Scriptie

Producten in een multisite CMS

Auteur	**Jonathan Karssen**
Studentnummer	**1630460**
Opleiding	**HBO-ICT**
Studierichting	**Informatica** **Software Engineering**
Onderwijsinstelling	**Hogeschool Utrecht**
Afstudeerbedrijf	**SalesCare**
Docentbegeleider	**Remco Ruijsenaars**
Bedrijfsbegeleider	**Tamer Saglambilek**
Eerste Examinator	**Jeroen Weber**

Versiebeheer

Versie	Status	Datum	Opmerkingen
0.1	Concept	12-9-2016	- Eerste versie
0.2	Concept	3-10-2016	- Opmerkingen docentbegeleider en leerteam verwerkt - Deelvraag 5 beantwoord - Conclusies en aanbevelingen toegevoegd
0.3	Concept	5-10-2016	- Managementsamenvatting en voorwoord toegevoegd - Opmerkingen leerteam verwerkt
0.4	Concept	13-10-2016	- Opmerkingen docentbegeleider verwerkt
0.5	Concept	17-10-2016	- Opmerkingen leerteam en familie verwerkt
0.6	Concept	20-10-2016	- Opmaak verbeterd en kleine verbeteringen aangebracht
1.0	Definitief	21-10-2016	- Opmerkingen docentbegeleider verwerkt

Voorwoord

Voor u ligt de scriptie ‘Producten in een multisite CMS’. Deze scriptie is geschreven tijdens mijn afstudeeropdracht voor de opleiding Informatica aan de Hogeschool Utrecht. De opdracht is uitgevoerd in opdracht van het online marketingbureau SalesCare. Van 30 mei tot en met 21 oktober 2016 ben ik bezig geweest met het onderzoeken en ontwikkelen van een oplossing om een grote hoeveelheid producten in een CMS voor websites toegankelijk te maken.

Via deze weg wil ik graag een aantal mensen bedanken voor hun hulp. Ten eerste wil ik mijn bedrijfsbegeleider Tamer Saglambilek hartelijk bedanken voor het aanbieden van de stageplek en het grote vertrouwen dat hij in mij heeft gehad de afgelopen periode. Ook wil ik mijn hartelijke dank uitspreken voor de begeleiding van Remco Ruijsenaars als docentbegeleider. Hij heeft me goed geholpen met feedback op het proces en op mijn scriptie.

Daarnaast wil ik mijn collega’s bedanken voor de samenwerking en de feedback op de aangedragen oplossingen. Zij hebben me vaak verder geholpen als ik ergens mee zat. Tot slot wil ik hartelijk mijn familie en vrienden bedanken. Zonder de gezellige avonden en diners was deze periode een stuk zwaarder geweest.

Ik wens u veel leesplezier.

Jonathan Karssen

Utrecht, oktober 2016

Managementsamenvatting

SalesCare heeft een klant in de technische industrie met de behoefte aan een Content Management Systeem (CMS), waarmee verschillende websites beheerd kunnen worden. SalesCare is bezig met het bouwen van dit CMS en is in een vergevorderd stadium om statische websites te kunnen beheren. De klant wil echter ook hun 500.000 producten op de websites kunnen laten zien. De student heeft onderzoek uitgevoerd om dit mogelijk te maken binnen het CMS van SalesCare.

De hoofdvraag voor dit onderzoek was als volgt:
"*Hoe kan de architectuur en het ontwerp van het multisite CMS van SalesCare aangepast worden om de nieuwe functionaliteiten mogelijk te maken en hoe kunnen de kwaliteitscriteria van deze uitbreiding het beste gegarandeerd worden door een goede teststrategie?*"

Om de vraag te beantwoorden zijn de volgende deelvragen opgesteld:

1. Wat is de huidige situatie van het CMS?
2. Wat zijn de benodigde functionaliteiten van de uitbreiding?
3. Wat zijn de gestelde eisen en kwaliteitscriteria aan het systeem?
4. Welke architectuur en ontwerpoplossing voldoen aan de gestelde eisen en kwaliteitscriteria?
5. Hoe kunnen de kwaliteitscriteria het beste getest worden?

Uit het onderzoek bleek dat er veel extra functionaliteiten nodig waren, die niet rechtstreeks met het tonen van de producten te maken hadden. Deze functionaliteiten waren echter wel belangrijk voor het beheren van de producten en voor de bestanden die bij de producten horen. Er is een Functioneel Ontwerp gemaakt (zie bijlage 1) om alle functionaliteiten op een rijtje te zetten.

Vervolgens is gekeken naar de kwaliteitscriteria waar het systeem aan moet voldoen. Uit de ISO 25010 norm zijn de belangrijkste kwaliteitscriteria gepakt die gelden voor het CMS. Hieruit bleek dat vooral aan de Performance Efficiency, Maintainability, Reliability en Security belangrijke eisen werden gesteld. Toen de kwaliteitscriteria waren vastgesteld, werd ontdekt dat de huidige architectuur van het systeem niet meer voldeed aan de kwaliteitscriteria. Hiervoor is eerst een oplossing gezocht door het project te splitsen naar een beheeromgeving en een omgeving om de websites te tonen op verschillende webservers.

Vervolgens is gekeken wat de beste oplossingen waren voor de uitbreiding van het CMS met de producten met het oog op de kwaliteitscriteria. Na een analyse is voor de volgende oplossing gekozen: De producteninformatie wordt opgeslagen in een aparte database en is bereikbaar met behulp van webservices. Om de performance van de productenpagina's te garanderen wordt er gebruikt gemaakt van een cache voor de categorieën op de frontend-webserver. Deze cache wordt elk uur ververst om de data zo recent mogelijk te houden.

Bij het testen van de kwaliteitscriteria is per criterium gekeken naar de beste oplossing. Zo is voor de performance eisen gekozen om loadtests uit te voeren. Met loadtests kan de performance van een systeem worden gemeten met een van tevoren bepaalde hoeveelheid gebruikers.

Uiteindelijk is de conclusie getrokken dat het systeem zoals het gebouwd is voldoet aan alle kwaliteitscriteria. Er zijn echter wel een aantal aanbevelingen die gedaan zijn voor de toekomst. Het is vooral belangrijk dat er getest blijft worden als er aanpassingen in het systeem gedaan worden. Ook is de aanbeveling gedaan om de tests uit te breiden met functionele en unit tests om een grotere hoeveelheid mogelijkheden te testen.

Inhoudsopgave

Begrippen- en afkortingenlijst

Begrip	Uitleg
API	Application Programming Interface. Maakt communicatie mogelijk tussen twee verschillende systemen.
CRM	Customer Relation Management. Onderdeel van het ERP, waarmee Stokvis alle klantgegevens beheert.
DAO	Database Access Object. Een abstractie laag, waarmee door een applicatie makkelijk gegevens uit een database gehaald kunnen worden.
ERP	Enterprise Resource Planning. Software waarmee Stokvis onder andere productprijzen en klantgegevens beheert.
Multisite CMS	Multisite Content Management Systeem. Een systeem waarin gebruikers meerdere websites kunnen beheren, bijvoorbeeld WordPress. Tenzij anders benoemd, verwijst dit naar het systeem wat gebouwd wordt door SalesCare.

Figurenlijst

Tabellenlijst

1. Inleiding

In de periode van 30 mei 2016 tot en met 28 oktober 2016 heeft Jonathan Karssen, 4e jaars student Informatica, zijn afstudeeropdracht uitgevoerd. In dit document wordt op deze periode teruggeblikt en de opdracht behandeld. De afstudeeropdracht is uitgevoerd bij het bedrijf SalesCare in Utrecht. De hoofdvraag voor de opdracht was als volgt:

> *"Hoe kan de architectuur en het ontwerp van het multisite CMS van SalesCare aangepast worden om de nieuwe functionaliteiten mogelijk te maken en hoe kunnen de kwaliteitscriteria van deze uitbreiding het beste gegarandeerd worden door een goede teststrategie?"*

1.1 Leeswijzer

Hieronder volgt een korte samenvatting van wat u in elk hoofdstuk kunt vinden.
Als eerste wordt in hoofdstuk 2 het bedrijf SalesCare besproken, dat de opdracht heeft aangeleverd. Hierna volgt in hoofdstuk 3 de aanleiding en de probleemstelling van de opdracht. Vervolgens wordt er een uitgebreidere omschrijving en de doelstelling van de opdracht gegeven in hoofdstuk 4.
Om de opdracht tot een succesvol einde te brengen is een onderzoek uitgevoerd naar de beste manier om de opdracht uit te voeren. De onderzoeksvragen en de aanpak van dit onderzoek vindt u in hoofdstuk 5 en de resultaten zijn te vinden in hoofdstuk 6. In hoofdstuk 7 wordt het gerealiseerde product besproken en wordt er vermeld wat er wel en niet is geïmplementeerd. Daarna volgen de conclusie en aanbevelingen in hoofdstuk 8. Tot slot eindigt het document met de bronnen en bijlagen.

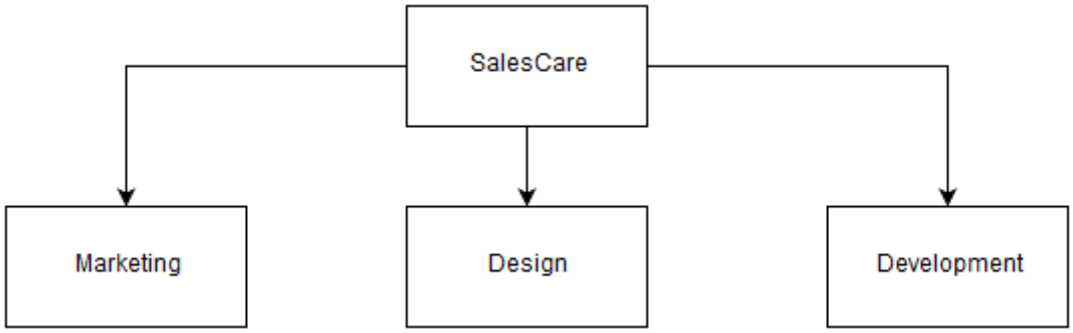
2. Bedrijf

2.1 SalesCare

De opdracht zal worden uitgevoerd bij SalesCare. SalesCare is een online marketingbureau voor bedrijven met online ambities en is gevestigd in Utrecht. Het bedrijf is opgericht in 2010 en heeft inmiddels 12 medewerkers. SalesCare helpt klanten met hun online advertentiestrategie. Daarnaast ontwerpen en ontwikkelen ze websites, maar kunnen ze ook bestaande websites optimaliseren. Op hun website (salescare.nl) stellen zij zich zelf zo voor:

> "*Sales Care is een jong en gedreven online marketingbureau. Vanuit ons kantoor in Utrecht werken wij hard om het uiterste uit de online marketingcampagnes van onze klanten te halen. Dagelijks zetten we nieuwe online marketingcampagnes uit, analyseren en optimaliseren we websites en ontwikkelen we nieuwe websites.*"

In figuur 1 staat een simpel organogram van SalesCare. De student valt onder het development-team van SalesCare.



Figuur 1: Organogram SalesCare

3. Probleem

3.1 Aanleiding

SalesCare heeft een klant, Stokvis Group, met vele dochterondernemingen. Voor deze dochterondernemingen moeten zij ongeveer zestig websites kunnen beheren in verschillende talen. Het huidige beheer van de websites is erg veel werk en dit willen zij makkelijker maken. Hiervoor willen zij een multisite en meertalig Content Management Systeem (CMS) gebruiken. Hiermee kunnen ze de websites vanaf één plek beheren. SalesCare is bezig met het maken van dit CMS.

3.2 Probleemstelling

De websites en hun statische pagina's kunnen ondertussen beheerd worden, maar voor de volgende release van het CMS moeten de producten van de Stokvis Group in het CMS beheerd kunnen worden. Deze producten moeten getoond en uiteindelijk besteld kunnen worden op de beheerde websites. Aangezien het om ongeveer 500.000 producten gaat, is de performance van het systeem een belangrijke kwestie. Bij een eerdere poging van een ander bedrijf was de laadtijd van een pagina met ongeveer 3000 producten 25 seconden. Dit zal uiteraard veel sneller moeten in het te realiseren systeem. De websites krijgen gezamenlijk naar verwachting een piekbelasting van 70 gebruikers tegelijkertijd. Het systeem zal dus zeker rekening met de performance moeten houden.

Ook de onderhoudbaarheid van het systeem is belangrijk. Het team developers is erg klein en dit team heeft ook andere werkzaamheden, dus kleine wijzigingen moeten snel doorgevoerd kunnen worden. Er zijn ook nog een aantal andere uitbreidingen gepland voor het systeem voor volgend jaar, dus het moet makkelijk zijn om deze toe te voegen.

4. Opdracht

4.1 Omschrijving

De afstudeeropdracht bestaat uit het onderzoek doen naar de beste oplossing, ontwerpen en ontwikkelen van de backend van een multisite en meertalig Content Management System (CMS). Zoals in de probleemstelling al genoemd, gaat het om een CMS waar producten voor verschillende websites moeten worden bijgehouden.

De websites die beheerd worden in het CMS moeten allemaal een deel van het assortiment van producten van de klant kunnen aanbieden. Stokvis Group verkoopt namelijk ongeveer 500.000 producten met ieder ongeveer 40 verschillende attributen. Deze producten moeten beheerd kunnen worden in het CMS, waarna de websites in het CMS verschillende dingen met deze producten kunnen doen. Zo moet elke website zelf een selectie kunnen maken welke producten ze tonen en moet er een makkelijke manier zijn om in deze producten te kunnen zoeken. Bij elk product is de informatie in meerdere talen beschikbaar en per website kan de prijs verschillen. Uiteindelijk is het de bedoeling dat deze producten ook kunnen worden besteld via de websites, maar dit valt buiten de scope van deze afstudeeropdracht.

De voorraad en prijzen van de producten worden door de klant beheerd in een eigen ERP-systeem, wat benaderd kan worden met behulp van een API. Het CMS zal deze informatie zelf niet opslaan en zal dus ook moeten communiceren met dit systeem om alle nodige informatie op te halen. Voor de meeste producten moeten gebruikers van een website ook bestanden kunnen downloaden. Deze bestanden worden ook in een ander systeem opgeslagen, namelijk YaPDM. Dit systeem moet wijzigingen in de bestanden kunnen melden bij het CMS. Het CMS zal hier dus een webservice voor moeten implementeren.

De performance van het systeem is erg belangrijk. Bij een systeem wat een andere leverancier had opgeleverd, was de laadtijd van een pagina met ongeveer 3000 producten ongeveer 25 seconden. Aangezien de performance van het systeem erg belangrijk is, zal de student in zijn scriptie onder andere onderzoek doen naar de beste manieren om de performance van het CMS zo goed mogelijk te maken door op een goede manier te testen.

4.2 Doelstelling

Het doel van de stage is dat de student aan het einde minimaal een ‘proof of concept’ kan presenteren van het productengedeelte van de backend van het CMS, maar het streven is een werkend product. Ook wordt verwacht dat het product goed wordt gedocumenteerd, zodat andere programmeurs makkelijk inzicht kunnen krijgen in het product. Daarnaast is het doel om de performance van het CMS aanvaardbaar te maken met behulp van een onderzoek.

5. Onderzoek

5.1 Onderzoeksvragen

Voor het onderzoek naar de oplossing van het probleem zijn de volgende hoofd- en deelvragen opgesteld.

5.1.1 Hoofdvraag

> "Hoe kan de architectuur en het ontwerp van het multisite CMS van SalesCare aangepast worden om de nieuwe functionaliteiten mogelijk te maken en hoe kunnen de kwaliteitscriteria van deze uitbreiding het beste gegarandeerd worden door een goede teststrategie?"

5.1.2 Deelvragen

1. Wat is de huidige situatie van het CMS?
2. Wat zijn de benodigde functionaliteiten van de uitbreiding?
3. Wat zijn de gestelde eisen en kwaliteitscriteria aan het systeem?
4. Welke architectuur en ontwerpoplossing voldoen aan de gestelde eisen en kwaliteitscriteria?
5. Hoe kunnen de kwaliteitscriteria het beste getest worden?

5.2 Aanpak

Voor de verschillende deelvragen zijn ook verschillende aanpakken nodig. Hieronder volgt per deelvraag een korte toelichting en de aanpak die de student zal nemen.

1. Wat is de huidige situatie van het CMS?
Voordat de uitbreiding van het CMS ontworpen kan worden, moet de huidige situatie van het CMS in beeld gebracht worden en moet duidelijk worden welke functionaliteiten nu al ondersteund worden. Aangezien er nauwelijks documentatie van het CMS is, zal de student zelf het bestaande project in beeld moeten brengen. Dit zal gedaan worden met behulp van een descriptief onderzoek. De student zal kijken naar de bestaande documentatie, de code en de database. Als het nodig is, kan de student om toelichting vragen van de programmeurs. De student zal in ieder geval de huidige architectuur en het huidige ERD tonen.

2. Wat zijn de benodigde functionaliteiten van de uitbreiding?
Om te bepalen wat er precies nodig is, zal de student alle functionaliteiten op een rijtje moeten zetten. Hij zal hiervoor een Functioneel Ontwerp maken, waarin in ieder geval alle user stories zijn opgeschreven. Verder zal de student per user story bepalen hoeveel extra informatie er nodig is om de nodige functionaliteit duidelijk te maken. De functionaliteiten zal de student halen uit een kwalitatief onderzoek met behulp van oude documentatie en gesprekken met developers van het CMS en de product-owner.

3. Wat zijn de gestelde eisen en kwaliteitscriteria aan het systeem?
Naast de functionele eisen zijn er ook andere eisen aan het systeem, zoals de performance. De student zal deze 'non-functional requirements' moeten vaststellen. De student zal eerst mogelijke kwaliteitscriteria zoeken met behulp van een literatuuronderzoek en dan bekijken welke van deze kwaliteitscriteria gelden voor dit project. De student zal de eisen voor deze kwaliteitscriteria uit bestaande documentatie en gesprekken met de opdrachtgever halen. De eisen zullen beschreven worden in het Technisch Ontwerp, zodat er rekening gehouden kan worden met de eisen bij het ontwerpen van de architectuur.

4. Welke architectuur en ontwerpoplossing voldoen aan de gestelde eisen en kwaliteitscriteria?
Nadat de eisen zijn vastgesteld kan gekeken worden naar de mogelijke oplossingen voor het probleem. Indien het nodig is, kan er deskresearch worden uitgevoerd om mogelijke oplossingen te vinden of te evalueren. Ook zal er kwalitatief onderzoek plaatsvinden door de mogelijke architecturen te bespreken met het hosting-bedrijf om de beste oplossing te vinden. De gekozen architectuur en ontwerp zullen vastgesteld worden in het Technisch Ontwerp, waar de gemaakte keuzes worden toegelicht.

5. Hoe kunnen de kwaliteitscriteria het beste getest worden?
De student zal moeten bepalen hoe de applicatie het beste getest kan worden, zodat vastgesteld kan worden dat de oplossing voldoet aan de gestelde eisen. Hij zal een literatuuronderzoek doen om de beste manieren te vinden om de gevonden kwaliteitscriteria te testen. De student zal vervolgens deze manieren gebruiken om de applicatie te testen. De resultaten van de tests zullen in een testrapport gemeld worden.

6. Resultaten

6.1 Deelvraag 1: Wat is de huidige situatie van het CMS?

Eerst is er geïnventariseerd of er documentatie aanwezig was over het huidige project. Al snel bleek dat de documentatie erg verouderd of überhaupt niet aanwezig was. Hierop is besloten om onderzoek te doen naar de functionaliteiten en de architectuur van het bestaande systeem. In het onderzoek is een kort overzicht gemaakt van het huidige systeem.

6.1.1 Functionaliteiten

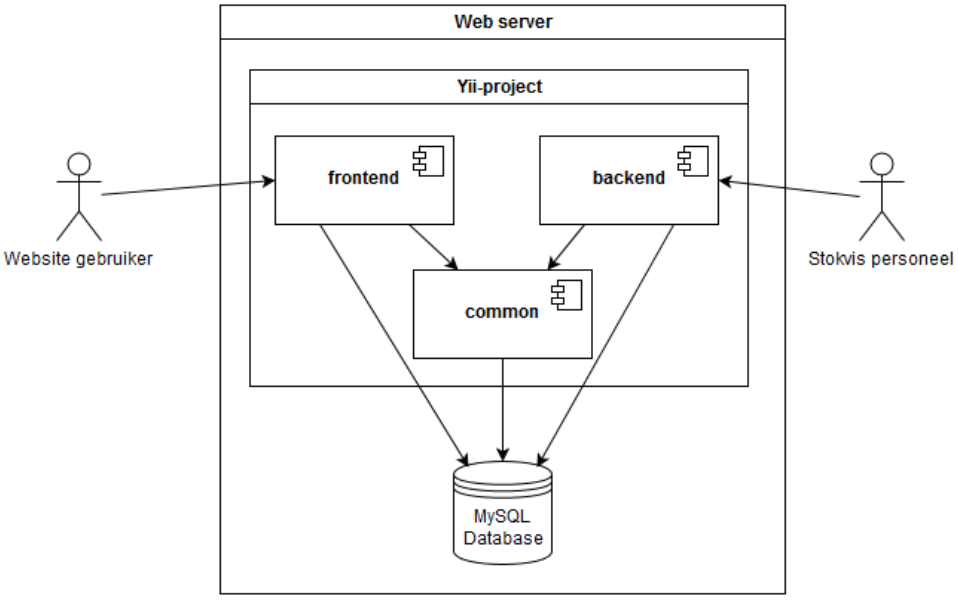
Het huidige CMS is in de laatste fases van de eerste oplevering. Bij deze oplevering kunnen er statische websites worden opgezet in een backend-dashboard en vervolgens worden de websites bezocht door gebruikers. Het dashboard is de beheer-omgeving van het CMS. De belangrijkste functionaliteiten die het dashboard heeft staan in tabel 1. Aangezien de student zelf de functionaliteiten moest opzoeken, zijn ze opgeschreven als simpele beschrijvingen. Er zit meer diepte in de functionaliteiten, maar dit is de basis. Uiteraard laat het systeem de beheerde gegevens zien in de websites als ze gepubliceerd worden.

Functionaliteit
Websites beheren
Pagina's beheren
Afbeeldingen beheren
Afbeelding-sliders beheren
Nieuwsberichten beheren
Evenementen beheren
Vacatures beheren
Backend gebruikers beheren

Tabel 1: Belangrijkste huidige functionaliteiten

6.1.2 Architectuur

Het CMS is gebouwd op het Yii2-framework. Dit is een "high-performance PHP framework best for developing Web 2.0 applications" (www.yiiframework.com). Het project is gebaseerd op het 'advanced template' van het framework, waarbij er een losse frontend- en backend-applicatie zijn die gebruik maken van een gemeenschappelijk deel. In figuur 2 staat de globale architectuur van het CMS.

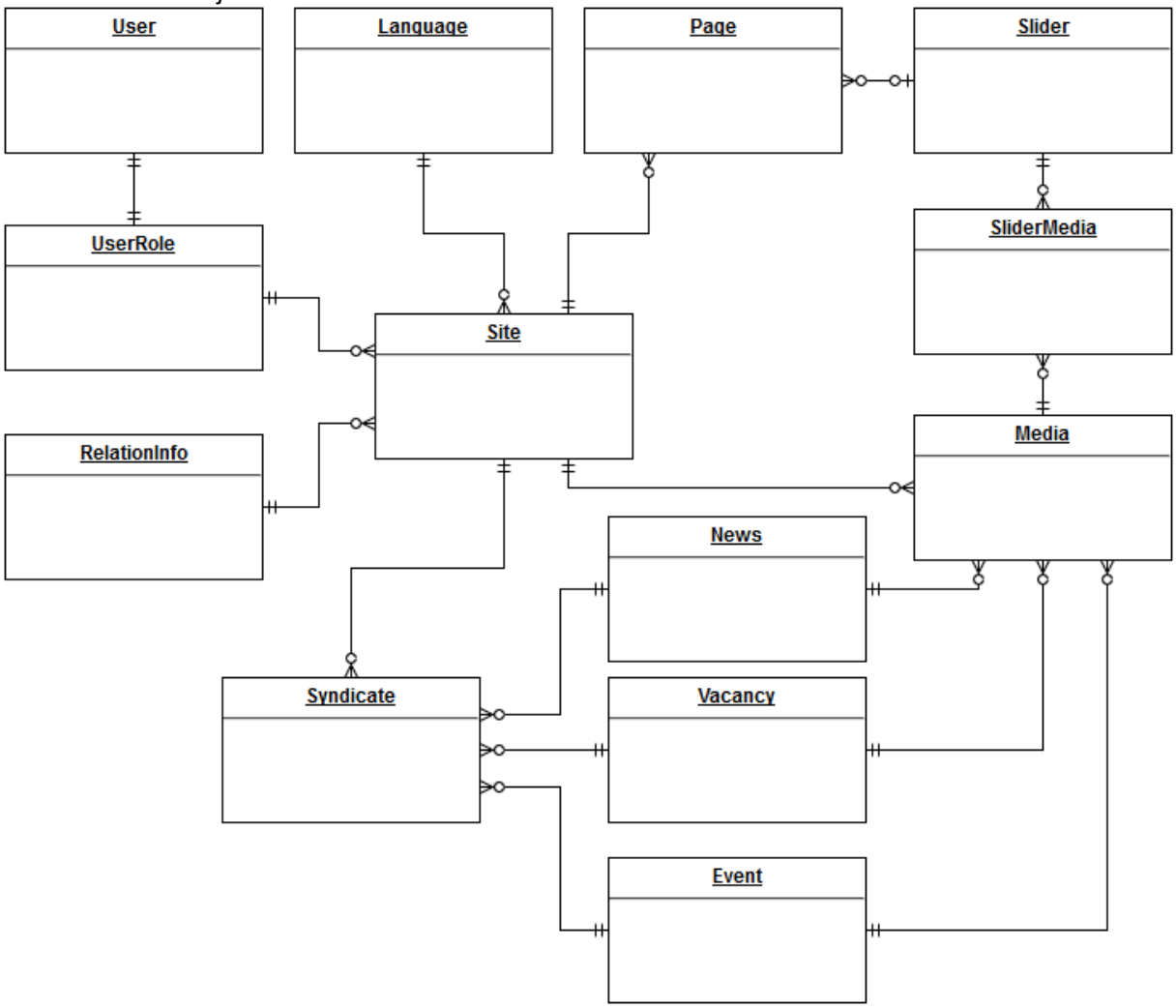


Figuur 2: Huidige architectuur

De applicaties maken gebruik van het ModelViewController-principe, waarmee domeinlogica gescheiden wordt van de presentatielogica. Er wordt één database gebruikt waaruit de applicaties hun data halen. Yii2 gebruikt hier standaard een DAO-implementatie voor, waar in dit project ook gebruik van is gemaakt.

6.1.3 Database design

Het databasemodel van het huidige CMS staat in figuur 3. De attributen zijn niet opgenomen om het model overzichtelijk te houden.



Figuur 3: Huidig databasemodel

6.1.4 Conclusie

Het huidige CMS bevat de functionaliteiten die nodig zijn om statische websites aan te maken en te beheren. Er kunnen websites met het systeem worden opgezet en ze kunnen worden bekeken door de website-bezoekers. De huidige structuur splitst het backend-dashboard en de frontend die de websites toont, zodat de verantwoordelijkheden van elke applicatie duidelijk zijn.

6.2 Deelvraag 2: Wat zijn de benodigde functionaliteiten van de uitbreiding?

Om de functionaliteiten vast te stellen die het productengedeelte van het CMS allemaal moeten bevatten, heeft de student als eerste gekeken naar bestaande documentatie. De grote lijnen van het systeem waren namelijk al beschreven. Het bleek dat alleen globaal de functionaliteiten waren beschreven die een website-bezoeker moest kunnen uitvoeren, maar dat er nog geen overzicht was over welke functionaliteiten er in het dashboard moesten komen. De student heeft na meerdere malen overleg met zowel de developers als de product-owner een Functioneel Ontwerp opgeleverd, waarin alle benodigde functionaliteiten stonden. Er is gekozen om de functionaliteiten als user stories op te schrijven, aangezien het development-team van SalesCare gebruik maakt van Scrum. Vervolgens zijn deze user stories geprioriteerd aan de hand van de MoSCoW-methode om duidelijk aan te geven welke functionaliteiten in ieder geval voltooid moeten zijn aan het einde van de afstudeerperiode.

Hieronder volgen de user stories per actor met de prioriteit die aan de user story is gegeven. Deze user stories zijn in het Functioneel Ontwerp verder uitgewerkt tot simpele use cases met, afhankelijk van de moeilijkheid van de user story, extra details en/of flowcharts. Het volledige Functioneel Ontwerp is te vinden in bijlage 2.

6.2.1 Websitegebruiker

Er zijn twee verschillende soorten websitegebruikers, namelijk:

- Anoniem
- Geregistreerd

Geregistreerde gebruikers krijgen in het ERP autorisaties toegewezen, waarmee ze toegang krijgen tot bepaalde functionaliteiten op de website. Een gebruiker kan meerdere verschillende autorisaties hebben en gebruikers worden per website geregistreerd. Dus als een gebruiker naar een andere website gaat zal hij opnieuw moeten registreren. Hij kan dan wel gekoppeld worden aan dezelfde gebruiker in het ERP met dus dezelfde autorisaties.

De volgende autorisaties worden bijgehouden in het ERP:

- Per gebruiker:
    - Type autorisatie voorraad
        - Verborgen
        - Wel/niet
        - Aantal
    - Type autorisatie prijzen
        - Verborgen
        - Bruto
        - Netto
- Per productgroep per gebruiker
    - Downloaden tekeningen (J/N)
    - Opvragen voorraad (J/N)
    - Opvragen prijs (J/N)
    - Orders plaatsen (J/N)
    - Opvragen offertes (J/N)
    - Opvragen orderstatus (J/N)
    - Opvragen orderhistorie (J/N)

Hieruit blijkt dus dat niet op elke website de bezoeker dezelfde functionaliteiten mag uitvoeren. Op basis van de autorisaties uit het ERP wordt bepaald of de gebruiker de user story mag uitvoeren.

Nr.	Omschrijving	Prioriteit
1.1	Als gebruiker wil ik producten kunnen bekijken	Must-have
1.1.1	- Als gebruiker wil ik de productcategorieën kunnen bekijken	
1.1.2	- Als gebruiker wil ik op eigenschappen kunnen filteren	
1.1.3	- Als gebruiker wil ik de prijs van producten kunnen bekijken	- Should-have
1.1.4	- Als gebruiker wil ik de voorraad van producten kunnen bekijken	- Should-have
1.1.5	- Als gebruiker wil ik de maattekeningen kunnen bekijken	
1.2	Als gebruiker wil ik files kunnen downloaden van een product	Must-have
1.2.1	- Als gebruiker wil ik files kunnen downloaden van een categorie	
1.3	Als gebruiker wil ik mijzelf kunnen registreren	Must-have
1.4	Als gebruiker wil ik mijn persoonlijke gegevens kunnen wijzigen	Should-have
1.5	Als gebruiker wil ik een wachtwoord vergeten-functie kunnen gebruiken	Should-have
1.6	Als gebruiker wil ik extra autorisaties kunnen aanvragen	Should-have
1.7	Als gebruiker wil ik een offerte kunnen aanvragen	Could-have
1.7.1	- Als gebruiker wil ik artikelen kunnen selecteren voor mijn offerte	
1.7.2	- Als gebruiker wil ik mijn huidige offerte kunnen inzien	
1.7.3	- Als gebruiker wil ik mijn offerte kunnen wijzigen	
1.7.4	- Als gebruiker wil ik mijn offerte kunnen plaatsen	
1.8	Als gebruiker wil ik een bestelling kunnen plaatsen	Could-have
1.8.1	- Als gebruiker wil ik artikelen kunnen selecteren voor mijn bestelling	
1.8.2	- Als gebruiker wil ik mijn huidige bestelling kunnen inzien	
1.8.3	- Als gebruiker wil ik mijn bestelling kunnen wijzigen	
1.8.4	- Als gebruiker wil ik mijn bestelling kunnen plaatsen	
1.9	Als gebruiker wil ik mijn order historie kunnen inzien	Could-have
1.9.1	- Als gebruiker wil ik een order kunnen herbestellen	
1.10	Als gebruiker wil ik een offerte kunnen omzetten in een bestelling	Could-have
1.11	Als gebruiker wil ik de bestelstatus van een order kunnen inzien	Could-have

Tabel 2: User stories websitegebruiker

6.2.2 Backend contactmanager

Nr.	Omschrijving	Prioriteit
2.1	Als contactmanager wil ik een overzicht kunnen opvragen van alle gebruikers die aandacht nodig hebben	Must-have
2.2	Als contactmanager wil ik nieuwe gebruikers kunnen koppelen aan ons CRM-systeem	Must-have
2.3	Als contactmanager wil ik nieuwe gebruikers kunnen weigeren	Must-have
2.4	Als contactmanager wil ik gegevens van een gebruiker kunnen aanpassen	Must-have

Tabel 3: User stories backend contactmanager

6.2.3 Backend admin

De backend admin heeft ook toegang tot alle andere backend user stories.

Nr.	Omschrijving	Prioriteit
3.1	Als admin wil ik een product kunnen beheren	Should-have
3.2	Als admin wil ik een productcategorie kunnen beheren	Should-have
3.3	Als admin wil ik een Excel-bestand met producteninformatie kunnen importeren	Must-have
3.3.1	- Als admin wil ik een new-products Excel-bestand kunnen importeren	
3.3.2	- Als admin wil ik een update-products Excel-bestand kunnen importeren	
3.3.3	- Als admin wil ik een delete-products Excel-bestand kunnen importeren	
3.4	Als admin wil ik productencaches kunnen resetten	Should-have
3.5	Als admin wil ik een productcatalogus kunnen beheren	Should-have
3.5.1	- Als admin wil ik een productcatalogus kunnen samenstellen	
3.6	Als admin wil ik attributen kunnen beheren	Should-have

Tabel 4: User stories backend admin

6.2.4 Backend author product

Nr.	Omschrijving	Prioriteit
4.1	Als product author wil ik productpagina's kunnen aanmaken en aanpassen	Must-have

Tabel 5: User stories backend author product

6.2.5 YaPDM

Nr.	Omschrijving	Prioriteit
5.1	Als YaPDM wil ik files kunnen beheren	Must-have
5.1.1	- addFile	
5.1.2	- deleteFile	
5.1.3	- updateFile	

Tabel 6: User stories YaPDM

6.2.6 Conclusie

Er zijn veel functionaliteiten nodig voor de uitbreiding van het CMS om de producten te ondersteunen. De belangrijkste functionaliteiten zijn onder andere: de producten tonen, de koppeling met YaPDM en de Excel-import tool om producten in de database te zetten. Alle nodige functionaliteiten zijn in het Functioneel Ontwerp (zie bijlage 2) gezet, waar nodig met extra details om de functionaliteit duidelijk te maken.

6.3 Deelvraag 3: Wat zijn de gestelde eisen en kwaliteitscriteria aan het systeem?

Naast de functionele eisen die zijn gesteld in het Functioneel Ontwerp, waren er ook technische eisen aan het project. De student heeft eerst onderzoek gedaan naar mogelijke kwaliteitscriteria. Vervolgens heeft de student kwaliteitscriteria opgesteld voor het project, waarna hij deze kwaliteitscriteria heeft besproken met de opdrachtgever. De opdrachtgever heeft de uiteindelijke kwaliteitscriteria goedgekeurd.

6.3.1 Mogelijke kwaliteitscriteria

ISO, de Internationale Organisatie voor Standaardisatie, heeft voor software een productkwaliteitsmodel uitgebracht (www.iso.org). Dit model, ISO/IEC 25010, benoemt 8 karakteristieken, waarmee software beoordeeld kan worden. Deze 8 karakteristieken hebben allemaal ook weer sub-karakteristieken. De karakteristieken zijn weergegeven in tabel 7 en tabel 8.

Functional Suitability	**Performance Efficiency**	**Compatibility**	**Portability**
Functional Completeness	Time behaviour	Co-existence	Adaptability
Functional Correctness	Resource Utilization	Interoperability	Installability
Functional Appropriateness	Capacity		Replaceability

Tabel 7: ISO 25010 Quality Attributes 1

Reliability	**Security**	**Maintainability**	**Usability**
Maturity	Confidentiality	Modularity	Appropriateness recognizability
Availability	Integrity	Reusability	Learnability
Fault Tolerance	Non-repudiation	Analysability	Operability
Recoverability	Accountability	Modifiability	User error protection
	Authenticity	Testability	User interface aesthetics
			Accessibility

Tabel 8: ISO 25010 Quality Attributes 2

Na overleg met de product-owner is vastgesteld welke kwaliteitscriteria het belangrijkst zijn voor het CMS. Uiteraard zijn de andere kwaliteitscriteria ook van belang, maar hiervoor is besloten om uit te gaan van een redelijke standaard zonder deze expliciet te benoemen.

6.3.2 Non-functional requirements

De belangrijkste requirements zijn in tabel 9 opgenomen met het bijbehorende kwaliteitsattribuut wat van belang is bij de eis. Deze requirements gelden voor de productie-omgeving als er getest wordt door SalesCare. De requirements voor de laadtijd (1.1-1.4) zijn gebaseerd op het onderzoek van Nah. Volgens haar onderzoek is de getolereerde wachttijd op een webpagina ongeveer 2 seconden. Bij het wachten op een download is de getolereerde wachttijd een stuk hoger, namelijk 10

seconden (Nah, 2004), waardoor requirement 1.4 een hogere laadtijd toestaat. De andere requirements komen voornamelijk van de product-owner en zijn business requirements.

Nr.	Requirement	Kwaliteitsattribuut
1.1	Het laden van een pagina zonder producten moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	Time behaviour
1.2	Het ophalen van alle producten van een categorie moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	Time behaviour
1.3	Het ophalen van de informatie van één product moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	Time behaviour
1.4	Het genereren en ophalen van een productdatasheet moet altijd binnen 10 seconden gebeuren.	Time behaviour
1.5	Het systeem moet minimaal 35 users op de frontend met een piekbelasting van 70 users tegelijkertijd aan kunnen zonder performanceverlies.	Capacity
2.1	De producteninformatie moet toegankelijk zijn voor andere systemen dan het CMS.	Modularity
2.2	Een programmeur van het CMS moet binnen twee uur de oorzaak van een fout in het systeem kunnen opsporen.	Analysability
2.3	Een nieuwe release moet binnen twee uur gevalideerd kunnen worden.	Testability
3.1	Als er een fout zit in pagina's met producten moeten de pagina's zonder producten nog steeds bereikbaar zijn.	Fault Tolerance
3.2	Als het ERP niet bereikbaar is, moet de andere productinformatie wel zonder vertraging getoond worden.	Fault Tolerance
4.1	Gevoelige data mag niet opgeslagen worden op dezelfde server als de websites.	Confidentiality

Tabel 9: Non-functional requirements

6.3.3 Constraints

Naast de kwaliteitscriteria zijn er ook een aantal eisen aan het project gesteld (zie tabel 10). Deze eisen komen vooral door het huidige systeem of zijn deadlines opgelegd door de product-owner.

Nr.	Constraint
1	Het CMS moet gebouwd worden in PHP met behulp van het framework Yii2.
2	Het CMS moet MySQL als databasemanagementsysteem gebruiken.
3	De uitbreiding van het CMS met producten moet klaar zijn voor 31 oktober 2016.

Tabel 10: Constraints

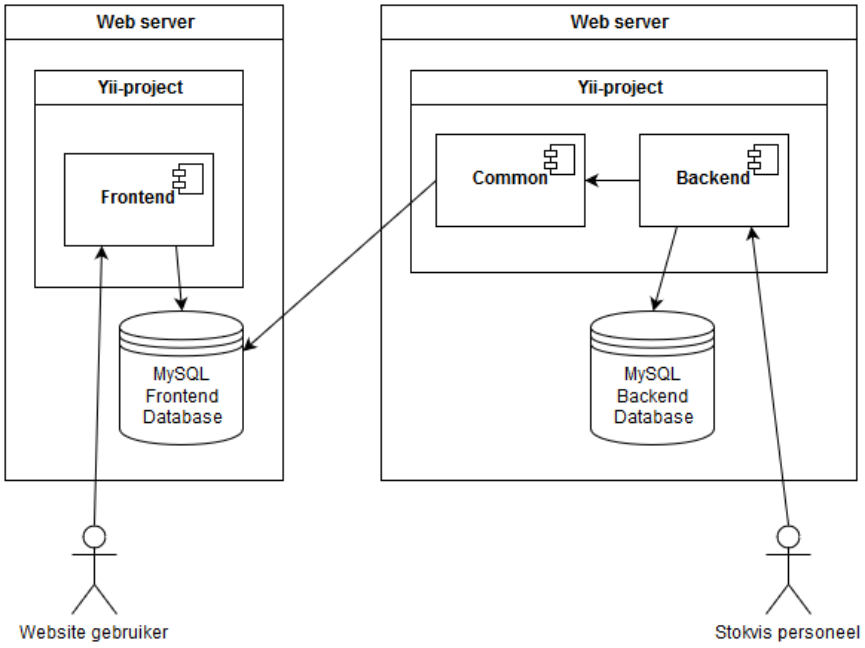
6.3.4 Conclusie

ISO, de Internationale Organisatie voor Standaardisatie, heeft een aantal kwaliteitscriteria benoemd, waarmee de kwaliteit van software kan worden aangetoond. In samenwerking met de product-owner zijn de belangrijkste kwaliteitscriteria en eisen opgeschreven. De belangrijkste kwaliteitscriteria, in het geval van het CMS, zijn vooral Performance Efficiency, Reliability, Security en Maintainability. Alle eisen zijn meegenomen in het Technisch Ontwerp (zie bijlage 3).

6.4 Deelvraag 4: Welke architectuur en ontwerpoplossing voldoen aan de gestelde eisen en kwaliteitscriteria?

Na het opstellen van de eisen in hoofdstuk 6.3, is gekeken naar manieren om de architectuur en het database-model uit te breiden. Tijdens dit onderzoek, bij het overleg met de hostingpartij over de architectuur, bleek dat er een belangrijke requirement was gemist. Requirement 4.1, gevoelige data mag niet opgeslagen worden op dezelfde server als de websites, was een eis die aan het begin van het CMS-project wel besproken was, maar daarna niet was meegenomen in de huidige architectuur. Hierdoor moest ten eerste de huidige architectuur aangepast worden, voordat goed gekeken kon worden naar de uitbreiding. De architectuur is toen aangepast naar de architectuur in figuur 4. Hierbij werd de backend database de oude database en werd er een extra frontend-database aangemaakt met de nodige informatie om de websites te laten functioneren, maar zonder gevoelige informatie of informatie die alleen nodig is voor het beheer van de websites zelf. Met behulp van de common-applicatie wordt de nodige data naar de frontend-database geschreven. Met deze architectuur wordt een deel van de data twee keer opgeslagen, maar dit zorgt wel voor een goed gescheiden architectuur van website-beheer en de websites zelf.

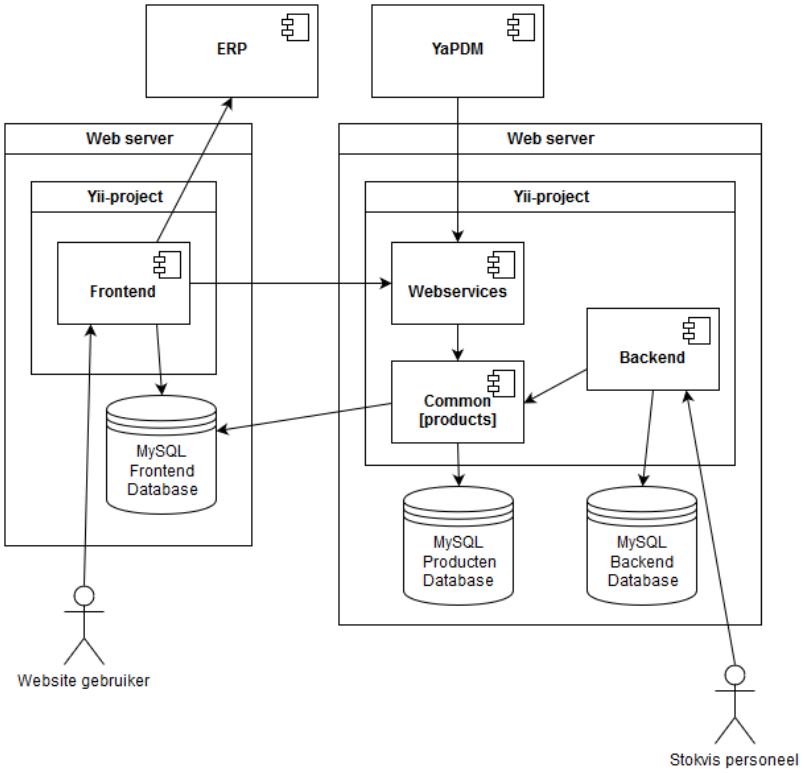
Op verzoek van de product-owner is de backend-webserver in de 'demilitarized zone' (DMZ) geplaatst van het bedrijf. Een DMZ is, net als bij de militaire term, een zone tussen twee gebieden in (Shinder, 2005). In dit geval is het een zone tussen het internet en het interne netwerk van het bedrijf. Dit betekent in dit geval dat de server achter een firewall zit en alleen bepaalde IP-adressen toegang hebben tot de server.



Figuur 4: Aangepaste huidige architectuur

6.4.1 Architectuur

In het Technisch Ontwerp (zie bijlage 3) was tot dan toe bij de mogelijke architecturen uitgegaan van de oude architectuur. Na de extra requirement te hebben meegenomen in de evaluatie van de mogelijke architecturen is gekozen voor de architectuur in figuur 5. Er is gekozen om een extra database op te nemen voor alle producten om goed te voldoen aan requirement 2.1. Door een aparte database op te nemen, is de data ook makkelijker beschikbaar te maken voor andere systemen zonder de informatie van het websitebeheer ook bloot te geven. Het grootste nadeel, de referentiële integriteit die verbroken wordt, kan opgevangen worden door de applicatie zelf.



Figuur 5: Gewenste architectuur

Verder wordt er een module products toegevoegd aan de common-applicatie. Hierin komen de Data Access Objects (DAO) van de producten, zodat zowel het productenbeheer in de backend-applicatie als de webservices bij de producteninformatie in de database kunnen. De webservices komen in een aparte webservices-applicatie in de backend-webserver. Yii zelf adviseert om webservices in een aparte applicatie te zetten, zodat er een aantal instellingen makkelijk kunnen worden ingesteld voor alle webservices. De webservices worden geïmplementeerd als een REST API. Deze keus is vooral gemaakt vanwege de goede support van Yii voor REST webservices en vanwege de relatief simpele requests die verstuurd zullen worden naar de API. Ook zal de API op het begin alleen door twee vertrouwde partijen, YaPDM en de frontend, gebruikt worden, waardoor er geen ingewikkelde validatie nodig is. Aangezien de webservices in de DMZ zitten, is er al een controle op IP-adres bij elke request. Ter extra beveiliging is er ook gebruik gemaakt van Basic Authentication om zeker te weten dat de webservices alleen gebruikt worden door vertrouwde partijen.

Naast de communicatie vanaf YaPDM, zal het CMS ook moeten communiceren met het ERP voor gebruikersgegevens en up-to-date productenprijzen. Hiervoor is al eerder een SOAP API geïmplementeerd in het ERP, wat de frontend-applicatie kan aanroepen.

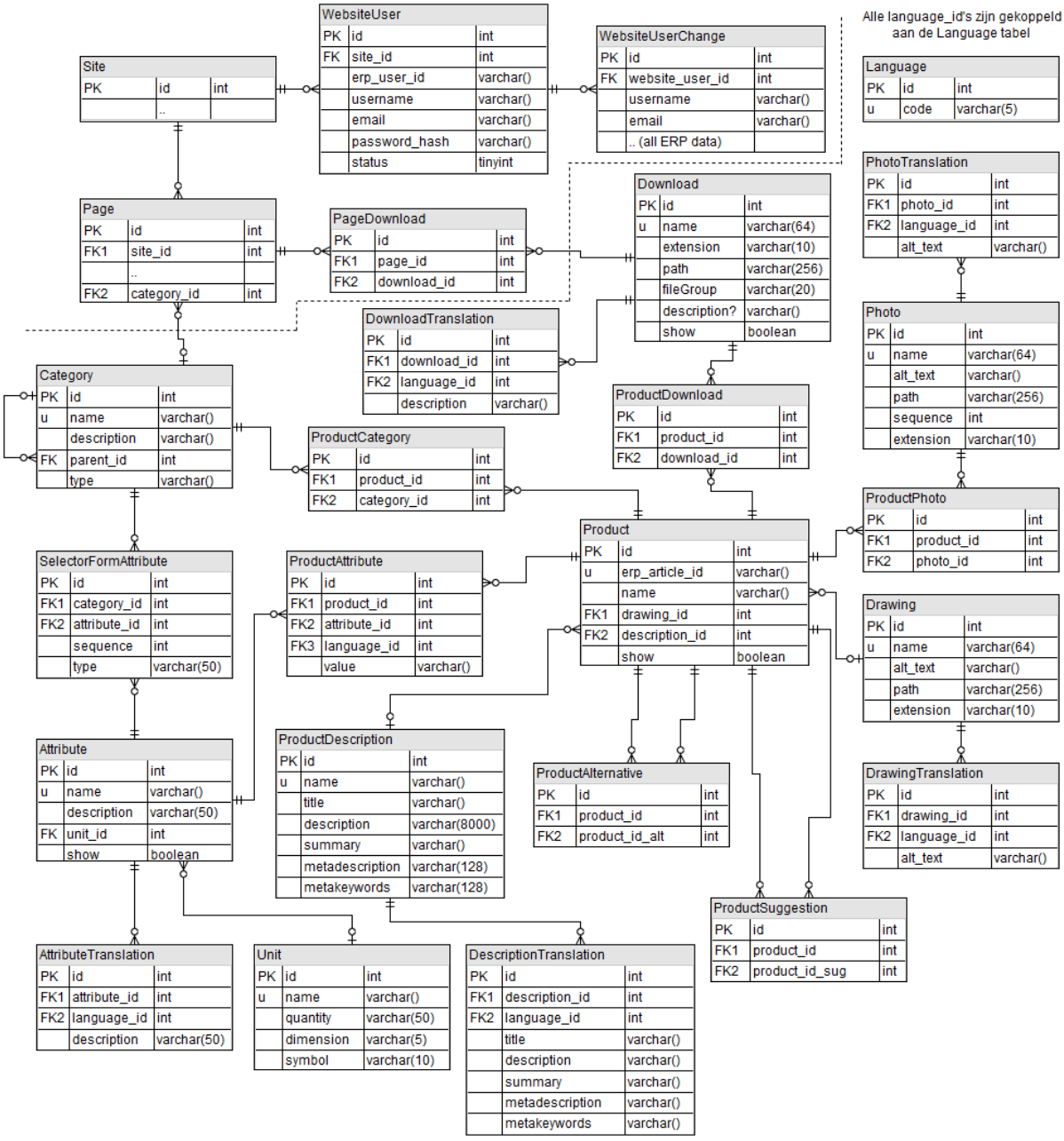
6.4.2 Databasemodel

Eén van de belangrijkste vragen van het project was hoe alle informatie van de producten opgeslagen moest worden. Tot dan toe werd de informatie niet centraal opgeslagen, maar werd er een database aangemaakt per website met alleen die producten. Er zijn verschillende iteraties van het ERD geweest, maar de uiteindelijke versie staat in figuur 6. Bovenin het ERD staan een aantal klassen die al in het bestaande ERD staan, namelijk Site, Page en Language. De overige tabellen zijn allemaal nieuw. Er is een stippellijn getekend om de scheiding aan te geven tussen de productendatabase en de backend-database. Alle tabellen boven de stippellijn worden in de backend- en/of frontend-database gezet en niet in de productendatabase.

Om de performance bij het zoeken zo snel mogelijk te maken, staan er indexen op alle ‘foreign keys’ en unieke velden. Hieronder volgt waar nodig een toelichting per tabel of per groep tabellen.

6.4.2.1 Attribute

Er zijn over alle producten verspreid ongeveer 250 verschillende attributen, waarvan de meeste maar voor een deel van de producten nodig zijn. Ook kunnen de waardes en namen van attributen verschillen per taal. Hierdoor is besloten om de attributen van producten in een aparte tabel op te slaan. Vertalingen van de attribuutnamen worden opgeslagen in een aparte tabel. Sommige attributen hebben ook een Unit, zoals bijvoorbeeld centimeter voor lengtes. De waardes van een attribuut worden per product opgeslagen in een koppeltabel met eventueel een language als de waarde maar voor één taal geldt. Aangezien de ‘value’-kolom vaak dezelfde waarde heeft, was het een optie om te normaliseren door de value in een eigen tabel te zetten om deze te hergebruiken. Er is toch besloten om dit niet te doen, omdat dit naar verwachting de performance negatief zou beïnvloeden.



Figuur 6: ERD

6.4.2.2 Category

Category is eigenlijk niets anders dan een verzameling producten. Een pagina kan een categorie hebben om de producten van die categorie op de website te tonen. Aangezien de frontend moet weten op welke attributen er makkelijk gezocht kan worden is er een koppeltabel toegevoegd tussen Category en Attribute: SelectorFormAttribute. In deze tabel wordt ook een type opgeslagen hoe de frontend-gebruiker dat attribuut ziet (bijvoorbeeld checkbox/dropdown) en de volgorde van de attributen.

6.4.2.3 ProductDescription

De ProductDescription-tabel is toegevoegd om de beschrijvingen van producten in op te slaan, aangezien de beschrijving vaak hetzelfde is voor enkele tientallen producten en er vertalingen van deze beschrijvingen opgeslagen moesten worden. Deze beschrijvingen worden voornamelijk gebruikt op de dynamische productpagina's.

6.4.2.4 Download, Drawing en Photo

Alle bestanden die bij producten horen, worden bewaard in het YaPDM-systeem. De database slaat alleen verwijzingen naar de bestanden op. Deze verwijzingen staan in de tabellen Download, Drawing en Photo afhankelijk van wat voor soort bestand het is. Drawing is de maatschets van het product, wat altijd op de pagina van een product te bekijken moet zijn. Photo slaat de foto's op van het product en Download bevat alle andere soorten bestanden. De normale downloads moeten ook op pagina's zonder producten gezet kunnen worden, waarvoor de koppeltabel tussen Download en Page is opgenomen.
Er had gekozen kunnen worden om alle bestanden in één tabel te zetten. Het is zo echter makkelijker om de beschrijvingen en alternatieve teksten van de foto's te beheren. Ook is zo waarschijnlijk het zoeken naar een bepaalde maatschets of bepaalde foto's een stuk sneller.

6.4.2.5 WebsiteUser

De frontend-gebruikers worden opgeslagen in WebsiteUser. De status van de gebruiker geeft aan of de gebruiker al goedgekeurd is. De meeste gegevens (naam, adres, etc.) worden normaal gesproken alleen opgeslagen in het ERP, zodat er één centrale plek is voor het klantenbeheer. Voordat de gegevens in het ERP komen of als de klant zijn gegevens wilt wijzigen, zullen de gegevens toch tijdelijk opgeslagen moeten worden in de database. Hiervoor is de extra tabel WebsiteUserChange opgenomen in het ERD. Wanneer de wijziging doorgevoerd is in het ERP door een medewerker van Stokvis verdwijnt de data weer uit het CMS.

6.4.2.6 Translation

Vertalingen van teksten worden opgeslagen in aparte tabellen in plaats van in dezelfde tabel. Hier zijn een aantal redenen voor. Ten eerste zijn de meeste websites alleen in het Nederlands beschikbaar, waarvoor dus geen van de vertalingen nodig is. De onnodige informatie voor deze websites is dus in aparte tabellen geplaatst. Ten tweede wilde de product-owner makkelijk een nieuwe taal kunnen aanmaken mocht het nodig zijn zonder veel werk voor de ontwikkelaars. In de huidige situatie hoeft alleen de taal aan de Language-tabel toegevoegd te worden om een nieuwe taal te kunnen ondersteunen.

6.4.3 Cache

Aangezien de producten in de backend worden ingevoerd en opgeslagen, voert de frontend een API-call uit naar de backend om productengegevens op te halen. Na een aantal eerste tests, bleek dat de performance hiervan niet goed genoeg was om dit bij elke pagina te doen. Bij een test-set van ongeveer 3.000 producten in de database, duurde het namelijk ongeveer 15 seconden om alle producten van een categorie op te halen. De performance issue lag vooral bij de informatie opzoeken in de database en niet in de connectie zelf. Na verbeteringen in het opzoeken is deze tijd verlaagd naar ongeveer 10 seconden, maar dit was nog steeds met maar 3.000 producten in de database. Door verwachtingen dat deze tijd weer zou oplopen als de database eenmaal helemaal gevuld zou zijn, is hierna besloten om de JSON-response van de API-call op te slaan als een soort cache op de frontend. Elk uur worden de API-calls opnieuw gestuurd om de JSON-bestanden te vervangen met de nieuwe data. Dit gebeurt alleen met de API-calls die een gehele categorie ophalen. Voor de pagina van één product wordt nog wel een API-call verstuurd. Het nadeel hiervan is dat wijzigingen in de producten niet meer rechtstreeks op de categoriepagina's op de websites staan, maar de verbetering in performance was significant.

6.4.4 Conclusie

Aan de hand van de kwaliteitscriteria is de gewenste architectuur opgesteld in figuur 5. Door het scheiden van de frontend- en backend-applicaties wordt gevoelige data beschermd en beïnvloed het aantal bezoekers op de ene applicatie de andere applicatie niet. Om de performance van de productenpagina's te garanderen wordt er gebruikt gemaakt van een cache voor de categorieën. De kwaliteitscriteria zullen getest worden in hoofdstuk 6.5 om de kwaliteit te garanderen.

6.5 Deelvraag 5: Hoe kunnen de kwaliteitscriteria het beste getest worden?

Nadat de kwaliteitscriteria waren opgesteld en de architectuur bepaald, is er onderzoek gedaan naar de beste manier om de kwaliteitscriteria te testen. Aangezien de aanpak flink kan verschillen per kwaliteitsattribuut is gekozen om per hoofdkwaliteitsattribuut te onderzoeken hoe de requirements van dat attribuut het beste getest kunnen worden.

6.5.1 Performance Efficiency

Requirements 1.1 tot en met 1.5 uit tabel 9 vallen onder het hoofdkwaliteitsattribuut Performance Efficiency. Er is begonnen door deskresearch te doen naar de verschillende manieren van het testen van performance.

6.5.1.1 Theoretisch kader

Er zijn vele bronnen die het belang aangeven om de performance van een applicatie te testen. Vaak wordt echter niet aangegeven wat de beste manier is om die performance te testen en hoe dat moet. Microsoft heeft een handleiding gemaakt voor performance testen van een webapplicatie (Meier, Farre, Bansode, Barber & Rea, 2007), waarin ze de verschillende manieren van testen benoemen en uitleggen. Naast standaard performance tests die naar bijvoorbeeld de response tijd kijken, benoemen zij ook unittests en drie subcategorieën van performance tests: loadtests, stresstests en capacity tests.

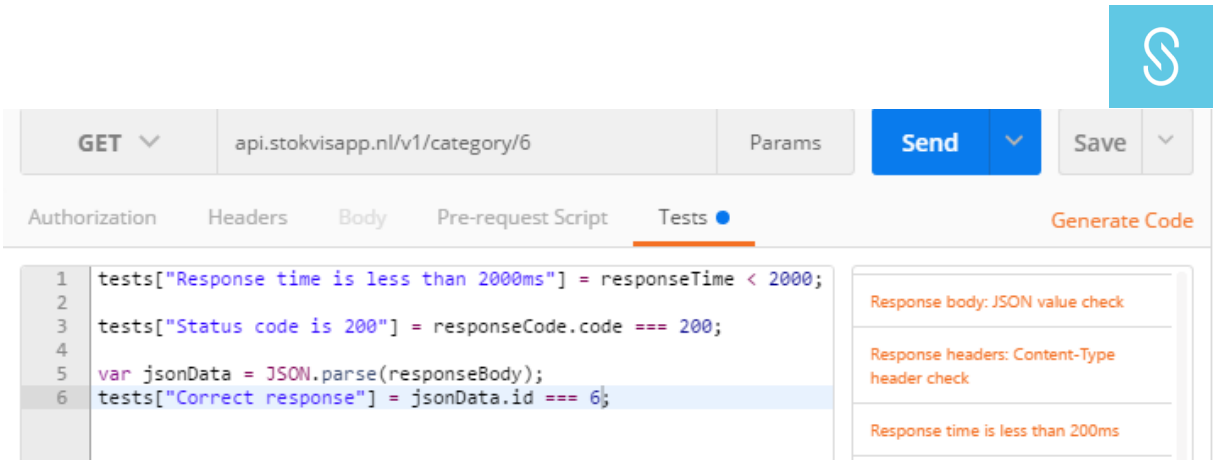
Unittests testen een samenhangend stuk code. Normaal gesproken worden unit tests vooral gebruikt om correctness te testen. Het is echter ook mogelijk om performance te testen met unittests (Meier et al, 2007). Het voordeel van deze tests is dat als er een probleem is, het direct duidelijk is waar het probleem zit. Een groot nadeel is dat er veel tests nodig zijn om alle code af te dekken en dat als de code verandert, de tests vaak ook moeten veranderen.

Loadtests kijken of het systeem het verwachte aantal bezoekers op één tijdstip tegelijk aankan zonder significant performanceverlies (Meier et al, 2007). Met loadtests kan dus beoordeeld worden of er problemen optreden als er meerdere bezoekers tegelijk een website bezoeken en of de hardware de hoeveelheid bezoekers aankan. Een nadeel van loadtests is dat de resultaten bij elke testrun kunnen verschillen en dat er dus met variatie in de response tijd rekening moet worden gehouden. Stresstests beoordelen het gedrag van het systeem wanneer er veel meer verkeer naar de websites gaat dan het maximum verwachte aantal bezoekers (Meier et al, 2007). Stresstests worden vooral gebruikt om te kijken of er fouten optreden die bij normaal gebruik niet voorkomen.

Capacity tests worden gebruikt om vast te stellen hoeveel gebruikers/transacties een systeem kan ondersteunen terwijl de performance doelen nog behaald worden (Meier et al, 2007).

6.5.1.2 Uitgevoerde tests

In eerste instantie werd tijdens de development vooral de werking van de webservices getest met behulp van Postman, een tool om API's mee te testen. Zelf noemen zij zich het Zwitserse zakmes onder de API-tools (www.getpostman.com), omdat de tool zo veel mogelijk maakt. Aangezien de app het ook mogelijk maakt om tests voor de response tijd toe te voegen bij elke request, was het erg makkelijk om hier ook de performance mee te testen (zie figuur 7). Met Postman zijn ook de eerste tests uitgevoerd, op basis waarvan (zie hoofdstuk 6.4.3) besloten is om een cache toe te voegen op de frontend-webserver.



Figuur 7: Voorbeeld Postman test

Postman is echter niet goed geschikt om de websites te testen. Zo wordt er geen javascript of CSS geladen bij de requests. Zonder de javascript werkt bijvoorbeeld de categoriepagina niet. Vervolgens is dus gekeken naar andere manieren om de performance-kwaliteitscriteria te testen. Om requirement 1.5 te testen, zal een loadtest moeten worden uitgevoerd om de performance met meerdere gebruikers tegelijk te kunnen testen. Bij loadtests kunnen ook direct de response tijden bekeken worden en worden geëvalueerd of ze voldoen aan de kwaliteitscriteria 1.1 tot en met 1.4.

Er is ook gekeken naar de mogelijkheid om unittests uit te voeren om de criteria te testen. Yii2, het framework waarop het CMS is gebouwd, ondersteunt standaard Codeception, een PHP testing framework, waarmee makkelijk unittests, functionele en acceptatie tests geschreven en uitgevoerd kunnen worden. Helaas werkte de standaard implementatie van het framework door de aanpassingen aan de architectuur niet goed meer in ons project. Aangezien het oplossen van dit probleem te veel tijd innam en de kwaliteitscriteria al getest werden met behulp van Postman en de loadtest, is er voor gekozen om geen unittests uit te voeren.

Als tool om de loadtests uit te voeren is gekozen voor Telerik Test Studio. Test Studio automatiseert het testen, waardoor je makkelijk GUI, performance en load testen kan uitvoeren (www.telerik.com/teststudio). Onder andere grote maatschappijen zoals Microsoft, HP en Dell gebruiken Test Studio om te testen. Met Test Studio stel je eerst een scenario op door acties uit te voeren in een browser. Deze acties neemt Test Studio op waarna het deze acties kan herhalen voor meerdere gebruikers tegelijkertijd tijdens de loadtest. Om alle kwaliteitscriteria betreffende performance te testen zijn drie verschillende scenario's opgesteld. Criteria 1.3 en 1.4 zijn samen in één scenario getest, omdat iemand altijd eerst naar een productenpagina zal gaan voor hij de datasheet zal downloaden.

1) Iemand kijkt naar pagina's zonder producten en klikt door verschillende pagina's heen (criterium 1.1).
2) Iemand kijkt naar een categoriepagina en zoekt door de producten (criterium 1.2).
3) Iemand gaat naar een productenpagina en download een datasheet (criteria 1.3 en 1.4).

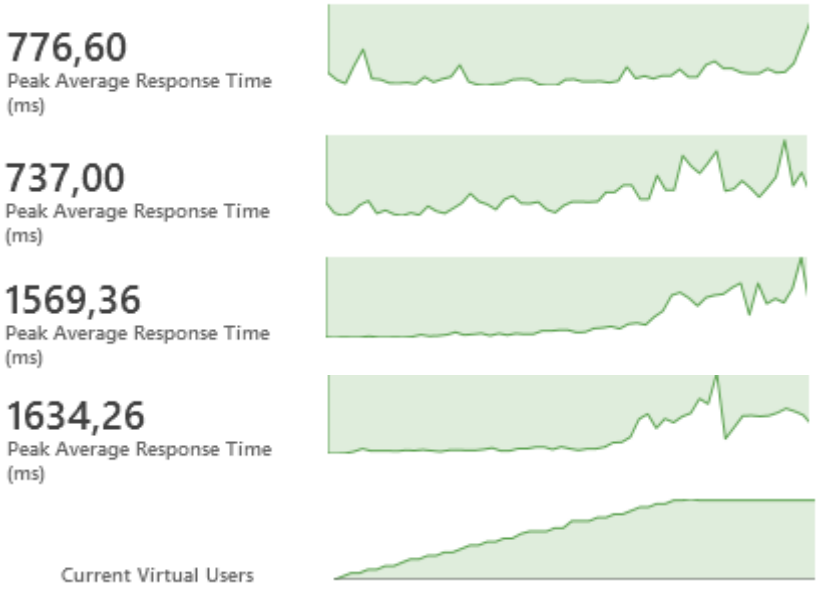
Deze scenario's zijn zowel los als gezamenlijk in een loadtest uitgevoerd. Gezamenlijk betekent dat 33% van de gebruikers scenario 1 doen, 33% scenario 2 en 33% scenario 3. Na een aantal keer testen met een variabele duur, is besloten om alle tests uit te voeren met een duur van 15 minuten, waarbij in de eerste 10 minuten het aantal gebruikers langzaam groeit van 1 naar het maximale aantal gebruikers. Elke 15 seconden wordt gekeken hoeveel tijd het kost om een response te krijgen. Tests met een langere duur leverden geen extra informatie op, waardoor voor deze tijdswaarden gekozen is. Alle tests zijn uitgevoerd op de productie-omgeving, aangezien deze op dat moment nog niet in gebruik was genomen. In tabel 11 staan de uiteindelijke resultaten van de tests. De volledige testresultaten zijn opgenomen in het testrapport in bijlage 4. Bij de resultaten is gekeken naar de hoogste gemiddelde response tijd, zowel van alle requests als per request.

Scenario	Testresultaat (35 gebruikers)	Testresultaat (70 gebruikers)
1	2801 ms (181 ms[1])	7410 ms (349 ms)
2	146 ms	246 ms
3	305 ms	477 ms
Gezamenlijk run 1	777 ms	737 ms
Gezamenlijk run 2	-	1569 ms
Gezamenlijk run 3	-	1634 ms

Tabel 11: Loadtest resultaten (hoogste gemiddelde response tijd)

Bij de resultaten kwam naar voren dat de grootste bottleneck qua performance op het moment ligt bij de afbeeldingen van de normale pagina's. De reden voor de bottleneck is dat dezelfde afbeeldingen nu heel vaak werden opgehaald. Toen dezelfde hoeveelheid gebruikers over drie verschillende websites met verschillende afbeeldingen werden verdeeld, was de hoogste gemiddelde laadtijd namelijk maar 2098 ms in plaats van 7410 ms. De verwachting is dus ook dat dit geen problemen zal opleveren in productie, aangezien de bezoekers over de websites verdeeld zullen zijn en dus niet allemaal dezelfde pagina's met dezelfde afbeeldingen zullen laden. Uiteraard zou het beter zijn om, indien mogelijk, deze bottleneck wel op te lossen.

Verder valt bij de resultaten op dat bij het gezamenlijke scenario de performance beter lijkt te worden bij meer gebruikers. Dit is enigszins misleidend, omdat gekeken wordt naar de hoogste gemiddelde responsetijd. In figuur 8 worden de grafieken getoond van de gemiddelde response tijd gedurende de tests. Hieruit wordt duidelijk dat alhoewel de hoogste gemiddelde response tijd tot ongeveer 1600 ms kan oplopen, de gemiddelde response tijd in het algemeen een stuk lager ligt. Bij het analyseren van de pagina-specifieke response tijden (zie bijlage 4, tabel 8) bleek dat ook bij deze tests de afbeeldingen van de normale pagina's het gemiddelde verhoogden. Wat ook opvalt is dat de pieken in response tijd eigenlijk pas beginnen als het aantal gebruikers bijna 70 is. Zolang de belasting van het systeem minder blijft dan de verwachte piekbelasting, lijken zich geen problemen voor te doen.



Figuur 8: Grafieken loadtests

Aangezien de hoogste gemiddelde response tijd dus niet het hele beeld geeft, is voor het beoordelen van de performance requirements gebruik gemaakt van het gemiddelde over de gehele testduur. Ook is gekeken naar de grafieken om te bepalen hoe de response tijden zich gedurende de tests

[1] Indien geen afbeeldingen worden geladen

gedroegen. In tabel 12 zijn de performance requirements beoordeeld met behulp van de testresultaten. Voor requirement 1.1 en 1.2 is gekeken naar het gemiddelde van de bijbehorende pagina's en afbeeldingen. Voor requirement 1.3 en 1.4 is gekeken naar de pagina specifieke response tijden in plaats van naar het gemiddelde van het gehele scenario. De tijden zijn genomen uit de drie testruns van het gezamenlijke scenario met 70 gebruikers.

Nr.	Requirement	Gemiddelde response tijd	Voldaan
1.1	Het laden van een pagina zonder producten moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	~609 ms	Ja
1.2	Het ophalen van alle producten van een categorie moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	~390 ms	Ja
1.3	Het ophalen van de informatie van één product moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	443 ms	Ja
1.4	Het genereren en ophalen van een productdatasheet moet altijd binnen 10 seconden gebeuren.	768 ms	Ja
1.5	Het systeem moet minimaal 35 users op de frontend met een piekbelasting van 70 users tegelijkertijd aan kunnen zonder significant performanceverlies.		Ja

Tabel 12: Requirements Performance Efficiency

6.5.2 Maintainability

Requirements 2.1 tot en met 2.3 uit tabel 9 vallen onder het hoofdkwaliteitsattribuut Maintainability. Deze requirements zijn minder makkelijk te meten dan bijvoorbeeld de performance. Er is gekozen om per requirement te kijken hoe deze het beste gevalideerd kan worden.

Ten eerste is gekeken naar requirement 2.1: De producteninformatie moet toegankelijk zijn voor andere systemen dan het CMS. Er zijn twee manieren in het systeem waarop de producteninformatie toegankelijk is voor andere systemen. In de architectuur is een aparte database opgenomen met de producteninformatie. Waar nodig zou dus een systeem toegang kunnen krijgen tot de database om informatie van producten te kunnen wijzigen. Daarnaast is de informatie ook verkrijgbaar via de webservices. Hiermee voldoet het systeem dus aan requirement 2.1.

Requirement 2.2 is niet makkelijk meetbaar. Zelfs als in het proces bij SalesCare de tijd werd opgenomen per bug, geeft dat de tijd voor het oplossen van de fouten en niet voor het vinden van de fouten. Daarom is gekozen om de vraag aan de programmeurs zelf te stellen in een korte enquête. De resultaten van deze enquête staan in tabel 13. Het nadeel aan de enquête is dat op het moment van afnemen de structuur nog maar kortgeleden is gerealiseerd, dus er is nog niet veel tijd geweest om de Maintainability te testen. Ook is de student één van de ondervraagden (Jonathan). Hierdoor zouden zijn resultaten vertekend kunnen zijn, omdat hij de nieuwe structuur zelf ontworpen en geïmplementeerd heeft en dus het systeem vrijwel volledig kent. Dit geeft echter ook een beeld van hoe onderhoudbaar het systeem is als de programmeur het systeem goed kent.

Vraag 1 en 2 gaan over requirement 2.2. Er blijkt dat er grote verschillen bestaan per programmeur. De reden hiervoor is waarschijnlijk dat de programmeurs vaak aan verschillende onderdelen werken. Stefan werkt vaak aan de lastigere onderdelen, zoals de afbeeldingen goed opslaan en het ondersteunen van websites met meerdere talen. Ian daarentegen werkt vaak aan kleinere

onderdelen, waardoor het vinden van fouten een stuk minder lang duurt. Iedereen blijft echter binnen de eisen van het kwaliteitscriterium.

Vraag	Ian	Jonathan	Stefan
1) Hoelang duurt het gemiddeld om de locatie van een fout te ontdekken?	10 minuten	5 minuten	30 minuten
2) Wat is de langste tijd die je hebt besteed aan het vinden van een fout?	30 minuten	1 uur	2 uur
3) Hoelang ben je gemiddeld bezig met testen om een release te valideren als je een bugfix hebt gedaan?	5 minuten	5 minuten	30 minuten
4) Hoelang ben je gemiddeld bezig met testen om een release te valideren als je een nieuwe feature hebt toegevoegd?	15 minuten	20 minuten	1,5 uur

Tabel 13: Resultaten enquête Maintainability

In de enquête zijn ook vragen gesteld om requirement 2.3 te kunnen evalueren. In het huidige project zijn namelijk nog geen geautomatiseerde tests geïmplementeerd. Elke programmeur moet dus zelf testen of hij geen bugs heeft geïntroduceerd. Uit de resultaten van de enquête blijkt dat er ook bij het testen grote verschillen zijn per programmeur. Dit heeft waarschijnlijk dezelfde redenen als bij het vinden van de oorzaak van een fout. Uiteindelijk is op basis van de enquête en de architectuur vastgesteld dat het systeem aan alle kwaliteitscriteria met betrekking tot Maintainability voldoet.

Nr.	Requirement	Voldaan
2.1	De producteninformatie moet toegankelijk zijn voor andere systemen dan het CMS.	Ja
2.2	Een programmeur van het CMS moet binnen twee uur de oorzaak van een fout in het systeem kunnen opsporen.	Ja
2.3	Een nieuwe release moet binnen twee uur gevalideerd kunnen worden.	Ja

Tabel 14: Requirements Maintainability

6.5.3 Reliability

Requirements 3.1 en 3.2 gaan allebei over zaken die fout kunnen gaan. Voor allebei deze requirements zijn tijdens het ontwerpen en ontwikkelen van het systeem bepaalde beslissingen genomen. Zo is voor requirement 3.1 besloten om bij het afhandelen van een paginabezoek, zodra duidelijk is dat het om een productenpagina gaat, de verdere afhandeling gescheiden te houden van de afhandeling van normale pagina's. Vervolgens is de requirement getest door opzettelijk fouten in de productenpagina's te introduceren, zodat deze niet meer getoond konden worden. Hieruit bleek dat de normale pagina's nog gewoon getoond konden worden.

De prijs- en voorraadinformatie van producten wordt opgehaald met behulp van AJAX, Asynchronous JavaScript And XML. Doordat de webservice van het ERP pas wordt aangeroepen nadat de rest van de pagina al geladen is, kan deze call geen vertraging opleveren voor de andere productinformatie. Hiermee voldoet het systeem aan requirement 3.2.

Nr.	Requirement	Voldaan
3.1	Als er een fout zit in pagina's met producten moeten de pagina's zonder producten nog steeds bereikbaar zijn.	Ja
3.2	Als het ERP niet bereikbaar is, moet de andere productinformatie wel zonder vertraging getoond worden.	Ja

Tabel 15: Requirements Reliability

6.5.4 Security

Requirement 4.1 valt onder de Security van het systeem. Zoals in hoofdstuk 6.4 benoemd, is vanwege deze requirement de gehele architectuur aangepast. Alle gevoelige data wordt ofwel in het ERP opgeslagen of in de backend database. Zo worden bijvoorbeeld van gebruikers alleen de naam en het email-adres opgeslagen op de frontend-webserver. Ook de voorraad- en prijsinformatie van de producten wordt niet opgeslagen op de webserver, maar wordt via webservice opgehaald bij het ERP. Al deze maatregelen hebben ervoor gezorgd dat als de frontend webserver wordt gehackt, er geen gevoelige data kan worden buitgemaakt.

Nr.	Requirement	Voldaan
4.1	Gevoelige data mag niet opgeslagen worden op dezelfde server als de websites.	Ja

Tabel 16: Requirements Security

6.5.5 Conclusie

Kwaliteitscriteria zijn erg verschillend en zijn soms lastig te testen. Daarom is per kwaliteitscriteria gekeken naar de beste manier om hem te testen. Bij de performance requirements is uiteindelijk besloten om loadtests uit te voeren. Er zijn geen unit tests uitgevoerd vanwege problemen met het gebruikelijke testframework voor Yii2 en een gebrek aan tijd voor deze tests. Met de loadtests kon de performance van het systeem beoordeeld worden bij het verwachte aantal bezoekers. De enige bottleneck die gevonden is zit bij de afbeeldingen van de normale pagina's. Als alle bezoekers naar dezelfde afbeeldingen kijken, zou de response tijd te hoog kunnen worden.

De andere kwaliteitscriteria waren minder makkelijk met vaste testmethoden te testen. Het waren voornamelijk eisen aan de architectuur en het ontwerp. Voor deze criteria is dus gekeken naar de invloed van de architectuur op de criteria. Ook is er een enquête uitgevoerd voor de Maintainability requirements om te beoordelen of het systeem aan deze eisen voldoet. Uiteindelijk is vastgesteld dat het systeem, zoals het ontworpen en gebouwd is, voldoet aan alle kwaliteitscriteria.

7. Gerealiseerde oplossing

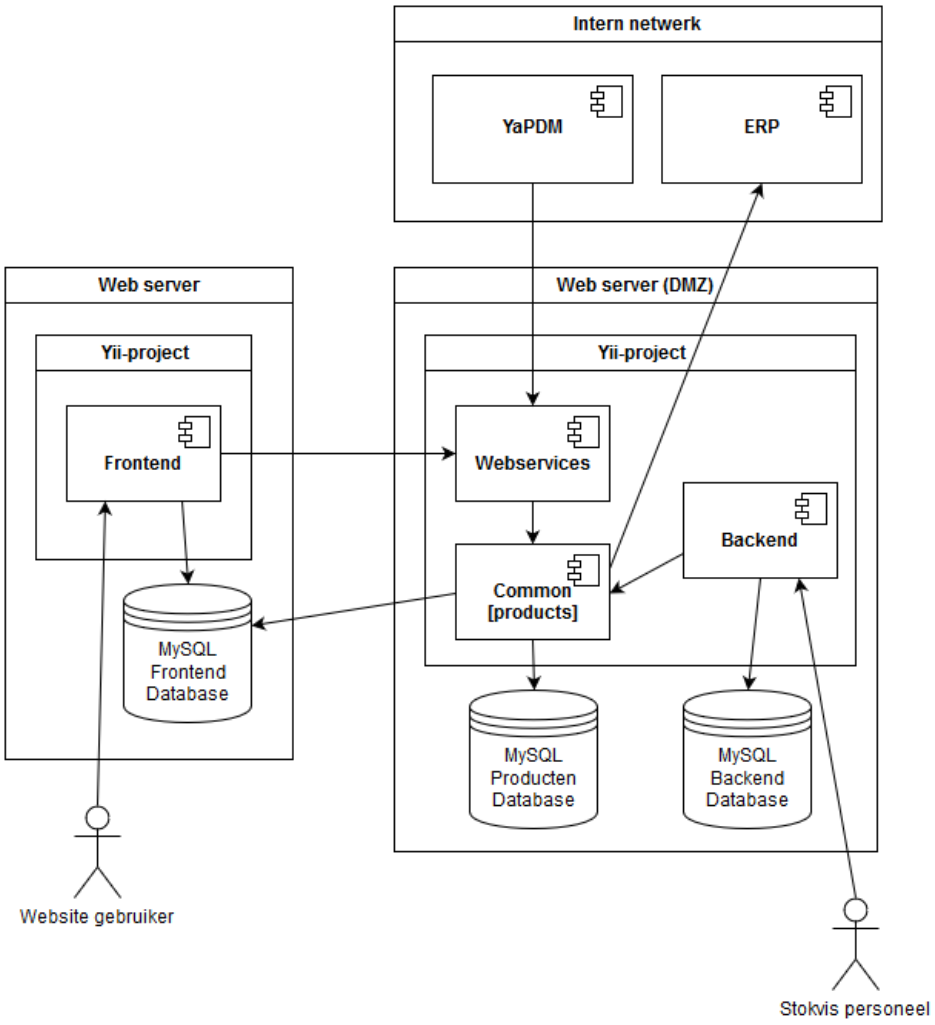
De resultaten van het onderzoek hebben geleid tot een gerealiseerde oplossing van het probleem. De gewenste functionaliteiten, die opgesteld zijn in hoofdstuk 6.2 als user stories, staan in tabel 17 met hun status. Zoals te zien, zijn in ieder geval alle Must-have user stories geïmplementeerd en een deel van de Should-have functionaliteiten. Ook wordt er verder gewerkt aan de webshop-functionaliteiten. Op moment van schrijven worden de producten in het systeem gezet en worden de huidige versies van de websites omgezet naar het CMS. Er is in ieder geval al één site zonder producten live (classyachtseurope.com) en de verwachting is dat het grootste deel van de websites in de loop van het jaar ook in gebruik zal worden genomen. Ook heeft de product owner al zijn interesse getoond in verdere uitbreidingen van het systeem.

Nr.	Omschrijving	Prioriteit	Geïmplementeerd
1.1	Als gebruiker wil ik producten kunnen bekijken	Must-have	Ja, met uitzondering van de ERP-data
1.2	Als gebruiker wil ik files kunnen downloaden van een product	Must-have	Ja
1.3	Als gebruiker wil ik mijzelf kunnen registreren	Must-have	Ja
1.4	Als gebruiker wil ik mijn persoonlijke gegevens kunnen wijzigen	Should-have	Nee
1.5	Als gebruiker wil ik een wachtwoord vergeten-functie kunnen gebruiken	Should-have	Ja
1.6	Als gebruiker wil ik extra autorisaties kunnen aanvragen	Should-have	Nee
1.7	Als gebruiker wil ik een offerte kunnen aanvragen	Could-have	Nee
1.8	Als gebruiker wil ik een bestelling kunnen plaatsen	Could-have	Nee
1.9	Als gebruiker wil ik mijn order historie kunnen inzien	Could-have	Nee
1.10	Als gebruiker wil ik een offerte kunnen omzetten in een bestelling	Could-have	Nee
1.11	Als gebruiker wil ik de bestelstatus van een order kunnen inzien	Could-have	Nee
2.1	Als contactmanager wil ik een overzicht kunnen opvragen van alle gebruikers die aandacht nodig hebben	Must-have	Ja
2.2	Als contactmanager wil ik nieuwe gebruikers kunnen koppelen aan ons CRM-systeem	Must-have	Ja
2.3	Als contactmanager wil ik nieuwe gebruikers kunnen weigeren	Must-have	Ja
2.4	Als contactmanager wil ik gegevens van een gebruiker kunnen aanpassen	Must-have	Ja
3.1	Als admin wil ik een product kunnen beheren	Should-have	Gedeeltelijk, een versimpelde versie
3.2	Als admin wil ik een productcategorie kunnen beheren	Should-have	Gedeeltelijk, een versimpelde versie
3.3	Als admin wil ik een Excel-bestand met producteninformatie kunnen importeren	Must-have	Ja

3.4	Als admin wil ik productencaches kunnen resetten	Should-have	Gedeeltelijk, het gebeurt elk uur automatisch
3.5	Als admin wil ik een productcatalogus kunnen beheren	Should-have	Nee
3.6	Als admin wil ik attributen kunnen beheren	Should-have	Ja
4.1	Als product author wil ik productpagina's kunnen aanmaken en aanpassen	Must-have	Ja
5.1	Als YaPDM wil ik files kunnen beheren	Must-have	Ja

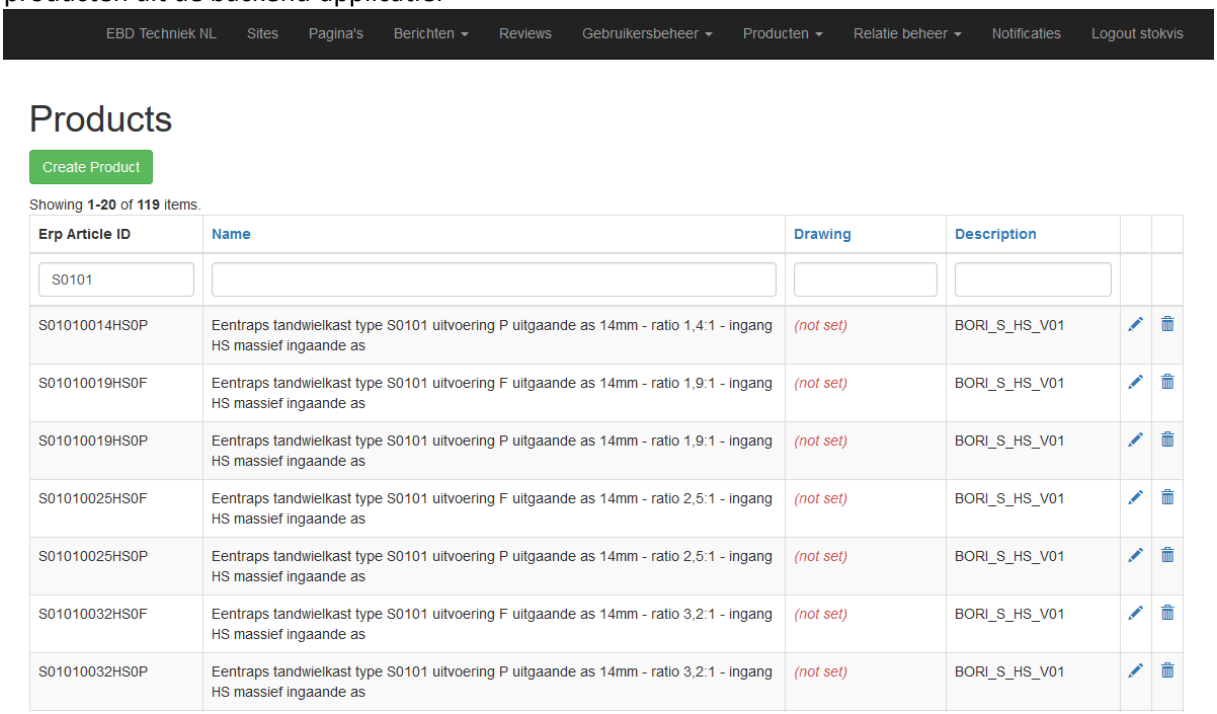
Tabel 17: Gerealiseerde user stories

Bij de realisatie is er een kleine wijziging geweest in de architectuur, zoals deze in hoofdstuk 6.4 is opgesteld. De nieuwe architectuur staat in figuur 9. Het bleek namelijk dat de webservices van het ERP niet publiekelijk toegankelijk waren, maar in het interne netwerk van het bedrijf waren geïmplementeerd. De webservices mochten voor de veiligheid alleen aangeroepen worden vanuit de DMZ-webserver. Hierdoor zijn de calls naar het ERP geïmplementeerd in de common-module in de backend-webserver. De frontend roept de webservices in het DMZ aan, die vervolgens de webservices van het ERP aanroepen. In de webservices van het DMZ wordt een extra check gedaan op de ingevoerde data om zeker te weten dat het ERP geldige requests krijgt.



Figuur 9: Gerealiseerde architectuur

Als voorbeeld van het CMS, staat in figuur 10 een afbeelding van de overzichtspagina van de producten uit de backend-applicatie.



Figuur 10: Backend product-overzichtspagina

Om een beter beeld te geven van het beheergedeelte van het CMS wordt hieronder een gedeelte van de folderstructuur gegeven van het Yii2-project dat op de backend-webserver staat.

- Backend
  - Controllers
  - Models
  - Modules
    - Products (bevat alle functionaliteiten die met producten te maken hebben)
      - Controllers
      - Models
      - Views
  - Views
  - Web
    - Uploads
      - Yapdm (bevat de YaPDM-files)
- Common
  - Models
    - Products
- Console
  - Migrations (bevat alle databasewijzigingen)
- Webservices
  - Common (voor functionaliteit die voor elke webservice-module nuttig kan zijn)
  - Modules
    - V1
      - Controllers
      - Models

8. Conclusies

"*Hoe kan de architectuur en het ontwerp van het multisite CMS van SalesCare aangepast worden om de nieuwe functionaliteiten mogelijk te maken en hoe kunnen de kwaliteitscriteria van deze uitbreiding het beste gegarandeerd worden door een goede teststrategie?*"

Met de antwoorden op de deelvragen is een groot deel van de hoofdvraag beantwoord. De bestaande architectuur bestond uit één webserver met een Yii2-project met een frontend en backend module. Nadat de kwaliteitscriteria waren opgesteld, bleek dat de bestaande architectuur niet voldeed aan criterium 4.1: Gevoelige data mag niet opgeslagen worden op dezelfde server als de websites. De architectuur van het CMS is vervolgens drastisch aangepast om ten eerste aan de kwaliteitscriteria te voldoen en ten tweede de nieuwe functionaliteiten mogelijk te maken. De nieuwe architectuur bestaat uit twee webservers: één bevat het beheersysteem van de websites en producten en één bevat de logica om de websites te tonen. Door het splitsen van de webservers hebben beide webservers een duidelijk doel en kunnen er extra beveiligingsmaatregelen worden genomen bij de beheer-webserver.

De opgestelde kwaliteitscriteria vormen een belangrijk doel voor het CMS om te halen. Eén van de belangrijkste eisen van het systeem lag in de performance. Door het gebruiken van een cache op de frontend-webserver om de producten van een categorie in op te slaan, wordt voorkomen dat lange querytijden bij de database de performance van de website verslechteren. Het enige nadeel van de cache is dat wijzigingen in producten niet meer direct live staan, maar dat de wijzigingen elk uur worden doorgevoerd.

Het testen van de kwaliteitscriteria verschilt erg per criterium. Sommige criteria zijn meer een eis aan de architectuur, waarvoor vaak geen tests te schrijven zijn. Sommige architectuurcriteria zijn bijvoorbeeld te testen door de koppeling te meten tussen verschillende modules. De opgestelde criteria waren echter niet op deze manier te meten. Deze criteria moeten dus stuk voor stuk beoordeeld worden om te kijken of het systeem aan het criterium voldoet. Daarnaast zijn er meetbare kwaliteitscriteria, zoals de performance eisen. De performance eisen kunnen goed getest worden met behulp van loadtests. Hierdoor wordt de performance gecontroleerd voor de juiste hoeveelheid gebruikers.

Het gerealiseerde systeem wordt nu naar tevredenheid gebruikt door de klant, al wordt er nog doorontwikkeld aan het CMS en zijn er nog toekomstplannen om het systeem verder uit te breiden.

9. Aanbevelingen

Er is een aantal verbeteringen, die nog niet in de afstudeerperiode geïmplementeerd zijn, die nog gedaan kunnen worden door SalesCare.

Ten eerste kwam uit de loadtests dat er een mogelijke bottleneck zit in de performance door het veelvoudig laden van afbeeldingen. De aanbeveling is om naar deze bottleneck te kijken en te zien of deze verholpen kan worden.

Daarnaast is het aan te raden om geautomatiseerd te gaan testen. Het is een aantal keer voorgekomen dat developers bij het oplossen van een bug een andere bug toevoegen. Een aantal van deze gevallen had voorkomen kunnen worden door functioneel testen. Ook is het aan te raden om unit tests toe te voegen. Hiervoor zou wel eerst Codeception werkend moeten worden gemaakt of er moet gekeken worden naar een ander test-framework.

Tot slot is het aanbevolen om een testomgeving op te zetten. Tot nu toe is er getest op de productie-omgeving aangezien deze nog niet echt in gebruik was genomen. Als de productie-omgeving daadwerkelijk in gebruik is, zouden de tests de performance negatief kunnen beïnvloeden. Het is dus slim om een extra testomgeving op te zetten met zoveel mogelijk dezelfde hardware als de productie-omgeving om zo te kunnen testen zonder de performance van de productie-omgeving aan te tasten.

Om de bovenstaande aanbevelingen te implementeren zou gekeken kunnen worden naar de mogelijkheden voor een ontwikkelstraat. Zo zou bijvoorbeeld de OTAP-ontwikkelstraat een verbetering kunnen zijn in het proces om nieuwe versies te accepteren voor ze op de productie-omgeving worden gezet. De afkorting OTAP betekent Ontwikkeling, Test, Acceptatie en Productie (ACC ICT BV). Door de stappen Test en Acceptatie toe te voegen aan het proces binnen SalesCare zou de kwaliteit van het CMS omhoog kunnen gaan.

10. Procesevaluatie

Tijdens de afstudeerperiode zijn er een aantal dingen geweest die beter hadden kunnen gaan.

De voornaamste verbetering zou zijn om eerder te beginnen met testen. Vanwege deadlines van de opdrachtgever zat er haast achter het ontwikkelen van de applicatie en zijn er geen tests geschreven totdat het grootste deel van het systeem al gebouwd was. Hierdoor konden de tests niet meer gebruikt worden om te kijken of de architectuur nog verbeterd kon worden, maar zijn ze gebruikt om te kijken of het systeem zoals het gebouwd is voldeed aan de eigenschappen. Als de tests eerder waren geschreven, hadden ze gebruikt kunnen worden om ook kleine variaties op de architectuur te beoordelen. Ook was er waarschijnlijk meer tijd geweest om verder te zoeken naar oplossingen voor het probleem met de unit tests. Misschien was er dan een mogelijkheid geweest om ook de unit tests te implementeren, wat een voordeel had kunnen zijn bij het implementeren.

Ook heeft het missen van de requirement om gevoelige data niet op dezelfde server op te slaan een kleine vertraging veroorzaakt. Er was in het technisch ontwerp rekening gehouden met de huidige structuur, waardoor er dus ook gekeken is naar oplossingen die niet voldeden aan de requirement. Als de requirement duidelijk was geweest vanaf het begin, waren mogelijk nog andere architecturen bekeken en had er dus niet gekeken te hoeven worden naar foute architecturen.

11. Bronnen

ACC ICT BV. (2016), *De OTAP-ontwikkelstraat: vier fases*, geraadpleegd op 13 oktober 2016, op https://www.acc-ict.com/de-otap-ontwikkelstraat-vier-fases/

ISO/IEC 25010:2011, *Systems and software engineering–Systems and software quality Requirements and Evaluation (SQuaRE)–Systems and software quality models,* geraadpleegd op 14 september 2016, op https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en:fig:4

Meier, J.D., Farre, C., Bansode, P., Barber, S. & Rea, D. (2007), *Performance Testing Guidance for Web Applications*, Microsoft, ISBN 978-0735625709

Nah, F. (2004), *A study on tolerable waiting time: how long are Web users willing to wait?*¸University of Nebraska-Lincoln, gepubliceerd door *Behavior and Information Technology* (2004, Vol. 23, No. 3, pp.153-163), geraadpleegd op 7 juli 2016, op http://sighci.org/uploads/published_papers/bit04/BIT_Nah.pdf

Postman. (2016), *Postman | Supercharge your API workflow*, geraadpleegd op 3 augustus 2016

SalesCare. (2016), *Sales Care Online Marketing*, geraadpleegd op 5 juli 2016, op https://salescare.nl

Shinder, D. (2005), *SolutionBase: Strengthen network defenses by using a DMZ*, geraadpleegd op 12 oktober 2016, op http://www.techrepublic.com/article/solutionbase-strengthen-network-defenses-by-using-a-dmz/

Steehouder, C.S, *Leren Communiceren*, Noordhoff Uitgevers, ISBN 978-90-01-54702-8 (of EAN 9789001788926)

Telerik. (2016), *Software Testing Tools, Automated Software Testing*, geraadpleegd op 19 september 2016, op www.telerik.com/teststudio

Yii. (2016), *Yii PHP Framework: Best for Web 2.0 Development*, geraadpleegd op 12 september 2015

Plan Van Aanpak

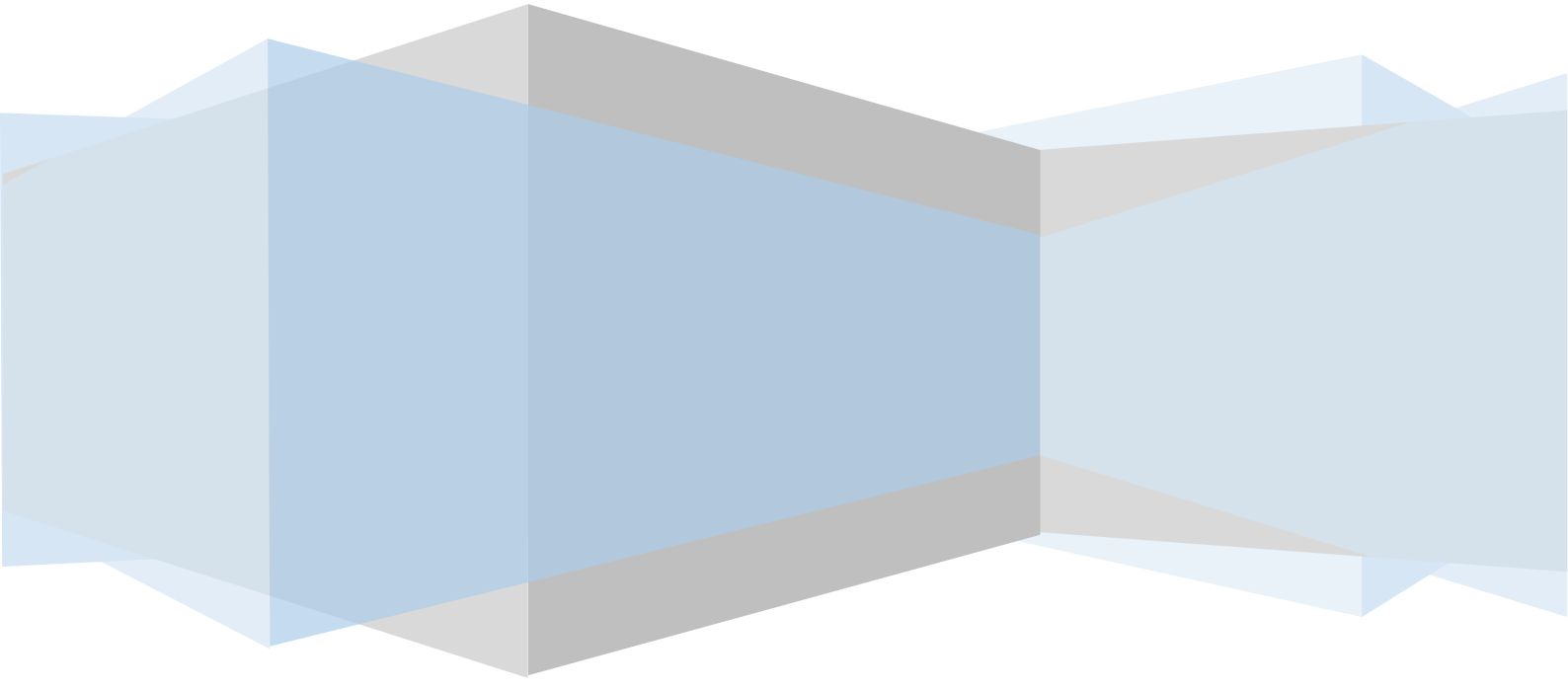
Producten in een multisite CMS

SalesCare

Jonathan Karssen

1630460

Informatica (Software Engineering)



Inhoudsopgave

Inleiding

In dit document wordt het Plan van Aanpak van Jonathan Karssen behandeld. Jonathan Karssen is een vierdejaars student van de opleiding Informatica in de richting Software Engineering aan de Hogeschool Utrecht. De afstudeeropdracht zal uitgevoerd worden bij het bedrijf SalesCare in Utrecht. De afstudeeropdracht betreft het maken van de backend van een Content Management Systeem (CMS) voor een klant van SalesCare. Tijdens de opdracht zal er onderzoek gedaan worden naar de performance eisen van het systeem en hoe deze het beste gegarandeerd en getest kunnen worden.

In dit document zullen ten eerste het bedrijf SalesCare en de probleemstelling worden besproken. Daarna volgt een omschrijving van de opdracht met de doelstelling, randvoorwaarden en de producten die opgeleverd zullen worden. Vervolgens wordt het onderzoek toegelicht met de onderzoeksvraag, de aanpak en een theoretisch kader. Verder volgen nog de technieken en methoden die de student zal gebruiken, de planning en de risico's. Het document wordt afgesloten met de contactgegevens van de betrokkenen en de bronvermeldingen.

1. Context

1.1 SalesCare

De opdracht zal worden uitgevoerd bij SalesCare. SalesCare is een online marketingbureau voor bedrijven met online ambities. Het bedrijf is opgericht in 2010 en heeft inmiddels 12 medewerkers. SalesCare helpt klanten met hun online advertentiestrategie. Daarnaast ontwerpen en ontwikkelen ze websites, maar kunnen ze ook bestaande websites optimaliseren. Op hun website (https://salescare.nl) stellen zij zich zelf zo voor:

> "*Sales Care is een jong en gedreven online marketingbureau. Vanuit ons kantoor in Utrecht werken wij hard om het uiterste uit de online marketingcampagnes van onze klanten te halen. Dagelijks zetten we nieuwe online marketingcampagnes uit, analyseren en optimaliseren we websites en ontwikkelen we nieuwe websites.*"

1.2 Taken

De student zal zich bij SalesCare alleen bezig houden met de taken van de afstudeeropdracht. Daaronder vallen onder andere het maken van documentatie over de opdracht en het ontwikkelen en programmeren van de gespecificeerde functionaliteiten en eisen. De student zal in principe geen taken of projecten uitvoeren die niet bij de afstudeeropdracht horen.

1.3 Andere projecten

De student zal zich in principe niet bezig houden met andere projecten van het bedrijf en deze zullen naar verwachting geen invloed hebben op het afstudeerproject.

2. Probleem

2.1 Aanleiding

SalesCare heeft een klant, Stokvis Group, met vele dochterondernemingen. Voor deze dochterondernemingen moeten zij ongeveer zestig websites kunnen beheren in verschillende talen. Het huidige beheer van de websites is erg veel werk en dit willen zij makkelijker maken. Hiervoor willen zij een multisite en multilingual Content Management Systeem (CMS) gebruiken. Hiermee kunnen ze de websites vanaf één plek beheren. SalesCare is bezig met het maken van dit CMS.

2.2 Probleemstelling

De websites en hun statische pagina's kunnen ondertussen beheerd worden, maar voor het volgende gedeelte van het CMS moeten de producten van de Stokvis Group in het CMS beheerd kunnen worden. Deze producten moeten getoond en uiteindelijk besteld kunnen worden op de beheerde websites. Aangezien het om ongeveer 500.000 producten gaat, zijn de performance en de onderhoudbaarheid van het systeem een belangrijke kwestie. Bij een eerdere poging van een ander bedrijf was de laadtijd van een pagina met ongeveer 3000 producten 25 seconden. Dit zal uiteraard veel sneller moeten in het te realiseren systeem.

3. Opdracht

3.1 Soort

Het soort opdracht van de afstudeeropdracht is een productopdracht. Dit betekent dat er een bruikbaar eindproduct of 'proof of concept' moet worden opgeleverd. De student zal wel eerst mogelijkheden onderzoeken die aan een oplossing kunnen bijdragen en het product moeten ontwerpen.

3.2 Omschrijving

De afstudeeropdracht bestaat uit het onderzoek doen naar de beste oplossing, ontwerpen en ontwikkelen van de backend van een multisite en multilingual Content Management System(CMS). Zoals in de probleemstelling al genoemd, gaat het om een CMS waar producten voor verschillende websites moeten worden bijgehouden.

De websites die beheerd worden in het CMS moeten allemaal een deel van het assortiment van producten van de klant kunnen aanbieden. Stokvis Group verkoopt namelijk ongeveer 500.000 producten met ieder ongeveer 40 verschillende attributen. Deze producten moeten beheerd kunnen worden in het CMS, waarna de websites in het CMS verschillende dingen met deze producten kunnen doen. Zo moet elke website zelf een selectie kunnen maken welke producten ze tonen en moet er een makkelijke manier zijn om in deze producten te kunnen zoeken. Bij elk product is de informatie in meerdere talen beschikbaar en per website kan de prijs verschillen. Uiteindelijk is het de bedoeling dat deze producten ook kunnen worden besteld via de websites, maar dit hoeft niet direct in deze stage te worden toegevoegd.

De voorraad en prijzen van de producten worden door de klant beheerd in een eigen ERP-systeem, wat benaderd kan worden met behulp van een API. Het CMS zal deze informatie zelf niet opslaan en zal dus ook moeten communiceren met dit systeem om alle nodige informatie op te halen. Voor de meeste producten moeten gebruikers van een website ook bestanden kunnen downloaden. Deze bestanden worden ook in een ander systeem opgeslagen, namelijk YaPDM. Dit systeem moet wijzigingen in de bestanden kunnen melden bij het CMS. Het CMS zal hier dus een webservice voor moeten implementeren.

De student zal het hele proces van het maken van software doorlopen. Dit betekent dat hij de eisen en functionaliteiten zal specificeren en de backend zal ontwerpen en ontwikkelen. Hij zal hierbij ook goed de koppeling met de front-end in het oog moeten houden met als belangrijkste onderdeel een filter om producten snel, gemakkelijk en gebruiksvriendelijk te sorteren.

De performance van het systeem is erg belangrijk. Bij een systeem wat een andere leverancier had opgeleverd, was de laadtijd van een pagina met ongeveer 3000 producten ongeveer 25 seconden. Aangezien de performance van het systeem erg belangrijk is, zal de student in zijn scriptie onder andere onderzoek doen naar de beste manieren om de performance van het CMS zo goed mogelijk te maken door op een goede manier te testen.

3.3 Doelstelling

Het doel van de stage is dat de student aan het einde minimaal een 'proof of concept' kan presenteren van het productengedeelte van de backend van het CMS, maar het streven is een werkend product. Ook wordt verwacht dat het product goed wordt gedocumenteerd, zodat andere programmeurs makkelijk inzicht kunnen krijgen in het product. Daarnaast is het doel om de performance van het CMS zo goed mogelijk te maken met behulp van een onderzoek.

3.4 Randvoorwaarden

Er zijn een aantal randvoorwaarden waardoor het project gelimiteerd is. Er kunnen tijdens de stage meer randvoorwaarden bijkomen.

3.4.1 Servers

De servers van SalesCare worden gehost bij het bedrijf Bizway. De servers draaien de volgende programma's/programmeertalen, waarmee de oplossing zal moeten werken:
Apache 2.4.18
MySQL 5.6.29
Php 7.0.3

3.4.2 Beveiliging

Aangezien er uiteindelijk bestellingen met het systeem zullen worden afgehandeld, mogen er uiteraard geen beveiligingslekken zijn.

3.5 Afbakening

Een groot deel van het CMS heeft te maken met het beheren van de websites en hun pagina's. De student zal zich niet bezighouden met deze functionaliteit en zich puur richten op de functionaliteit die nodig is voor de producten. Uiteraard hoort de koppeling tussen de rest van het CMS en de producten hier wel bij.

3.6 Producten

Tijdens de stage zullen er een aantal producten worden opgeleverd. Hieronder volgt een tabel met deze producten en de criteria waarmee de producten beoordeeld kunnen worden.

Product	Criteria
Plan van Aanpak	Zie afstudeerleidraad
Contract afstudeeropdracht	Zie afstudeerleidraad
Functioneel Ontwerp (FO)	Onderbouwd, overzichtelijk, toetsbaar
Technisch Ontwerp (TO)	Onderbouwd, overzichtelijk, toetsbaar
Geïmplementeerde oplossing	Voldoet aan de specificaties uit het FO en TO
Testrapport	Overzichtelijk, toetsbaar
Scriptie	Zie afstudeerleidraad

4 Onderzoek

4.1 Onderzoeksvragen

Voor het onderzoek naar de oplossing van het probleem zijn de volgende hoofd- en deelvragen opgesteld.

4.1.1 Hoofdvraag

> "Hoe kan de architectuur en het ontwerp van het multisite CMS van SalesCare aangepast worden om de nieuwe functionaliteiten mogelijk te maken en hoe kunnen de kwaliteitscriteria van deze uitbreiding het beste gegarandeerd worden door een goede teststrategie?"

4.1.2 Deelvragen

6. Wat is de huidige situatie van het CMS?
7. Wat zijn de benodigde functionaliteiten van de uitbreiding?
8. Wat zijn de gestelde eisen en kwaliteitscriteria aan het systeem?
9. Welke architectuur en ontwerpoplossing voldoen aan de gestelde eisen en kwaliteitscriteria?
10. Hoe kunnen de kwaliteitscriteria het beste getest worden?

4.2 Aanpak

Voor de verschillende deelvragen zijn ook verschillende aanpakken nodig. Hieronder volgt per deelvraag een korte toelichting en de aanpak die de student zal nemen.

1. Wat is de huidige situatie van het CMS?
Voor gekeken kan worden naar de uitbreiding van het CMS moet gekeken worden hoe het CMS er nu uitziet en welke functionaliteiten nu al ondersteund worden. Aangezien de documentatie van het CMS niet altijd goed is bijgewerkt, zal de student zelf het bestaande project in beeld moeten brengen. Dit zal gedaan worden met behulp van een descriptief onderzoek. De student zal kijken naar de bestaande documentatie, de code en de database. Als het nodig is, kan de student om toelichting vragen van de programmeurs. De student zal in ieder geval de huidige architectuur en het huidige ERD tonen.

2. Wat zijn de benodigde functionaliteiten van de uitbreiding?
Om te bepalen wat er precies nodig is, zal de student alle functionaliteiten op een rijtje moeten zetten. Hij zal hiervoor een Functioneel Ontwerp maken, waarin in ieder geval alle user stories zijn opgeschreven. Verder zal de student per user story bepalen hoeveel extra informatie er nodig is om de nodige functionaliteit duidelijk te maken. De functionaliteiten zal de student halen uit een kwalitatief onderzoek met behulp van oude documentatie en gesprekken met developers van het CMS en de product-owner.

3. Wat zijn de gestelde eisen en kwaliteitscriteria aan het systeem?
Naast de functionele eisen zijn er ook andere eisen aan het systeem, zoals de performance. De student zal deze 'non-functional requirements' moeten vaststellen. De student zal eerst mogelijke kwaliteitscriteria zoeken met behulp van een literatuuronderzoek en dan bekijken welke van deze kwaliteitscriteria gelden voor dit project. De student zal de eisen voor deze kwaliteitscriteria uit bestaande documentatie en gesprekken met de opdrachtgever halen. De eisen zullen beschreven worden in het Technisch Ontwerp, zodat er rekening gehouden kan worden met de eisen bij het ontwerpen van de architectuur.

4. Welke architectuur en ontwerpoplossing voldoen aan de gestelde eisen en kwaliteitscriteria?
Nadat de eisen zijn vastgesteld kan gekeken worden naar de mogelijke oplossingen voor het probleem. Indien het nodig is, kan er deskresearch worden uitgevoerd om mogelijke oplossingen te

vinden of te evalueren. Ook zal er kwalitatief onderzoek plaatsvinden door de mogelijke architecturen te bespreken met het hosting-bedrijf om de beste oplossing te vinden. De gekozen architectuur en ontwerp zullen vastgesteld worden in het Technisch Ontwerp, waar de gemaakte keuzes worden toegelicht.

5. Hoe kunnen de kwaliteitscriteria het beste getest worden?
De student zal moeten bepalen hoe de applicatie het beste getest kan worden, zodat vastgesteld kan worden dat de oplossing voldoet aan de gestelde eisen. Hij zal een literatuuronderzoek doen om de beste manieren te vinden om de gevonden kwaliteitscriteria te testen. De student zal vervolgens deze manieren gebruiken om de applicatie te testen. De resultaten van de tests zullen in een testrapport gemeld worden.

4.3 Theoretisch kader

Er zijn een aantal dingen van belang bij het onderzoek. Ten eerste is het nuttig om het multisite Content Management Systeem te definiëren. Een multisite CMS is een systeem dat gegevens moet beheren voor meerdere websites. Onder deze gegevens vallen onder andere pagina's van de website, nieuwsberichten, evenementen en vacatures. Dit betekent dat alle url's van de verschillende websites naar het CMS verwijzen, wat de goede pagina toont voor die url. De websites en pagina's kunnen door medewerkers in het CMS aangemaakt en beheerd worden. In het geval van het CMS van SalesCare kunnen de websites ook verschillende talen hebben. Het is nu echter de vraag hoe de producten van de Stokvis Group ook beschikbaar kunnen worden gemaakt op deze websites. Om dit duidelijk te maken is het eerste deel van de hoofdvraag opgenomen; Hoe kan de architectuur en het ontwerp van het multisite CMS van SalesCare aangepast worden om de nieuwe functionaliteiten mogelijk te maken?

Natuurlijk heeft het CMS ook eisen waar het systeem aan moet voldoen. In het boek *Software Requirements*(Wiegers en Beatty, 2013) worden een aantal verschillende definities van software eisen benoemd in het Engels. Eén van deze definities (Sommerville en Sawyer, 1997) gaat als volgt: "Requirements are a specification of what should be implemented. They are descriptions of how the system should behave, or of a system property or attribute. They may be a constraint on the development process of the system." Aan de hand van deze definitie kunnen de belangrijkste kwaliteitscriteria voor het CMS benoemd worden.

4.3.1 Kwaliteitscriteria

Verreweg het belangrijkste kwaliteitscriterium voor het CMS is de performance van het systeem. Uit eerdere tests is gebleken dat ongeveer 3000 producten met al hun attributen ophalen uit de database erg veel tijd kost. Bij een eerder gemaakt proof-of-concept was de laadtijd hiervan ongeveer 25 seconden. Aangezien uit eerder onderzoek (Nah, 2004) blijkt dat de getolereerde wachttijd van een webgebruiker op een pagina ongeveer twee seconden is, is het garanderen van de performance zeker een prioriteit. Naast het ophalen van een selectie van producten zal ook het ophalen van de informatie van één product vaak voorkomen.

Naast de performance zullen er zeer waarschijnlijk ook nog andere eisen gesteld worden. In het boek *Non-Functional Requirements in Software Engineering* (Chung, Nixon, Yu & Mylopoulos, 2000) worden bijvoorbeeld de volgende kwaliteitscriteria van een softwaresysteem benoemd die niet met performance te maken hebben: correctness, maintainability, verifiability, expandability, flexibility, interoperability, portability en reusability. De student zal in het onderzoek nog moeten kijken welke eisen het CMS op deze punten heeft.

4.3.2 Testen

Er zijn verschillende manieren om kwaliteitscriteria te testen. Microsoft heeft een handleiding gemaakt voor performance testen van een web applicatie (Meier, Farre, Bansode, Barber & Rea,

2007), waarin ze de verschillende manieren van testen benoemen. Naast standaard performance tests die naar bijvoorbeeld de response tijd kijken, benoemen zij ook unittests en drie subcategorieën van performance tests: loadtests, stresstests en capacity tests (Meier et al, 2007).

Unittests testen een samenhangend stuk code. In het geval van het CMS zouden deze tests bijvoorbeeld de model-klassen kunnen testen. Normaal gesproken worden unit tests vooral gebruikt om correctness te testen. Het is echter ook mogelijk om de performance te testen met unittests (Meier et al, 2007). Deze mogelijkheid zal zeker onderzocht moeten worden, want hiermee zouden bijvoorbeeld de twee bottlenecks goed getest kunnen worden zonder tegelijk de afhandeling op de frontend mee te testen.

Loadtests kijken of het systeem het maximum aantal verwachte bezoekers op één tijdstip tegelijk aankan (Meier et al, 2007). Het CMS verwacht niet heel veel bezoekers tegelijk, aangezien de websites erg specialistisch zijn. De student zal nog moeten beoordelen of de loadtests ook voor weinig bezoekers een voordeel opleveren naast de gewone performance tests.

Stresstests beoordelen het gedrag van het systeem wanneer er veel meer verkeer naar de websites gaan dan het maximum verwachte aantal bezoekers (Meier et al, 2007). Aangezien deze situatie niet verwacht wordt, zullen deze tests waarschijnlijk niet uitgevoerd worden.

Capacity tests worden gebruikt om vast te stellen hoeveel gebruikers/transacties een systeem kan ondersteunen terwijl de performance doelen nog behaald worden (Meier et al, 2007). Ook hierbij is de vraag hoe nuttig deze tests zijn voor het CMS. Aangezien het aantal verwachte bezoekers goed is te voorspellen en niet in de nabije toekomst zal veranderen, zal het doel van deze tests waarschijnlijk al behaald worden door middel van loadtests.

Uiteraard zijn er nog andere soorten tests die niet specifiek op de performance zijn gericht. In het onderzoek zal verder worden gekeken naar welke manier van testen het beste geschikt is voor het CMS. Hiervoor is het tweede deel van de hoofdvraag opgenomen; hoe kunnen de kwaliteitscriteria van deze uitbreiding het beste gegarandeerd worden door een goede teststrategie?

5. Technieken en methoden

5.1 Technieken

Tijdens de stage zullen in ieder geval de volgende technieken gebruikt worden door de student. Het is mogelijk dat er nog andere technieken worden gebruikt, die naar voren komen tijdens het onderzoek.

- PHP
- MySQL
- Yii2 (PHP-framework)
- API

In het bestaande gedeelte van het CMS worden PHP en MySQL gebruikt. Zeer waarschijnlijk zal de student hier ook gebruik van moeten maken om de koppeling tussen het CMS en het productengedeelte goed werkend te krijgen. Yii2 is het framework waarop het CMS is gebouwd, dus daar geldt hetzelfde voor. De student heeft nog geen ervaring met Yii2 zelf, maar wel met een ander MVC-framework Symfony. Waarschijnlijk zal het framework dus niet heel veel problemen opleveren.

Om de koppeling te maken tussen de productendatabase van de klant en het CMS zal de student in ieder geval de API van de klant moeten aanroepen. Zeer waarschijnlijk zal hij ook zelf een API moeten opzetten, waarmee het voor de klant mogelijk is om door te geven dat er wijzigingen zijn.

Er zullen ook tests moeten worden uitgevoerd op het CMS. Aangezien er onderzoek gedaan wordt naar de verschillende manieren van testen, zijn er nog geen specifieke testtechnieken opgenomen.

5.2 Methoden

Er zijn een aantal methoden die gebruikt zullen worden tijdens de opdracht.

- Agile / SCRUM
- Watervalmethode
- MoSCoW-methode

Het SalesCare development-team werkt met een variatie op de SCRUM-methode voor het ontwikkelen van software. Dit betekent dat ze in korte sprints van ongeveer twee weken werken, waarin elke keer een deel van het werk wordt opgepakt, waarna opnieuw wordt bekeken wat de status van het werk is en wat de prioriteiten zijn van de taken in de product backlog. Ook wordt er elke dag kort besproken wat elk lid van het team gaat doen. Er worden echter geen SCRUM-verslagen gemaakt.

Voor de opdracht zal de student een combinatie gebruiken van deze SCRUM-methode en de watervalmethode. Aangezien de opdracht technisch erg gecompliceerd is, zal de student eerst de gehele situatie goed moeten analyseren. Hij zal hiervoor een functioneel en technisch ontwerp maken, zodat het duidelijk is wat het systeem moet doen en hoe het technisch in elkaar moet zitten. Nadat dit duidelijk is, zal de student overgaan op hetzelfde systeem als het SalesCare development-team. Dit betekent dat hij nadat de structuur duidelijk is, de verschillende functionaliteiten in sprints zal implementeren.

De student zal bij het prioritizeren van functionaliteiten gebruik maken van de MoSCoW-methode. Dit zal te zien zijn in het functioneel ontwerp, waar user stories benoemd worden als Must-have, Should-have, Could-have of Would-like.

6. Planning

6.1 Activiteiten

Er zijn verschillende activiteiten voor zowel de opdracht zelf als voor het onderzoek. Deze zijn hieronder op een rijtje gezet met een geschat aantal uren wat er voor nodig zal zijn. Om het inplannen van de activiteiten makkelijk te maken, wordt het aantal uren geschat in een meervoud van acht. Het totaal aantal uren van de start van de stage tot de afstudeerzitting is 920 uur.

6.1.1 Opdracht

Activiteit	Beschrijving	Geschat aantal uren
Plan van Aanpak maken	In het Plan van Aanpak wordt de aanpak beschreven waarmee de afstudeerstage zal worden aangepakt.	40
Onderzoek doen naar opdracht	Voor er begonnen kan worden aan het daadwerkelijk ontwerpen van het product zal de student zich voorbereiden door het bestaande project te onderzoeken	32
Functioneel Ontwerp maken	In het Functioneel Ontwerp worden alle functionaliteiten die het systeem moet hebben op een rijtje gezet en toegelicht.	40
Technisch Ontwerp maken	In het Technisch Ontwerp worden alle technische eisen van het systeem toegelicht en wordt weergegeven hoe het systeem de functionaliteiten technisch gaat vervullen.	80
Product ontwikkelen	Het realiseren van de gevonden oplossing uit het Technisch Ontwerp. Indien er iets anders wordt geïmplementeerd dan in het Functioneel of Technisch Ontwerp dit veranderen in de ontwerpen en toelichten.	360
Product testen en testrapport maken	Om te zorgen dat het product goed functioneert zal het goed getest moeten worden. Aangezien er waarschijnlijk op meerdere manieren getest moet worden vanwege het onderzoek is hier extra tijd voor gerekend.	80
Uitloop	In het geval er problemen optreden of er meer tijd voor een andere activiteit nodig is, is er extra uitloop tijd meegenomen, zodat er niet onmiddelijk een probleem ontstaat.	120

6.1.2 Onderzoek

Activiteit	Beschrijving	Geschat aantal uren
Deelvraag 1 beantwoorden	Onderzoek doen naar het antwoord op deelvraag 1 en dit verwerken in de scriptie.	16
Deelvraag 2 beantwoorden	Onderzoek doen naar het antwoord op deelvraag 2 en dit verwerken in de scriptie.	16
Deelvraag 3 beantwoorden	Onderzoek doen naar het antwoord op deelvraag 3 en dit verwerken in de scriptie.	16
Deelvraag 4 beantwoorden	Onderzoek doen naar het antwoord op deelvraag 4 en dit verwerken in de scriptie.	32
Deelvraag 5 beantwoorden	Onderzoek doen naar het antwoord op deelvraag 5 en dit verwerken in de scriptie.	40
Scriptie schrijven	Met de antwoorden van de 5 deelvragen een antwoord vinden op de hoofdvraag en dit verwerken in de scriptie.	40

6.2 Deadlines

Hieronder volgen een aantal deadlines, gesteld vanuit de Hogeschool Utrecht.

Tijdstip	Actie
30 Mei 2016	Start stage
26 Augustus 2016	Inleveren afstudeercontract
13 September 2016	Inleveren 1e conceptscriptie
4 Oktober 2016	Inleveren 2e conceptscriptie
25 Oktober 2016	Inleveren scriptie en bedrijfsbeoordeling
7-11 November 2016	Afstudeerzitting

6.3 Agenda

In de agenda staat per dag aangegeven aan welke activiteiten wordt gewerkt. Het is goed mogelijk dat er dingen in de planning verschuiven of dat er meer overlap tussen activiteiten zal plaatsvinden.

Legenda

	Onderzoeken
	Ontwerpen
	Ontwikkelen
	Testen
	Deadline
	Uitloop
START	Het begin van een activiteit
EIND	Het einde van een activiteit

6.3.1 Mei/Juni

Zondag	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag
29	30 **START** Afstudeer-project	31 **START** Onderzoek doen naar opdracht	1	2	3 **EIND** Onderzoek doen naar opdracht	4
5	6 **START** Plan van Aanpak maken	7	8	9	10 **EIND** Plan van Aanpak maken	11
12	13 **START** Functioneel Ontwerp maken	14	15	16	17 **EIND** Functioneel Ontwerp maken	18
19	20 **START** Deelvraag 1	21 **EIND** Deelvraag 1	22 **START** Deelvraag 2	23 **EIND** Deelvraag 2	24 **START** Deelvraag 3	25
26	27 **EIND** Deelvraag 3	28 **START** Deelvraag 4	29	30		

6.2.2 Juli

Zondag	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag
					1 **EIND** Deelvraag 4	2
3	4 **START** Technisch Ontwerp maken	5	6	7	8	9
10	11	12	13	14	15 **EIND** Technisch Ontwerp maken	16
17	18 **START** Product ontwikkelen	19	20	21	22 **START** Deelvraag 5	23
24	25	26	27	28	29	30
31						

6.2.3 Augustus

Zondag	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag
	1	2	3	4	5 Deelvraag 5	6
7	8	9	10	11	12	13
14	15	16	17	18	19 Deelvraag 5	20
21	22	23	24	25	26 Inleveren afstudeer-contract	27
28	29	30	31			

6.2.4 September

Zondag	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag
				1	2 Deelvraag 5	3
4	5	6	7	8	9	10
11	12	13 Inleveren 1e concept-scriptie	14	15	16 **EIND** Deelvraag 5	17
18	19 **START** Product testen en testrapport	20	21	22	23 Product testen en testrapport	24
25	26 **START** Scriptie schrijven	27	28	29	30 **EIND** Scriptie schrijven	

6.2.5 Oktober

Zondag	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag
						1
2	3 Product ontwikkelen	4 Inleveren 2e concept-scriptie	5	6	7 **EIND** Product ontwikkelen	8
9	10 Product testen en testrapport	11	12	13	14 **EIND** Product testen en testrapport	15
16	17 **START** Uitloop	18	19	20	21	22
23	24	25 Inleveren scriptie en beoordeling	26	27	28	29
30	31					

6.2.6 November

Zondag	Maandag	Dinsdag	Woensdag	Donderdag	Vrijdag	Zaterdag
		1	2	3	4 **EIND** Uitloop	5
6	7	8	9	10	11	12
	Afstudeerzitting					

7. Risico's

Er zijn verschillende risico's bij deze opdracht. Deze worden hieronder genoemd en toegelicht.

Risico	Toelichting	Preventie/Oplossingen
Afwijken van de planning	Als er vertraging is of er moeilijkheden optreden bij een onderdeel, zou het kunnen dat er niet genoeg tijd is om het project af te kunnen maken.	Het belangrijkste is ten eerste dat er goed gepland wordt. Zolang voor alles ruim de tijd wordt ingepland, is er ruimte voor uitloop. Ten tweede is het belangrijk om zodra er wijzigingen zijn in de tijd die voor iets nodig is, deze ook door te voeren in de planning. Hierdoor wordt het risico op vertraging geminimaliseerd.
De student heeft niet de benodigde kennis/vaardigheden	Het zou kunnen dat de student op een gegeven moment niet verder kan, omdat hij iets niet weet of niet snapt.	De student kan eerst op zoek gaan op Google om te kijken of de antwoorden daar te vinden zijn. Als het alsnog niet duidelijk is, kan de student zijn vraag aan de bedrijfsbegeleider voorleggen.

8. Contactgegevens

8.1 Bedrijfsbegeleider

Tamer Saglambilek
E-mailadres: tamer@salescare.nl
Telefoonnummer: 030 7400 999

8.2 Docentbegeleider

Remco Ruijsenaars
E-mailadres: remco.ruijsenaars@hu.nl
Telefoonnummer: 0624738896

8.3 Afstudeerder

Jonathan Karssen
E-mailadres: jonathan.karssen@student.hu.nl
Telefoonnummer: 0615080217

9. Bronvermelding

Chung, L., Nixon, B.A., Yu, E. & Mylopoulos, J. (2000), *Non-Functional Requirements in Software Engineering*, Springer Science+Business Media, geraadpleegd op 7 juli 2016, op https://books.google.nl/books?id=MNrcBwAAQBAJ

Meier, J.D., Farre, C., Bansode, P., Barber, S. & Rea, D. (2007), *Performance Testing Guidance for Web Applications*, Microsoft, geraadpleegd op 8 juli 2016, op http://perftestingguide.codeplex.com/releases/view/6690

Nah, F. (2004), *A study on tolerable waiting time: how long are Web users willing to wait?*¸University of Nebraska-Lincoln, geraadpleegd op 7 juli 2016, op http://sighci.org/uploads/published_papers/bit04/BIT_Nah.pdf

Steehouder, C.S, *Leren Communiceren*, Noordhoff Uitgevers, ISBN 978-90-01-54702-8 (of EAN 9789001788926)

Wiegers, K. & Beatty, J. (2013), *Software Requirements* (3e editie), Pearson Education, geraadpleegd op 7 juni 2016, op https://scholar.google.nl/scholar?oi=bibs&cluster=17436007402552466256&btnI=1

Bijlagen

Bijlage 1: Concept Projectomschrijving

Het betreft een multi-site / multi-language CMS voor een klant die internationaal opereert.
De klant heeft 500.000 producten in een database waar we via een API mee kunnen communiceren.

Per website moet de klant de gewenste producten kunnen weergeven middels selectoren moet er gefilterd kunnen worden. De producten gaan tot 7 lagen diep. Zie afbeeldingen.

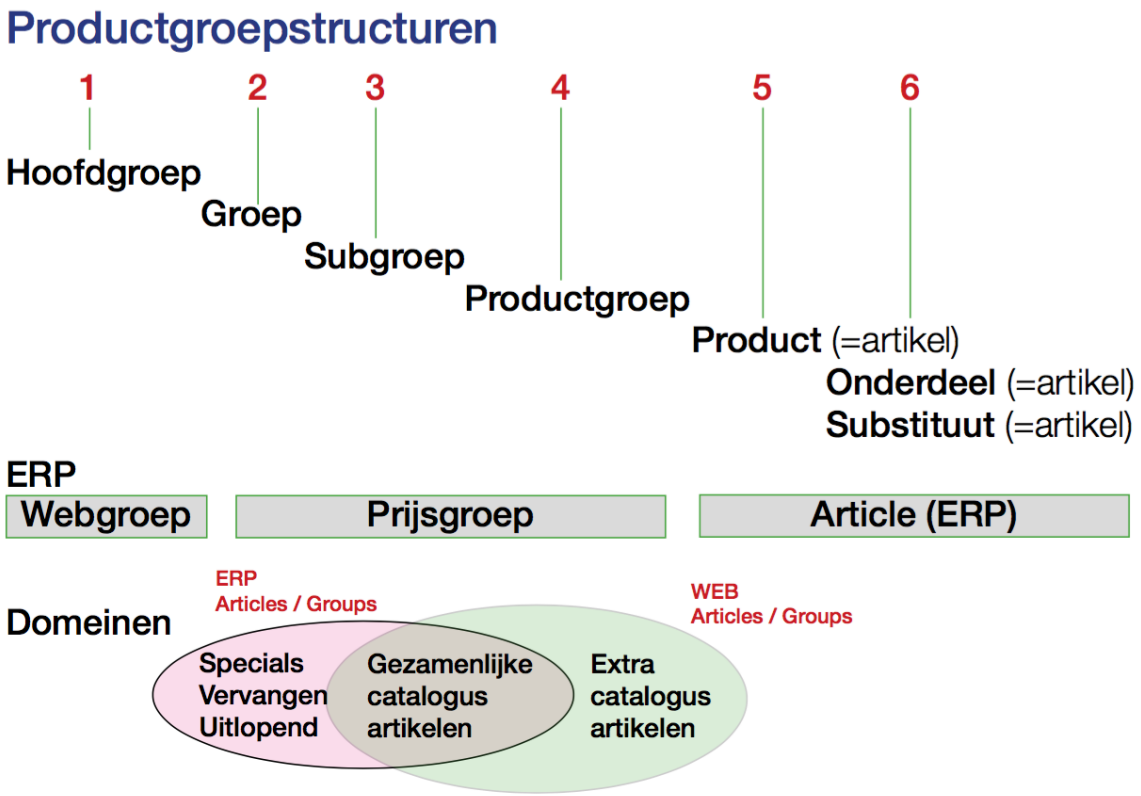
Per hoofdgroep wordt het aan een domein gekoppeld. Sommige producten komen in meerdere productgroepen voor en alle producten hebben pdf downloads beschikbaar die automatisch worden gegenereerd. Productinformatie is ook in meer talen beschikbaar.

Wij zijn van mening dat we dit met verschillende koppeltabellen kunnen oplossen. Enkel moet een schaalbare oplossing zijn omdat nog slechts 40% van de producten is ingevoerd.

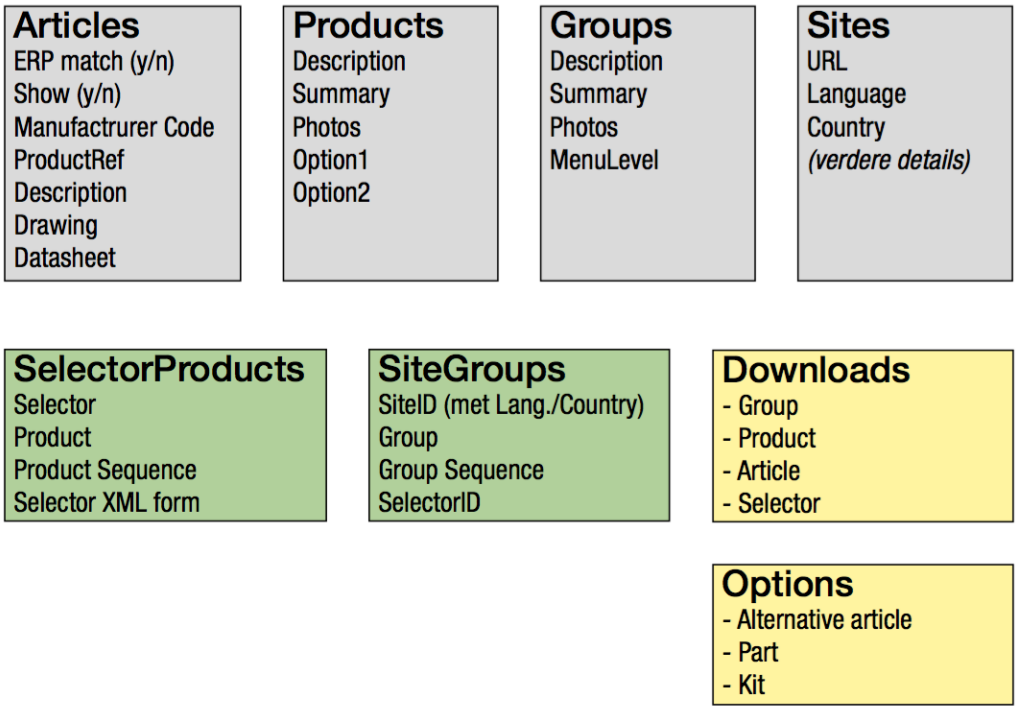
Er zijn totaal 60 websites die gebruik moeten maken van dit systeem, dat wil zeggen dat het erg zwaar kan worden. Informatie moet ook snel en flexibel opgezocht kunnen worden.
Een zoekfunctie moet er komen per website, die moet zoeken in de voor hem bepaalde informatie / producten. Denk hierbij ook aan caching en informatie management.

Klanten kunnen inloggen om de voor hen verschillende downloads te bekijken en mogelijk voor hun vastgestelde prijzen in te zien. Dit is verschillend per klant en gebaseerd op een kortingspercentage die is vastgesteld in hun ERP systeem waarmee we kunnen communiceren.

Onderstaande afbeelding is een overzicht van de huidige structuur.



Database schema



Functioneel Ontwerp

Producten in het CMS

Stokvis Group

Versiebeheer

Versie	Datum	Auteur	Opmerkingen
0.1	10-6-2016	Jonathan Karssen	- Initiële versie
0.2	16-6-2016	Jonathan Karssen	- Opmerkingen verwerkt van de vergadering op 14-6
0.3	21-6-2016	Jonathan Karssen	- Opmerkingen van overleg binnen SalesCare verwerkt
0.4	29-6-2016	Jonathan Karssen	- Opmerkingen van Marcel (Stokvis Group) verwerkt - Flowcharts voor een aantal usecases toegevoegd - Overzicht van productpagina's toegevoegd als hoofdstuk 4 - Userstory 3.7 'Attributen beheren' toegevoegd

Inhoudsopgave

Inleiding

In dit document worden de functionaliteiten beschreven voor het productengedeelte van de multi-CMS voor Stokvis Group. Dit document is bedoeld voor de developers van SalesCare en de ‘product owner’ van Stokvis Group om duidelijk te maken wat de gebruikers met het systeem moeten kunnen.

1. Uitgangspunten

1.1 Actors

Hier volgen de verschillende gebruikers van het systeem, ook wel actors genoemd.

1.1.1 Frontend

- Website gebruikers
    - Anoniem
    - Geregistreerd (moet eerst goedgekeurd zijn door stokvis admin)

Geregistreerde gebruikers krijgen in ERP door middel van autorisaties toegang tot bepaalde functionaliteiten. Een gebruiker kan meerdere verschillende autorisaties hebben per productcategorie en gebruikers worden per website geregistreerd. Dus als een gebruiker naar een andere website gaat zal hij opnieuw moeten registreren. Hij kan dan wel gekoppeld worden aan dezelfde gebruiker in het ERP.

Autorisaties:

- Per gebruiker:
    - Type autorisatie voorraad
        - Nee
        - Ja
        - Aantal
    - Type autorisatie prijzen
        - Verborgen
        - Bruto
        - Netto
- Per productgroep per gebruiker
    - Downloaden tekeningen (J/N) (Dit gaat om alle downloads van producten, niet om de afbeeldingen of maatschetsen of productcatalogi)
    - Opvragen voorraad (J/N)
    - Opvragen prijs (J/N)
    - Orders plaatsen (J/N)
    - Opvragen offertes (J/N)
    - Opvragen orderstatus (J/N)
    - Opvragen orderhistorie (J/N)

1.1.2 Backend

De volgende rollen/rechten moeten gedefinieerd zijn in de back-end van het CMS.Een user moet meerdere Author-rollen kunnen hebben (voor meerdere sites). Een deel van deze rollen bestaan al in het huidige systeem. De onderstreepte rollen moeten nieuw toegevoegd worden.

- Admin
    - Kan alles beheren van alle sites
- Moderator
    - Kan alle nieuwe en geupdate pagina's (tekstueel) aanpassen/goedkeuren voor publicatie. Het betreft de volgende pagina typen:
        - Content
        - Productgroep
        - Berichten (nieuws / evenementen / vacatures / reviews)
- <u>Contactmanager</u>
    - Kan alle geregistreerde front-end gebruikers beheren
- <u>Author product</u>
    - Rechten per website voor aanmaken en aanpassen product pagina's

- Author content
    - Rechten per website voor aanmaken en aanpassen content pagina's
- Author bericht nieuws
- Author bericht evenementen
- Author bericht vacatures
- Author bericht reviews

1.1.3 Stokvis backend

Stokvis gaat een aparte omgeving gebruiken voor file-management. Deze omgeving, YaPDM, zal alle downloads en afbeeldingen van de producten bevatten. Via webservices zal het CMS bereikbaar moeten zijn voor wijzigingen in deze files.

- YaPDM

1.2 Prioriteiten

De prioriteiten zijn gesteld via de MoSCoW methode (Zie https://nl.wikipedia.org/wiki/MoSCoW-methode). Deze prioriteiten zijn gesteld voor deel 2 van het CMS-project. In deel 3 kunnen een groot deel van de prioriteiten veranderen.

Prioriteit	Beschrijving
Must-have	Deze eisen moeten in het eindresultaat terugkomen. Zonder deze user stories is het product niet bruikbaar.
Should-have	Deze user stories zijn zeer gewenst, maar zonder is het product wel bruikbaar.
Could-have	Deze eisen zullen alleen aan bod komen als er tijd genoeg is.
Would-like (Won't-have)	Deze eisen zullen in dit project niet aan bod komen maar kunnen in de toekomst, bij een vervolgproject, interessant zijn.

2. User stories

Hieronder volgen de user stories per actor met de prioriteit die aan de user story is gegeven. Deze user stories worden in hoofdstuk 3 uitgewerkt tot use cases.

2.1 Website gebruiker

Niet elke gebruiker mag elke user story hieronder doen. Op basis van autoriteiten uit het ERP wordt bepaald of de gebruiker de user story mag doen.

Nr	Omschrijving	Prioriteit
1.1	Als gebruiker wil ik producten kunnen bekijken	Must-have
1.1.1	- Als gebruiker wil ik de productcategorieën kunnen bekijken	
1.1.2	- Als gebruiker wil ik op eigenschappen kunnen filteren	
1.1.3	- Als gebruiker wil ik de prijs van producten kunnen bekijken	- Should-have
1.1.4	- Als gebruiker wil ik de voorraad van producten kunnen bekijken	- Should-have
1.1.5	- Als gebruiker wil ik de maattekeningen kunnen bekijken	
1.2	Als gebruiker wil ik files kunnen downloaden van een product	Must-have
1.2.1	- Als gebruiker wil ik files kunnen downloaden van een categorie	
1.3	Als gebruiker wil ik mijzelf kunnen registreren	Must-have
1.4	Als gebruiker wil ik mijn persoonlijke gegevens kunnen wijzigen	Should-have
1.5	Als gebruiker wil ik een wachtwoord vergeten-functie kunnen gebruiken	Should-have
1.6	Als gebruiker wil ik extra autorisaties kunnen aanvragen	Should-have
1.7	Als gebruiker wil ik een offerte kunnen aanvragen	Could-have
1.7.1	- Als gebruiker wil ik artikelen kunnen selecteren voor mijn offerte	
1.7.2	- Als gebruiker wil ik mijn huidige offerte kunnen inzien	
1.7.3	- Als gebruiker wil ik mijn offerte kunnen wijzigen	
1.7.4	- Als gebruiker wil ik mijn offerte kunnen plaatsen	
1.8	Als gebruiker wil ik een bestelling kunnen plaatsen	Could-have
1.8.1	- Als gebruiker wil ik artikelen kunnen selecteren voor mijn bestelling	
1.8.2	- Als gebruiker wil ik mijn huidige bestelling kunnen inzien	
1.8.3	- Als gebruiker wil ik mijn bestelling kunnen wijzigen	
1.8.4	- Als gebruiker wil ik mijn bestelling kunnen plaatsen	
1.9	Als gebruiker wil ik mijn order historie kunnen inzien	Could-have
1.9.1	- Als gebruiker wil ik een order kunnen herbestellen	
1.10	Als gebruiker wil ik een offerte kunnen omzetten in een bestelling	Could-have
1.11	Als gebruiker wil ik de bestelstatus van een order kunnen inzien	Could-have

2.2 Back-end contactmanager

Nr	Omschrijving	Prioriteit
2.1	Als contactmanager wil ik een overzicht kunnen opvragen van alle gebruikers die aandacht nodig hebben	Must-have
2.2	Als contactmanager wil ik nieuwe gebruikers kunnen koppelen aan ons CRM systeem	Must-have
2.3	Als contactmanager wil ik nieuwe gebruikers kunnen weigeren	Must-have
2.4	Als contactmanager wil ik gegevens van een gebruiker kunnen aanpassen	Must-have

2.3 Back-end admin

De back-end admin heeft ook toegang tot alle andere back-end user stories.

Nr	Omschrijving	Prioriteit
3.1	Als admin wil ik een product kunnen beheren	Must-have
3.2	Als admin wil ik een productcategorie kunnen beheren	Must-have
3.3	Als admin wil ik een Excel-bestand met producteninformatie kunnen importeren	Must-have
3.3.1	- Als admin wil ik een new-products Excel-bestand kunnen importeren	
3.3.2	- Als admin wil ik een update-products Excel-bestand kunnen importeren	
3.3.3	- Als admin wil ik een delete-products Excel-bestand kunnen importeren	
3.4	Als admin wil ik productencaches kunnen resetten	Should-have
3.5	Als admin wil ik een productcatalogus kunnen beheren	Should-have
3.5.1	- Als admin wil ik een productcatalogus kunnen samenstellen	
3.6	Als admin wil ik attributen kunnen beheren	Must-have

2.4 Back-end author product

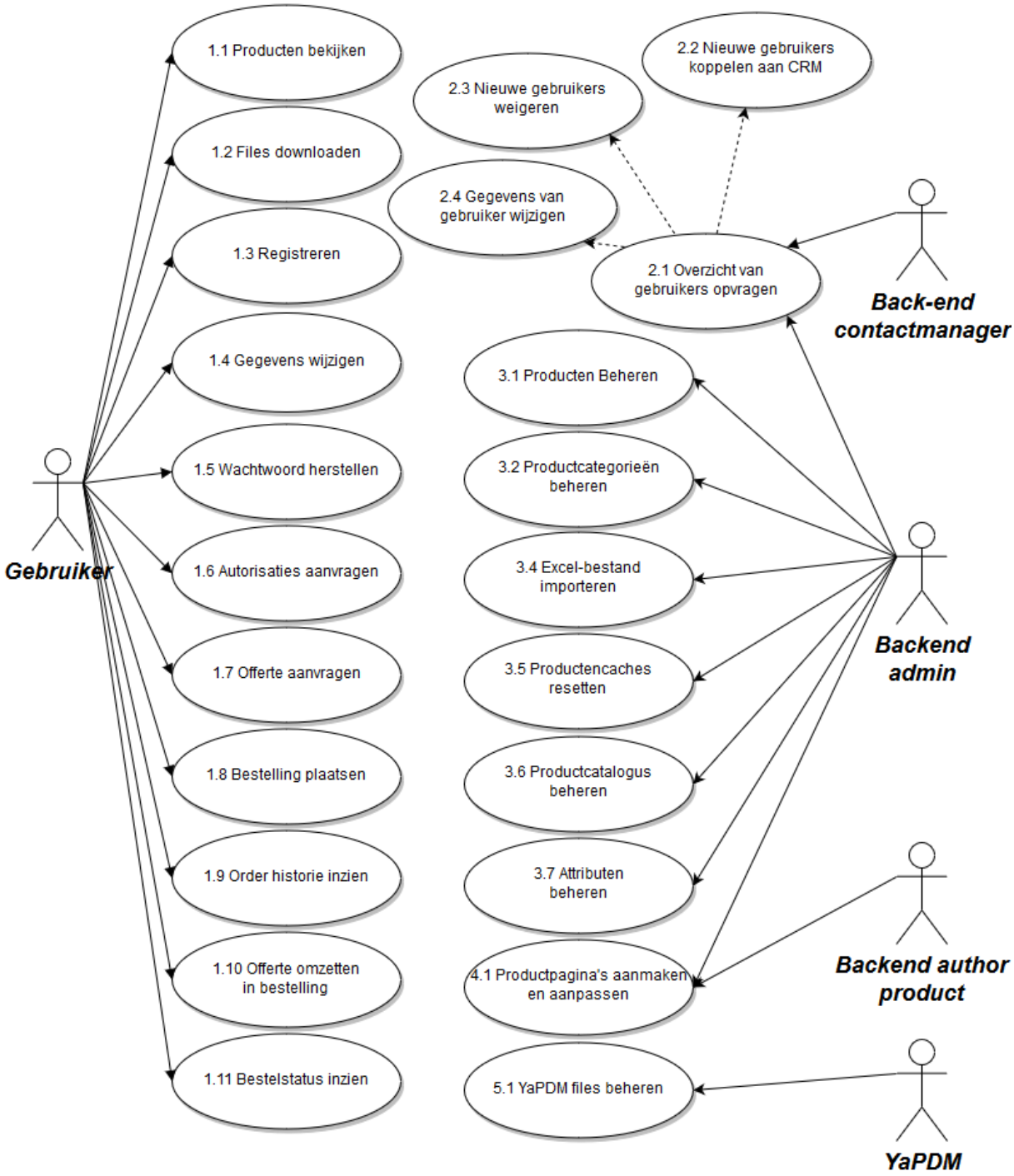
Nr	Omschrijving	Prioriteit
4.1	Als product author wil ik productpagina's kunnen aanmaken en aanpassen	Must-have

2.5 YaPDM

Nr	Omschrijving	Prioriteit
5.1	Als YaPDM wil ik files kunnen beheren	Must-have
5.1.1	- addFile	
5.1.2	-deleteFile	
5.1.3	- updateFile	

3. Use cases

3.1 Use case diagram



Hieronder volgen de korte use case beschrijvingen. Er is voor gekozen om deze vrij simpel te houden met waar het nodig is extra details of flowcharts. Eventueel kunnen ze later verder worden uitgebreid.

3.2 Website gebruiker

Use case	1.1
Naam	Producten bekijken
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik producten kunnen bekijken
Omschrijving	De gebruiker wilt producten bekijken en de details ervan bekijken. Hij kan zoeken in de verschillende productcategorieën en selecteren op basis van selectors. Afhankelijk van de rechten die de gebruiker heeft, kan hij de prijs (netto/bruto/niet), voorraad (aanwezig/aantal/niet) en maattekeningen (wel/niet) zien en files downloaden (wel/niet).

Use case	1.2
Naam	Files downloaden
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik files kunnen downloaden van een product
Omschrijving	De gebruiker moet files kunnen downloaden van zowel categorieën als producten. De gebruiker moet voor sommige files rechten hebben om de bestanden te kunnen downloaden. Naast de opgeslagen files moet ook van elk product een dynamische datasheet gedownload kunnen worden. De datasheet moet alle informatie die op de website staat bevatten.

Use case	1.3
Naam	Registreren
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik mijzelf kunnen registreren
Omschrijving	De gebruiker wilt zich registreren. Hij vult hiervoor een formulier in met de benodigde gegevens (onder andere ook gewilde autorisaties en/of functie), waarna hij een bevestigings-email krijgt om de email te verifiëren. Als de email is geverifieerd zal een stokvis medewerker de gebruiker nog moeten koppelen aan het ERP-systeem en de benodigde autorisaties geven (use case 3.5).

Use case	1.4
Naam	Gegevens wijzigen
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik mijn persoonlijke gegevens kunnen wijzigen
Omschrijving	De gebruiker wilt zijn gegevens wijzigen. Hij vult hiervoor een formulier in met de benodigde gegevens. Een stokvis medewerker zal deze gegevens nog moeten goedkeuren (use case 3.7).

Use case	1.5
Naam	Wachtwoord herstellen
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik een wachtwoord vergeten-functie kunnen gebruiken
Omschrijving	Als de gebruiker zijn wachtwoord is vergeten moet hij een manier hebben om zijn wachtwoord te wijzigen. Als de gebruiker de functie aanroept moet hij zijn gebruikersnaam/email invullen, waarna hij een e-mail krijgt met een link om zijn wachtwoord te veranderen. Deze link verloopt na 15 minuten.

Use case	1.6
Naam	Autorisaties aanvragen
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik extra autorisaties kunnen aanvragen
Omschrijving	Als de gebruiker extra autorisaties wilt, moet hij een verzoek kunnen sturen naar een stokvis medewerker om deze autorisaties te krijgen. De stokvis medewerker kan het verzoek beoordelen en eventueel de autorisaties wijzigen in het ERP-systeem.

Use case	1.7
Naam	Offerte aanvragen
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik een offerte kunnen aanvragen
Omschrijving	Als de gebruiker autorisatie heeft, kan hij producten vanaf de productpagina's in een winkelmandje plaatsen en een offerte-aanvraag opsturen. Hij kan eventueel eerst de offerte wijzigen en producten weer weghalen.

Use case	1.8
Naam	Bestelling plaatsen
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik een bestelling kunnen plaatsen
Omschrijving	Als de gebruiker autorisatie heeft, kan hij producten vanaf de productpagina's in een winkelmandje plaatsen en een bestelling plaatsen. Hij kan eventueel eerst de bestelling wijzigen en producten weer weghalen.

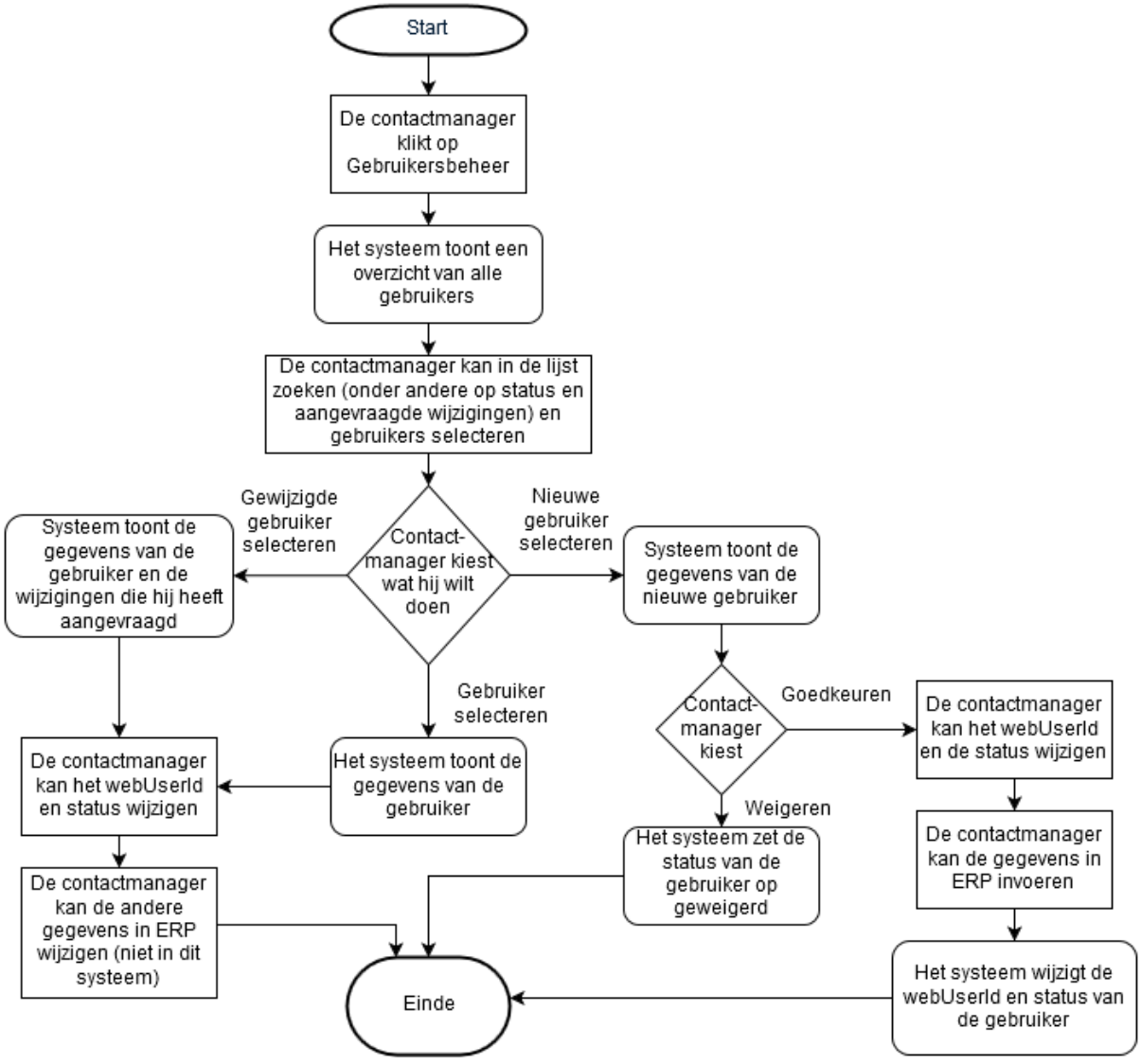
Use case	1.9
Naam	Order historie inzien
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik mijn order historie kunnen inzien
Omschrijving	Als de gebruiker autorisatie heeft, kan hij een overzicht zien van alle vorige offertes en bestellingen van het bedrijf waar hij bij hoort. Eventueel kan hij offertes omzetten in bestellingen (use case 1.10) of bestellingen kopiëren om nog een keer dezelfde producten/aantallen te bestellen als hij autorisatie hiervoor heeft.

Use case	1.10
Naam	Offerte omzetten in bestelling
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik een offerte kunnen omzetten in een bestelling
Omschrijving	Als de gebruiker mag bestellen, kan hij bij de order historie een offerte-aanvraag waar al op gereageerd is door een stokvis medewerker omzetten in een bestelling, waardoor er een nieuwe bestelling wordt aangemaakt met dezelfde producten/aantallen.

Use case	1.11
Naam	Bestelstatus inzien
Actor	Gebruiker
Samenvatting	Als gebruiker wil ik de bestelstatus van een order kunnen inzien
Omschrijving	Als gebruiker wil ik als ik de autorisatie heb de details van een order kunnen zien, waaronder de bestelstatus.

3.3 Back-end contactmanager

Use case	2.1
Naam	Overzicht van gebruikers opvragen
Actor	Contactmanager
Samenvatting	Als contactmanager wil ik een overzicht kunnen opvragen van alle gebruikers die aandacht nodig hebben
Omschrijving	De stokvis medewerker wilt een overzicht zien van alle nieuw geregistreerde gebruikers, zodat hij deze per stuk kan gaan koppelen aan het ERP systeem (use case 3.5) of weigeren (use case 3.6). Ook moet de medewerker de gebruikers zien die hun gegevens willen wijzigen(use case 1.4) of extra autoriteiten hebben aangevraagd(use case 1.6).



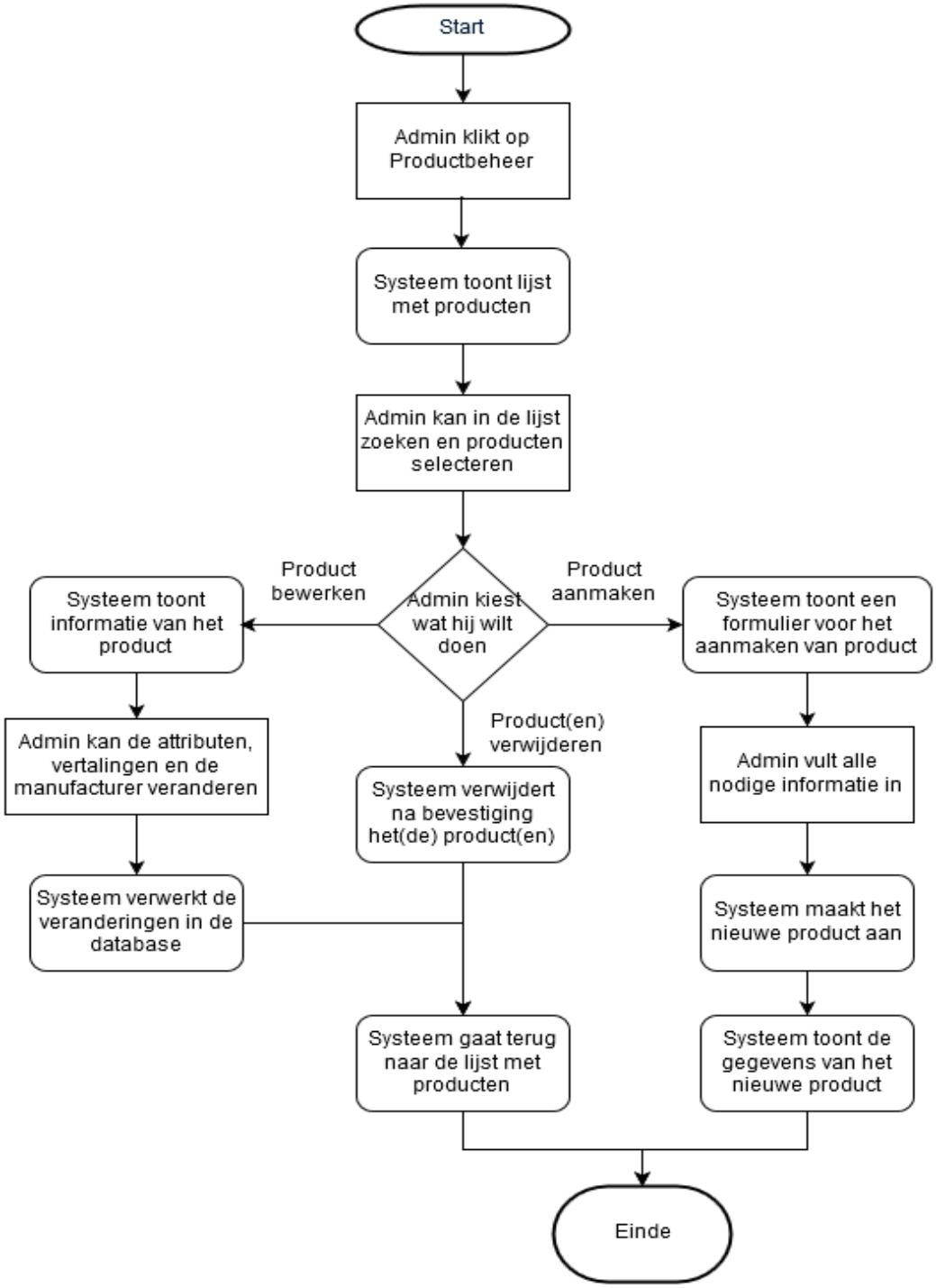
Use case	2.2
Naam	Nieuwe gebruikers koppelen aan CRM
Actor	Contactmanager
Samenvatting	Als contactmanager wil ik nieuwe gebruikers kunnen koppelen aan ons CRM systeem
Omschrijving	De stokvis medewerker kan de gegevens zien van een nieuwe registratie. Hij voert de gegevens in bij het CRM systeem, waaronder een WebuserID. Als de gebruiker al in het CRM-systeem staat, verandert hij juist het WebuserID in het CMS. Dit WebuserID wordt later gebruikt door de website om de gegevens en autorisaties van de gebruiker op te halen. Als de medewerker klaar is, wordt een mail gestuurd naar de gebruiker om te laten weten dat de registratie helemaal klaar is.

Use case	2.3
Naam	Nieuwe gebruikers weigeren
Actor	Contactmanager
Samenvatting	Als contactmanager wil ik nieuwe gebruikers kunnen weigeren
Omschrijving	Als de gebruiker geen toegang moet krijgen tot de website, moet de medewerker de gebruiker kunnen weigeren. De gebruiker moet wel bewaard blijven, maar hij wordt niet gekoppeld aan het ERP en krijgt geen rechten.

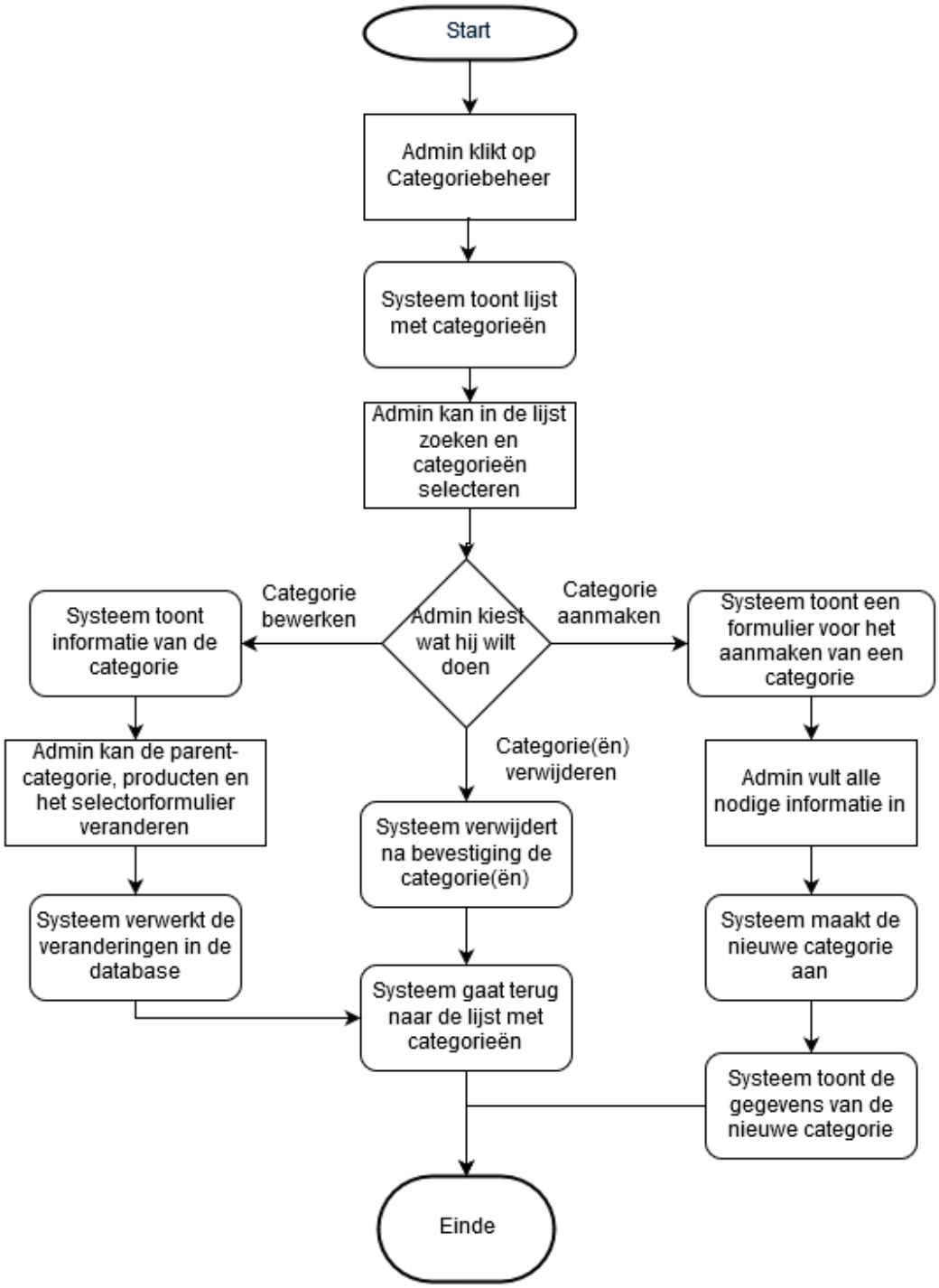
Use case	2.4
Naam	Gegevens van gebruiker wijzigen
Actor	Contactmanager
Samenvatting	Als contactmanager wil ik gegevens van een gebruiker kunnen aanpassen
Omschrijving	Als de medewerker de gegevens van een gebruiker wilt veranderen, bijvoorbeeld omdat de gebruiker zijn gegevens wilde wijzigen of extra autoriteiten heeft aangevraagd, moet hij dit kunnen doen. De medewerker zal eerst naar de pagina van de gebruiker gaan, door ofwel het overzicht van de wijzigingaanvragen of door te zoeken op WebuserID (misschien ook andere dingen?). Hierna ziet hij zowel de huidige gegevens als de (indien aanwezig) aangevraagde veranderingen.

3.4 Back-end admin

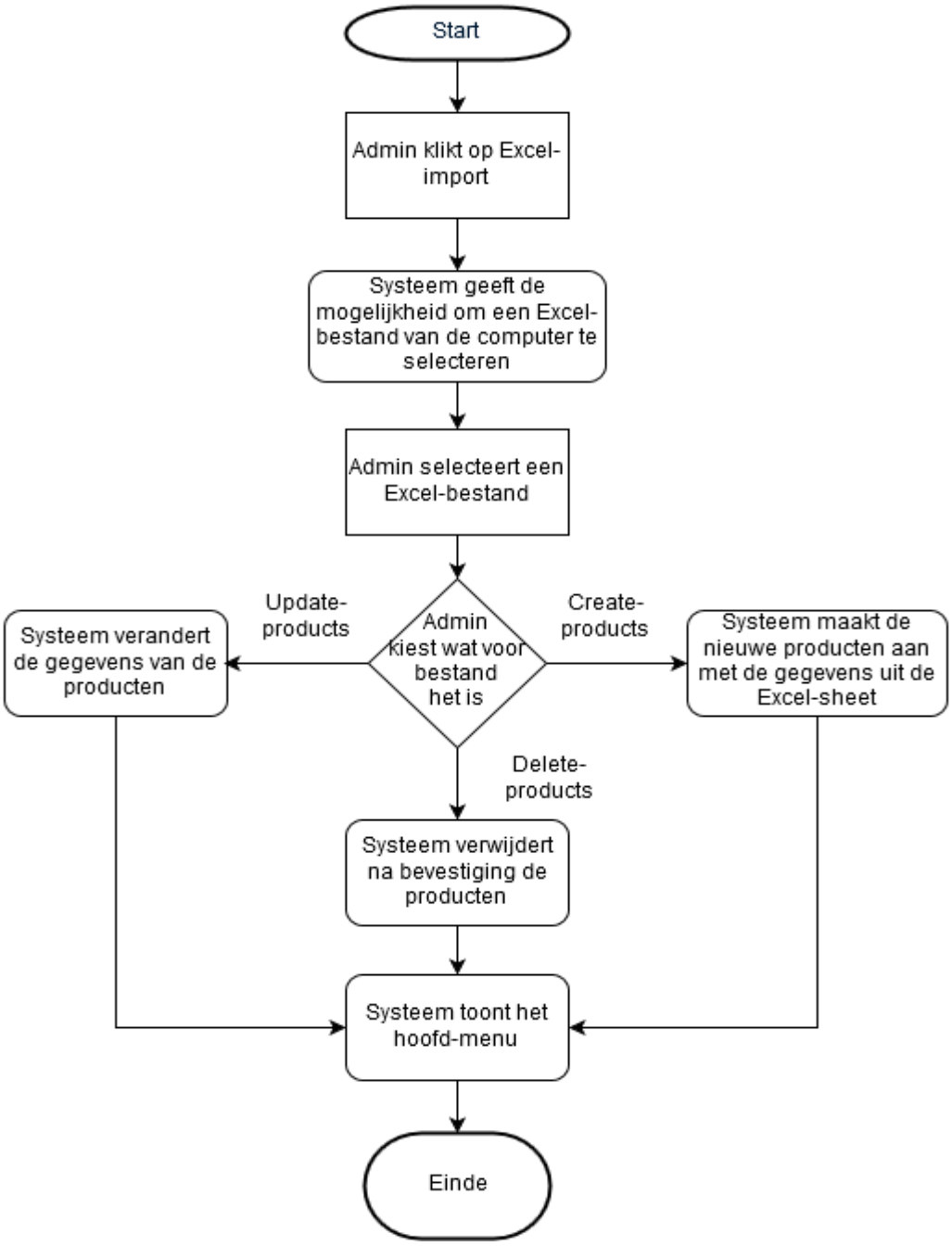
Use case	3.1
Naam	Producten beheren
Actor	Admin
Samenvatting	Als admin wil ik een product kunnen beheren
Omschrijving	Als admin wil ik producten kunnen toevoegen, wijzigen en verwijderen. Hierbij moet ik de attributen, alternatieve producten, beschrijvingen in meerdere talen en de manufacturer kunnen wijzigen. Productbeheer kan ook via Excel-imports (use case 3.5) om makkelijk grote hoeveelheden producten te beheren.
Preconditie	Admin is ingelogd



Use case	3.2
Naam	Productcategorieën beheren
Actor	Admin
Samenvatting	Als admin wil ik een productcategorie kunnen beheren
Omschrijving	Als admin wil ik categorieën kunnen toevoegen, wijzigen en verwijderen. Dit betekent dat ik onder andere producten aan de categorieën kan toewijzen, een parentcategorie kan toewijzen en een beschrijving van de categorie kan schrijven. Het toewijzen van de producten moet naast het toevoegen met artikelnummer ook kunnen door alle producten met bepaalde attributen te selecteren. Er kunnen verschillende typen categorieën zijn (semantische productboom, merken, branches, modules/oplossingen). Nadat de admin de categorie heeft geselecteerd, kan hij de attributen zien die alle producten van de categorieën hebben. Deze kan hij selecteren en in volgorde zetten om het selectorformulier voor de categorie te maken. Per attribuut kan ook gekozen worden hoe de gebruiker een keuze kan maken uit het attribuut (multicheckbox/checkbox/dropdown). Het systeem geeft een standaardkeuze gebaseerd op het aantal mogelijke values voor dat attribuut. True/false mogelijk: checkbox(boolean), 2-4 mogelijke values: multicheckbox, 5+ values: dropdown. Dit selectorformulier wordt op de website getoond aan de gebruiker en hiermee kan de gebruiker de producten doorzoeken. De categorieën moeten beschikbaar zijn voor alle websites om te selecteren om producten te tonen.


Start
Admin klikt op Categoriebeheer
Systeem toont lijst met categorieën
Admin kan in de lijst zoeken en categorieën selecteren
Admin kiest wat hij wilt doen
Categorie bewerken
Categorie aanmaken
Categorie(ën) verwijderen
Systeem toont informatie van de categorie
Admin kan de parent-categorie, producten en het selectorformulier veranderen
Systeem verwerkt de veranderingen in de database
Systeem verwijdert na bevestiging de categorie(ën)
Systeem gaat terug naar de lijst met categorieën
Systeem toont een formulier voor het aanmaken van een categorie
Admin vult alle nodige informatie in
Systeem maakt de nieuwe categorie aan
Systeem toont de gegevens van de nieuwe categorie
Einde

Use case	3.3
Naam	Excel-bestand importeren
Actor	Admin
Samenvatting	Als admin wil ik een Excel-bestand met producteninformatie kunnen importeren
Omschrijving	De meeste productinformatie wordt beschikbaar gemaakt met Excel-bestanden. Als admin wil ik een Excel-bestand met de nodige informatie van producten kunnen importeren om de producten beschikbaar te maken in het CMS. Er moeten drie verschillende soorten Excel-bestanden geïmporteerd kunnen worden: • Nieuwe producten in het CMS zetten • Bestaande producten wijzigen • Bestaande producten verwijderen uit het CMS



Use case	3.4
Naam	Productencaches resetten
Actor	Admin
Samenvatting	Als admin wil ik productencaches kunnen resetten
Omschrijving	Indien er gebruik gemaakt wordt van een cache om producten in op te slaan, zodat er minder vaak calls naar de database hoeven te worden gedaan, moeten deze kunnen worden gereset. Hierdoor moet er weer een call naar de database worden gedaan, zodat de laatste informatie getoond wordt op de website(s).

Use case	3.5
Naam	Productcatalogus beheren
Actor	Admin
Samenvatting	Als admin wil ik een productcatalogus kunnen beheren
Omschrijving	Als admin wil ik een productcatalogus kunnen samenstellen en in pdf-formaat beschikbaar maken voor een website. Een catalogus wordt aan één of meerdere categorie(ën) gekoppeld en kan op meerdere websites worden gebruikt.

Use case	3.6
Naam	Attributen beheren
Actor	Admin
Samenvatting	Als admin wil ik attributen kunnen beheren
Omschrijving	De attributen van producten moeten beheerd kunnen worden. Zo moeten in ieder geval de naam, de vertalingen en de unit veranderd kunnen worden van het attribuut.

3.5 Back-end author product

Use case	4.1
Naam	Productpagina's aanmaken en aanpassen
Actor	Author product
Samenvatting	Als product author wil ik productpagina's kunnen aanmaken en aanpassen
Omschrijving	De product author moet productpagina's kunnen aanmaken en wijzigen. Alle wijzigingen moeten net als bij normale pagina's eerst worden goedgekeurd door een admin voordat ze live gaan. De product author kan alleen teksten en foto's op de pagina wijzigen. Een lijst met producten met behulp van een selectie op een pagina zetten kan alleen door een admin gedaan worden. Als een admin deze use case uitvoert, moet hij dus wel de selectie kunnen wijzigen. De admin kan ook kiezen of hij van de selectie de lijst met producten op de pagina wilt zetten of een lijst met alle downloads. Zie hoofdstuk 4 voor een overzicht van de pagina's/wireframes die gemaakt moeten kunnen worden.

3.6 YaPDM

Use case	5.1
Naam	YaPDM files beheren
Actor	YaPDM
Samenvatting	Als YaPDM wil ik files kunnen beheren
Omschrijving	Het YaPDM file-beheersysteem moet API-calls kunnen doen naar het systeem om filepaths van bestanden te kunnen geven. De volgende calls moeten worden geïmplementeerd: • addFile

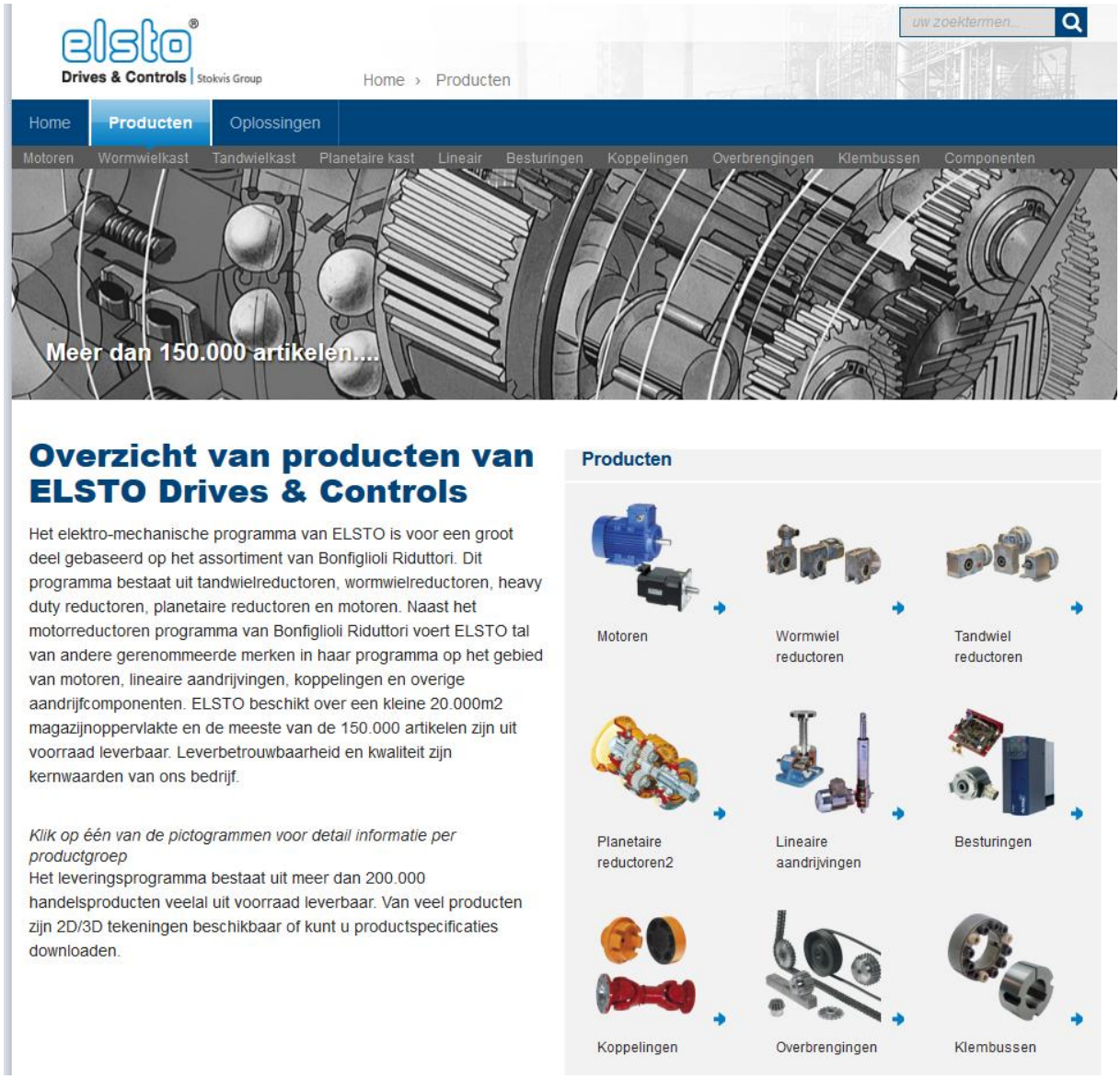
- deleteFile
- updateFile

4. Productpagina's

Voor producten zijn een aantal verschillende pagina's nodig. Hieronder staan ze op een rijtje met een screenshot voor de duidelijkheid:

4.1 Hoofdpagina

Deze pagina wordt net als andere contentpagina's gemaakt, maar wel met een ander type pagina. Eventueel moet er naar een mogelijkheid gekeken worden om plaatjes van productcategorieën te gebruiken in plaats van deze apart te moeten importeren in het normale contentpagina-systeem.

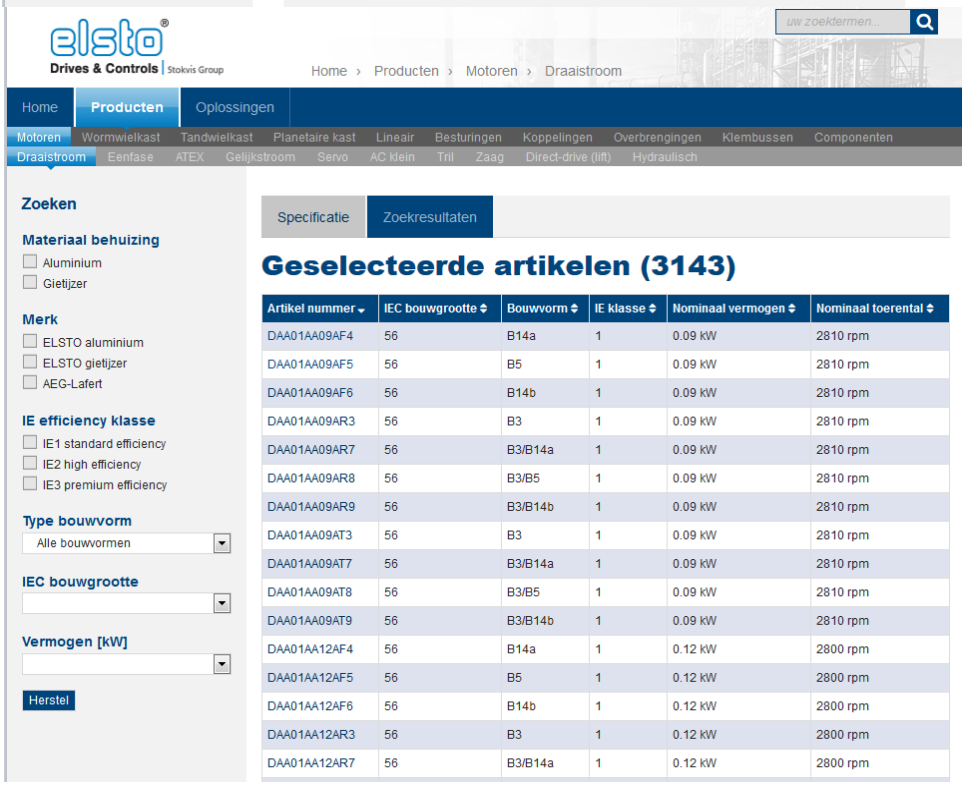


4.2 Categorieënpagina

Deze pagina is een subpagina van de hoofdpagina of van een andere categorieënpagina en toont verschillende (sub)categorieën. Ook deze pagina wordt net als een normale contentpagina gemaakt. Weer geldt dat er misschien een manier gevonden moet worden om afbeeldingen van daadwerkelijke productcategorieën te gebruiken.

4.3 Selectorpagina

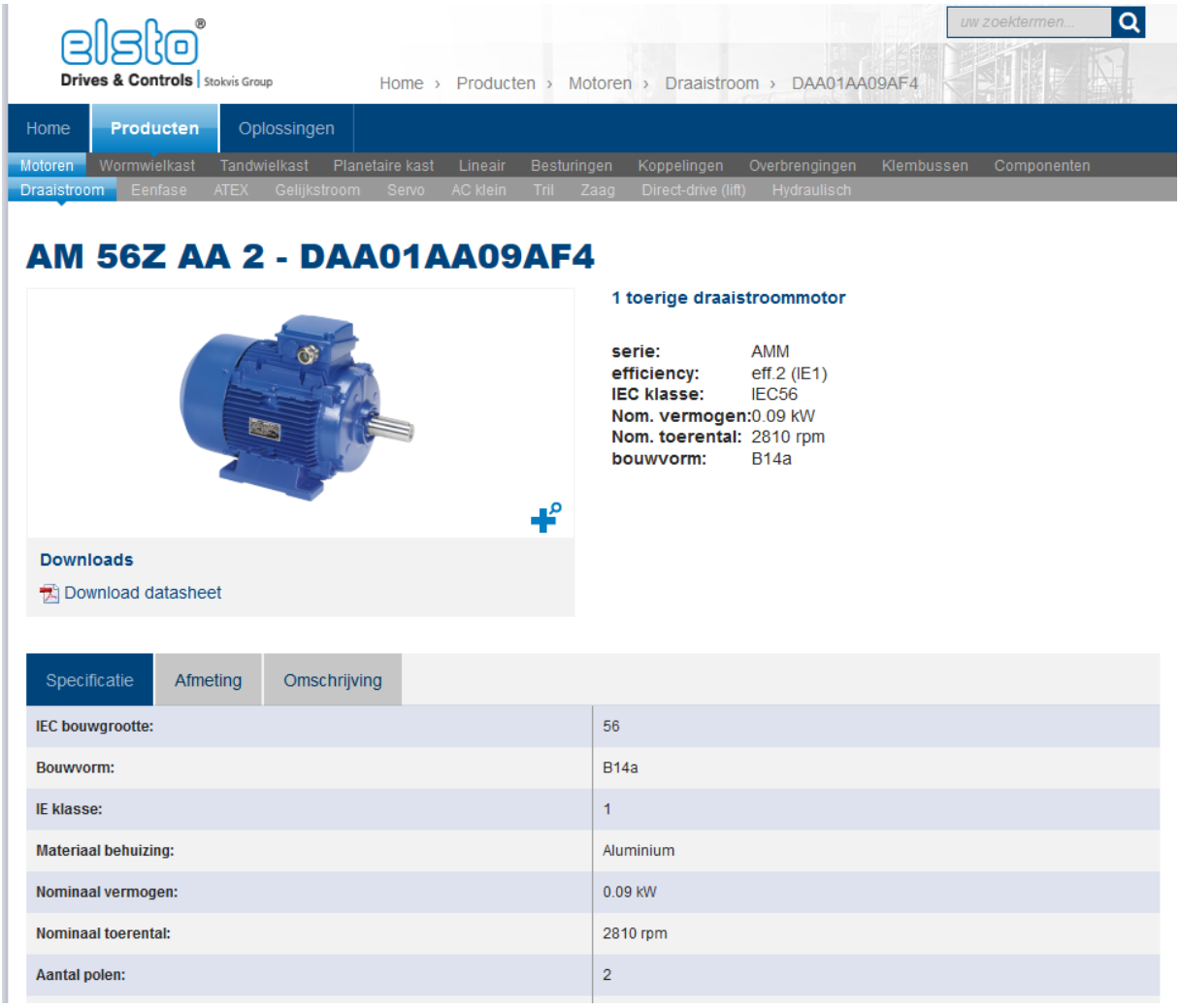
Dit is een nieuw soort pagina. Bij deze pagina moet een productselectie geselecteerd zijn. Een product author kan de tekst bij de specificatie aanpassen, maar verder niks. De rest (de selector waarmee gezocht kan worden, het plaatje van de categorie, de downloads van de categorie en de producten zelf) wordt allemaal automatisch uit de selectie gehaald. Er is een admin nodig om de selectie op de pagina te zetten of te wijzigen.



Artikel nummer	IEC bouwgrootte	Bouwvorm	IE klasse	Nominaal vermogen	Nominaal toerental
DAA01AA09AF4	56	B14a	1	0.09 kW	2810 rpm
DAA01AA09AF5	56	B5	1	0.09 kW	2810 rpm
DAA01AA09AF6	56	B14b	1	0.09 kW	2810 rpm
DAA01AA09AR3	56	B3	1	0.09 kW	2810 rpm
DAA01AA09AR7	56	B3/B14a	1	0.09 kW	2810 rpm
DAA01AA09AR8	56	B3/B5	1	0.09 kW	2810 rpm
DAA01AA09AR9	56	B3/B14b	1	0.09 kW	2810 rpm
DAA01AA09AT3	56	B3	1	0.09 kW	2810 rpm
DAA01AA09AT7	56	B3/B14a	1	0.09 kW	2810 rpm
DAA01AA09AT8	56	B3/B5	1	0.09 kW	2810 rpm
DAA01AA09AT9	56	B3/B14b	1	0.09 kW	2810 rpm
DAA01AA12AF4	56	B14a	1	0.12 kW	2800 rpm
DAA01AA12AF5	56	B5	1	0.12 kW	2800 rpm
DAA01AA12AF6	56	B14b	1	0.12 kW	2800 rpm
DAA01AA12AR3	56	B3	1	0.12 kW	2800 rpm
DAA01AA12AR7	56	B3/B14a	1	0.12 kW	2800 rpm

4.4 Productpagina

Deze pagina toont de informatie van het product. Deze wordt automatisch gebouwd door het CMS en kan niet in het CMS veranderd worden. Bij het tabblad Afmeting wordt de maattekening getoond en alle maten van het product. Bij het tabblad Omschrijving wordt de beschrijving van het product getoond. Als iemand is ingelogd en de juiste rechten heeft voor het bepaalde product moet hij ook de prijs, voorraad en downloads kunnen zien. Later zal ook een knop nodig zijn om het product toe te voegen aan een offerte/bestelling.



Onderin het scherm worden ook alternatieve producten getoond.

Bijlage 3: Technisch Ontwerp

Technisch Ontwerp

Producten in het CMS

Stokvis Group

Versiebeheer

Versie	Datum	Auteur	Opmerkingen
0.1	10-6-2016	Jonathan Karssen	Initiële versie met alleen het ERD
0.2	14-7-2016	Jonathan Karssen	Uitgebreid met requirements en mogelijke architecturen
0.3	21-7-2016	Jonathan Karssen	Gekozen architectuur toegevoegd

Inhoudsopgave

1. Requirements

1.1 Functional requirements

Het systeem moet voldoen aan alle eisen die gesteld zijn in het Functioneel Ontwerp.

1.2 Non-functional requirements

De belangrijkste requirements zijn hieronder opgenomen met het bijbehorende kwaliteitsattribuut wat van belang is bij de eis. Deze requirements gelden voor de productie-omgeving als er getest wordt door SalesCare.

Nr	Requirement	Kwaliteitsattribuut
1.1	Het laden van een pagina zonder producten moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	Time behaviour
1.2	Het ophalen van alle producten van een categorie moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 15 seconden duren.	Time behaviour
1.3	Het ophalen van de informatie van één product moet in 95% van de gevallen binnen 2 seconden gebeuren. In de overige 5% mag het maximaal 4 seconden duren.	Time behaviour
1.4	Het genereren en ophalen van een productdatasheet moet altijd binnen 10 seconden gebeuren.	Time behaviour
1.5	Het systeem moet minimaal 35 users op de frontend met een piekbelasting van 70 users tegelijkertijd aan kunnen zonder performanceverlies.	Capacity
2.1	De producteninformatie moet ook toegankelijk zijn voor andere systemen dan het CMS.	Modularity
2.2	Een programmeur van het CMS moet binnen twee uur de oorzaak van een fout in het systeem kunnen opsporen.	Analysability
2.3	Een nieuwe release moet binnen twee uur gevalideerd kunnen worden.	Testability
3.1	Als er een fout zit in pagina's met producten moeten de pagina's zonder producten nog steeds bereikbaar zijn.	Fault Tolerance
3.2	Als het ERP niet bereikbaar is, moet de andere productinformatie wel zonder vertraging getoond worden.	Fault Tolerance
4.1	Gevoelige data mag niet bereikbaar zijn vanaf de websites.	Confidentiality

1.3 Constraints

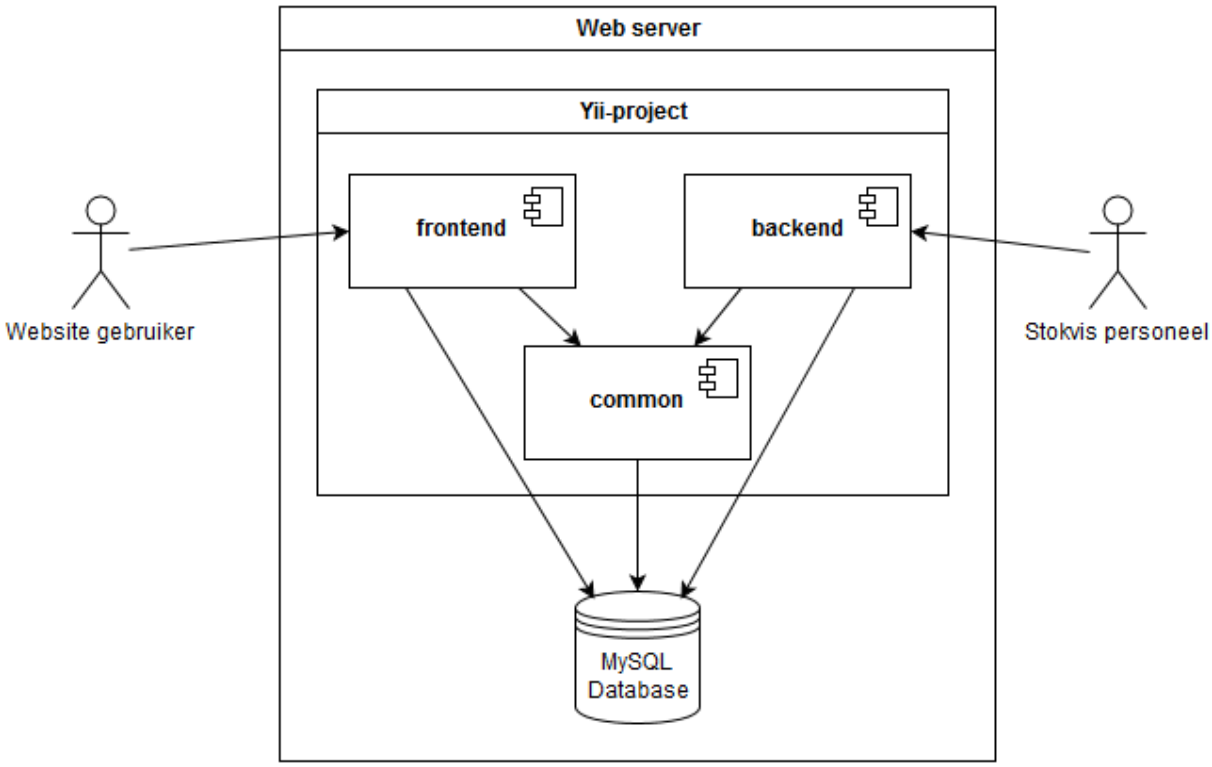
Hieronder volgen de constraints van het systeem.

Nr	Constraint
1	Het CMS moet gebouwd worden in PHP met behulp van het framework Yii2.
2	Het CMS moet MySQL als database management systeem gebruiken.
3	Het productenbeheer en productenpagina's moeten klaar zijn voor 31 oktober 2016.

2. Architectuur

2.1 Huidige architectuur

De huidige architectuur (zie figuur 2.1) is de standaard architectuur van het Yii2 advanced template. Het Yii2-project bestaat uit drie applicaties: frontend, backend en common. Al deze applicaties gebruiken het ModelViewController-principe. De common-applicatie wordt echter alleen gebruikt door andere applicaties om gemeenschappelijke modellen en functionaliteiten in op te slaan en wordt niet gebruikt door users buiten het systeem.

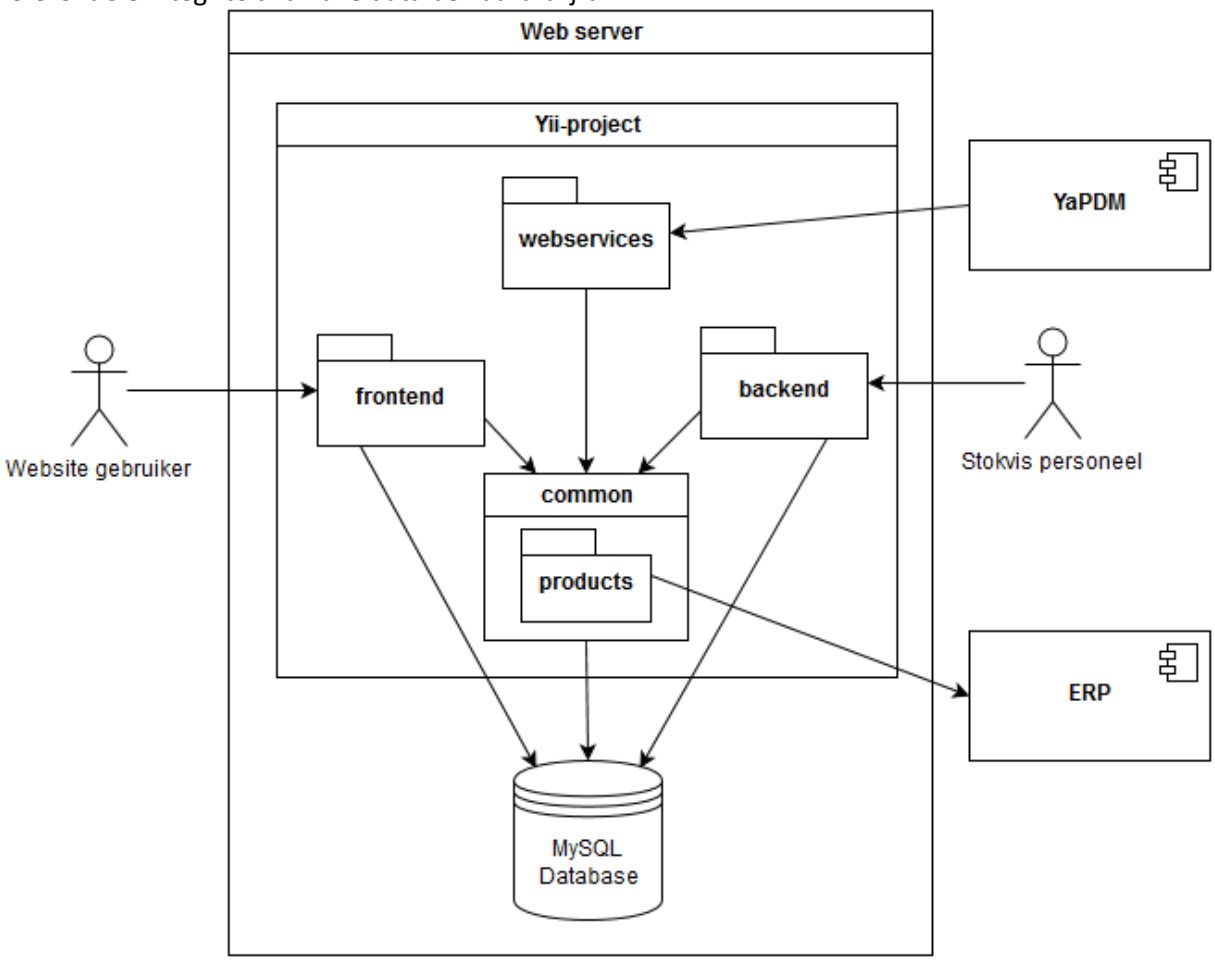


Figuur 2.1: Huidige architectuur

2.2 Mogelijke keuzes

2.2.1 Eén project met één database

De makkelijkste variant is de nieuwe functionaliteiten in het bestaande project toe te voegen. Er wordt een module products toegevoegd in de common-applicatie om de productenfunctionaliteiten te groeperen en de overzichtelijkheid te behouden in het project. Mocht het nodig zijn kunnen er ook productmodules in de frontend- en backendapplicaties toegevoegd worden. Er is een aparte webservice-applicatie toegevoegd voor YaPDM, aangezien Yii2 zelf aanraadt om een aparte applicatie toe te voegen voor webservices. Alleen de module products zal communiceren met het ERP om prijs- en voorraadgegevens op te halen.
De productentabellen staan in dezelfde database als de bestaande tabellen. Dit zorgt ervoor dat de referentiële integriteit van alle data bewaard blijft.

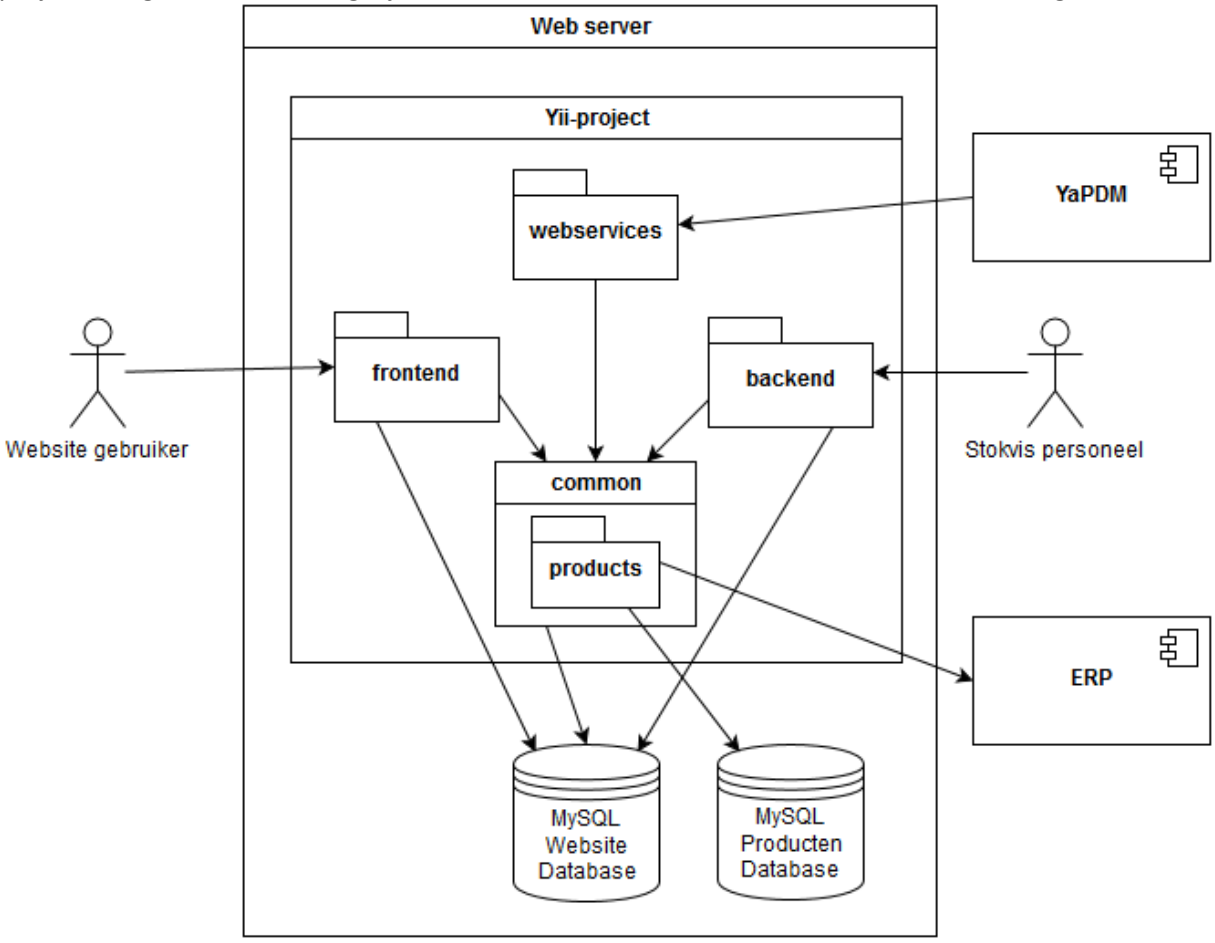


Hieronder volgen de plus- en minpunten van deze architectuur:

+	Makkelijke integratie in het bestaande project (door middel van Yii2-submodules)
+	Geen webservices nodig om producten op te halen
+	Referentiële integriteit in de database
-	Mogelijk onoverzichtelijk (zowel het project als de database)
-	Grootte van de database
-	Niet makkelijk mogelijk om productenbeheer los te koppelen

2.2.2 Eén project met twee databases

Deze variant lijkt erg veel op de eerste oplossing. Er wordt echter een tweede database gebruikt om de informatie van de producten in op te slaan. De applicatie zal zelf de referentiële integriteit moeten bewaren door ook de andere database te checken als er bepaalde wijzigingen in een database optreden. Verder levert het gebruik van twee databases geen problemen op voor het Yii-project, aangezien Yii zelf mogelijkheden biedt om meerdere databases te definiëren en gebruiken.



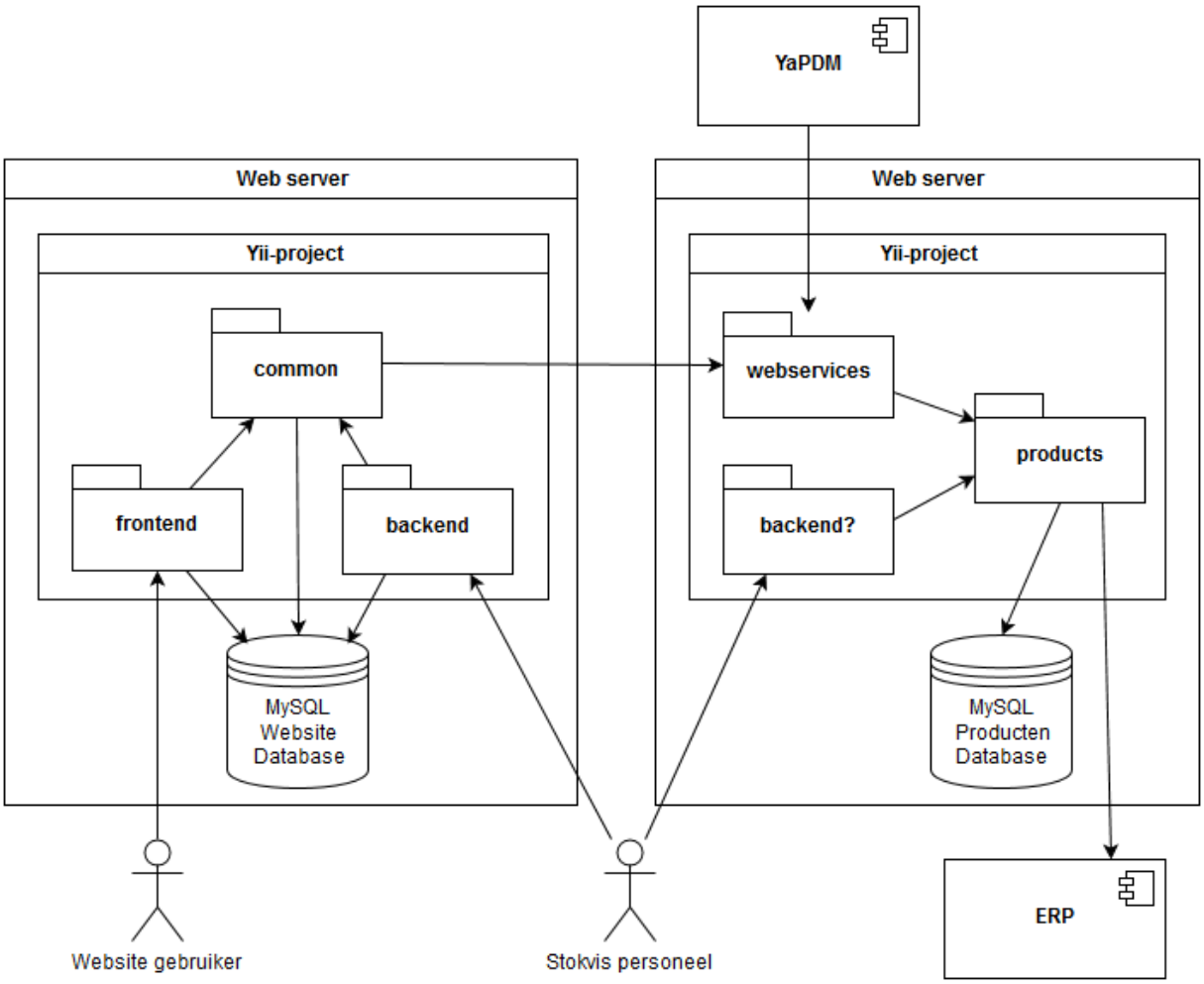
Hieronder volgen de plus- en minpunten van deze architectuur:

+	Makkelijke integratie in het bestaande project (door middel van Yii2-submodules)
+	Geen webservices nodig om producten op te halen
+	Mogelijkheid om de databases op aparte servers te zetten
+	Aparte back-ups per database mogelijk
+	Productendatabase kan eventueel aan andere systemen gekoppeld worden
-	Mogelijk onoverzichtelijk project
-	Meer werk om de refentiële integriteit te bewaren in de applicatie
-	Niet makkelijk mogelijk om productenbeheer los te koppelen

2.2.3 Twee projecten met twee databases

De andere mogelijkheid is om het productenbeheer in een geheel apart systeem te maken. Uiteraard zal een groot deel van de functionaliteiten sowieso in het bestaande project moeten worden gemaakt, zoals alle acties die een front-end gebruiker moet kunnen uitvoeren. Het productenbeheer, het maken van de selecties en de API die beschikbaar moet zijn voor YaPDM zou echter in een apart project kunnen worden gemaakt. Dit zou dan echter wel ook bereikbaar moeten zijn voor het normale CMS-project.

Eventueel zou de backend voor het productbeheer wel in het bestaande Yii-project gemaakt kunnen worden. Dit zou betekenen dat wijzigingen aan de producten ook doorgevoerd worden met behulp van webservices. Hierdoor zouden er geen twee omgevingen nodig zijn voor de Stokvis Group medewerkers. Dit zou echter weer goed beveiligd moeten worden, zodat er geen onbevoegde wijzigingen in de producten kunnen worden doorgegeven via de webservices. Dit zou waarschijnlijk ook betekenen dat er maar één Yii-package nodig is waarin zowel de webservices als de logica in kan staan.



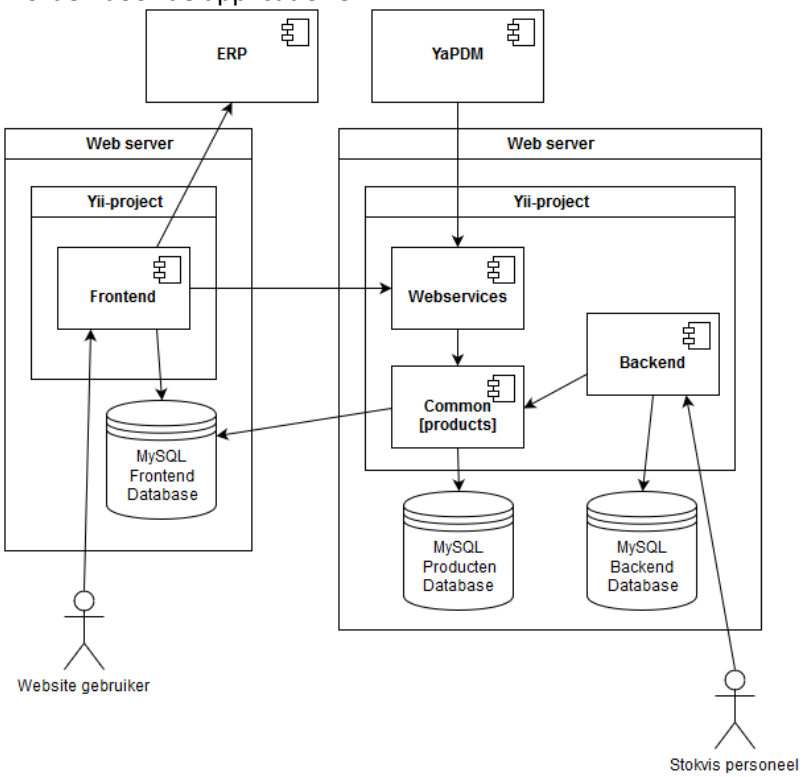
Hieronder volgen de plus- en minpunten van deze architectuur:

+	Duidelijke scheiding tussen productenCMS en websiteCMS
+	Bij mogelijke fouten in het productenCMS is het makkelijk uit te schakelen, terwijl de rest van de website nog wel werkt
+	Mogelijkheid om de systemen op aparte servers te zetten
+	Aparte back-ups mogelijk

+	Productendatabase kan eventueel aan andere systemen gekoppeld worden
-	Webservices nodig om producten op te halen, kan laadtijd vergroten
-	Aparte authenticatie nodig
-	Meer werk aangezien er webservices gebouwd moeten worden ipv calls naar classes
-	Meer werk om de refentiële integriteit te bewaren in de applicatie
-	Kan onduidelijkheid opleveren welke functionaliteit in welk systeem hoort

2.3 Gemaakte keuzes

Uiteindelijk is gekozen voor een variant op de het architectuurmodel met twee servers en twee databases. Er is gekozen om een extra database op te nemen voor alle producten om goed te voldoen aan requirement 2.1. Door een aparte database op te nemen, is de data ook makkelijker beschikbaar te maken voor andere systemen zonder de informatie van het websitebeheer ook bloot te geven. Het grootste nadeel, de referentiële integriteit die verbroken wordt, kan opgevangen worden door de applicatie zelf.

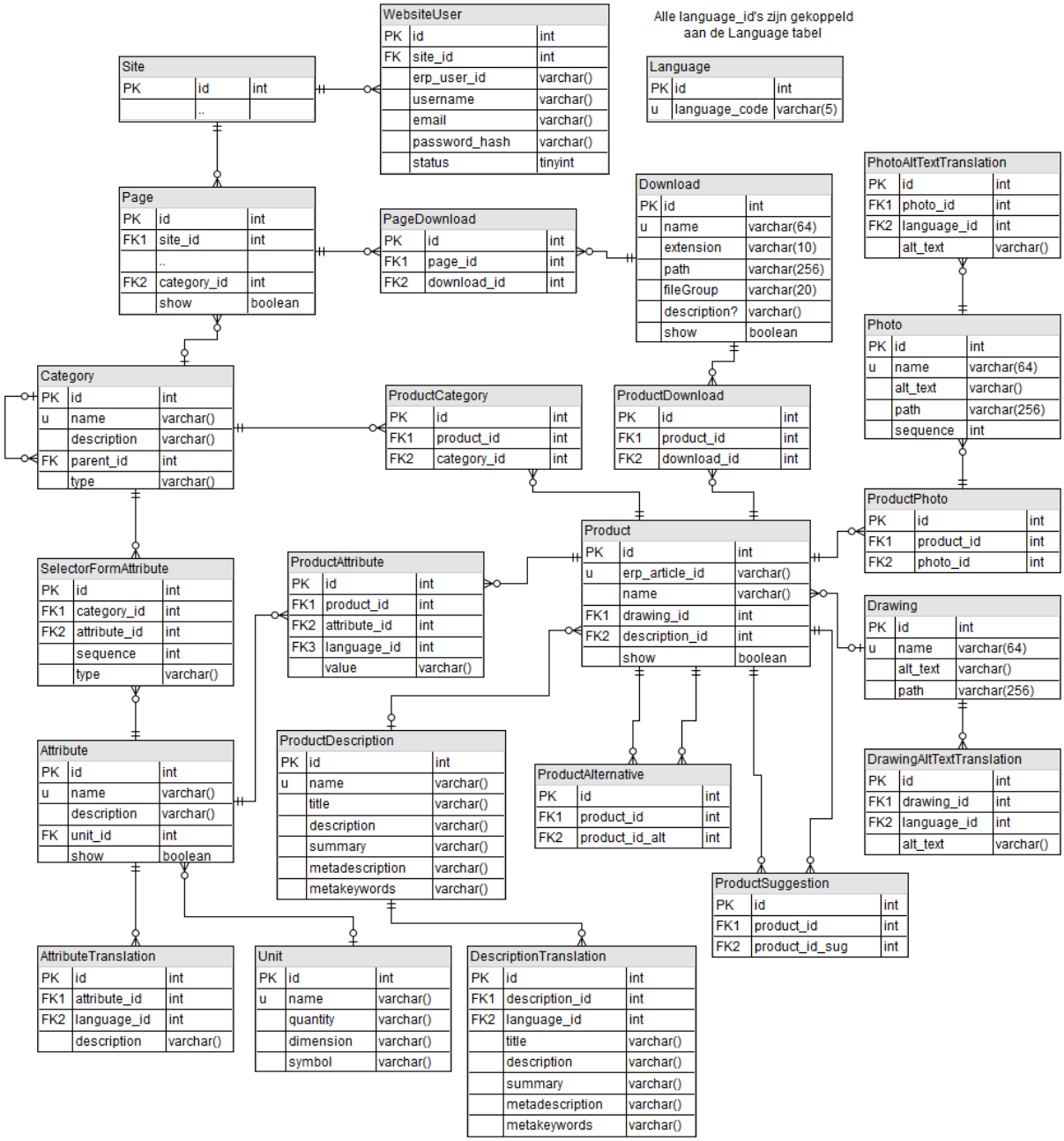


Verder wordt er een module products toegevoegd aan de common-applicatie. Hierin komt de DAO van de producten, zodat zowel het productenbeheer in de backend-applicatie als de webservices bij de producteninformatie in de database kunnen. De webservices komen in een aparte webservices-applicatie in de backend-webserver. Yii zelf adviseert om webservices in een aparte applicatie te zetten, zodat er een aantal instellingen makkelijk kunnen worden ingesteld voor alle webservices. De webservices worden geïmplementeerd als een REST API. Deze keus is vooral gemaakt vanwege de goede support van Yii voor REST webservices en vanwege de relatief simpele requests die verstuurd zullen worden naar de API. Ook zal de API op het begin alleen door twee vertrouwde partijen, YaPDM en de frontend, gebruikt worden, waardoor er geen ingewikkelde validatie nodig is.

Naast de communicatie vanaf YaPDM, zal het CMS ook moeten communiceren met het ERP voor gebruikersgegevens en up-to-date productenprijzen. Hiervoor is een API geïmplementeerd in het ERP, wat de frontend-applicatie kan aanroepen.

3. Databasemodel

3.1 ERD



3.2 Toelichting

Bovenin het ERD staan een aantal klassen die al in het bestaande ERD staan, namelijk Site, Page en Language. De overige tabellen zijn allemaal nieuw. Hieronder volgt waar nodig een toelichting per tabel of per groep tabellen.

3.2.1 Attribute

Er zijn over alle producten verspreid ongeveer 250 verschillende attributen, waarvan de meeste maar voor verschillende delen van de producten nodig zijn. Ook kunnen de waardes en namen van attributen verschillen per taal. Hierdoor is besloten om de attributen van producten in een aparte tabel op te slaan. De waardes van een attribuut worden per product opgeslagen in een koppeltabel

met eventueel een language als de waarde maar voor één taal geldt. Vertalingen van de attribuutnamen worden opgeslagen in een aparte tabel. Sommige attributen hebben ook een Unit, zoals bijvoorbeeld centimeter voor lengtes.

3.2.2 Category

Category is eigenlijk niks anders dan een verzameling producten. Een pagina kan een categorie hebben om de producten van die categorie op de website te tonen. Aangezien de frontend moet weten op welke attributen er makkelijk gezocht kan worden is er een koppeltabel toegevoegd tussen Category en Attribute: SelectorFormAttribute. In deze tabel wordt ook een type opgeslagen hoe de frontend-gebruiker dat attribuut ziet (bijvoorbeeld checkbox/dropdown) en de volgorde van de attributen.

3.2.3 ProductDescription

De ProductDescription-tabel is toegevoegd om de beschrijvingen van producten in op te slaan, aangezien de beschrijving vaak hetzelfde is voor enkele tientallen producten en er vertalingen van deze beschrijvingen opgeslagen moesten worden. Deze beschrijvingen worden voornamelijk gebruikt op de dynamische productpagina's.

3.2.4 Download, Drawing en Photo

Alle bestanden die bij producten horen, worden bewaard in het YaPDM-systeem. De database slaat alleen verwijzingen naar de bestanden op. Deze verwijzingen staan in de tabellen Download, Drawing en Photo afhankelijk van wat voor soort bestand het is. Drawing is de maatschets van het product, wat altijd op de pagina van een product te bekijken moet zijn. Photo slaat de foto's op van het product en Download bevat alle andere soorten bestanden. De normale downloads moeten ook op pagina's zonder producten gezet kunnen worden, waarvoor de koppeltabel tussen Download en Page is opgenomen.
Er had gekozen kunnen worden om alle bestanden in één tabel te zetten. Het is zo echter makkelijker om de beschrijvingen en alternatieve teksten van de foto's te beheren. Ook is zo waarschijnlijk het zoeken naar een bepaalde maatschets of bepaalde foto's een stuk sneller.

3.2.5 WebsiteUser

De frontend-gebruikers worden opgeslagen in WebsiteUser. De status van de gebruiker geeft aan of de gebruiker al goedgekeurd is. De meeste gegevens (naam, adres etc) worden normaal gesproken alleen opgeslagen in het ERP, zodat er één centrale plek is voor het klantenbeheer. Voordat de gegevens in het ERP komen of als de klant zijn gegevens wilt wijzigen, zullen de gegevens toch tijdelijk opgeslagen moeten worden in de database. Hiervoor is de extra tabel WebsiteUserChange opgenomen in het ERD. Wanneer de wijziging doorgevoerd is in het ERP door een medewerker van Stokvis verdwijnt de data weer uit ons systeem.

Hieronder volgt de tabel met alle ERP-data die opgeslagen wordt:

firstName
lastName
Bedrijfsnaam
Functie
Geboortedatum
Telefoonnummer
Faxnummer
Adres1?
Adres2?

Bijlage 4:Testrapport

Tests

Er is gekozen om alleen loadtests uit te voeren.
Om alle requirements te testen zijn drie verschillende scenario's opgesteld.

1) Iemand kijkt naar pagina's zonder producten en klikt door verschillende pagina's heen.
2) Iemand kijkt naar een categoriepagina, zoekt naar een specifiek product en gaat uiteindelijk naar die productenpagina.
3) Iemand gaat naar een productenpagina en download een datasheet.

Deze scenario's zijn zowel los als gezamenlijk in een loadtest uitgevoerd. Gezamenlijk betekent dat 33% van de gebruikers scenario 1 doen, 33% scenario 2 en 33% scenario 3. Alle tests zijn uitgevoerd met een duur van 15 minuten, waarbij in de eerste 10 minuten het aantal gebruikers langzaam groeit van 1 naar het totaal aantal gebruikers. De scenario's zijn twee keer uitgevoerd. Eén keer met een maximum van 35 gebruikers en één keer met een maximum van 70 gebruikers.

Scenario 1

Hieronder volgt het 'User Profile' van scenario 1. Dit zijn alle URL's die de loadtest aanroept per gebruiker bij dit scenario.

http://ebdtechniek.nl/js/all.js
http://ebdtechniek.nl/fonts/icomoon.ttf?8zrw9e
http://ebdtechniek.nl/uploads/media/EBD/2016/08/f7cc9123526af1214f74ffd895814c88.png
http://ebdtechniek.nl/css/site/css
http://ebdtechniek.nl/contact
http://ebdtechniek.nl/uploads/media/EBD/2016/08/c5eb858c1ad136d446e1f141b3d07e80.png
http://ebdtechniek.nl/uploads/media/EBD/2016/08/df7886f6875cf646c4015113ddce70b6.gif
http://ebdtechniek.nl/uploads/media//EBD/2016/08/medium/61d874e1a24fa1a76c717786fb7426e6.png
Think time
http://ebdtechniek.nl/contact
Think time
http://ebdtechniek.nl/producten/liften/huisliften
http://ebdtechniek.nl/uploads/media/EBD/2016/08/e1c47cd55c33d461dbcdb8bca95750f9.png
http://ebdtechniek.nl/uploads/media/EBD/2016/08/413f7bbc9f9dbd41c64bfcc30e3abd96.png
http://ebdtechniek.nl/uploads/media/EBD/2016/08/b885a6b5b92d68521c5e70d46c97db36.png

Tabel 1: User Profile scenario 1

Scenario 2

User Profile

http://producten-test.nl/tandwiel
http://producten-test.nl/css/site.css
http://producten-test.nl/js/all.js
http://producten-test.nl/js/Product.js
http://producten-test.nl/uploads/media/tes/2016/08/ad101d477aa7c40d0efa4774cce6a979.gif
http://producten-test.nl/tandwiel?limit=10
http://producten-test.nl/fonts/icomoon.ttf?8zrw9e
http://producten-test.nl/uploads/media/tes/2016/08/87d149af0e3c62e7fae5a506ba86c997.png
Think time
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F
Think time

http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24
Think time
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24&offset=20
Think time
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24&offset=20&sort=ASC&sortBy=101
Think time
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24&offset=30&sort=ASC&sortBy=101

Tabel 2: User Profile scenario 2

Scenario 3

User Profile

http://producten-test.nl/favicon.ico
http://producten-test.nl/tandwiel?product=S01010014HS0P
Think time
http://producten-test.nl/tandwiel?product=S01010014HS0P&datasheet

Tabel 3: User Profile scenario 3

Gezamenlijk scenario

In het gezamenlijke scenario worden alle drie de bovenstaande scenario's tegelijk uitgevoerd in plaats van maar één scenario.

Resultaten

Scenario 1

Iemand kijkt naar pagina's zonder producten en klikt door verschillende pagina's heen.

Url	Average response time 35 gebruikers (ms)	Average response time 70 gebruikers (ms)
http://ebdtechniek.nl/js/all.js	983	1721
http://ebdtechniek.nl/fonts/icomoon.ttf?8zrw9e	277	437
http://ebdtechniek.nl/uploads/media/EBD/2016/08/f7cc9123526af1214f74ffd895814c88.png	3844	8349
http://ebdtechniek.nl/css/site/css	422	682
http://ebdtechniek.nl/contact	368	501
http://ebdtechniek.nl/uploads/media/EBD/2016/08/c5eb858c1ad136d446e1f141b3d07e80.png	3249	7293
http://ebdtechniek.nl/uploads/media/EBD/2016/08/df7886f6875cf646c4015113ddce70b6.gif	264	378
http://ebdtechniek.nl/uploads/media//EBD/2016/08/medium/61d874e1a24fa1a76c717786fb7426e6.png	1878	3604
http://ebdtechniek.nl/	373	502
http://ebdtechniek.nl/producten/liften/huisliften	425	533
http://ebdtechniek.nl/uploads/media/EBD/2016/08/e1c47cd55c33d461dbcdb8bca95750f9.png	292	483

http://ebdtechniek.nl/uploads/media/EBD/2016/08/413f7bbc9f9dbd41c64bfcc30e3abd96.png	3071	6693
http://ebdtechniek.nl/uploads/media/EBD/2016/08/b885a6b5b92d68521c5e70d46c97db36.png	3084	6453

Tabel 4: Resultaten scenario 1

Aangezien de gemiddelde response tijd zo veel hoger was bij afbeeldingen, is vervolgens ook een test uitgevoerd zonder de afbeeldingen om te kijken of het probleem echt bij de afbeeldingen ligt. Deze resultaten staan hieronder.

Url	Average response time 35 gebruikers (ms)	Average response time 70 gebruikers (ms)
http://ebdtechniek.nl/js/all.js	134	233
http://ebdtechniek.nl/fonts/icomoon.ttf?8zrw9e	94	140
http://ebdtechniek.nl/css/site/css	48	89
http://ebdtechniek.nl/contact	169	287
http://ebdtechniek.nl/	141	236
http://ebdtechniek.nl/producten/liften/huisliften	207	308

Tabel 5: Resultaten scenario 1 zonder afbeeldingen

Scenario 2

Iemand kijkt naar een categoriepagina, zoekt naar een specifiek product en gaat uiteindelijk naar die productenpagina.

Url	Average response time 35 gebruikers (ms)	Average response time 70 gebruikers (ms)
http://producten-test.nl/uploads/media/tes/2016/08/ad101d477aa7c40d0efa4774cce6a979.gif	44	74
http://producten-test.nl/tandwiel	127	173
http://producten-test.nl/js/Product.js	50	102
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F	127	177
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20 diameter=24&offset=20&sort=ASC&sortBy=101	130	188
http://producten-test.nl/css/site.css	91	124
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20 diameter=24	133	177
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20 diameter=24&offset=30&sort=ASC&sortBy=101	125	180
http://producten-test.nl/fonts/icomoon.ttf?8zrw9e	52	81

http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20 diameter=24&offset=20	131	176
http://producten-test.nl/uploads/media/tes/2016/08/87d149af0e3c62e7fae5a506ba86c997.png	53	72
http://producten-test.nl/js/all.js	211	264
http://producten-test.nl/tandwiel?limit=10	123	175

Tabel 6: Resultaten scenario 2

Scenario 3

Iemand gaat naar een productenpagina en download een datasheet.

Url	Average response time 35 gebruikers (ms)	Average response time 70 gebruikers (ms)
http://producten-test.nl/favicon.ico	50	61
http://producten-test.nl/tandwiel?product=S01010014HS0P	200	224
http://producten-test.nl/tandwiel?product=S01010014HS0P&datasheet	371	395

Tabel 7: Resultaten scenario 3

Gezamenlijk scenario

In dit scenario voert 33% scenario 1 uit, 33% scenario 2 en 33% scenario 3. Het scenario is drie keer uitgevoerd met 70 gebruikers.

Url	Average response time 35 gebruikers (ms)	Average response time 70 gebruikers run 1 (ms)	Average response time 70 gebruikers run 2 (ms)	Average response time 70 gebruikers run 3 (ms)
Peak average response time	777	737	1569	1634
http://producten-test.nl/fonts/icomoon.ttf?8zrw9e	76	133	194	156
http://ebdtechniek.nl/fonts/icomoon.ttf?8zrw9e	88	115	411	319
http://producten-test.nl/js/Product.js	89	146	208	195
http://ebdtechniek.nl/contact	188	249	617	697
http://ebdtechniek.nl/uploads/media/EBD/2016/08/c5eb858c1ad136d446e1f141b3d07e80.png	829	959	1372	1095
http://producten-test.nl/tandwiel?product=S01010014HS0P	253	301	527	502
http://ebdtechniek.nl/uploads/media/EBD/2016/08/413f7bbc9f9dbd41c64bfcc30e3abd96.png	737	929	1019	826
http://producten-test.nl/tandwiel?product=S01010014HS0P&datasheet	447	503	950	851

http://ebdtechniek.nl/uploads/media/EBD/2016/08/df7886f6875cf646c4015113ddce70b6.gif	78	109	218	144
http://ebdtechniek.nl/js/all.js	295	342	364	366
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F	181	249	529	476
http://producten-test.nl/uploads/media/tes/2016/08/ad101d477aa7c40d0efa4774cce6a979.gif	68	130	157	111
http://ebdtechniek.nl/uploads/media//EBD/2016/08/medium/61d874e1a24fa1a76c717786fb7426e6.png	459	557	821	667
http://producten-test.nl/favicon.ico	87	128	344	274
http://ebdtechniek.nl/uploads/media//EBD/2016/08/medium/d43a63daf534557576fc7e4c354f5c59.png	567	681	962	993
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24&offset=20&sort=ASC&sortBy=101	165	272	601	547
http://ebdtechniek.nl/producten/tilhulpen/serverliften	221	297	737	828
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24&offset=20	149	195	542	455
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24&offset=30&sort=ASC&sortBy=101	164	202	755	718
http://ebdtechniek.nl/uploads/media/EBD/2016/08/b885a6b5b92d68521c5e70d46c97db36.png	856	946	1129	920
http://producten-test.nl/css/site.css	137	221	227	300
http://producten-test.nl/uploads/media/tes/2016/08/87d149af0e3c62e7fae5a506ba86c997.png	96	145	201	177
http://ebdtechniek.nl/css/site.css	125	156	395	340
http://ebdtechniek.nl/uploads/media/EBD/2016/08/f7cc9123526af1214f74ffd895814c88.png	1039	1139	1396	1109
http://producten-test.nl/js/all.js	298	378	379	380
http://producten-test.nl/tandwiel?limit=10&Reductor%20uitvoering=F&Uitgaande%20as%20diameter=24	156	240	787	923
http://producten-test.nl/tandwiel?limit=10	151	207	378	298
http://ebdtechniek.nl/	182	248	537	465
http://ebdtechniek.nl/producten/liften/huisliften	214	296	650	601

http://ebdtechniek.nl/uploads/media/EBD/2016/08/e1c47cd55c33d461dbcdb8bca95750f9.png	111	105	185	233
http://producten-test.nl/tandwiel	157	246	373	402

Tabel 8: Resultaten scenario gezamenlijk

Conclusie

Bij de resultaten kwam naar voren dat de grootste bottleneck qua performance op het moment ligt bij de afbeeldingen van de normale pagina's. Dit komt waarschijnlijk omdat het dezelfde afbeeldingen waren die nu heel vaak werden opgehaald. Toen dezelfde hoeveelheid gebruikers over drie verschillende websites met afbeeldingen werden verdeeld, was de hoogste gemiddelde laadtijd namelijk maar 2098 ms in plaats van 7410 ms. De verwachting is dus ook dat dit geen problemen zal opleveren in productie, aangezien de bezoekers over alle websites verdeeld zullen zijn.

Scenario	Testresultaat (35 gebruikers)	Testresultaat (70 gebruikers)
1	2801 ms (181 ms*)	7410 ms (349 ms*)
2	146 ms	246 ms
3	305 ms	477 ms
Gezamenlijk run 1	777 ms	737 ms
Gezamenlijk run 2	-	1569 ms
Gezamenlijk run 3	-	1634 ms

Loadtest resultaten (hoogste gemiddelde response tijd op een tijdstip)