven werken met Linux servers, omdat deze goedkoper zijn dan Windows servers. De afhankelijkheid van Windows kan, afhankelijk van de situatie, dus een groot nadeel zijn.

4.3.2 C++

C++ is een programmeertaal dat als extensie op de C-taal is ontstaan is. Over verloop van tijd heeft de taal veel ontwikkelingen meegemaakt en tegenwoordig is het object georiënteerd. C++ draait op verschillende platformen, waaronder Windows, MAC OS en Unix[26].

Ten eerste is een voordeel van C++ draagbaarheid, dat wil zeggen dat het platform onafhankelijk is. De gebruiker kan de applicatie draaien op verschillende besturingssystemen of interfaces. Ook als programmeur kan het een goede uitkomst bieden. Stel voor dat de ontwikkelaar van Windows naar Linux overstapt, dan kan de applicatie ook daar nog steeds draaien.

Ten tweede is C++ object georiënteerd waarbij elementen zoals classes[27], overerving[27], polymorfisme[27], abstractie en encapsulatie[28] van pas komen. Deze zorgen ervoor dat de code herbruikbaar is en dat de applicatie betrouwbaar is.

Ten derde maakt C++ gebruik van het multi-paradigma-principe. In dit geval verwijst paradigma naar de manier van programmeren, waarbij verschillende programmeertechnieken gebruikt kunnen worden. Het gaat hierbij om de logica, structuur en procedure van het programma.

Ten vierde geeft C++ de verantwoordelijkheid van het memory management aan de ontwikkelaar. Het maakt gebruik van een concept van Dynamic Memory Allocation[29] waarbij pointers gebruikt worden. Dit kan echter ook een nadeel zijn, omdat de ontwikkelaars verantwoordelijk is voor het memory management en niet een Garbage Collector.

Allereerst is een van de nadelen van C++ dat er geen ingebouwde Garbage Collector is om automatisch overbodige data te verwijderen. De ontwikkelaar heeft volledige verantwoordelijkheid voor het memory management. Door eventuele onopgemerkte bugs in het programma zou dit kunnen leiden tot *memory leaks* wat potentieel kan leiden tot het vastlopen van het hele systeem.

[25] Integrated Development Environment
[26] C++ Tutorial - Tutorialspoint. (z.d.). C++ Tutorial - Tutorialspoint
[27] H-SE-OOPR-1-15-2017: 2017 Object orientation, Den Haag P3, Programming and Testing (studiemateriaal BlackBoard)
[28] Abstraction and Encapsulation
[29] What is Dynamic Memory Allocation?

Verder maakt C++ gebruik van pointers[30] om waardes van een variabele in op te slaan. Deze nemen relatief veel geheugen in beslag. Het concept van pointers kan moeilijk te omvatten zijn waardoor ze verkeerd toegepast worden. Resulterend kan dit leiden tot een systeemcrash.

4.3.3 Java

Java is een programmeertaal dat class based is, dit is een vorm van object georiënteerd programmeren[27] (OOP) waarbij overerving plaatsvindt door het definiëren van objectklassen. Daarnaast is het gebaseerd op de Write Once Run Anywhere-benadering[31]. Java wordt gebruikt voor verschillende doeleinden, zoals het ontwikkelen van software voor desktop, web, mobiel en enterprise. Bovendien is Java niet alleen de programmeertaal, maar heeft ook een heel ecosysteem. Deze bestaat uit de Java Development Kit (JDK), Java Runtime Environment (JRE)[32] en de Integrated Development Environment (IDE).

Als eerst is een van de voordelen van Java dat het Object Oriented Programming (OOP) toepast. Het is een codeerconcept waarin niet alleen het datatype en de structuur daarvan wordt gedefinieerd, maar ook het geheel van bijbehorende functies. Op deze manier wordt de datastructuur een object dat gemanipuleerd kan worden om relaties tussen objecten te maken. In tegenstelling tot een ander concept, genaamd *procedural programming*, zorgt OOP ervoor dat variabelen en functies gegroepeerd kunnen worden op basis van context. Hierbij wordt er als het ware een label aan gegeven waarbij het refereert naar functies binnen de context van een specifiek object. De redenen die OOP een voordeel maken zijn de herbruikbaarheid van objecten, het zorgt ervoor dat applicaties meer georganiseerd zijn en het bevordert de onderhoudbaarheid van de code.

Verder is Java een high-level taal, wat inhoudt dat het meer op de menselijke taal lijkt, in tegenstelling tot low-level talen die meer lijken op machinetaal. Bij een high-level taal moeten er compilers gebruikt worden om de ‘menselijke’ kant naar machinetaal te vertalen. Het maakt development eenvoudiger dankzij de makkelijke leesbaarheid en manier van code schrijven.

In tegenstelling tot sommige andere programmeertalen is Java goedkoper om te draaien, omdat er geen specifieke hardware infrastructuur benodigd is. De servers kunnen draaien op elk soort besturingssysteem.

Daarnaast is Java platformonafhankelijk, dit houdt in dat wanneer een Java-applicatie gebouwd is Windows, het alsnog op andere platforms kan draaien zolang deze Java Virtual Machine (JVM)[32] ondersteunt. JVM zorgt in dit geval voor het abstractieniveau tussen de code en de hardware.

Bovendien heeft Java automatic memory management (AMM), waarbij de ontwikkelaar niet zelf code hoeft te schrijven voor geheugen gerelateerde taken. Bij AMM wordt een garbage collector gebruikt dat kan detecteren wanneer er niet meer naar een object verwezen wordt binnen de applicatie. Op basis van deze detectie worden de objecten uit het computergeheugen verwijderd wat dus voorkomt dat er een *memory leak* is.

Als laatst is er sprake van multithreading bij Java. Een thread is een kleine verwerkingseenheid waarbij de processor zo goed mogelijk wordt benut. Java zorgt ervoor dat deze threads tegelijk uitgevoerd kunnen worden. De threads zijn onafhankelijk van elkaar wat betekent dat wanneer er bij een thread een exception optreedt dat de rest geen hinder daarvan krijgt.

Echter zijn er naast voordelen ook nadelen bij Java. Ten eerste is de performance bij een Java-applicatie relatief slecht. Doordat er gecompileerd moet worden en gebruik gemaakt wordt van een virtuele machine heeft dit negatieve impact op de performance. Ten tweede kan de garbage

[30] Pointers in C++
[31] What Is "Write Once and Run Anywhere" Feature of Java?
[32] JDK vs JRE vs JVM: Key Differences

collector voor prestatieproblemen zorgen, aangezien alle threads gestopt moeten worden tijdens het memory managementproces. Ook kunnen er *deadlocks* optreden wanneer meerdere threads toegang willen krijgen tot dezelfde bron.

4.3.4 Python

Python is een high-level programmeertaal waarbij de nadruk ligt op leesbaarheid van de code. Net als Java, C# en C++ benut Python het concept van Object Oriented Programming. Bovendien staat Python erom bekend dat deze gemakkelijk aan te leren is en het is open-source. Er is een veelzijdige toepassing met Python zoals Web Development, Network Programming en Game Development.

Python is een geïnterpreteerde taal, wat inhoudt dat er geen compilatie plaatsvindt en lijkt op een normaal tekstbestand. De code kan overal geschreven worden, zelfs in een kladblok applicatie. Echter is het gebruikelijk om voor Python PyCharm, PyDev of KomodoIDE te gebruiken om daarin Python code te schrijven.

Een aantal van de voordelen van Python zijn:

- Een simpele, duidelijke syntax, waardoor het makkelijk aan te leren is.
- Dynamische types (van variabelen)
- Geen gebruik van brackets[33]
- Automatische geheugenallocatie
- Veel ondersteunende libraries

Naast de voordelen van Python zijn er ook nadelen. Over het algemeen zijn de prestaties van applicaties relatief gezien langzaam. Voor bepaalde toepassingen zoals machine learning is Python een goede keuze. Echter is het voor het ontwikkelen van mobiele applicaties minder geschikt, daarvoor is het beter om andere programmeertalen zoals Java of React Native te gebruiken.

Een aantal nadelen van Python zijn:

- Python kan relatief traag zijn en voor grote en complexe applicaties onpraktisch zijn.
- Het is een high-level programmeertaal wat het minder geschikt maakt voor hardware applicaties.
- For bepaalde taken kan een impliciete geheugen allocatie nadelig zijn.
- Hoewel de simpliciteit van Python een van de voordelen is kan dit ook een nadeel zijn. Het is dan voor ontwikkelaars die hieraan gewend zijn moeilijker om over te stappen naar een moeilijkere syntax zoals bij Java of C#.
- Hoge geheugenconsumptie.
- Minder geschikt voor mobiele applicaties.

4.3.5 PHP

PHP[34] is een open-source programmeertaal dat voornamelijk gebruikt wordt voor het ontwikkelen van applicaties. De syntax van PHP komt redelijk overeen met C. Daarbij is de toepassing van PHP breed, zo kunnen er verschillende soorten applicaties mee gebouwd worden. Zoals voor een Content Management Systeem, web development, het verwerken van afbeeldingen, grafische interface design en data-analyse. Echter staat PHP het meest bekend in de web development

[33] Brackets
[34] Pros and cons of PHP web development

wereld. Veel PHP-apps draaien op een webserver zoals Apache. Daarnaast maakt PHP gebruik van het OOP-concept.

PHP heeft een aantal voordelen, ten eerste dat het veilig is en relatief goedkoop om in gebruik te nemen. PHP werkt samen met HTML om dynamische elementen op een webpagina te tonen. Bovendien maakt PHP onderdeel uit van de LAMP[35]-stack. Het is een softwarebundel dat voor Linux, Apache, MySQL en PHP staat. Hiermee worden hoog presterende webapplicaties geleverd.

Ten tweede is PHP erg flexibel en aanpasbaar. Het kan voor heel veel toepassingen gebruikt worden in een grote variëteit aan context, aangezien het voor veel besturingssystemen beschikbaar is. Met andere woorden, PHP is platformonafhankelijk. Bovendien draait het op de meest populaire webservers en is compatibel met een groot aantal relationele databasemanagementsystemen. Mede dankzij de grote flexibiliteit en aanpasbaarheid heeft PHP een groot ecosysteem.

Ten derde is leercurve laag bij PHP, het is namelijk eenvoudig om te gebruiken. De reden hiervoor is dat PHP flexibel is en het leerproces ondersteunt dankzij een structuur wat intuïtief is. Programmeurs die bekend zijn met programmeertalen zoals C# of Java zullen PHP snel op kunnen pakken.

Als laatst heeft PHP een grote hoeveelheid aan libraries en frameworks beschikbaar. Dit opent meer mogelijkheden voor PHP waarbij er meer logica en consistentie is. Het gebruik van libraries en frameworks kan zelfs nadelen van de taal tenietdoen.

Naast de voordelen van PHP zijn er ook een aantal nadelen. Het draait iets trager dan sommige programmeertalen. Dat komt doordat PHP een interpreterende taal is. Talen die gebruikmaken van een compiler[36] zijn ‘sneller’, omdat hierbij de code direct door de computer vertaald wordt. Het vertaalproces gaat van voor de mens leesbare code naar leesbare code voor de machine (nullen en enen). Terwijl bij een interpreterende taal voor hetzelfde proces nog tussenkomende software voor benodigd is, de ‘vertaler’.

Bovendien is de pure programmeertaal PHP zonder framework inconsistent. Er ontbreekt namelijk een structuur en strikte architectuur waardoor de ontwikkelaars op een zodanige manier kunnen coderen wat hen het beste uitkomt. Dit kan leiden tot verschillende code structuren onder PHP-ontwikkelaars met als resultaat dat code slecht onderhoudbaar is. Dit probleem zou opgelost kunnen worden door onder andere frameworks en het OOP-concept te gebruiken.

4.3.6 Verantwoording programmeertaal

Op basis van de besproken programmeertalen en in het kader van de afstudeeropdracht (web development) lijken C#, Java, Python en PHP geschikte kandidaten. Kijkend naar deze vier mogelijkheden heb ik vanuit de opleiding ervaring opgedaan met Java en C# en tijdens de stage van het derde studiejaar heb ik met PHP gewerkt. Dit zijn voor mij persoonlijk de talen waar ik uit zou willen kiezen, echter moet ik ook rekening houden met de bedrijfssituatie bij Bitman.

Bij Bitman wordt er met MacOS, Linux, MySQL, PHP, CodeIgniter en Laravel gewerkt, dit is te beschouwen als het ecosysteem van Bitman. Aangezien de helpdesktool uiteindelijk verder wordt ontwikkeld binnen Bitman is het een onverstandige keuze om voor C# en ASP.NET te kiezen, omdat het hierbij een vereiste is om het Windows besturingssysteem te gebruiken. Bij het kiezen van C# en ASP.NET zou Bitman moeten investeren in een Microsoft licentie en daarvoor aanpassingen moeten aanbrengen in hun ecosysteem. Java, Python en PHP daarentegen zijn

[35] LAMP-stack
[36] What Are Compilers And Programming Languages

platformonafhankelijk waardoor ze veel betere keuzes zullen zijn, kijkend naar de bedrijfssituatie van Bitman.

Aangezien er voor de afstudeeropdracht een webapplicatie gebouwd gaat worden is de keuze gevallen op PHP. De voornaamste reden is dat het deel uitmaakt van de LAMP-stack, waarbij alle ingrediënten voor het maken van een webapplicatie in staan. Daarnaast heeft PHP veel meer frameworks beschikbaar in vergelijking met Python en Java. Naast deze genoemde redenen is uit de gesprekken met de opdrachtgever en bij het opstellen van de afstudeeropdracht gebleken dat de applicatie in PHP met Laravel gebouwd moet gaan worden.

Framework

Nu de keuze gevallen is op PHP is het tijd om te kijken naar welke frameworks er hiervoor bestaan. De frameworks die het meest in gebruik genomen worden zijn Laravel, CodeIgniter, Symfony, CakePHP, Yii, Zend Framework, Phalcon, FuelPHP en Slim[37]. Er zal per framework redenen benoemd worden om deze te gebruiken. Op basis van de redenen zal er verantwoord worden waarom voor een bepaald framework gekozen is.

4.3.7 Laravel

Laravel is een framework dat goed kan omgaan met complexe webapplicaties. In vergelijking met andere frameworks doet Laravel dit sneller en veiliger. Daarnaast wordt het gehele ontwikkelproces vereenvoudigd door het verminderen va de complexiteit van gebruikelijke taken, zoals sessies, routing, queuing en authenticatie. Een aantal redenen om voor Laravel te kiezen zijn:

- Uitgebreide documentatie en tutorials.
- Het is geschikt om te gebruiken voor het ontwikkelen van applicaties met complexe backend requirements, zowel kleine als grote applicaties. Dankzij het introduceren van Homestead is het installeren van Laravel makkelijk gemaakt. Het bestaat uit een all-in-one vagrant box.
- Het is een framework met veel features voor het aanpassen van complexe apps. Denk hierbij aan naadloze data migratie, MVC-architectuur ondersteuning, security, routing, view template engine, authenticatie en nog veel meer.
- Laravel is erg krachtig, samen met de snelheid en security past het goed in het kader van verwachtingen van een moderne webapplicatie.

4.3.8 CodeIgniter

CodeIgniter staat erom bekend dat het een kleine footprint[38] heeft en gaat gepaard met een gemakkelijke installatie met minimale configuratievereisten. Het is een simpel en krachtig framework, waarbij het gehele framework inclusief de documentatie slechts 2MB groot is. Het is met name geschikt voor het ontwikkelen van dynamische websites, aangezien het veel voorgebouwde modules bevat die helpen bij het construeren van sterke en herbruikbare componenten. De redenen om CodeIgniter te gebruiken zijn:

- Het is een lichtgewicht en eenvoudig PHP-framework dat gemakkelijk te installeren is in tegenstelling tot andere frameworks.
- De belangrijkste kenmerken zijn een MVC-architectuur, goede foutafhandeling, ingebouwde security tools en simpele en uitmuntende documentatie. Daarnaast kunnen er schaalbare apps mee worden gemaakt.
- In vergelijking met andere frameworks is CodeIgniter sneller. Aangezien het een licht framework is met solide prestaties is het een goede keuze wanneer men een lichte applicatie wil

[37] 10 Popular PHP frameworks to consider
[38] Het aantal geheugen wat een applicatie inneemt op het computergeheugen.

ontwikkelen dat op een bescheiden aantal servers draait. Let wel op: de releases bij CodeIgniter zijn wel onregelmatig. Dit betekent dat het minder geschikt is voor applicaties waarbij een hoog niveau van security van belang is.

4.3.9 Symfony

Symfony is een framework dat al een stuk langer bestaat dan de meeste PHP-frameworks. Het bestaat sinds 2005 waaruit blijkt dat het een volwassen en betrouwbaar platform is. Het framework is een extensieve PHP framework en houdt zich aan de standaarden van PHP en het web. Een aantal redenen om Symfony te gebruiken:

- Symfony is de perfecte keuze als het gaat om het ontwikkelen van grootschalige enterprise projecten. Het is gemakkelijk te installeren en configureren op de meeste platforms.
- De beschikbaarheid van herbruikbare libraries en componenten waarmee taken zoals authenticatie, templating, object configuratie en meer gedaan worden. Bovendien voldoen de componenten met de best practices van web en design patterns en zorgen voor integratie met andere libraries.
- Symfony is erg flexibel en kan integreren met grotere frameworks zoals Drupal. Symfony en Laravel hebben veel overeenkomende en unieke features.
- Groot ecosysteem met veel documentatie en ondersteuning van de community.

4.3.10 CakePHP

Net als Symfony heeft CakePHP de tand des tijds doorstaan. Het bestaat al ruim een decennium en is nog steeds een populair framework simpelweg omdat het na verloop van tijd mee is gegaan met de veranderingen. Bij elke nieuwe release zijn er nieuwe features bijgekomen waardoor het aantal gebruikers nog steeds groot is. De nieuwste versie van het framework heeft features geïntroduceerd zoals verbeterde modulariteit en een grotere mogelijkheid om extra standalone libraries te maken. Een aantal redenen om voor CakePHP te kiezen:

- CakePHP is simpel en gemakkelijk te installeren, zo is er alleen een webserver en een kopie van het framework nodig.
- Het is een goede keuze voor commerciële applicaties dankzij de security features zoals preventie van SQL-injectie, input validatie, cross-site request forgery (CSRF) en cross-site scripting (XSS) bescherming.
- Belangrijkste kenmerken zijn een modern framework, snelle builds, overerving, validatie en security. Bovendien beschikt CakePHP over een goede documentatie, veel support portals en premium support vanuit het moederbedrijf Cake Development Corporation.

4.3.11 Yii

Het Yii-framework, een afkorting voor “Yes, it is!”, is in feite simpel en evolutionair. Het is een component-based PHP framework, met hoge performance, voor het ontwikkelen van moderne webapplicaties. Het is framework is gebaseerd op het Don’t Repeat Yourself (DRY) concept. Een aantal redenen waarom Yii populair is:

- Een gemakkelijk installatieproces. Verder heeft het robuuste security features dat het framework geschikt maakt voor e-commerce projecten, portals, CMS, fora en meer.
- Het heeft een goede snelheid en prestatie, is goed uitbreidbaar, en het voorkomt dat ontwikkelaars complexe, repetitieve SQL-statements moeten schrijven. Zo kunnen developers de database modelleren op basis van objecten.
- Het is enorm uitbreidbaar, zo is het zelfs mogelijk op de kern van de code aan te passen naar eigen wens. Het komt wel met een grote leercurve wanneer men nog onbekend is met dit framework.

4.3.12 Zend Framework

Het Zend Framework is een compleet object georiënteerd framework. Het feit dat er features zijn zoals interfaces en overerving maken het uitbreidbaar. Het framework is met de agile methodologie gebouwd, wat helpt bij het leveren van hoge kwaliteit applicaties voor ondernemingen. Zend is erg aanpasbaar en houdt zich aan de best practices van PHP, wat het geschikt maakt om project-specifieke functionaliteiten toe te voegen. Een aantal redenen om Zend te gebruiken:

- Het is een goed framework voor complexe, enterprise-level projecten. Voor grote ICT-afdelingen en banken is zelf een voorkeur om dit framework te gebruiken.
- Belangrijke kenmerken zoals MVC-componenten, simpele cloud API, data encryptie en sessiemanagement.
- Zend kan met externe libraries integreren en van deze libraries kunnen alleen de benodigde componenten gebruikt worden. Het framework bevat bovendien erg goede documentatie en heeft een grote community. Voor het ontwikkelen van mobiele applicaties heeft dit framework een hoge leercurve.

4.3.13 Phalcon

Phalcon is een full-stack PHP framework dat gebruikmaakt van het MVC-pattern en oorspronkelijk in C en C++ is geschreven. Sinds de eerste release van 2012 heeft dit framework constant updates gekregen. Phalcon is een extensie van C, maar het is niet nodig om kennis van C te hebben om dit framework te gebruiken. De redenen waarom men voor Phalcon kiest:

- Phalcon is makkelijk te installeren en geschikt voor het ontwikkelen van zeer configureerbare webapplicaties die overeenkomen met de richtlijnen voor enterprise development.
- Belangrijke kenmerken zoals hoge uitvoeringssnelheid, asset management, object-relational mapping (ORM), hoge security en caching.
- In tegenstelling tot andere frameworks focust Phalcon zich op het optimaliseren van performance. Dit wordt mogelijk gemaakt dankzij de efficiënte geheugenutilisatie.
- Er is wel een klein nadeel van Phalcon: de bugs worden niet snel opgelost waardoor het minder geschikt is voor applicaties waarbij security een grote rol speelt.

4.3.14 FuelPHP

FuelPHP is een flexible, full-stack framework dat voor het eerst is uitgebracht in 2011. Het maakt niet alleen gebruik van het MVC-pattern, maar ook van de opvolger ervan, het HMVC-pattern op architectuurniveau. HMVC staat voor Hierarchical-Model-View-Controller[39]. Daarnaast voegt het framework een extra class toe dat bekend staat als de Presenter-class (bij MVC ViewModel). De Presenter ligt tussen de Controller en View lagen in en verzorgt de logica voor het genereren van views. Een aantal redenen waarom men kiest voor FuelPHP:

- FuelPHP helpt ontwikkelaars om end-to-end weboplossingen te leveren die divers zijn qua grootte en complexiteit.
- Belangrijke kenmerken zoals de HMVC-implementatie, URL-routing systeem, kwetsbaarheidsprotectie en een caching-systeem.
- Het heeft een uitgebreide security benadering wat een goede optie is voor projecten waarbij security van groot belang is.

4.3.15 Slim

Slim is het framework dat gebruikt wordt voor het ontwikkkelen van simpele, maar krachtige webapplicaties. Het is een micro-framework en het idee hierachter is dat het uitermate geschikt is

[39] HMVC structure

voor kleine applicaties die niet de kenmerken van een full-stack framework nodig hebben. Een aantal punten waar Slim geschikt voor is:

- Het is gemakkelijk aan te leren. PHP-ontwikkelaars gebruiken Slim om RESTful API's en webservices te ontwikkelen.
- Belangrijke kenmerken zoals URL-routing, sessies, cookie encryptie, client-side HTTP caching en meer.
- Voor kleine webapplicaties die niet alle features van een full-stack framework nodig hebben is Slim een goede keuze. Bovendien is er regelmatige onderhoud en een gebruiksvriendelijke documentatie.

4.3.16 Verantwoording framework

De kern van de afstudeeropdracht is om een deel van het knowledge based systeem te bouwen met focus op de functionaliteit van de decision trees. Het is een full-stack opdracht waarbij zaken zoals de database, de front-end en de backend aan het licht komen. Op basis van deze kern zijn er een aantal frameworks die het meest voor de hand liggen, respectievelijk Laravel en Symfony. De kenmerken van deze frameworks komen voor een deel overeen met elkaar. Persoonlijk heb ik al ervaring opgedaan met Symfony tijdens mijn derdejaars stage. In het kader van beroepstaak *Gf: leren leren: voorbereiden op volgende studiefase en beroep* wil ik kiezen voor Laravel. Het is namelijk een framework waar ik nog niet eerder mee heb gewerkt en daar zou ik graag mee willen leren werken. Dit stemt tevens overeen met wat er vanuit Bitman vereist wordt, namelijk dat de afstudeeropdracht uitgevoerd zal worden in het Laravel framework.

4.4 Overzicht keuze programmeertaal

Er zijn voor de keuze van de programmeertaal een aantal voorwaarden die ik heb opgesteld waaraan voldaan moet worden. Hieronder geef ik in een tabel een overzicht met hulp van een checklist.

Voorwaarde	**C#**	**C++**	**Java**	**Python**	**PHP**
Opensource ondersteuning.	✓	✓	✓	✓	✓
Platformonafhankelijkheid	✗	✓	✓	✓	✓
Object georiënteerd.	✓	✓	✓	✓	✓
Learning-curve is laag.	✓	✗	✓	✓	✓
Groot ecosysteem met veel ondersteuning vanuit de community. Denk aan libraries	✓	✓	✓	✓	✓
Goede ondersteuning web development.	✓	✗	✓	✓	✓

5. Werkzaamheden

In dit hoofdstuk zullen alle werkzaamheden per fase besproken gaan worden. Daarnaast zullen gemaakte keuzes onderbouwd worden. Voor de aantoning van de beroepstaken wordt verwezen naar hoofdstuk 6 Evaluatie onder het kopje **Beroepstaken**.

5.1 Voorbereidings-en onderzoeksfase

5.1.1 Maken plan van aanpak

Tijdens de voorbereidingsfase is het enorm belangrijk om een plan van aanpak te maken. Door dit te doen heb ik voor mijzelf een goed overzicht gekregen van alle stappen en processen die bij het afstudeertraject komen kijken. Hierin heb ik allerlei zaken beschreven zoals de te gebruiken technieken, randvoorwaarden, risicoanalyse, onderzoeksmethoden en de planning.

De te gebruiken technieken die in het plan van aanpak staan kunnen beschouwd worden als de technische beperkingen tijdens het afstudeertraject. In de randvoorwaarden staan zaken zoals de tijdsindeling per week, aandachtspunten en een afbakening van het traject. Door deze randvoorwaarden in kaart te brengen ben ik mij bewust geworden van de verhouding tussen de tijd van het afstudeertraject en de opdracht. Daarnaast zorgt het ervoor dat ik te allen tijde zal gaan prioriteren op de documentatie indien de gelegenheid zich voordoet.

Door een risicoanalyse te doen ben ik mij bewust geworden van verscheidene risico's die op zouden kunnen treden tijdens het afstudeertraject. Door de risico's inzichtelijk te maken is hier beter op te anticiperen en indien deze toch optreden kunnen ze gemitigeerd worden. Tijdens het afstuderen zullen er een aantal keren onderzoek gedaan worden. Hierbij heb ik de onderzoeksmethoden beschreven waarmee ik kan verwachten hoe ik het aan zal pakken.

Als laatste de planning, deze is van tevoren opgesteld en kan tussentijds veranderen. Door het gehele afstudeertraject in fasen op te delen heb ik een overzicht gemaakt waarbij ik in de gaten kan houden hoe de progressie ten behoeve van het traject verloopt.

5.1.2 Verkenning

Tijdens de oriëntatie en verkenning ben ik begonnen met het opstellen van templates voor de documenten. Hierbij maakte ik een universele template voor alle documenten om zo tijd te kunnen besparen. Nadat dit gedaan was begon ik aan de bedrijfsoriëntatie waarbij ik de organisatiestructuur in kaart bracht aan de hand van een organogram. Dankzij deze bedrijfsoriëntatie ben ik te weten gekomen wat de werkwijze bij Bitman is en hoe de verhoudingen tussen het management en de medewerkers staan. Bovendien heb ik zo een beter beeld wat mijn positie is binnen het bedrijf als afstudeerder.

De volgende stap na de bedrijfsoriëntatie was het verkennen van het huidige systeem waar T-Mobile gebruik van maakt. Ik kreeg toegang tot de testversie van het systeem vanuit de opdrachtgever, deze versie staat vrijwel gelijk aan de live versie. Zo kon ik gerust mijn gang gaan en van alles uitproberen zonder dat er wat met het live systeem zou gebeuren. Het was uitermate belangrijk om het huidige systeem te zien en uit te proberen om zo grip te krijgen op de werking ervan. Bovendien maakte ik met hulp van een diagram een overzicht van hoe een van de bestaande decision trees in elkaar steekt. Zo kon ik het principe van de decision tree goed begrijpen.

Voor het opzetten van de ontwikkelomgeving heb ik zoveel mogelijk zelfstandig informatie opgezocht en de Laravel documentatie gelezen. Het configureren lukte tot op een zekere mate zelfstandig, toen ik niet verder kwam kreeg ik hulp van een van mijn technische begeleiders. Hierbij zetten we een lokaal testproject op waarbij we konden verifiëren dat de omgeving correct zou werken.

5.1.3 Onderzoek

Het eerste wat gedaan moest worden voordat er aan het onderzoek begonnen zou worden is om de onderzoeksvragen op te stellen. Het doel van dit onderzoek was om erachter te komen hoe de vernieuwde troubleshooter bij zou kunnen dragen aan het ondersteunen van de klantenservice van verschillende organisaties. De hierbij opgestelde vragen waren als volgt:

Hoofdvraag: Hoe kan de vernieuwde troubleshooter de klantenservice van verschillende organisaties ondersteunen en deze verbeteren?

Deelvragen:

1. Wat is het doel waarmee de troubleshooter gebruikt wordt?
2. Hoe zit de huidige troubleshooter in elkaar?
3. Wat zijn de requirements van de stakeholders aan de troubleshooter?
4. Welke bestaande decision tree oplossingen zijn er?
5. Welke componenten/elementen van het huidige systeem kunnen gebruikt worden voor het te bouwen systeem?

De hoofdvraag moet een bijdrage leveren aan (een deel van) de oplossing voor de probleemstelling, en de deelvragen geven samen antwoord op de hoofdvraag. Op basis van de probleemstelling die is opgesteld in hoofdstuk 3 onder kopje 3.1: Probleemstelling is de hoofdvraag opgesteld. Vervolgens werden de deelvragen opgesteld op basis van de hoofdvraag. De deelvragen moeten samen stap voor stap antwoord geven op de hoofdvraag. Aangezien er voor het onderzoek een open hoofdvraag wordt gesteld betreft het een kwalitatief onderzoek[40].

Nadat de onderzoeksvragen waren opgesteld kon er een begin worden gemaakt aan het onderzoek. Zoals uit het plan van aanpak beschreven is wordt er tijdens het onderzoek gebruik gemaakt van een aantal onderzoeksmethoden. Respectievelijk zijn dit deskresearch, inventarisatieonderzoek en toegepast onderzoek. In de externe bijlage '**A.1.4 Plan van Aanpak Quang Hang - 17069866.docx**' is meer informatie te vinden over hoe de onderzoeksmethodieken worden toegepast. Het complete onderzoek is na te lezen in de externe bijlage '**A.1.3 Onderzoeksrapport Quang Hang - 17069866.docx**'. In de scriptie zal voornamelijk motivatie gegeven worden met betrekking tot het onderzoek.

De eerste deelvraag gaat over wat het doel is waarmee de troubleshooter gebruikt wordt. Dit lijkt in eerste instantie vanzelfsprekend, maar toch moet hier onderzoek naar gedaan worden. Het is namelijk zo dat er geen aannames gedaan mogen worden, want dan wordt er niet kritisch gewerkt conform beroepstaak *Gc: kritisch, onderzoekend en methodisch werken*. De aanpak bij deze deelvraag was door naar de huidige troubleshooter te kijken en de decision tree in kaart te brengen middels een diagram. Er werd gekeken naar de structuur binnen de troubleshooter en hoe het bijdraagt aan de werkwijze van een callcentermedewerker.

De tweede deelvraag had als doel om te kijken naar hoe de troubleshooter werkt en in elkaar zit. Hierbij werd er concreet gekeken naar wat de werkwijze van een callcentermedewerker is in dit proces. De aanpak hierbij was door een voorbeeld te nemen van een bepaald geval waarbij de troubleshooter te pas komt. Het is namelijk zo dat er ook informatiepagina's zijn waarbij er helemaal geen troubleshooter is. Het beantwoorden en onderzoeken van deze deelvraag werd mede mogelijk gemaakt door het feit dat ik toegang had tot de testversie van het huidige systeem dat vrijwel gelijkstaat met de live versie.

[40] Baarda, B. (2014). Dit is onderzoek! incl. toegang tot Prepzone (2de editie). Noordhoff. blz. 21

De derde deelvraag heeft betrekking op de requirements. Hoe deze tot stand zijn gekomen zal twee paragrafen verder te lezen zijn. Bij deze deelvraag was het van belang om te kijken naar welke requirements onmisbaar zouden zijn om een werkend deelproduct op te leveren.

De vierde deelvraag had als doel om te kijken welke bestaande softwareoplossingen er eventueel al zouden bestaan met betrekking tot decision trees voor customer support. Na wat deskresearch bleken er bestaande softwareoplossingen voor te bestaan. Vervolgens was het zaak om per oplossing te kijken naar wat de belangrijkste kenmerken hiervan zijn. Uit het onderzoek bleek dat er overeenkomende kenmerken waren tussen de softwareoplossingen. Hieruit kon de conclusie getrokken worden dat deze kenmerken cruciaal zijn en zullen dus deel uitmaken van het antwoord op de hoofdvraag.

Bij de vijfde en laatste deelvraag werd er onderzocht of er componenten of elementen van het huidige systeem eventueel hergebruikt zouden kunnen worden. Eerst werd er deskresearch gedaan naar het legacy framework Fusebox en het moderne framework Laravel. Verder werd er een blik geworpen op de structuur en code van het huidige systeem, dit valt onder de onderzoeksmethodiek toegepast onderzoek. Er wordt namelijk gekeken hoe het in de praktijk is gebouwd.

Op basis van de bevindingen uit het onderzoeksrapport was de conclusie als volgt. De troubleshooter kan de klantenservice van verschillende organisaties verbeteren door het gebruik van intuïtieve decision trees, integratie met bestaande systemen en het bieden van een self-service voor klanten. De volledige details van het onderzoeksrapport zijn te vinden in de externe bijlage ‘**A.1.3 Onderzoeksrapport Quang Hang - 17069866.docx**’.

5.1.4 Stakeholderanalyse

Voor de stakeholderanalyse heb ik het Handboek Requirements van Nicole de Swart geraadpleegd. In dit boek staat hoe een stakeholderanalyse gedaan kan worden. Hierbij worden de belanghebbenden in drie groepen ingedeeld, dat zijn respectievelijk bedrijfsdoel, systeem en project. Het indelen in drie groepen is van belang omdat het een algemene beeldvorming geeft over belanghebbenden die bij een bepaalde groep horen. Vervolgens komt het identificeren van deze belanghebbenden en dat kan gedaan worden door bij elke groep een aantal standaardvragen te beantwoorden. In het functioneel ontwerp ‘**A.1.2 Functioneel ontwerp Quang Hang - 17069866.docx**’ staan deze vragen allemaal beschreven.

5.1.5 Achterhalen requirements

Voor het achterhalen van de requirements heb ik gebruik gemaakt van een aantal elicitatietechnieken. Welke dat zijn is in de volgende paragraaf te lezen. Vervolgens zullen de requirements gespecificeerd worden, waarna de prioritering komt. Bovendien werd er bij het opstellen van de requirements een onderscheid gemaakt tussen de functionele en niet-functionele requirements. Verder zal het gaan over hoe de use cases tot stand zijn gekomen en de bijbehorende use case beschrijvingen. Voor alle details over onder andere de requirements, de use cases en stakeholderanalyse wordt verwezen naar ‘**A.1.2 Functioneel ontwerp Quang Hang - 17069866.docx**’.

5.1.6 Elicitatietechniek

Er zijn verschillende elicitatietechnieken[41] en respectievelijk zijn dit interviewen, prototype maken, workshops houden en observeren. De elicitatietechnieken die ik heb gebruikt zijn interviewen en observeren. De reden hiervoor is dat deze met het thuiswerken het meest praktisch zijn. Daarnaast vereist een workshop veel tijd en middelen om deze te organiseren, indien er niet genoeg tijd en middelen voor zijn is het beter om voor een alternatieve elicitatietechniek te kiezen[42]. Bij het interviewen betrok ik de opdrachtgever en een medewerker van T-Mobile (een van de administrators bij T-Mobile). Uit de opdrachtomschrijving kon ik een eerste set aan requirements

[41] Swart, N. (2017). Handboek Requirements. Reed Business Education – Hoofdstuk 16: Elicitatietechnieken
[42] Swart, N. (2017). Handboek Requirements. Reed Business Education – blz. 214-215

achterhalen. In een gesprek met de opdrachtgever heb ik aanvullende requirements op kunnen stellen. Zo kon ik feedback krijgen en wist ik gelijk of er verbeterpunten waren.

Daarnaast wilde ik weten van een medewerker, een administrator, van T-Mobile wat zij van het huidige systeem vinden en of er verbeteringen mogelijk zijn. Een gebruiker van het systeem is namelijk bekend met het systeem en kan aangeven of aanpassingen gewenst zijn. Bovendien komen eventuele gebreken van het oude systeem zo aan de orde. Door de medewerker via de mail te interviewen kon ik erachter komen of er vanuit deze gebruiker requirements zijn die aangepast moeten worden of dat er nieuwe requirements voor het systeem zijn. Ik maakte in deze mail duidelijk dat het om het troubleshooter-gedeelte ging van het systeem. Zo kon ik gelijk de scope afbakenen ten behoeve van de opdracht.

De andere elicitatietechniek, het observeren, heb ik kunnen doen dankzij de toegang die ik tot de testversie van het systeem kreeg vanuit de opdrachtgever. Ik kreeg toegang tot de omgeving waar een callcentermedewerker mee werkt en de beheeromgeving waar de administrators mee werken. In de omgeving van de callcentermedewerker oriënteerde ik op met name de informatiepagina's die een troubleshooter bevatten. Er was een account voor mij aangemaakt door de opdrachtgever waardoor ik het systeem zodanig kon navigeren net als hoe een medewerker dit zou doen. Wegens het thuiswerken was deze mogelijkheid een goed alternatief ten opzichte van een fysiek bezoek bij T-Mobile.

5.1.7 Specificering van requirements

Eenmaal toen de requirements in kaart gebracht waren kon er begonnen worden aan het specificeren hiervan. Het voornaamste doel is om de requirements voor alle belanghebbenden duidelijk te maken. Bij het specificeren probeerde ik zoveel mogelijk om het juiste detailniveau te kiezen. Dat wil zeggen dat er niet te veel informatie in een requirement gegeven wordt, dit bevordert namelijk de leesbaarheid. Een goede leesbaarheid leidt ertoe dat een requirement sneller begrepen wordt door de stakeholders. Tijdens het maken van de requirements hield ik er rekening mee dat er eventueel nog extra details aan de requirements toegevoegd zouden kunnen worden. Het is namelijk onmogelijk om een perfecte requirementsspecificatie te maken. Zo kan het zijn dat er tijdens het ontwikkelen blijkt dat een bepaalde requirement nog wat specifieker beschreven kan worden. Indien dat het geval is zal gebruik gemaakt worden van het SMART[43]-principe.

5.1.8 Prioritering van requirements

De prioriteringstechniek dat toegepast is voor de requirements is de MoSCoW-prioritering[44]. Door de requirements volgens deze techniek te prioriteren is er een overzicht van welke requirements noodzakelijk zijn voor de werking van het deelproduct. Zo kan er tijdens de ontwikkelfase rekening mee worden gehouden en komen de functionaliteiten ten behoeve van deze requirements als eerst.

5.1.9 Use cases

Voor de use cases begon ik met use case diagrammen te maken. Ik was hierbij kritisch en probeerde mij zoveel mogelijk scenario's uit te beelden die bij het doorlopen van een troubleshooter te pas komen. Op basis hiervan maakte ik een eerste diagram waarbij ik met een van de technische begeleiders overleg pleegde via Slack[45]. Op basis van de feedback bleek dat het nog onduidelijk was en dat het beter zou zijn om het diagram overzichtelijker te maken en relaties duidelijker aan te geven door middel van een uitleg boven de lijnen. Ik nam de feedback tot mij en bedacht toen dat het in het kader van duidelijkheid en meerdere scenario's het beste zou zijn om het diagram op te splitsen in subdiagrammen per scenario. Na het proces van verbetering stuurde ik de diagrammen opnieuw op naar de technische begeleider die er vervolgens goedkeuring voor gaf.

[43] SMART
[44] Swart, N. (2017). Handboek Requirements. Reed Business Education – blz. 226-227
[45] Slack

Na de goedkeuring van de use case diagrammen begon ik met de use cases te beschrijven. Een use case beschrijving kent drie detailniveaus: een contourschets, detailschets of volledig beschreven. Ik heb gekozen voor het niveau volledig beschreven. Dit heb ik gedaan omdat ik een zo gedetailleerd mogelijk beschrijving wilde maken en het past bij de use case diagrammen die ik had gemaakt. Hierbij hield ik namelijk met zoveel mogelijk scenario's rekening.

5.1.10 Verantwoording

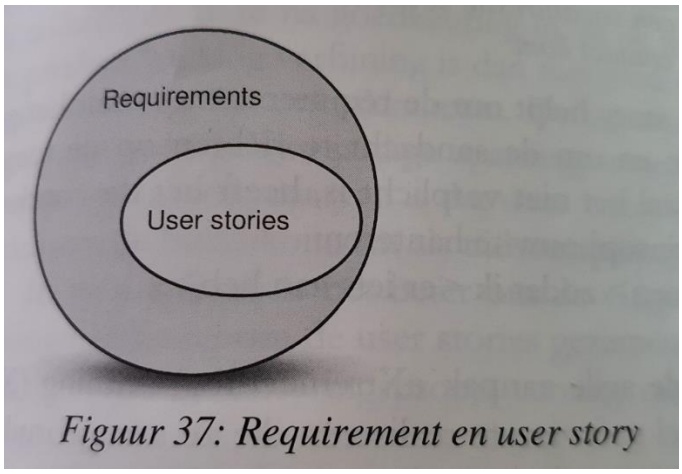
Qua tijdspanne ben ik iets afgeweken van de vooraf gestelde tijd van rond 20 dagen. De reden hiervoor was dat ik verwachtte dat er al wat documentatie was gemaakt van het huidige systeem. Dit bleek echter niet zo te zijn, waardoor ik meer tijd moest steken in het zelfstandig oriënteren op de bestaande software. Bij deze oriëntatie maakte ik tegelijk diagrammen waar nodig om zo een betere grip te hebben om de werking van de software. Bij het opzetten van de ontwikkelomgeving heb ik zoveel mogelijk zelf gedaan aan de hand van de documentatie van Laravel. Toen ik er zelf niet uitkwam heb ik hulp ingeschakeld van mijn technische begeleider.

5.2 Ontwerpfase

5.2.1 Functioneel ontwerp

Het functioneel ontwerp gaat over wat het systeem moet gaan doen. In het kader van de watervalmethode valt het onder de fase basisontwerp[46]. De gemaakte stakeholderanalyse, de opgestelde requirements en de use cases in de vorige fase zijn allemaal opgenomen in het functioneel ontwerp. Daarnaast zijn er nog user stories gemaakt en is er onderscheid tussen functionele en niet-functionele requirements gemaakt.

Als eerst de user stories, deze beschrijven de requirements vanuit het oogpunt van de gebruiker[47]. Het is een van de manieren om een requirement tot uitdrukking te brengen. Iedere user story is een representatie van een requirement, maar niet iedere requirement is een user story. De overeenkomst tussen een requirement en een user story is dat ze beide de behoeften van de business weergeven en ze hebben beide geen vaste omvang. De relatie tussen een requirement en user story is in de figuur hieronder gevisualiseerd.



Figuur 4: Relatie tussen requirement en user story

De reden om user stories te maken was om zo de wensen van de gebruikers, als ontwikkelaar zijnde, beter te kunnen begrijpen. Het maken van deze user stories zorgde er voor mij voor dat ik een beter beeld kreeg over de verschillen tussen de eindgebruikers. Er zijn namelijk twee soorten gebruikers van de troubleshooter, de tweedelijns supportmedewerkers en de support agents. Daarnaast zorgt het toepassen van user stories ervoor dat onnodige overhead, dat wil zeggen

[46] Software-ontwikkeling: welke methoden zijn er en waarin verschillen ze?
[47] Swart, N. (2017). Handboek Requirements. Reed Business Education. Blz. 134

overbodige informatie, geëlimineerd wordt en dat gebruikers betrokken worden in het requirementsproces.

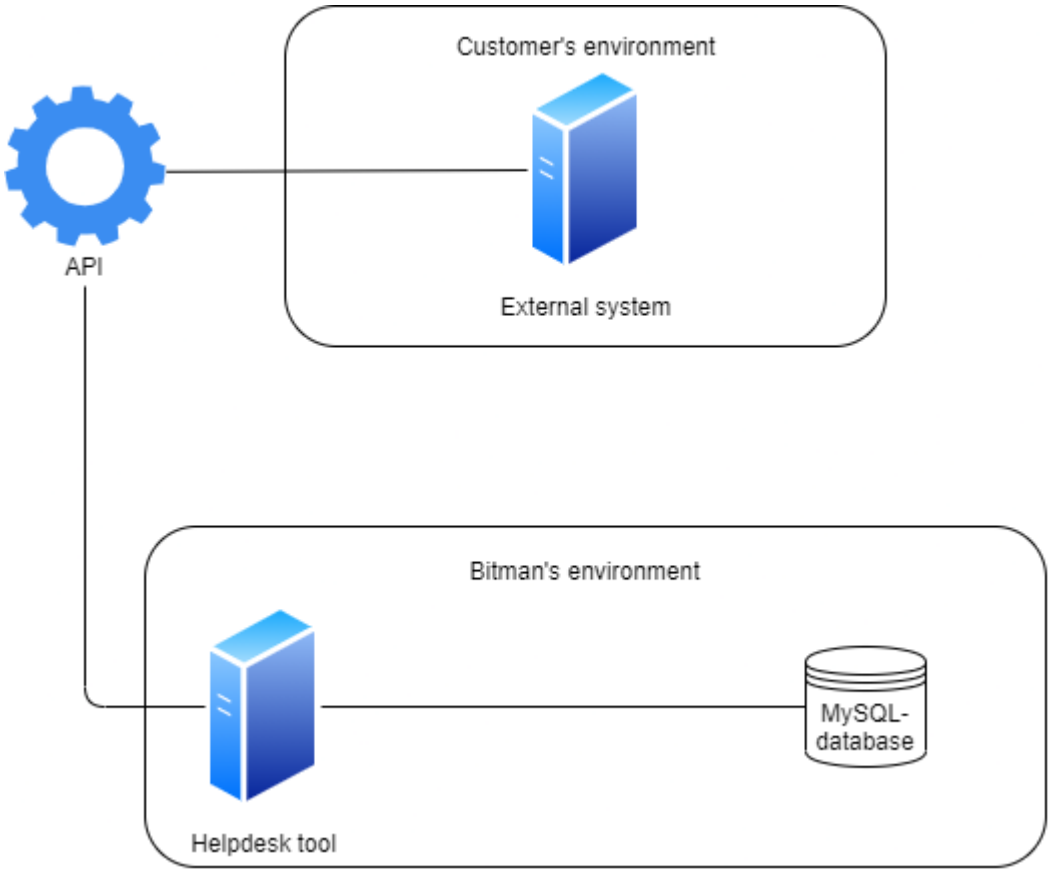
Ten tweede het onderscheid tussen functionele en niet-functionele requirements. Functionele requirements geven aan wat (functionaliteit) het systeem moet bevatten. Niet-functionele requirements zijn kwaliteitseisen waaraan het systeem moet voldoen. Samen vormen ze de softwarequirements. Het is belangrijk om deze te onderscheiden, omdat de ontwikkelaar dan een duidelijk overzicht heeft van wat voor functionaliteiten er in het systeem moeten komen en welke kwaliteitseisen deze moet hebben. Bovendien zou het zonder onderscheid een wirwar van softwarerequirements worden wat zou kunnen leiden tot meer tijdsconsumptie.

5.2.2 Technisch ontwerp

Het technisch ontwerp is een document waarin staat hoe het systeem tot stand zal komen en valt onder de derde fase van de watervalmethode[46]. Ik heb de indeling van dit document opgesteld op basis van het gesprek met de technische begeleider. We keken samen naar welke zaken het beste besproken konden worden in het technisch ontwerp. De reden dat ik hulp raadpleegde was omdat ik mijn eigen bevindingen wilde vergelijken met wat de technische begeleider verwachtte. Bovendien had ik nog weinig ervaring met het maken van een technisch ontwerp ten opzichte van het functioneel ontwerp.

Globale architectuur

Verder ging ik nog in overleg met de opdrachtgever om erachter te komen wat de positie van het systeem in het geheel is. Uit dit overleg kon ik opmaken dat de troubleshooter een ondersteuning zal vormen boven op de systemen van de organisaties. In feite is het dus zo dat de troubleshooter geen vervanging zal zijn voor de systemen van de klantenservice, maar het zal een aanvulling zijn. Op basis hiervan maakte ik een architectuurdiagram dat een globaal overzicht weergeeft tussen de omgevingen waarin de troubleshooter zich zal bevinden. Deze architectuurdiagram is in onderstaand figuur te zien en in de bijlage ‘**A.1.5 Technisch ontwerp Quang Hang - 17069866.docx**’ te vinden.



Figuur 5: Globale architectuurdiagram

Vervolgens keek ik naar hoe het huidige systeem te werk gaat om te zien hoe T-mobile met de bestaande helpdesktool werkt. Wat mij opviel was dat de klantenservicemedewerkers niet vanuit hun eigen systeem de helpdesktool aanroepen, maar dat ze met een aangemaakte account op de troubleshooter inloggen. Hierbij bedacht ik dat het juist handiger zou zijn als ze direct vanuit hun eigen systeem de troubleshooter zouden kunnen aanroepen. Dit zou de gebruiksvriendelijkheid kunnen bevorderen en ook hoeven er dan geen accountgegevens opgeslagen te worden.

Op grond daarvan ging ik uitzoeken wat een goede oplossing zou kunnen zijn om twee applicaties met elkaar te kunnen laten communiceren. Ik kwam uit op het gebruik van een API[48], dat staat voor Application Programming Interface. Simpelweg gezegd zorgt het ervoor dat er directe communicatie mogelijk is tussen verschillende applicaties. In dit geval het systeem van de klantenservice die met de troubleshooter zal communiceren. Dit idee heb ik voorgesteld aan de technische begeleider en opdrachtgever en ze vonden het beiden een goed idee. Op basis hiervan ben ik doorgegaan met het idee om een API te gebruiken voor de communicatie tussen de applicaties.

Database

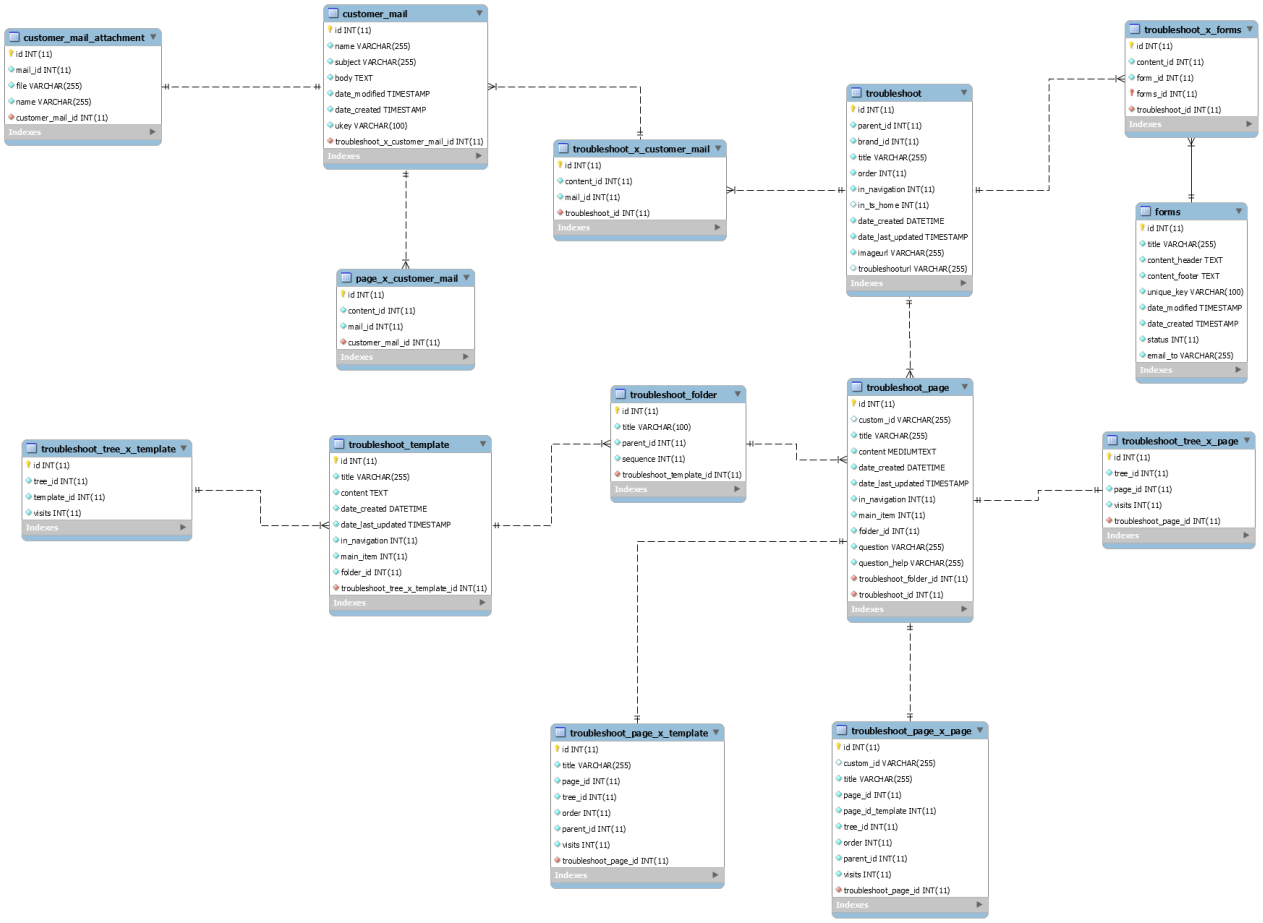
De volgende stap was om een ontwerp van de database te maken. Het gaat hierbij specifiek om een klassendiagram dat de tabellen omvat die gerelateerd zijn aan de data van de troubleshooter. De daadwerkelijke database is veel groter, maar in het kader van de scope van de afstudeeropdracht heb ik mij beperkt tot de data van de troubleshooter. In overleg met de opdrachtgever bleek dat het van belang is dat de data intact blijft bij de migratie naar het nieuwe systeem. Resulterend besloot ik om een klassendiagram te maken van de troubleshooter data.

Samen met de technische begeleider besprak ik de tools die toegepast zouden kunnen worden om een klassendiagram te kunnen genereren van de bestaande database. Er bestaan een aantal tools

[48] Wat is een API en wat kan je ermee?

voor zoals Dataedo, MySQL Workbench, dbdesc en Elasoft SqlSpec[49]. Aangezien ik MySQL Workbench al op mijn laptop geïnstalleerd had staan heb ik gekozen om deze tool te gebruiken. De eerste reden voor de keuze van het genereren van een klassendiagram van de database is het feit dat er helemaal geen bestaande documentatie is. De tweede reden is dat het om een bestaande database gaat en dat het een kwestie gaat zijn om de data te migreren.

Na het genereren van het klassendiagram van de database bleek dat er geen relaties[50] tussen tabellen waren gelegd middels Foreign Keys. Dit is het in het kader van een relationele database en referentiële integriteit[50] onjuist. Daardoor ging ik na het genereren van het klassendiagram de relaties tussen tabellen vaststellen en maken om zo de referentiële integriteit van de database te bevorderen. Het klassendiagram van de database is in onderstaand figuur en in de bijlage '**A.1.5 Technisch ontwerp Quang Hang - 17069866.docx**' te vinden.



Figuur 6: Database klassendiagram

De werkwijze was als volgt, ik ging in MySQL Workbench per tabel na wat de variabelen waren en hoe ze logischerwijs met andere tabellen gekoppeld zouden kunnen worden. Op basis van de data en variabelen die in de tabellen stonden kon ik vaststellen wat de relaties tussen deze tabellen zouden moeten zijn.

Al met al heb ik voor het klassendiagram van de database een deel gedocumenteerd en het andere deel ontworpen waarbij ik de relaties tussen de tabellen heb gemaakt. Deze ontbraken namelijk in

[49] Database documentation tools

[50] H-SE-DBPR1-1-15-DH: 2017 DH Database Applications, DBDS Database design (studiemateriaal BlackBoard)

het gegenereerde klassendiagram. Bovendien komt het vastleggen van de relaties tussen de tabellen de integriteit van de database ten goede.

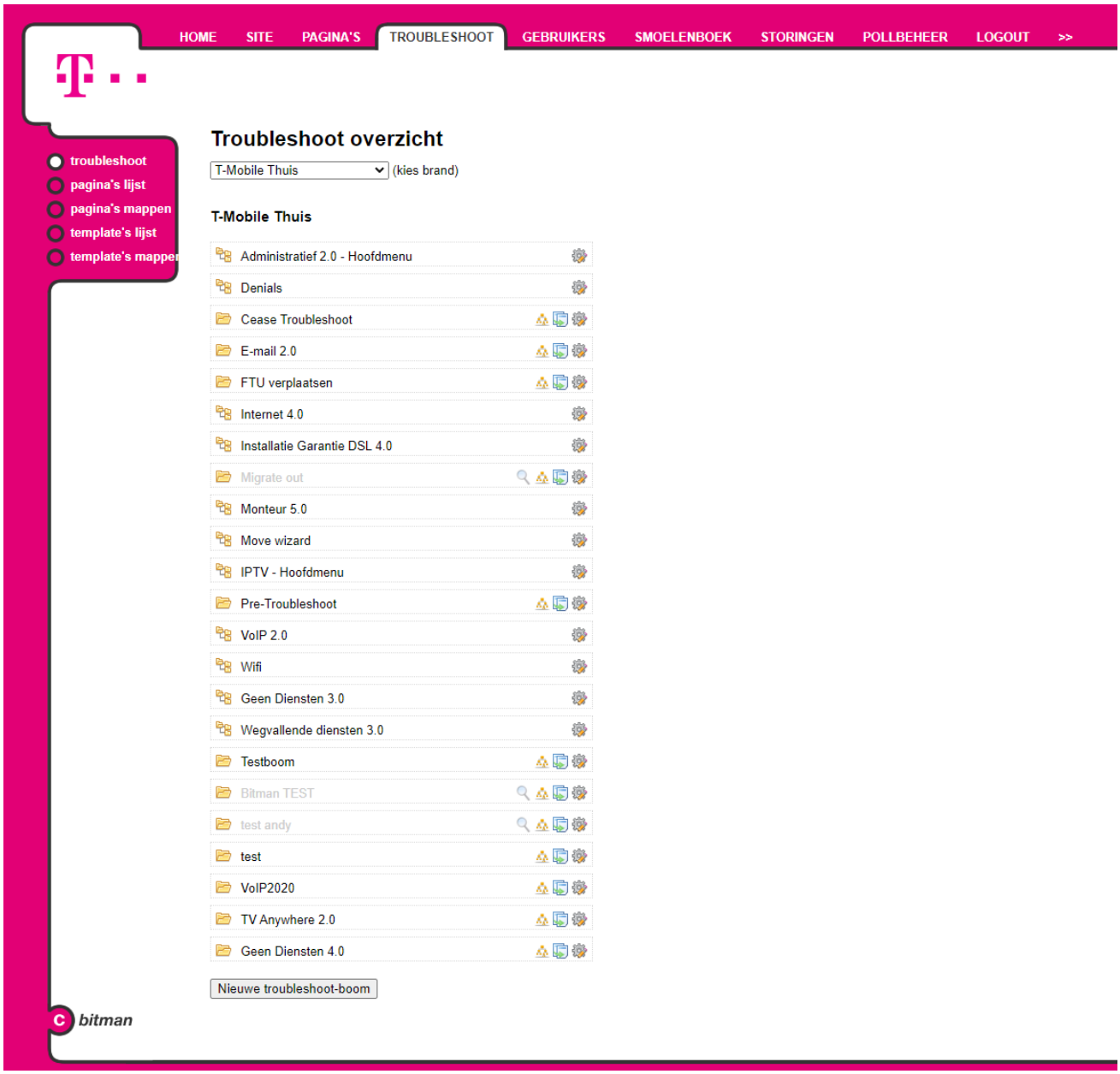
UI-design

Bij het maken en ontwerpen van de nieuwe user interface[51] (UI) waren er een aantal punten die centraal stonden, namelijk simpliciteit en universaliteit en het moet intuïtief zijn. Het is belangrijk om bij het ontwerpen van een user interface het zo simpel mogelijk te houden, zo kan de gebruiker zijn of haar doel sneller bereiken. Door het intuïtief te maken zullen handelingen voor de gebruiker voor zich spreken en kan er efficiënter gewerkt worden. De universaliteit verwijst naar het feit dat de troubleshooter door meerdere soorten organisaties te gebruiken moet zijn.

Op basis van de centrale punten die in de vorige paragraaf zijn er een aantal veranderingen gemaakt in het nieuwe UI-design ten opzichte van het huidige design. Een ander belangrijk punt is dat er voor de troubleshooter onderscheid is tussen twee omgevingen, namelijk die van de beheerders en de standaardomgeving van de supportmedewerkers. Ik zal per omgeving het verschil tonen tussen het huidige ontwerp en het nieuwe UI-design. Voor meer ontwerpen verwijs ik naar de externe bijlage '**A.1.5 Technisch ontwerp Quang Hang - 17069866.docx**'.

Als eerst de UI van de beheeromgeving. Het eerste wat de beheerder ziet van het troubleshootergedeelte is een overzicht van alle troubleshooters. Het is mogelijk om een bepaalde brand te kiezen middels de dropdown en om een nieuwe troubleshoot-boom aan te maken met de knop onderaan het scherm. Wat opvallend is aan het huidige ontwerp is dat de breedte van het scherm niet wordt benut wat leidt tot een lange verticale lijst indien er veel items zijn. Onderstaand figuur weergeeft hoe de huidige interface eruitziet.

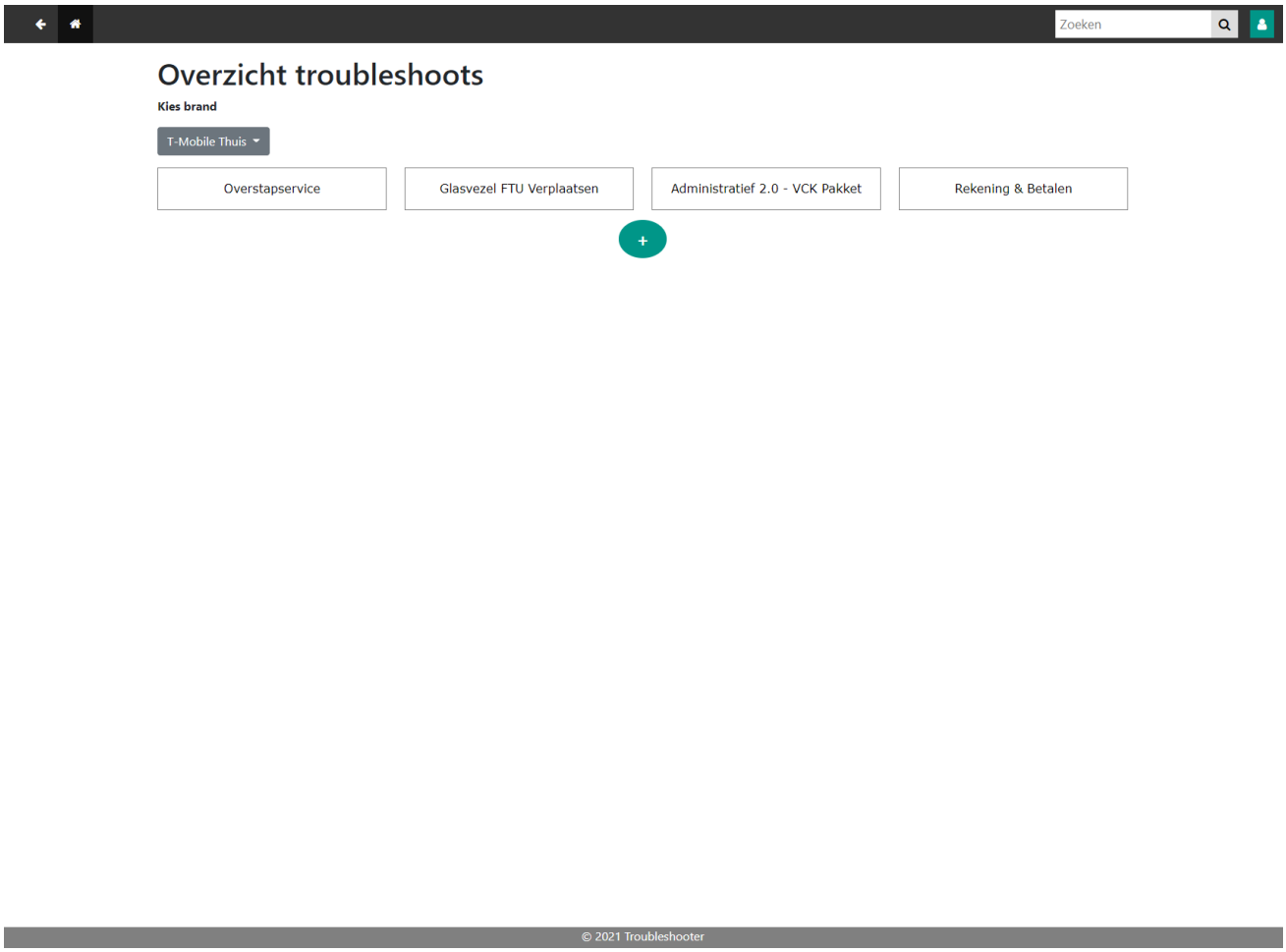
[51] User Interface Design



Figuur 7: Overzicht van troubleshoots in de oude user interface

De onderstaande figuur laat het nieuwe ontwerp zien van de user interface. Een belangrijk verschil tussen het nieuwe en oude ontwerp is dat de troubleshooters horizontaal getoond worden van links naar rechts. Indien er zoveel data in komt te staan zoals in het oude ontwerp dan zal er per rij vier items getoond worden. De ruimte op het scherm wordt zo beter benut en is ten opzichte van het oude ontwerp eenvoudiger. Bovendien is de knop om een nieuwe troubleshoot-boom toe te voegen veranderd in een knop met een plus logo. Dit bevordert zowel de simpliciteit en het instinct van de gebruiker.

Een ander nieuw element in het nieuwe ontwerp is de navigatiebalk bovenaan het scherm. Hierin staan links een terug-en homeknop. Er zijn in deze knoppen icoontjes gebruikt in plaats van tekst om het zo eenvoudig mogelijk te maken voor de gebruiker. Rechts van de navigatiebalk staan een zoekbalk en een profielknop. Met de zoekbalk kan de gebruiker alle informatie met betrekking tot de troubleshooters opzoeken. De gebruiker kan met de profielknop zijn of haar eigen profiel bekijken of uitloggen. Deze navigatiebalk is ontworpen voor zowel de beheer-als standaardomgeving.

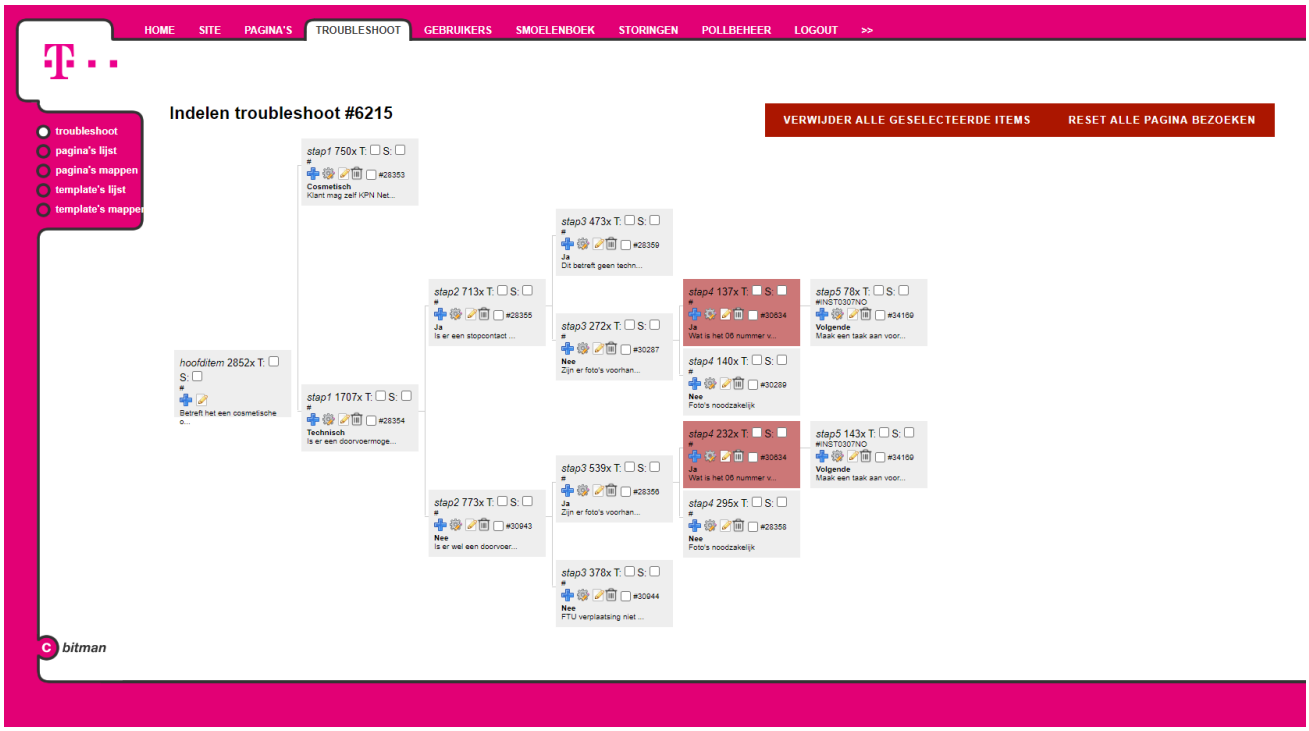


Figuur 8: Overzicht troubleshoots in de nieuwe user interface

Wanneer de beheerder op een van de troubleshooters klikt zal er een nieuwe pagina verschijnen die de decision tree van de betreffende troubleshooter toont. Per stap is er een mogelijkheid om er een nieuw item aan toe te voegen, het te bewerken en te verwijderen. In onderstaand figuur is de huidige interface van deze pagina te zien.

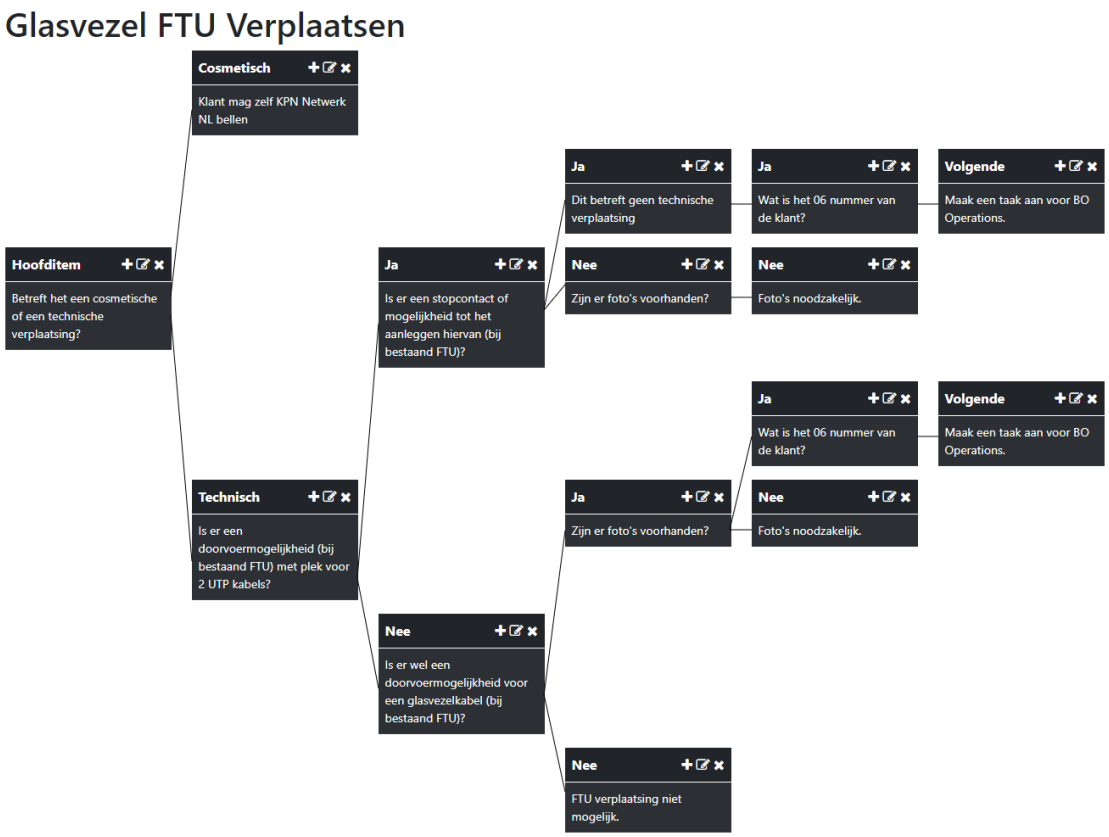
De huidige interface bevat per item veel informatie. Ik heb met de technische begeleider besproken wat de checkboxen rechtsboven, respectievelijk T en S betekenen. Uit dit gesprek bleek dat de technische begeleider goed moest nadenken wat deze inhouden en hij wist hier niet gelijk antwoord op. Hieruit leidde ik af dat te veel informatie in een item niet goed is. De eindgebruiker zal hier wellicht nog minder raad mee weten, zeker wanneer het een persoon betreft die net nieuw met het systeem werkt.

Daarnaast zijn er vier knoppen voor drie functionaliteiten, het toevoegen, bewerken en verwijderen van een item. In de huidige interface is er voor het bewerken van een item onderscheid gemaakt tussen de titel van de pagina en de gekoppelde pagina.



Figuur 9: Decision tree in oude ontwerp

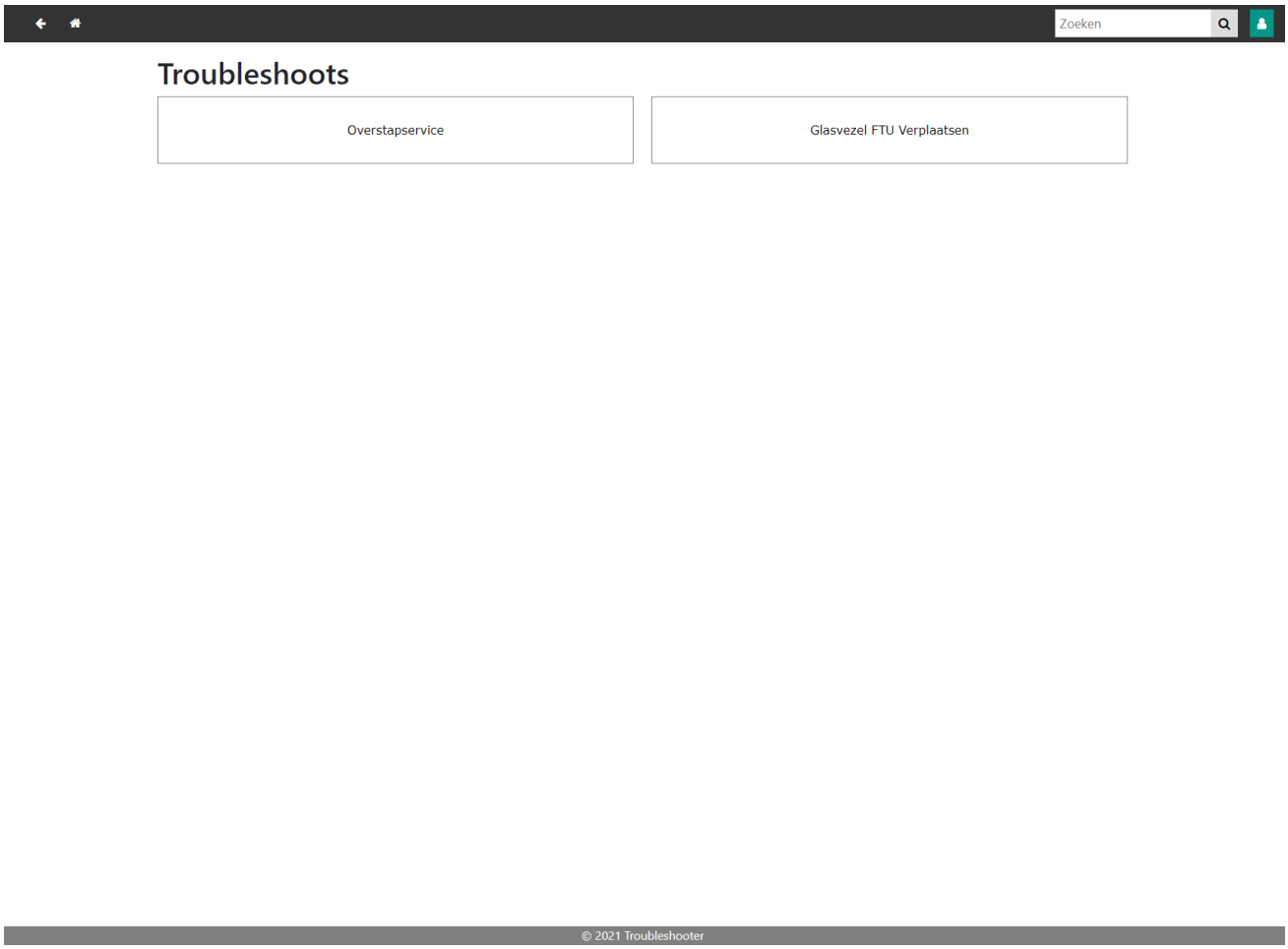
Onderstaand figuur weergeeft het nieuwe ontworpen design voor de decision tree-pagina. Zoals in de vorige paragraaf is vermeld bevatte de oude interface per item veel informatie. Dit zou voor de eindgebruiker tot onduidelijkheid kunnen leiden wat het werken minder efficiënt zou kunnen maken. Resulterend is het nieuwe design voor de interface vereenvoudigd waarbij de hoeveelheid informatie per item is verminderd. De functionaliteiten voor het toevoegen, bewerken en verwijderen zijn gereduceerd tot drie knoppen. De icoontjes zijn een plusje, een pen en een kruisje wat respectievelijk toevoegen, bewerken en verwijderen inhoudt. Daarnaast zijn de checkboxen verdwenen in het nieuwe ontwerp in het kader van eenvoud.



Figuur 10: Decision tree in het nieuwe ontwerp

Ten tweede de ontwerpen van de standaardomgeving waar supportmedewerkers mee zullen werken. Het eerste wat een supportmedewerker ziet op de troubleshooter is een overzicht van alle troubleshooters. Het verschil tussen de beheer en standaardomgeving is hierbij dat er geen brands gekozen kunnen worden en dat er geen knop is om een troubleshooter toe te voegen. Dit zijn functionaliteiten waar alleen een beheerder toegang tot heeft. Dit heb ik afgeleid uit gesprekken met de opdrachtgever en de technische begeleider.

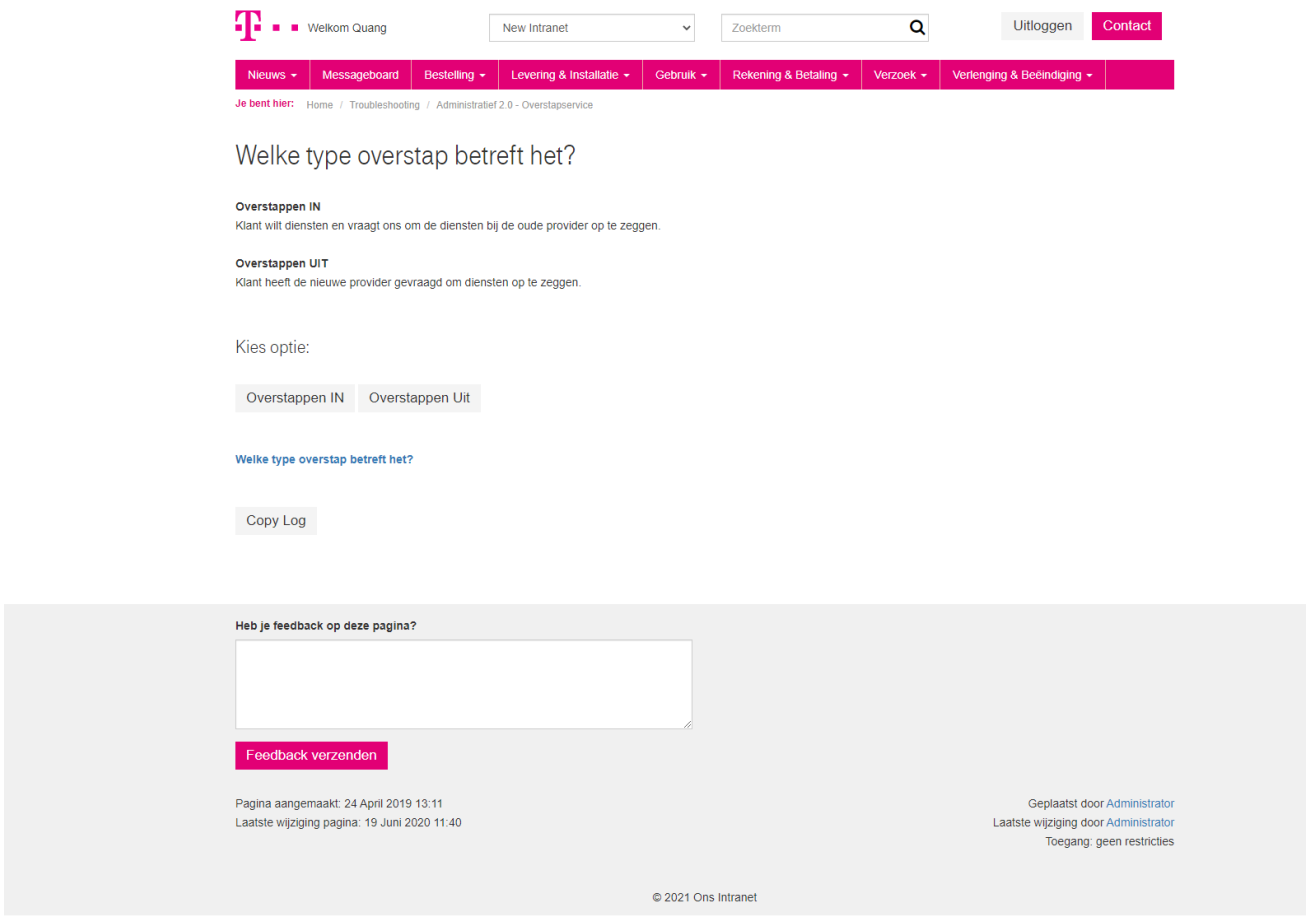
Onderstaande figuur toont deze nieuwe overzichtspagina. In het ontwerp staan er twee troubleshooters waarbij er per rij maximaal vier items getoond zullen worden. Indien er bijvoorbeeld 24 items zijn zullen er in totaal zes rijen zijn met ieder vier items.



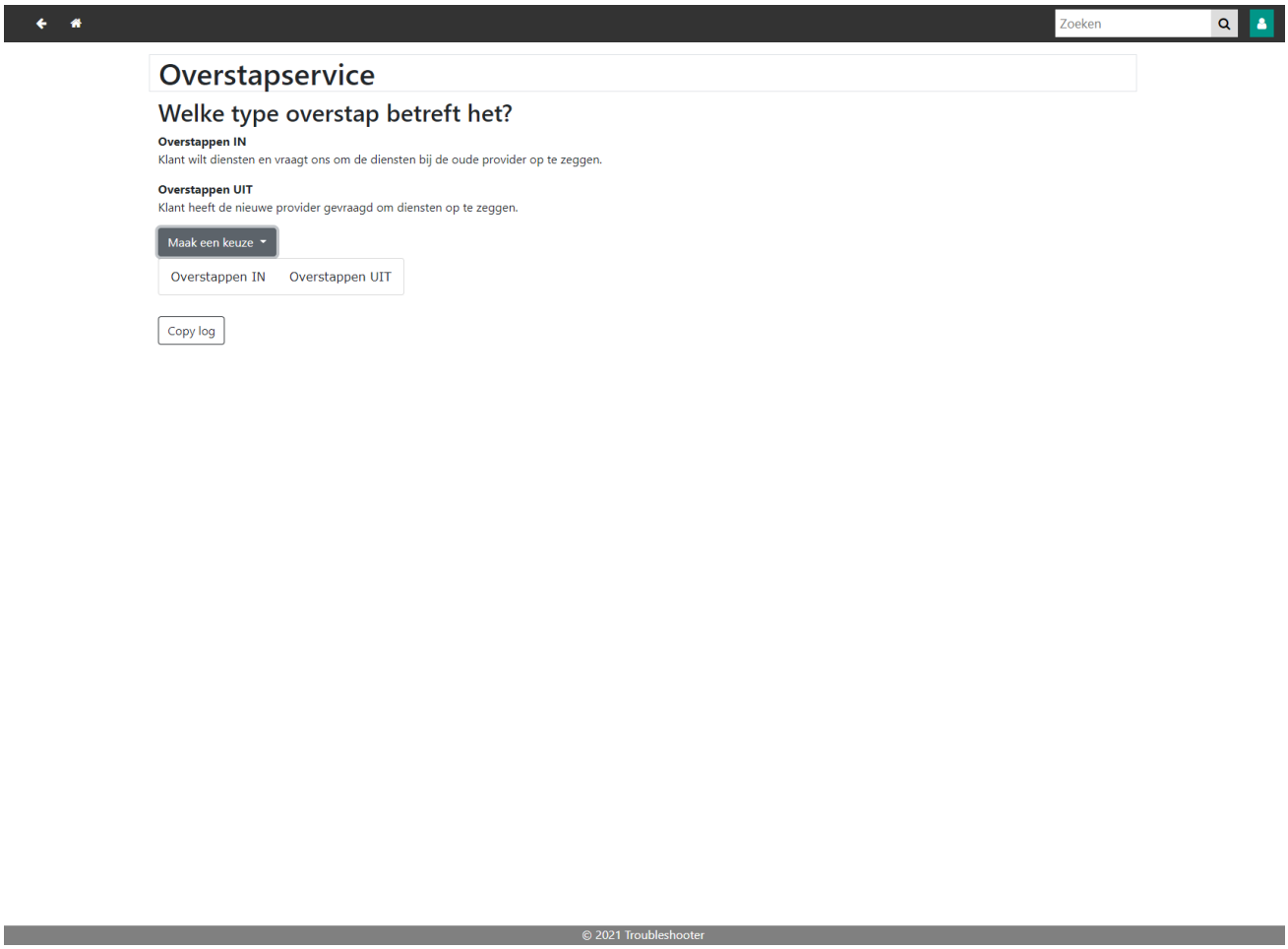
Figuur 11: Nieuw ontworpen overzichtspagina voor de standaardomgeving

Daarnaast heb ik in het kader van universaliteit gekozen om een overzichtspagina te ontwerpen voor alle troubleshooters in de standaardomgeving. De helpdesktool moet in de toekomst namelijk door verschillende soorten organisaties te gebruiken zijn. De huidige interface heeft geen overzichtspagina van troubleshooters, maar er moet via bepaalde informatiepagina's genavigeerd worden om bij een troubleshooter te komen. Niet alle informatiepagina's hebben een troubleshooter, maar aangezien de afstudeeropdracht zich richt op de troubleshooters en de decision trees is de rest buiten beschouwing gelaten.

Onderstaande twee figuren laten een bepaalde troubleshooter zien met als eerst de huidige interface en daarna het nieuwe ontwerp. Het grootste verschil tussen het oude en nieuwe ontwerp is dat de te maken keuze in het nieuwe design verstopt zit in een dropdown menu. Dit is gedaan in het kader van modernisering en het intuïtief maken van de interface.



Figuur 12: De eerste pagina van een bepaalde troubleshooter in de huidige interface



Figuur 13: Eerste pagina van troubleshooter in nieuw ontwerp

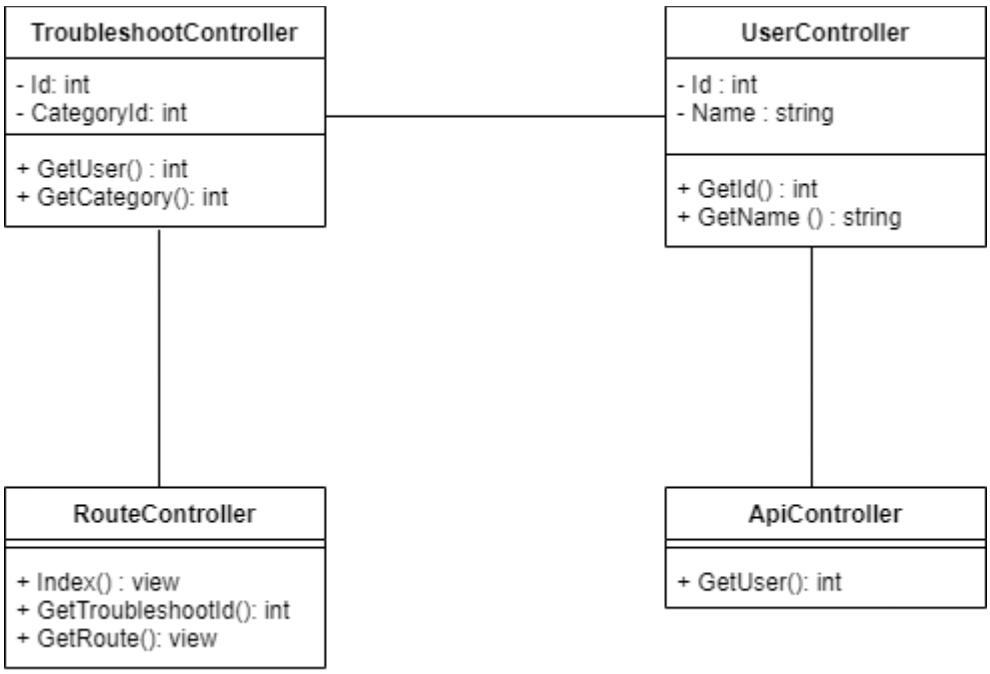
Object model diagram

Bij het ontwerpen van onderstaande diagram is per controller[52] rekening gehouden met een van de SOLID-principes[53], dat is de S. Deze letter staat voor 'Single-responsiblity' wat inhoudt dat een object of klasse slechts één verantwoordelijkheid moet hebben. Zo heeft de UserController de verantwoordelijkheid om alle acties die gerelateerd zijn aan de gebruiker te verwerken. Denk hierbij aan het controleren welke gebruiker toegang wilt krijgen tot de troubleshooter en het ophalen van de naam van de gebruiker. Dit principe geldt ook voor de rest van de controllers.

In onderstaande klassendiagram staat de TroubleshootController centraal. Deze haalt de gegevens op van de gebruiker middels de methode 'GetUser'. Het id van de gebruiker wordt bepaald in de UserController middels de methode 'GetId', waarbij deze methode het id van de gebruiker ophaalt via de ApiController. Het idee van de ApiController is dat deze de id van de gebruiker bepaalt aan de hand van een uniek id dat voor elke klant anders zal zijn.

De RouteController verzorgt de logica van welke pagina getoond moet worden. De standaardmethode dat uitgevoerd zal worden is de index. Deze laat de standaard overzichtspagina zien waar de gebruiker begint in de troubleshooter. De troubleshooter wordt bepaald aan de hand van de 'GetTroubleshootId' methode en de 'GetRoute' methode zal bepalen naar welke pagina genavigeerd wordt per stap.

[52] Controllers Laravel
[53] H-SE-PRPR-1-16-DH-2018: 2018 Advanced Programming Den Haag, Programming and Testing 2 (studiemateriaal BlackBoard)



5.2.3 Verantwoording

Ik heb er bewust voor gekozen om de ontwerpen van de pagina's gelijk in HTML en CSS te bouwen in combinatie met Bootstrap[54] en het Laravel framework. De reden hiervoor is dat ik dan gelijk goed in kon komen met de werking van het framework in combinatie met de views van het MVC-pattern. Bovendien had ik al ervaring met het Bootstrap framework waardoor ik in relatief korte tijd de ontwerpen voor de pagina's kon opzetten binnen Laravel.

Van het functioneel ontwerp heb ik de volgende onderdelen zelf ontworpen en gemaakt: de stakeholderanalyse, de softwarerequirements en de use cases. De user stories heb ik gebaseerd op het bestaande systeem en zijn in feite documentatie daarvan. Om deze te bekijken verwijs ik naar de externe bijlage '**A.1.2 Functioneel ontwerp Quang Hang - 17069866.docx**'. Van het technisch ontwerp heb ik alles zelf ontworpen, behalve het klassendiagram van de database. Dit klassendiagram is gegenereerd op basis van de bestaande database. Ik heb hier echter wel verbeteringen aan gemaakt door relaties te maken tussen tabellen om de referentiële integriteit van de database te verbeteren. Om deze te bekijken verwijs ik naar de externe bijlage '**A.1.5 Technisch ontwerp Quang Hang - 17069866.docx**'.

De tijd die benodigd was voor de ontwerpfase week iets af van de vooraf gestelde tijd van circa 25 dagen. Het kwam namelijk wel eens voor dat ik feedback kreeg van de opdrachtgever waarbij er nog verbeteringen mogelijk waren in de tussenproducten zoals de designs, het functioneel en technisch ontwerp. Het was niet een eenmalig proces, maar duurde soms een aantal keren.

5.3 Ontwikkelfase

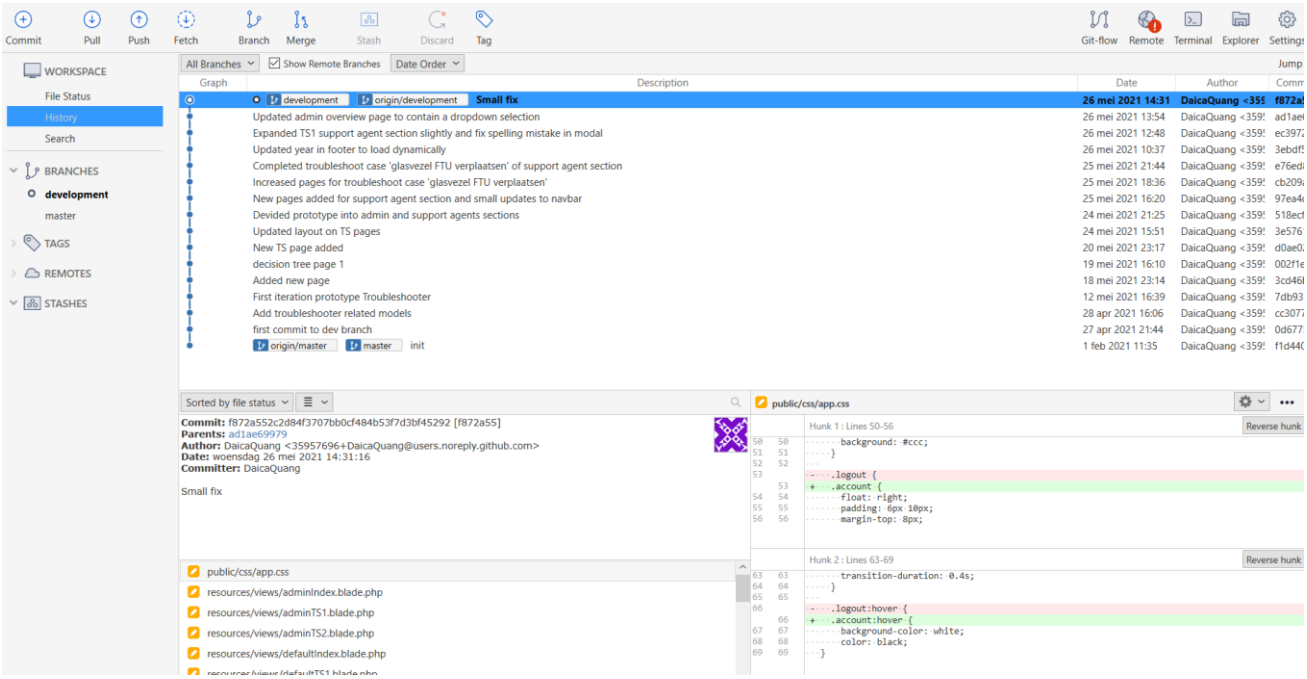
De ontwikkelfase zal alle werkzaamheden beschrijven die gerelateerd zijn aan het ontwikkelen van de applicatie. Vanaf deze fase wordt er volgens de SCRUM-methodiek gewerkt en hoe hier mee omgegaan is staat in paragraaf 4.2 Ontwikkelmethode. De ontwikkelfase bestaat uit tweewekelijkse sprints met in totaal drie sprints. Alvorens te beginnen aan het coderen ging ik de documentatie van

[54] Bootstrap

Laravel[55] globaal doornemen, omdat ik dan een betere grip heb op wat ik moet doen en dan ben ik ingelezen op het framework.

5.3.1 Algemene werkwijze

Voor het versiebeheer van de te bouwen applicatie maakte ik gebruik van Bitbucket en SourceTree. Bitbucket wordt bij Bitman gebruikt voor het beheren van de code en ik gebruikte de SourceTree software om een directe GIT-verbinding met Bitbucket te maken. De reden waarom ik SourceTree gebruikte is omdat ik een grafische interface fijner vind werken dan een Command Line Interface[56]. Onderstaande screenshot toont hoe de interface van SourceTree eruitziet.



Figuur 14: SourceTree interface

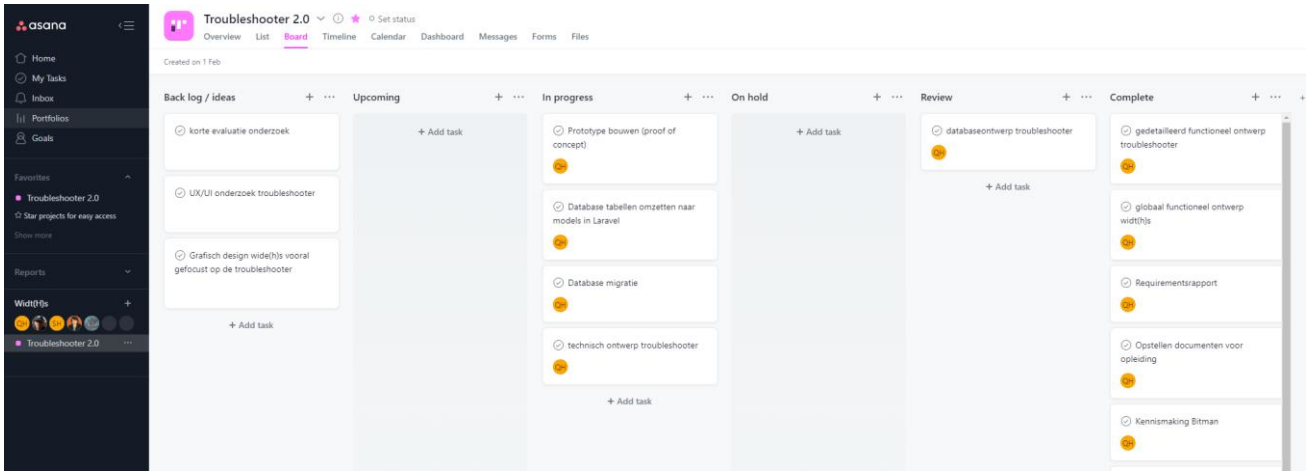
Voordat ik met het ontwikkelen begon maakte ik naast de master branch[57] een development branch aan. De master branch had als doel om alleen complete functionaliteiten te bevatten en in de development branch kon alles ontwikkeld worden. Een functionaliteit werd pas als compleet gezien wanneer deze na de sprint review goed bevonden werd.

Mijn standaard werkwijze was om elke keer een commit[58] te maken naar de development branch wanneer de werkdag voorbij was of wanneer een bepaalde functionaliteit af was. De reden om na elke werkdag een commit te maken was om rekening te houden met het slechtste scenario. Stel voor dat mijn laptop ineens niet meer zou werken, dan zou de code op de ontwikkelomgeving van Bitman staan waardoor ik op een nieuw systeem verder zou kunnen werken.

Het project managementstool waar Bitman mee werkt is Asana. Deze tool bevat een bord waar allerlei taken op geplaatst kunnen worden en er is de mogelijkheid om verschillende kolommen aan te maken. Ik maakte in totaal zes kolommen aan, deze zijn respectievelijk Backlog, Upcoming, In progress, On hold, Review en Complete. Ik zal deze in de volgende paragraaf kort toelichten.

[55] Laravel docs
[56] What is Command Line Interface (CLI)?
[57] Git Branching - Branches in a Nutshell
[58] Git commit

Voor de ‘Backlog’ kolom werd gebrainstormd en alle ideeën werden hierin gezet. Vervolgens maakte ik daar bijbehorende taken van om de werkzaamheden te kunnen concretiseren. De kolom ‘Upcoming’ had als doel om aan te geven welke taken in de aankomende sprint gedaan zouden worden. Nadat de betreffende sprint van start ging stonden de taken waaraan gewerkt werd in de ‘In progress’ kolom. Indien ik ergens niet uitkwam en er niet gelijk hulp geboden kon worden zette ik een taak in de ‘On hold’ kolom om aan te geven dat de taak tijdelijk gepauzeerd stond. De ‘Review’ kolom gaf taken aan die klaarstonden om gereviewed te worden voor het eind van de sprint. Als laatste de ‘Complete’ kolom dat aangaf welke taken klaar waren. Deze konden pas als af gedefinieerd worden indien de feedback van de opdrachtgever positief was en er goedkeuring was. Hieronder is een screenshot te zien van Asana.



Figuur 15: Asana

5.3.2 Sprint 1

Het thema dat centraal stond in sprint 1 was de database. De bedoeling hierbij was om op basis van het opgestelde klassendiagram de tabellen om te zetten naar models in Laravel volgens de MVC-structuur. Daarnaast was de migratie van de data het doel in deze sprint.

De werkzaamheden die gericht waren op het omzetten van het klassendiagram naar models ging als volgt. Er werd per tabel een model aangemaakt Laravel. Zo werd een tabel met de naam customer_mail als CustomerMail in de models map geplaatst. Ditzelfde principe gold voor de rest van de tabellen. Daarnaast was het van belang om de relaties[59] van de models goed te definiëren, aangezien dat de relationele integriteit van de database ten goede komt. Verder was het cruciaal om de variabelen van de tabellen het juiste datatype[60] mee te geven.

Tijdens deze sprint liep ik tegen een probleem aan. Dat was het feit dat ik thuis geen connectie kon maken met de database. Hierop besloot ik direct om hulp te vragen aan mijn technische begeleider via Slack. We probeerden samen om tot een oplossing te komen. Wat een oplossing kon zijn was om te kijken of het probleem lag bij mijn thuis IP-adres. Ik stuurde mijn IP-adres door naar de technische begeleider, zodat hij handmatig dit adres kon toevoegen aan de toegangslijst. Echter was het probleem nog niet opgelost en het bleek uiteindelijk dat ik alleen via een VPN[61]-verbinding zou kunnen maken vanaf huis. Aangezien deze hulp alleen op kantoor aangeboden kon worden besloot ik om de werkzaamheden ten behoeve van de migratie tijdelijk stop te zetten. Hierop ging ik in overleg met de opdrachtgever hoe ik het beste verder kon gaan. Op basis van het gesprek is

[59] Relationships
[60] Computer Programming - Data Types
[61] Wat is VPN?

besloten om verder te werken aan de taken van de volgende sprint, aangezien ik toen al halverwege de eerste sprint was.

5.3.3 Requirements

Op basis van het gesprek met mijn bedrijfsmentor is gebleken dat de data dat gemigreerd gaat worden naar de nieuwe applicatie een op een beschikbaar moet zijn. Resulterend was de requirement voor deze sprint als volgt: *De data mag bij de migratie niet verloren gaan.* Hiermee werd bedoeld dat de data intact diende te blijven na de migratie.

5.3.4 Review en retrospective

De review werd aan het eind van de sprint online gehouden via Microsoft Teams. Bij de review waren mijn opdrachtgever en technische begeleider betrokken. Hierbij hield ik een presentatie over hoe de progressie tijdens de eerste sprint is gegaan. Daarbij toonde ik hoe het omzetten van het klassendiagram naar de models is gegaan en besprak ik een aantal problemen waar ik tegen aanliep. Na de presentatie kreeg ik gelijk feedback van de opdrachtgever en de technische begeleider. Deze feedback vormde de basis van de retrospective. Tijdens de retrospective bleek dat de initiële opdracht te ambitieus was en niet haalbaar zou zijn. Op basis hiervan heb ik samen met de opdrachtgever besloten om een Proof of Concept[62] te ontwikkelen.

5.3.5 Verantwoording

In de eerste sprint is een deel van de werkzaamheden gelukt, namelijk het omzetten van het database klassendiagram naar models in Laravel. Het andere deel wat helaas niet gelukt is, is de migratie. Een van de belemmeringen was dat ik pas halverwege de sprint erachter kwam dat ik geen connectie kon maken met de database vanaf huis. Het omzetten naar de database modellen nam ongeveer vier dagen in beslag en pas nadat dit gedaan was keek ik naar de migratie en kwam dit probleem naar voren. Mijn technische begeleider kon dit alleen verhelpen op kantoor, omdat ik een VPN van Bitman moest gebruiken. Doordat ik niet gelijk verder kon werken nam ik contact op met de bedrijfsmentor om te hoe ik het beste verder kon gaan. We besloten gezamenlijk dat ik het beste verder kon werken aan de taken van de volgende sprint.

Daarnaast was op basis van de retrospective besloten om de doelstelling van de afstudeeropdracht een klein beetje aan te passen. De applicatie zal niet meer opgeleverd worden als een deelsysteem, maar als een Proof of Concept. Het doel is dus verschoven van de oplevering van een deelsysteem naar de aantoning van een IT-oplossing door een werkend product te demonstreren binnen Bitman.

5.3.6 Sprint 2

Het thema van de tweede sprint was de algemene backend bouwen. Bij het ontwikkelen van de backend is rekening gehouden met het SOLID-principe. De code is zo gebouwd dat elke klasse één verantwoordelijkheid heeft en dat hetzelfde geldt voor elke functie binnen die klasse. Verder is de code zo gebouwd dat het uitbreidbaar is. Het belangrijkste was dus om een blauwdruk te maken waar verder op gebouwd kan worden door andere ontwikkelaars.

In het kader van het MVC-pattern werd alle code met betrekking tot de logica in de map ‘controllers’ geplaatst. Zo werd de logica met betrekking tot de troubleshooter in de ‘TroubleshootController’ gezet. Het belangrijkste voor de Proof of Concept was het werkend maken van de routes naar de verschillende pagina’s. Hiervoor maakte ik gebruik van de documentatie van Laravel en kon ik deze logica maken. De models van het MVC-pattern waren al gedaan in de eerste sprint, tijdens deze sprint werden de controllers opgezet.

[62] Wat is een Proof of Concept (PoC)?

Aangezien er een Proof of Concept ontwikkeld zou worden was het van belang om gebruik te maken van mocking[63]. Voor de functionaliteit van de API-aanroep werd HTTP-fake[64] gebruikt waarbij de JSON API responses werden getest. Door dit te testen kon er gecontroleerd worden of de output overeen zou komen met wat er verwacht werd.

5.3.7 Requirements

De requirement die bepaalde wanneer deze sprint af zou zijn is de volgende technische requirement uit het functioneel ontwerp:

- T2: De tool moet uitbreidbaar zijn voor verschillende organisaties

In het kader van het SMART-principe heb ik deze requirement verder gespecificeerd. De tool verwees naar de helpdesktool waar de troubleshooter onderdeel van is. Uitbreidbaar hield in dat er één backend gebouwd werd dat de basis vormde zodat meerdere organisaties het zouden kunnen gebruiken. Er werd met organisaties specifiek bedoeld organisaties die gebruikmaken van een klantenservice. In de SMART-formulering ziet was de requirement als volgt: *De troubleshooter moet zo gebouwd zijn dat meerdere soorten organisaties met een klantenservice deze kunnen gebruiken.*

5.3.8 Review en retrospective

Aan het eind van de tweede sprint werd weer een online review gehouden via Microsoft Teams. Ik hield een presentatie over wat er gedaan was, welke problemen er waren en er was een kleine demonstratie over hoe ik de backend had opgezet. Aan het eind van de presentatie was er gelegenheid voor de opdrachtgever en technische begeleider om vragen te stellen en feedback te geven. Er was voornamelijk positieve feedback op een paar kleine punten na die ik meenam naar de volgende sprint.

5.3.9 Verantwoording

In deze sprint waren alle geplande werkzaamheden hiervoor uiteindelijk gelukt. Dat waren het bouwen van de algemene backend, het uitbreidbaar opzetten van de code en het testen van de API-responses. Er waren relatief weinig problemen tijdens deze sprint en wanneer ik ergens vastliep probeerde ik altijd in eerste instantie het zelf op te lossen. Wanneer dit niet lukte nam ik contact op met de technische begeleider die mij goed kon ondersteunen.

5.3.10 Sprint 3

Tijdens deze sprint was het doel om de backend met de front-end te combineren en die werkend te maken. Ik had tijdens de ontwikkelfase de front-end al gebouwd met HTML en CSS met Bootstrap en deze in de views geplaatst op basis van het MVC-pattern. De benodigde pagina's voor het Proof of Concept waren dus al klaar en deze moest ik combineren met de backend om een deels werkend product te maken.

5.3.11 Requirements

Tijdens deze sprint was de volgende functionele requirement van het functioneel ontwerp het punt van aandacht:

- F1: De tool moet ondersteuning bieden aan callcenter medewerkers om klanten te helpen.

Deze requirement is verder gespecifieerd volgens het SMART-principe. De tool verwees naar de helpdesktool waar de troubleshooter onderdeel van is. Een klant is een persoon die een dienst afneemt of een product heeft gekocht van een organisatie die klantenservice aanbiedt. De nieuw geformuleerde requirement was *'De troubleshooter moet ondersteuning bieden aan supportmedewerkers zodat zij hun klanten kunnen helpen.'*

[63] Mocking
[64] HTTP Fake

Naast de bovenstaande requirement waren er user stories van het functioneel ontwerp die voor deze sprint werden gebruikt. Deze user stories waren in twee categorieën gesplitst op basis van de gebruikersgroepen. Dat zijn de tweedelijns supportmedewerkers, ook bekend als de beheerders, en de supportmedewerkers. Deze user stories werden naar een requirement gespecifieerd op basis van het SMART-principe.

User stories beheerders

User story	Requirement
Als tweedelijns supportmedewerker wil ik een nieuwe troubleshooting wizard kunnen aanmaken, zodat klantenservice medewerkers onze klanten kunnen helpen.	De troubleshooter moet de mogelijkheid bieden aan de beheerder om een troubleshooting item aan te maken.
Als tweedelijns supportmedewerker wil ik de troubleshooting wizard structuur goed kunnen zien, zodat ik het troublehooting proces kan begrijpen.	De troubleshooter moet een overzichtelijke structuur tonen van de troubleshooting decision tree*. *De datastructuur van te volgen stappen in het troubleshooting proces.
Als tweedelijns supportmedewerker wil ik de bestaande items kunnen bewerken in de wizard, zodat ik geen nieuwe items aan hoef te maken bij het ontvangen van feedback van de klantenservice medewerkers.	Een item van de decision tree moet te bewerken zijn door de beheerder.
Als tweedelijns supportmedewerker wil ik zien wat de vorige en volgende stappen zijn, zodat ik de structuur van de wizard kan begrijpen bij het creëren of bewerken van stappen in de wizard.	De decision tree moet overzichtelijk zijn waarbij de stappen duidelijk te zien zijn voor de beheerder.

User stories supportmedewerkers

User story	Requirement
Als support agent wil ik de klant kunnen helpen bij het oplossen van een probleem, zodat ik mijn ticket kan afronden.	De troubleshooter moet ondersteuning bieden aan supportmedewerkers zodat zij hun klanten kunnen helpen
Als support agent wil ik duidelijke instructies hebben om de klant te helpen, zodat ik het probleem gemakkelijk kan oplossen.	De informatie binnen de decision tree moet duidelijk zijn.
Als support agent wil ik bij het ondervragen van de klant de feedback gemakkelijk kunnen indienen, zodat ik door kan gaan in het troubleshooting proces.	Bij stappen in de decision tree waarbij gegevens moeten worden genoteerd, moet dit ingediend kunnen worden.

5.3.12 Review

Aangezien dit de laatste sprint was, was er alleen een review en geen retrospective. Deze laatste review vond op kantoor plaats met de opdrachtgever en de technische begeleider was er online bij via Teams. Tijdens deze review gaf ik een presentatie over de gedane werkzaamheden en een korte demonstratie van het Proof of Concept van de troubleshooter. Na afloop van de presentatie gaf ik de opdrachtgever de gelegenheid om zelf het Proof of Concept uit te proberen.

5.3.13 Verantwoording

Tijdens deze sprint was er ongeveer twee dagen besteed aan het concretiseren van de gekozen requirements en user stories. De keuze werd gemaakt uit de opgestelde requirements en user stories uit het functioneel ontwerp. Hierbij werd uiteraard gekeken naar welke requirement of user story de hoogste prioriteit hadden. Nadat dit gedaan was werd de rest van de tijd besteed aan het koppelen van de backend met de frontend. Dit ging redelijk goed doordat ik de documentatie van Laravel al had geraadpleegd tijdens de verkennings-en ontwerpfase.

5.3.14 Beroepstaken

Tijdens de ontwikkelfase kwamen een aantal beroepstaken aan bod zoals ik in het afstudeerplan heb benoemd. Ik zal ze hieronder benoemen en wat ik daarvoor gedaan heb.

D1: Realiseren van software

Tijdens het ontwikkelen van het Proof of Concept heb ik deze beroepstaak kunnen aantonen. Dat is op basis van het programmeren volgens het SOLID-principe, het MVC-pattern en het opzetten van zowel de backend als de frontend. Ik begon eerst met de backend waarbij ik de basisfunctionaliteiten inbouwde en vervolgens combineerde ik de backend met de frontend om de pagina's werkend te maken. Tijdens het realiseren van het Proof of Concept maakte ik een aantal tussenproducten die tijdens de sprint reviews gedemonstreerd werden. Hierop volgde feedback waarbij een aantal verbeteringen en aanpassingen doorgevoerd werden. Bovendien heb ik de API-responses getest om te controleren of de werking hiervan klopte.

Gb: (Internationaal) samenwerken

Deze beroepstaak kwam elke sprint aan bod. Zo besprak ik samen met mijn opdrachtgever wat het thema van elke sprint was en welke requirements behandeld zouden worden. Daarnaast maakten we gebruik van het project managementstool Asana. Hierin werden de taken waaraan gewerkt zou

worden voor de sprint in de kolom ‘upcoming’ geplaatst. Wanneer er daadwerkelijk aan de taak gewerkt werd, dan werd deze in de kolom ‘in progress’ gezet. Door dit te doen kon de opdrachtgever direct zien met welke taken ik bezig was. Indien ik ergens niet uitkwam schakelde ik hulp in van mijn technische begeleider.

Daarnaast was er wekelijks een bespreking met mijn opdrachtgever over de voortgang halverwege de sprint. Door deze besprekingen wekelijks te doen konden eventuele problemen gedetecteerd worden en opgelost worden. Bovendien kon er dan gekeken worden of de sprint wel haalbaar zou zijn en deze indien nodig aangepast kon worden.

Als laatst was er aan het eind van elke sprint een review waarin ik aan mijn opdrachtgever en technische begeleider presenteerde hoe de werkzaamheden gegaan zijn en besprak ik eventuele problemen waar ik tegenaan liep.

Gc: Kritisch, onderzoekend en methodisch werken

Elke sprint pakte ik requirements uit het functioneel ontwerp. Deze waren af en toe nog te algemeen waardoor ik deze specifieker ging maken met gebruik van het SMART-principe. Daarnaast werkte ik volgens een redelijk vaste werkwijze in de ontwikkelfase, zoals beschreven in paragraaf 5.3.1 ‘**Algemene werkwijze**’.

Gf: Leren leren: voorbereiden op volgende studiefase en beroep

Door aan het eind van elke sprint een review te houden met een daaropvolgende retrospective kon ik mij goed inwerken en wennen aan hoe er in het bedrijfsleven met SCRUM omgegaan kan worden. Bovendien kon ik van mijn fouten leren op basis van de feedback die ik kreeg en kon ik deze verbeteren in een volgende sprint. Zo werd er aan mij verteld dat ik af en toe iets eerder aan de bel mocht trekken als ik problemen had of ergens vastliep. Ik heb namelijk soms de neiging om te veel zelfstandig het probleem proberen op te lossen waardoor ik zo nu en dan meer tijd nodig had.

6. Evaluatie en reflectie

Achteraf gezien ben ik redelijk tevreden over hoe ik heb gefunctioneerd tijdens het afstudeertraject. Ik verwachtte van tevoren dat de uitdaging nog groter zou zijn in verband met het op afstand werken. De communicatie tussen mij, de bedrijfsmentor en technische begeleiders verliep over het algemeen goed. Ik heb tijdens het afstudeertraject veel kunnen leren, zo leerde ik werken met het Laravel framework waar ik nog niet eerder mee heb gewerkt. Het eindresultaat wat ik heb kunnen opleveren is een proof of concept waarin getoond wordt hoe de nieuwe troubleshooter eruit zou komen te zien.

Er zijn voor mij ook een aantal belangrijke leerpunten. Als eerst dat ik bij een probleem niet te lang zelfstandig het probleem moet proberen op te lossen, maar dat ik eerder aan de bel moet trekken. Ten tweede heb ik ervaren hoe het is om op afstand voor een bedrijf te werken en hierdoor ben ik beter voorbereid hoe het zal zijn om later (op afstand) te werken.

6.1 Afwijkingen afstudeerplan

Er is bij het uitvoeren van het afstudeertraject enigszins afgeweken van het afstudeerplan. Oorspronkelijk was het de bedoeling om een nieuw deelsysteem ten behoeve van de troubleshooter en de decision tree op te leveren. Dit bleek een te ambitieuze opdracht waardoor ik samen met mijn bedrijfsmentor besloten heb om een Proof of Concept op te leveren waarbij ik met een werkend deelproduct kon aantonen of de IT-oplossing geschikt zou zijn voor het doel wat Bitman wilde bereiken.

6.2 Aanpak

Terugkijkend naar het gehele afstudeertraject ben ik grotendeels tevreden over de aanpak en planning die ik van tevoren heb opgesteld. Achteraf gezien was ik meer tijd kwijt aan de verkennings-en onderzoeksfase en de ontwerpfase. De reden hiervoor was dat ik verkeerd had ingeschat hoeveel tijd ik kwijt zou zijn hieraan waardoor ik een beetje achter begon te lopen op de planning. Gelukkig heb ik dit bespreekbaar gemaakt met de bedrijfsmentor waardoor we tot een passende oplossing zijn gekomen.

6.3 Thuiswerken

Deze hele afstudeertraject is in het midden van de coronapandemie afgelegd waardoor het thuiswerken de norm was. Als afstudeerder vond ik het een lastige periode om af te studeren. Je wilt je namelijk betrokken voelen bij het bedrijf. In mijn ervaring is dat het beste te doen wanneer ik de bedrijfscultuur goed kan meemaken. Echter werkte ik maximaal een keer per week op kantoor waardoor ik dit gevoel miste. Het zorgde er soms ook voor dat ik mij minder gemotiveerd voelde, omdat ik mij niet heel erg betrokken voelde. Uiteraard deed ik mijn best om gemotiveerd te blijven.

Bovendien kreeg ik niet altijd direct antwoord wanneer een vraag had en deze online via Slack stelde. Dat is begrijpelijk omdat mijn collega’s en technische begeleider vaak met hun eigen werkzaamheden bezig waren. Het zou anders geweest zijn als ik gewoon de hele week op kantoor zou kunnen werken met mijn collega’s. Dan had ik direct een vraag kunnen stellen en konden ze mij sneller helpen. Als er geen pandemie geweest zou zijn dan denk ik dat ik het afstuderen heel anders zou ervaren en had ik wellicht meer dan een proof of concept kunnen opleveren.

6.4 Beroepstaken

Een beroepstaak houdt in dat ik als afstudeerder aan kan tonen dat ik in het bezit ben van bepaalde competenties. Zo moet ik laten zien dat ik de theoretische kennis, die tijdens de studie opgedaan is, in de praktijk kan toepassen bij een afstudeeropdracht zoals deze. In de volgende paragrafen zal ik toelichten welke beroepstaken ik gedaan heb en wat de bijbehorende werkzaamheden hierbij waren.

6.4.1 A1: Analyseren probleemdomein & opstellen doelstelling

Deze beroepstaak heb ik aangetoond tijdens de verkenning waarbij ik mij oriënteerde op de afstudeeropdracht. Hierbij kon ik een probleemstelling opstellen waarbij ik deze in een domein kon plaatsen. Het was duidelijk geworden dat het domein zich in de klantenservice omgeving bevond. Vervolgens kon ik op basis van de probleemstelling een passende doelstelling opstellen. Nadat dit klaar was begon ik met het maken van het plan van aanpak. Hierin maakte ik de planning, beschreef ik de randvoorwaarden en maakte ik een risicoanalyse. Wat ik hiervoor heb opgeleverd is het plan van aanpak die ik samen met de opdrachtgever besprak. Zo konden we samen kijken of de opgestelde planning wel realistisch en op orde zou zijn. Alle details van het plan van aanpak zijn in de externe bijlage '**A.1.4 Plan van Aanpak Quang Hang - 17069866.docx**' te vinden.

6.4.2 A4: Uitvoeren van onderzoek

Deze beroepstaak heb ik aangetoond bij het uitvoeren van het onderzoek. Alvorens ik begon met het onderzoek stelde ik de hoofdvraag op. Voordat ik verder ging met de deelvragen was ik kritisch en keek ik of de hoofdvraag niet te kort of te lang was. Vervolgens ging ik de deelvragen opstellen en telkens evalueerde ik of de deelvragen wel antwoord geven op de hoofdvraag. Verder heb ik zelfstandig deskresearch gedaan naar de verschillende softwareoplossingen waarbij ik kritisch keek naar de kenmerken van elke tool. Vervolgens vergeleek ik de kenmerken met elkaar en keek ik of er overeenkomsten waren. Op basis van de overeenkomende kenmerken kon ik concluderen dat deze van belang zijn. Het gehele onderzoek is te vinden in de externe bijlage '**A.1.3 Onderzoeksrapport Quang Hang - 17069866.docx**'.

6.4.3 C1: Ontwerpen software

Deze beroepstaak heb ik aangetoond aan de hand van de ontwerpen die in de opgeleverde tussenproducten (functioneel & technisch ontwerp) staan. In de scriptie staat de motivatie achter de ontwerpen beschreven bij de werkzaamheden. Van het functioneel ontwerp heb ik de stakeholderanalyse, de softwarerequirements en de use cases zelf achterhaald en ontworpen. De user stories waren gedocumenteerd op basis van de huidige troubleshooter.

In het technisch ontwerp heb ik alles zelf ontworpen behalve het klassendiagram van de database. Dit klassendiagram heb ik relationeel verbeterd door de relaties tussen de tabellen te maken. De kwaliteit stelde ik vast door met de opdrachtgever en technische begeleider deze deelproducten te bespreken. De deelproducten zijn te vinden in de externe bijlages '**A.1.2 Functioneel ontwerp Quang Hang - 17069866.docx**' en '**A.1.5 Technisch ontwerp Quang Hang - 17069866.docx**'.

6.4.4 D1: Realiseren van software

Deze beroepstaak heb ik tijdens de ontwikkelfase toegepast waarbij ik volgens het SOLID-principe en het MVC-pattern heb gecodeerd. Hierbij werden een aantal requirements geïmplementeerd die betrekking hadden op de uitbreidbaarheid van de code en de gebruikerservaring. Per sprint leverde ik een deelproduct op die vervolgens in de sprint reviews werden besproken. De feedback die ik kreeg leidde tot eventuele verbeteringen en aanpassingen. Deze deelproducten vormden samen het opgeleverde Proof of Concept. Tijdens de ontwikkelfase heb ik de API-responses getest om te controleren of de functie volgens verwachting werkte.

6.4.5 Gb: (Internationaal) samenwerken

Deze beroepstaak werd constant toegepast tijdens het afstudeertraject. Voor de voorbereidings-en onderzoeksfase was het van belang dat ik mijzelf kon oriënteren op het huidige systeem. Dit gaf ik aan bij de bedrijfsmentor die vervolgens toegang regelde voor mij. In de ontwerpfase stuurde ik het functioneel ontwerp op naar de opdrachtgever om feedback te krijgen. Op basis hiervan moest ik een aantal dingen concreter beschrijven en aanpassen. Hetzelfde gold voor het technisch ontwerp die ik op basis van de feedback nog kon verbeteren.

Tijdens de ontwikkelfase is deze beroepstaak toegepast aan de hand van de sprint reviews en de retrospectives. In de reviews communiceerde ik de voortgang door middel van een presentatie en

een korte demonstratie van de deelproducten. Vervolgens kreeg ik terugkoppeling dat de basis vormde voor de retrospective waarin gekeken werd wat er verbeterd kon worden. Bovendien had ik wekelijks contact met mijn bedrijfsmentor om de voortgang en eventuele problemen halverwege de sprints te bespreken. Vanwege de coronapandemie werd er voornamelijk thuis gewerkt en de communicatie verliep via Slack en Microsoft Teams.

6.4.6 Gc: Kritisch, onderzoekend en methodisch werken

In de voorbereidings-en onderzoeksfase heb ik deze beroepstaak toegepast door kritisch te kijken naar welke programmeertalen en frameworks geschikt zouden kunnen zijn voor het uitvoeren van de afstudeeropdracht. Op basis van de gevonden programmeertalen deed ik onderzoek naar welke voor-en nadelen deze hadden. Op basis van dit onderzoek maakte ik een weloverwegen keuze waarbij ik keek naar een aantal factoren, zoals documentatie, open-source en het ecosysteem.

Daarnaast was er in de onderzoeksfase een onderzoeksrapport wat ik had gemaakt met de bijbehorende hoofd-en deelvragen. Hierbij was het enorm belangrijk om geen aannames te maken, maar om echt kritisch te zijn.

Voor de ontwerpfase heb ik deze beroepstaak toegepast in het functioneel ontwerp waarbij ik voor de requirements onderscheid maakte tussen de functionele en technische requirements. Ik maakte een stakeholderanalyse waarbij ik op basis van een methodiek uit het Handboek Requirements de belanghebbenden kon identificeren.

Voor de ontwikkelfase heb ik deze beroepstaak toegepast door de requirement indien nodig verder te specificeren indien volgens het SMART-principe. Bovendien werkte ik volgens een algemene werkwijze zoals beschreven in paragraaf 5.3.1 ‘**Algemene werkwijze**’.

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

Aan het eind van elke sprint in de ontwikkelfase werd een retrospective gehouden waarbij gekeken werd wat er goed en fout ging en welke verbeteringen er mogelijk waren. Een positief punt bleek dat ik goed zelfstandig kon werken, echter was dit in een aantal gevallen ook een punt ter verbetering. Wanneer ik een probleem had probeerde ik soms te lang om het zelfstandig om te lossen en vroeg ik later pas om hulp. Wat ik hierbij beter had kunnen doen is om eerder aan de bel te trekken.

Verder heb ik dankzij het afstudeertraject mij in een bijzondere situatie, de coronapandemie, kunnen voorbereiden voor een latere baan. Het is mogelijk dat het thuiswerken nog deels de norm zal blijven na de pandemie, waarbij ik heb geleerd om vanuit huis te werken en online contact te houden met mijn collega’s.

Begrippenlijst

Begrip	Betekenis
Object-oriented programming (OOP)	Het is een manier van programmeren dat gebruikmaakt van het concept van klassen en objecten. Elk object en data structuur heeft eigen attributen en eigenschappen. Bijvoorbeeld: een auto (object) heeft attributen zoals merk, kleur en bouwwjaar.
Procedural Programming	Het is een programmeermodel dat is afgeleid van imperatief programmeren, gebaseerd op het concept van de producedureaanroep. Deze procedures bestaan uit een reeks van computationele stappen die moeten worden uitgevoerd.
Memory leak	Gebeurtenis waarbij een applicatie het geheugen van de computer onjuist beheert. Het gevolg hiervan kan zijn dat een applicatie of het systeem helemaal vast komt te lopen.
Deadlock	Dit is een situatie waarbij twee computerprocessen van elkaars componenten afhangen. Indien de processen aan deze componenten gekoppeld zijn kan het voorkomen dat de processen op elkaar blijven wachten waardoor het systeem vastloopt. Voorbeeld: • Proces 1 vraagt component B aan van proces 2. • Component B is gelocked terwijl proces 2 draait. • Proces 2 vereist dat component A van proces 1 klaar is met draaien. • Component A is gelocked terwijl proces 1 draait.

Literatuurlijst

1. Swart, N. (2017). Handboek Requirements. Reed Business Education.
2. Baarda, B. (2014). Dit is onderzoek! incl. toegang tot Prepzone (2de editie). Noordhoff.
3. Randen, H. J. (2015). Aan de slag met Scrum (1ste ed.). Academic Service.
4. H-SE-OOPR-1-15-2017: 2017 Object orientation, Den Haag P3, Programming and Testing (studiemateriaal BlackBoard)
5. H-SE-DBPR1-1-15-DH: 2017 DH Database Applications, DBDS Database design (studiemateriaal BlackBoard)
6. H-SE-PRPR-1-16-DH-2018: 2018 Advanced Programming Den Haag, Programming and Testing 2 (studiemateriaal BlackBoard)
7. B. (2021, 28 januari). A Tour of C# - C# Guide. Microsoft Docs. https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/
8. Szecsei, S. (2020, 2 juli). ASP.NET – Pros and Cons You Ought to Know. PopArt Studio. https://www.popwebdesign.net/popart_blog/en/2020/03/asp-net-pros-and-cons-you-ought-to-know/
9. Pros and Cons of Using C# as Your Backend Programming Language. (2017, 11 januari). Pros and Cons of Using C# as Your Backend Programming Language. https://agilites.com/pros-and-cons-of-using-c-as-your-backend-programming-language.html
10. C++ Tutorial - Tutorialspoint. (z.d.). C++ Tutorial - Tutorialspoint. https://www.tutorialspoint.com/cplusplus/index.htm
11. Team, D. (2020, 3 februari). Advantages and Disadvantages of C++ | Make your Next Move! DataFlair. https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/
12. GeeksforGeeks. (2020, 31 december). Advantages and Disadvantages of C++. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-c/
13. Team, D. (2021, 17 februari). Pros and Cons of Java | Advantages and Disadvantages of Java. DataFlair. https://data-flair.training/blogs/pros-and-cons-of-java/
14. Editor. (2020, 28 februari). The Good and the Bad of Java Programming. AltexSoft. https://www.altexsoft.com/blog/engineering/pros-and-cons-of-java-programming/
15. Cary, I. (2020, 7 november). Python: Pros and Cons. Scholarlyoa.Com. https://scholarlyoa.com/python-pros-and-cons/
16. Pros & Cons of Using Python For Web Development. (z.d.). Thepythonguru.Com. https://thepythonguru.com/pros-cons-of-using-python-for-web-development/
17. Korsun, J. (2021, 3 maart). The 16 Most Important Pros and Cons of using Python for Web Development. Django Stars Blog. https://djangostars.com/blog/python-web-development/
18. The Pros and Cons of Programming with PHP and .NET | Informatics Inc. (z.d.). All Things Internet | Informatics Inc. https://www.informaticsinc.com/blog/september-2018/pros-and-cons-php-and-net/amp
19. Solutions, M. (2018, 6 juni). Advantages and Disadvantages of PHP Frameworks - Mindfire Solutions. Medium. https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-php-frameworks-c046d50754e5
20. Pros and cons of PHP web development. (z.d.). SapientPro. https://sapient.pro/blog/pros-and-cons-of-php-web-development/
21. GeeksforGeeks. (2020a, november 10). Advantages and Disadvantages of PHP. https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/

22. Morris, W. (2021, 9 maart). 8 Best PHP Frameworks for Web Developers. Hostinger Tutorials. https://www.hostinger.com/tutorials/best-php-framework
23. Bhatia, S., Patel, R., Blalock, B., E., J., Mahajan, R., Cyrus, M., Owen, T., Southard, L., Barnett, E., Whitney, A., Medina, M., Purcell, G., Love, E., & Bernal, R. (z.d.). 10 Best PHP Frameworks for Web Development in 2021 [Updated]. Hackr.Io. https://hackr.io/blog/best-php-frameworks
24. 10 Popular PHP frameworks to consider. (2018, 21 november). Raygun Blog. https://raygun.com/blog/top-php-frameworks/
25. Emerce. (2017, 24 januari). Software-ontwikkeling: welke methoden zijn er en waarin verschillen ze? https://www.emerce.nl/best-practice/software-ontwikkeling-welke-methoden-zijn-er-en-waarin-verschillen-ze
26. McCormick, M. (2012). Waterfall vs. Agile Methodology. Waterfall vs. Agile Methodology, 5–6. http://www.mccormickpcs.com/images/Waterfall_vs_Agile_Methodology.pdf
27. Wat is een API en wat kan je ermee? (z.d.). Salesforce NL Blog. https://www.salesforce.com/nl/blog/2019/10/wat-is-een-api.html
28. Siegel, C. (2021, 12 april). What is an API? API Friends. https://apifriends.com/api-management/what-is-an-api/
29. Database Documentation tools for MySQL - DBMS Tools. (z.d.). DBMS Tools. https://dbmstools.com/categories/database-documentation-tools/mysql
30. Guided Decisions for Customer Service Management https://docs.servicenow.com/bundle/quebec-customer-service-management/page/product/customer-service-management/concept/guided-decisions.html
31. Britton, M (1997-2002). Getting started with PHP Fusebox https://download.itadmins.net/Programming/PHP/%5BDeveloper%20Shed%20Network%5D%20Server%20Side%20-%20PHP%20-%20Scratching%20the%20Surface%20-%20Getting%20Started%20with%20PHP%20Fusebox.pdf
32. What is Elasticsearch? (z.d.). Elastic. Geraadpleegd op 9 februari 2021, van https://www.elastic.co/what-is/elasticsearch
33. Kinsta. (2021, 28 mei). What Is A Content Management System (CMS)? https://kinsta.com/knowledgebase/content-management-system/
34. A. (2020, 12 februari). Overview of ASP.NET Core MVC. Microsoft Docs. https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0
35. Introduction What Is Vue.Js? (z.d.). What Is Vue.Js? Geraadpleegd op 11 februari 2021, van https://vuejs.org/v2/guide/
36. Installation - Laravel - The PHP Framework For Web Artisans. (z.d.). Laravel Docs. Geraadpleegd op 15 februari 2021, van https://laravel.com/docs/8.x
37. Introduction to HTML - Learn web development | MDN. (2021, 5 mei). Introduction to HTML. https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML
38. MySQL :: MySQL 8.0 Reference Manual :: 1.2.1 What is MySQL? (z.d.). What Is MySQL? Geraadpleegd op 15 februari 2021, van https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
39. Atlassian. (z.d.). Git bash: Definition, commands, & getting started. Geraadpleegd op 19 februari 2021, van https://www.atlassian.com/git/tutorials/git-bash
40. Atlassian. (z.d.-b). Sourcetree | Free Git GUI for Mac and Windows. SourceTree. Geraadpleegd op 19 februari 2021, van https://www.sourcetreeapp.com/
41. Atlassian. (z.d.-a). Bitbucket | The Git solution for professional teams. Bitbucket. Geraadpleegd op 19 februari 2021, van https://bitbucket.org/product/

42. What Is an IDE? (z.d.). Codecademy. Geraadpleegd op 12 februari 2021, van https://www.codecademy.com/articles/what-is-an-ide
43. Rungta, K. (2021, 28 mei). Difference between Abstraction and Encapsulation. Difference between Abstraction and Encapsulation. https://www.guru99.com/difference-between-abstraction-and-encapsulation.html
44. GeeksforGeeks. (2020, 8 februari). What is Dynamic Memory Allocation? https://www.geeksforgeeks.org/what-is-dynamic-memory-allocation/
45. Singh, C. (2017, 11 september). Pointers in C++. Beginnersbook.Com. https://beginnersbook.com/2017/08/cpp-pointers/
46. What Is “Write Once and Run Anywhere” Feature of Java? (z.d.). What Is “Write Once and Run Anywhere” Feature of Java? Geraadpleegd op 16 februari 2021, van https://www.w3schools.in/java-questions-answers/write-once-run-anywhere-wora/
47. Rungta, K. (2021, 23 april). JDK vs JRE vs JVM: Key Differences. JDK vs JRE vs JVM: Key Differences. https://www.guru99.com/difference-between-jdk-jre-jvm.html
48. Techopedia. (2017, 2 februari). Bracket. Techopedia.Com. https://www.techopedia.com/definition/18050/bracket
49. Education, I. C. (2021, 30 maart). LAMP Stack. IBM. https://www.ibm.com/cloud/learn/lamp-stack-explained
50. Wilson, L. I. (2020, 31 mei). What Are Compilers And Programming Languages - The Startup. Medium. https://medium.com/swlh/crafting-compilers-intro-47d7a6985665
51. HMVC structure. (z.d.). DotKernel PSR-15 Middleware Applications. Geraadpleegd op 18 februari 2021, van https://www.dotkernel.com/docs/hmvc-structure/
52. W. (z.d.). SMART doelen: voorbeelden uitgewerkt! Wat kun je ermee? De Steven training | coaching | outplacement. Geraadpleegd op 18 februari 2021, van https://www.desteven.nl/leerdoelen/smart-leerdoelen
53. S. (z.d.-a). Welcome to your new HQ. Slack. Geraadpleegd op 2 maart 2021, van https://slack.com/intl/en-nl/
54. Wat is een API en wat kan je ermee? (2019, 23 oktober). Salesforce NL Blog. https://www.salesforce.com/nl/blog/2019/10/wat-is-een-api.html
55. What is User Interface Design? (z.d.). The Interaction Design Foundation. Geraadpleegd op 15 maart 2021, van https://www.interaction-design.org/literature/topics/ui-design
56. Controllers - Laravel - The PHP Framework For Web Artisans. (z.d.). Laravel. Geraadpleegd op 17 maart 2021, van https://laravel.com/docs/8.x/controllers
57. Otto, M. J. T. (z.d.). Introduction. Bootstrap. Geraadpleegd op 18 maart 2021, van https://getbootstrap.com/docs/5.0/getting-started/introduction/
58. What is CLI. (z.d.). W3schools. Geraadpleegd op 19 april 2021, van https://www.w3schools.com/whatis/whatis_cli.asp
59. Git - Branches in a Nutshell. (z.d.). Git. Geraadpleegd op 19 april 2021, van https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell
60. Atlassian. (z.d.). Git Commit | Atlassian Git Tutorial. Geraadpleegd op 19 april 2021, van https://www.atlassian.com/git/tutorials/saving-changes/git-commit
61. Eloquent: Relationships - Laravel - The PHP Framework For Web Artisans. (z.d.). Laravel. Geraadpleegd op 20 april 2021, van https://laravel.com/docs/8.x/eloquent-relationships
62. Computer Programming - Data Types - Tutorialspoint. (z.d.). Tutorialspoint. Geraadpleegd op 20 april 2021, van https://www.tutorialspoint.com/computer_programming/computer_programming_data_types.htm

63. Kamp, R. (2020, 14 december). VPN: je internetverbinding versleutelen. Consumentenbond. https://www.consumentenbond.nl/veilig-internetten/veiliger-internetten-met-een-vpn

64. P. (2020, 4 november). Wat is een Proof of Concept (PoC)? agile4all. https://www.agile4all.nl/wat-is-een-proof-of-concept-poc/

65. Mocking - Laravel - The PHP Framework For Web Artisans. (z.d.). Laravel. Geraadpleegd op 7 mei 2021, van https://laravel.com/docs/8.x/mocking

A. Bijlagen

A.1 Externe bijlagen

A.1.1 Afstudeerplan Quang Hang - 17069866.docx

A.1.2 Functioneel ontwerp Quang Hang - 17069866.docx

A.1.3 Onderzoeksrapport Quang Hang - 17069866.docx

A.1.4 Plan van Aanpak Quang Hang - 17069866.docx

A.1.5 Technisch ontwerp Quang Hang - 17069866.docx