Eindverslag

Content management systeem

Versie 1.5

Hogeschool	Saxion hogeschool Deventer
Opleiding	HBO-ICT
Auteur	Idris Oncul (427041)
Begeleider	Dick Heijink
Opdrachtgever	Stefan Arts
Kwartiel	4.4 + 5.1 +5.2
Datum	20-01-2020
Plaats	Witteveen en Bos

VOORWOORD

Voor u ligt het afstudeerverslag ‘Contentmanagementsysteem’. Dit verslag heb ik geschreven tijdens mijn afstudeerperiode voor de opleiding HBO-ICT Software engineering aan de Hogeschool Saxion te Deventer, in opdracht van Witteveen en Bos. Van mei 2019-februari 2020 heb ik me beziggehouden met het onderzoek, de ontwikkeling en het schrijven van het eindverslag.

Samen met mijn afstudeerbegeleider, Leon Roeterdink, heb ik het probleem van Witteveen en Bos in kaart gebracht. Na uitvoering van mijn onderzoek heb ik de deelvragen beantwoord, waardoor ik de hoofdvraag kon beantwoorden. Door het antwoord op de hoofdvraag was het gelijk duidelijk wat Witteveen en Bos zocht en kon ik beginnen met de ontwikkelingsfase.

Bij dezen wil ik mijn afstudeerbegeleider bedanken voor zijn goede hulp gedurende de periode. Ook wil ik alle collega’s van mijn groep GEO en Data Application bedanken voor de fijne samenwerking. Daarnaast wil ik Dick Heijink danken. Zonder zijn begeleiding en de gesprekken had ik dit eindverslag waarschijnlijk niet voltooid.

Ik wens u veel leesplezier toe.

Idris Öncül

Deventer, 14 januari 2020

Versiebeheer

Versie	Auteurs	Datum	Opmerking
0.1	Idris Öncül	20-11-2016	Opzetten sjabloon.
0.2	Idris Öncül	10-07-2019	Invoegen onderdelen Plan van aanpak
0.3	Idris Öncül	11-07-2019	Invoegen onderdelen introductie Invoegen onderdelen onderzoek
0.4	Idris Öncül	07-08-2019	Uitbereiding Onderzoek Invoegen onderdelen realisatie
0.5	Idris Öncül	18-09-2019	Reviewpunten Dick heijnink Reviewpunten Leon Roeterdink
0.6	Idris Öncül	20-09-2019	Uitbereiding onderzoek Uitbereiding realisatie Invoegen onderdeel aanbevelingen
0.7	Idris Öncül	3-10-2019	Spelfouten en grammatica controle
0.8	Idris Öncül	11-10-2019	Reviewpunten Leon Roeterdink
0.9	Idris Öncül	11-10-2019	Invoegen bijlage Api endpoints
1.0	Idris Öncül	11-10-2019	Opmaak voor tussentijdse inlevering
1.1	Idris Öncül	22-10-2019 / 27-10-2019	Reviewpunten Dick Heijnink Reviewpunten Willem Prakken
1.2	Idris Öncül	1-11-2019 / 05-01-2020	Verslag uitbereid.
1.3	Idris Oncul	13-01-2020/ 15-01-2020	Reviewpunten Dick Heijnink Reviewpunten Leon Roeterdink
1.4	Berthijn(scribbs)	14-01-2020	Spelling, grammatica controle
1.5	Idris Oncul	14-01-2020/ 20-01-2020	Reviewpunten Berthijn verwerkt.

Inhoudsopgaven

1. INLEIDING

Witteveen en Bos schrijft veel rapportages tijdens het gehele ontwerptraject van grote infrastructurele projecten, zoals milieueffectrapporten (MER's) en ontwerpstudies. De informatie in deze rapportages komt vanuit verschillende disciplines, waardoor de verslagen lastig te begrijpen zijn voor personen zonder enige vorm van kennis van de betreffende discipline.

Vanwege de digitalisering en toenemende vraag om transparantie vanuit de samenleving worden delen van het rapport gepubliceerd door gebruik te maken van webapplicaties. De moeilijk te begrijpen complexe informatie wordt aangevuld met beelden, zoals afbeeldingen, grafieken en kaartmateriaal, zodat de gebruiker meer inzicht krijgt in de geplande wijzigingen.

Momenteel wordt dit soort webapplicaties gemaakt in opdracht van specialisten, waarbij de tekstuele content wordt aangeleverd in een .CSV-bestand. Aan de hand van deze tekstuele content wordt een webapplicatie ontwikkeld.

Deze werkwijze is echter om meerdere redenen niet optimaal. Ten eerste kost het de specialisten tijd om de .CSV-bestanden op te stellen. Hierdoor kan foutieve of verouderde tekstuele content aangeleverd worden.

Ten tweede kan deze content bij foutief aangeleverde documenten niet door de specialist aanpast worden. De aanpassingen worden doorgegeven aan een ontwikkelaar, die het vervolgens kan publiceren.

Ten derde zijn geen standaarden afgesproken. Hierdoor kunnen bijvoorbeeld de databasemodellen voor een gelijksoortige webapplicatie verschillen. Dit betekent dat een ontwikkelaar soms moet zoeken naar welke gegevens in welke tabel worden opgeslagen.

Al deze werkzaamheden kunnen vereenvoudigd worden, zodat de ontwikkelaars de resterende tijd kunnen gebruiken om andere toepassingen te ontwikkelen en de specialisten hun tijd bijvoorbeeld kunnen besteden aan het opstellen van een andere rapportage.

2. HET BEDRIJF

Witteveen en Bos is een ingenieurs- en adviesbureau dat complexe vraagstukken oplost en de maatschappij vooruithelpt. Grote opgaven waaraan Witteveen en Bos kan bijdragen, zijn bijvoorbeeld energietransitie, klimaatadaptatie, overstromingsproblematiek, gezonde steden, circulaire economie en grootschalige vervanging van infrastructuur. Witteveen en Bos adviseert en helpt opdrachtgevers de juiste keuzes te maken.

Bij deze internationale organisatie werken ruim 1100 ingenieurs en adviseurs in 11 landen, verdeeld over 19 kantoren. Witteveen en Bos is actief op vier werkterreinen, te weten:

- gebouwde omgeving;
- delta's, kusten en rivieren;
- energie, water en milieu;
- infrastructuur en mobiliteit.

Per jaar voert Witteveen en Bos op verschillende werkterreinen gemiddeld 3000 projecten uit. In deze projecten streeft het bureau naar een maximale waardecreatie. Dit betekent duurzame oplossingen voor de maatschappij, klantwaarde voor zijn opdrachtgevers en kansen voor talentontwikkeling voor zijn medewerkers.

Witteveen en Bos is in 1946 opgericht door Willem Gerrit Witteveen en Goosen Siger Bos om een bijdrage te leveren aan de grote maatschappelijke uitdaging van die tijd: de wederopbouw van Nederland na de Tweede Wereldoorlog.

Meer informatie over Witteveen en Bos kan gevonden worden op de webpagina: https://www.witteveenbos.com/nl/.

3. OPDRACHT

3.1 AANLEIDING

Zoals beschreven in de inleiding kan een ontwikkelaar content aanpassen. Dit was ook het geval bij de interactieve webapplicatie die gerealiseerd werd voor het MER ‘Dijkversterking IJsseldijk Zwolle-Olst’; dit rapport is beschikbaar via http://zwolle-olst.plusplatform.nl/.

Tijdens de publicatie van deze webapplicatie bleek dat de aangeleverde content verouderd was, waardoor bijna alle content aangepast moest worden.

Gezien de grote impact hiervan, waarbij veel tekstuele content aangepast moest worden, is er voor dit project voor gekozen om een backend te ontwikkelen waarbij de ingevulde tekstuele content aangepast kan worden door specialisten.

Na deze gebeurtenis vroegen de ontwikkelaars en de specialisten hoe dit in de toekomst voorkomen kan worden. Daaruit bleek dat het meerwaarde biedt om een contentmanagementsysteem te ontwikkelen waarbij de specialisten zelf content kunnen aanmaken en/of wijzigen.

Figuur 1: Interactieve webapplicatie

3.2 DEFINITIE

Zoals beschreven in de aanleiding (paragraaf 3.1) is de werkwijze zoals het nu gaat niet de efficiëntste manier van werken. Witteveen en Bos wil daarom het beheer en/of de publicatie van content overlaten aan zijn specialisten.

Het doel van de afstudeeropdracht is een applicatie te ontwikkelen waarmee de content van een interactieve webapplicatie gevuld kan worden. Daarnaast moet het mogelijk zijn rapportages gebruikersvriendelijk te schrijven, zodat deze eenvoudig kunnen worden ontsloten voor webtoepassingen. Hierdoor kan de content voor een interactieve webapplicatie snel en eenvoudig worden gevuld.

De webapplicatie moet het mogelijk maken om de inhoud (tekst) van rapportages te koppelen aan beeld. Dit wordt ontwikkeld door niet ‘documentgericht’ te werken, maar juist ‘tekstblokgericht’. Tekstblokken kunnen onder elkaar getoond worden, waardoor het voor de gebruiker lijkt dat hij of zij alsnog in één document werkt.

Aangezien de applicatie tekstblokgericht is, kunnen kaartmaterialen, afbeeldingen en thema’s gekoppeld worden aan tekstuele content. Hierdoor is het mogelijk om webpagina’s te voorzien van inhoud en beeld.

De opdracht is verdeeld in twee onderdelen. Het eerste onderdeel is het ontwerp van een datamodel waarin alle informatie van een project kan worden opgeslagen en dat gebruikt kan worden om rapporten te creëren.

Als tweede is de systematiek geïntegreerd in een contentmanagementsysteem (ook wel afgekort als CMS-systeem). Het CMS-systeem wordt gebruikt om nieuwe content toe te voegen en bestaande content aan te passen of te verwijderen.

3.3 AFBAKENING

De scope van de afstudeeropdracht beperkt zich tot het management van content om de eerdergenoemde rapporten te digitaliseren en de aanbieding van content om de verschillende rapporten te visualiseren. Visualisatie van een interactieve webapplicatie valt buiten de scope van de opdracht.

4 ONDERZOEK

4.1 HOOFD- EN DEELVRAGEN

Tijdens dit onderzoek stond de digitalisatie van het MER centraal. Daarom luidt de hoofdvraag:

> Hoe kunnen specialisten de content van een MER publiceren als een interactieve presentatie?

Uit de hoofdvraag kunnen de volgende twee deelvragen worden afgeleid:

- Welke functionaliteiten en beeldmaterialen worden gebruikt om een MER op een interactieve manier te presenteren?
- Welke open source CMS-systemen zijn beschikbaar in de programmeertalen Python en/of PHP? En kan eventueel worden uitgeweken naar andere programmeertalen?

Deelvraag 1

Om deze vraag te beantwoorden, is bureauonderzoek uitgevoerd naar al eerder ontwikkelde interactieve webapplicaties. De bestaande webapplicaties zijn beoordeeld op de verschillende soorten content. Aan de hand van de bevindingen zijn gesprekken gehouden met medeontwikkelaars en met collega’s die het kaartmateriaal samenstellen, zodat een juiste keuze kon worden gemaakt.

Deelvraag 2

Om deze vraag te beantwoorden, is bureauonderzoek verricht naar de bestaande soorten CMS-systemen. Het onderzoek was gericht op het bepalen van de werking van CMS-systemen en toepasbaarheid van het product. De CMS-systemen zijn gefilterd op programmeertaal, uitstraling, moeilijkheidsgraad van uitbreidingen en GitHub-ranking.

4.2 WELKE FUNCTIONALITEITEN EN BRONNEN WORDEN GEBRUIKT OM EEN INTERACTIEVE PRESENTATIE TE PUBLICEREN?

Om een goed onderzoek te verrichten, is het noodzakelijk in kaart te brengen wie de gebruikers van de applicaties worden. Daarnaast moet duidelijk worden welke functionaliteiten noodzakelijk zijn, zodat tijdens het onderzoek het juiste CMS-systeem kan worden onderzocht.

4.2.1 WERKWIJZE

Om het onderzoek in goede banen te leiden, zijn eerst de al eerder ontwikkelde interactieve webapplicaties bekeken.

Aan de hand van deze bevindingen heeft de onderzoeker gesproken met Stefan Arts en Pieter-Bas de Visser. Deze gesprekken zijn gehouden om te controleren of de applicatie alle benodigde onderdelen bevat en deze waar nodig aan te vullen, zodat belangrijke onderdelen niet vergeten werden.

4.2.2 INTERVIEW UITWERKING

Stefan Arts is aardwetenschapper van de groep GEO en Data Application. Het gesprek met de heer Arts had voornamelijk betrekking op de ontwikkeling van kaartmaterialen. Aangezien de heer Arts een groot bijdrage heeft geleverd aan eerder ontwikkelde interactieve webapplicaties, kon hij de stappen van het begin tot eind toelichten. De heer Arts legde hoofdzakelijk uit hoe kaartmaterialen tot stand komen. Hierbij moet worden gedacht aan de ontwikkeling en publicatie van kaartmaterialen.

Pieter-Bas van Visser is afdelingshoofd van de groep BIM Vormgeving en simulatie infra. Hij houdt zich voornamelijk bezig met het ontwerp van 3D-modellen en plattegronden. De heer van Visser heeft daarnaast veel kennis van het opstellen van een notitie reikwijdte en detailniveau (NRD) en een MER. Het gesprek met de heer van Visser richtte zich voornamelijk op het verloop van het ontwikkelingsproces van deze rapporten. Hierdoor wist de onderzoeker welke disciplines aan deze rapporten werken.

4.2.3 RESULTAAT VAN ONDERZOEKSFUNCTIONALITEITEN

4.2.3.1 REQUIREMENTS

Op basis van de gesprekken en de op een andere manier verzamelde informatie zijn uiteindelijk veertien ‘requirements’ geselecteerd. Deze requirements zijn samen met de afstudeerbegeleider geprioriteerd volgens de MoSCoW-methode.

R.nr	Requirement	Prioriteit
R.01	De actor moet gebruik kunnen maken van het systeem	Should
R.02	De actor kan een nieuwe pagina aanmaken	Must
R.03	De actor kan een basispagina samenstellen	Must
R.04	De actor kan een verkenningsrapport samenstellen	Must
R.05	De actor kan een kaartvariant samenstellen	Must
R.06	De actor kan een nieuw rapport aanmaken	Must
R.07	De actor kan een rapport typen	Must
R.08	De actor kan een rapportbron toevoegen	Must
R.09	De actor kan een afbeelding toevoegen	Must
R.10	De actor kan html-tekst toevoegen	Must
R.11	De actor kan een thema selecteren	Should
R.12	De actor kan zich met zijn bestaande Witteveen en Bos-account registeren	Could
R.13	De actor kan versiebeheer toepassen	Could
R.14	De tweede lezer kan geschreven tekst publiceren	Could

4.3 WAT IS EEN CONTENTMANAGEMENTSYSTEEM?

Een CMS-systeem is een softwaretoepassing, meestal een webapplicatie, die het mogelijk maakt voor gebruikers om zonder veel technische kennis, documenten of gegevens op internet te publiceren.

CMS-systemen zijn ontwikkeld om content te beheren, op te slaan en te publiceren. Hierdoor is het voor kleine bedrijven eenvoudiger geworden om complexe sites te beheren en de informatie up-to-date te houden.

Volgens BuiltWith is 65% van de websites op internet momenteel opgebouwd in een CMS-systeem.[2] Tegenwoordig zijn er twee soorten CMS-systemen:

- traditioneel CMS-systeem;
- headless CMS-systeem.

4.3.1 TRADITIONELE CONTENTMANAGEMENTSYSTEEM

Traditionele CMS-systemen bestaan al een aantal jaren. Zij maken het mogelijk om zonder enige programmeerervaring gegevens te publiceren en deze op een goede manier te presenteren op de website. Gebruikers kunnen door middel van drag-and-drop (slepen en neerzetten) eenvoudig een website opzetten.

4.3.2 HEADLESS CONTENTMANAGEMENTSYSTEEM

De headless-structuur is deels een antwoord op de manier hoe webcontent zich heeft ontwikkeld. Lange tijd werd webcontent alleen geleverd aan een browser, maar tegenwoordig wordt deze content aan steeds meer platforms geleverd, zoals smartphones en IOT-apparaten.

In tegenstelling tot bij het traditionele CMS-systeem is er bij een headless CMS-systeem geen website, maar wordt de content gepresenteerd in bijvoorbeeld JSON-format. Hierdoor is het mogelijk niet één platform te voorzien van de juiste gegevens, maar meerdere platforms te koppelen aan gegevens die zijn ingevoerd in het headless CMS-systeem.

4.4 ONDERZOEK

4.4.1 ONDERZOCHTE CONTENTMANAGEMENTSYSTEMEN

Tijdens het onderzoek naar bestaande CMS-systemen bleek er veel van deze systemen bestaan; de meeste zijn traditioneel. Aangezien niet alle CMS-systemen kunnen worden geanalyseerd, is ervoor gekozen het onderzoek te beperken tot de vijf populairste open source CMS-systemen.

De populariteit van een CMS-systeem is bepaald met bureauonderzoek naar de informatie die op internet te vinden is, de GitHub-beoordelingen en het aantal keer dat een ‘repository’ ‘geforkt’ is.

De onderzochte CMS-systemen zijn:

- WordPress
- Drupal
- Django CMS
- Wagtail
- Strapi

4.4.1.1 WORDPRESS

WordPress is een van de bekendste CMS-systemen. Het heeft een gebruikersvriendelijke backend en er bestaan meer dan 100.000 plug-ins die geïnstalleerd kunnen worden. De taal waar WordPress op is gebaseerd, is PHP. WordPress wordt voornamelijk gebruikt door webloggers en webshops.[3]

WordPress is op GitHub beoordeeld met 12.533 sterren en is 7598 keer geforkt.[4]

Figuur 2: WordPress

DRUPAL

Drupal is een CMS-systeem dat is gebaseerd op de programmeertaal PHP. Drupal wordt, net als WordPress, vaak gebruikt door webloggers. Vergeleken met WordPress is Drupal iets uitgebreider. Drupal is sterk in het verdelen van rollen aan gebruikers. Hierdoor kan een gebruiker van bijvoorbeeld Financiën alleen bij de documenten die horen bij de categorie ‘Financiën’.[5]

Drupal is op GitHub beoordeeld met 3267 sterren en is 1680 keer geforkt.[6]

Figuur 3: Drupal

4.4.1.2 DJANGO CMS

Django CMS is vergelijkbaar met WordPress, maar is gebaseerd op een andere programmeertaal, namelijk Python. Een van de nuttigste functionaliteiten is de mogelijkheid de content aan de voorkant aan te passen. Zo kan tekst worden gewijzigd terwijl de website kan worden bekeken door dubbel te klikken op het element. Het adminpaneel hoeft hiervoor niet te worden geopend. Django CMS wordt gebruikt door veel grote bedrijven, zoals National Geographic, L’Oréal en NASA.[7]

Django CMS is op GitHub beoordeeld met 6822 sterren en is 2298 keer geforkt.[8]

Figuur 4:Django CMS

4.4.1.3 WAGTAIL

Figuur 5: Wagtail

Wagtail is, net als Django CMS, een open source CMS-systeem dat is gebaseerd op de programmeertaal Python en dat gebruikmaakt van het Django-framework. De userinterface (UI) van Wagtail is overzichtelijker dan die van Django CMS. Wagtail heeft een grotere community dan Django CMS, waardoor het sneller ontwikkeld wordt dan Django CMS. Een sterke kant van Wagtail is voornamelijk de rich-texteditor. Hierdoor kan een webpagina op een overzichtelijke manier worden ingericht met verschillende componenten.[9]

Wagtail is op GitHub beoordeeld met 7311 sterren en is 1590 keer geforkt.[10]

4.4.1.4 STRAPI

Figuur 6: Strapi

Strapi is een headless CMS-systeem dat gebaseerd is op het JavaScript-framework Node.js. Strapi richt zich niet op het creëren van een frontend, zoals de eerdergenoemde CMS-systemen, maar de content wordt aangeleverd door middel van een application programming interface(API). Strapi is een van de betere headless CMS-systemen en heeft een grote community. Strapi maakt, net als WordPress, gebruik van plug-ins.[11]

Strapi is op GitHub beoordeeld met 14.809 sterren en is 1686 keer geforkt.[12]

4.5 ONDERZOEKSOPZET

4.5.1 WERKWIJZE

Zoals eerder genoemd, is in dit onderzoek voornamelijk gebruikgemaakt van bureauonderzoek om de verschillende CMS-systemen te analyseren.

Het onderzoek is onder te verdelen in twee stappen.

Eerst is onderzocht welke open source CMS-systemen er bestaan. Hiervoor is gebruikgemaakt van bureauonderzoek, waarvan de uitkomsten in een vergelijkend onderzoek zijn bekeken. Hier is voor gekozen, omdat er nog geen kennis was over bestaande CMS-systemen en over wat hiermee gecreëerd kan worden.

Daarna is voor elk CMS-systeem twee dagen vrijgehouden waarin de CMS-systemen werden geïnstalleerd en werd geprobeerd deze aan te passen naar de eisen en wensen van Witteveen en Bos. Hierbij is niet alleen beoordeeld of de systemen voldoen aan deze eisen en wensen, maar ook waar ze in het algemeen aan moet voldoen. De volgende onderdelen zijn bekeken:

- Wat is de uitstraling van het betreffende CMS-systeem?
- Is het CMS-systeem overzichtelijk?
- Is hierover informatie te vinden op internet?
- Hoe ziet de WYSIWYG-editor eruit?
- Hoe moeilijk is het om het CMS-systeem aan te passen?
- Kan het adminpaneel aangepast worden, zodat de functionaliteiten verwerkt kunnen worden?
- Kan tekstuele content bijvoorbeeld gekoppeld worden aan andere content, zoals afbeeldingen of kaartmateriaal?

4.6 RESULTATEN

In dit hoofdstuk worden de resultaten van het onderzoek besproken.

4.6.1 CONTENTMANAGEMENTSYSTEEM

Zoals eerder benoemd, zijn vijf open source CMS-systemen onderzocht. De programmeertalen van de eerste vier systemen zijn PHP en Python. Dit was een wens van de organisatie, maar geen harde eis. Daarom zijn ook andere talen bekeken. Eén CMS-systeem sprong eruit met de ontwikkelingstaal JavaScript, waarbij gebruik wordt gemaakt van het framework Node.js. Hierna staan de uitslagen van de onderzochte CMS-systemen.

4.6.1.1 WORDPRESS

WordPress is een veelgebruikt framework. De uitstraling is overzichtelijk en WordPress is zorgvuldig opgebouwd. WordPress wordt door velen gebruikt, omdat het eenvoudig is om content te publiceren en op te maken. Op internet is veel informatie te vinden over hoe een gebruiker een website kan maken.

De aanpassingen waar Witteveen en Bos het voor kan gebruiken, kunnen worden gerealiseerd door een plug-in te ontwikkelen. Na de installatie van WordPress kan de plug-in geïnstalleerd worden.

Tekstuele content kan getypt en gepresenteerd worden door gebruik te maken van pagina en/of blogs. Aan elke pagina of blog kan content, zoals kaartmateriaal, toegevoegd worden.

4.6.1.2 DRUPAL

De uitstraling van Drupal was na het werken met WordPress matig. Drupal wordt voornamelijk gebruikt om websites te genereren. Door informatie te zoeken op internet was het eenvoudiger om hiermee te werken. De documentatie op de website van Drupal is overzichtelijk en volledig. Om meerdere functionaliteiten te verkrijgen, is gebruikgemaakt van modules. Dit kunnen publieke modules zijn, waarvan een aantal beschikbaar is, maar er is ook een mogelijkheid om zelf modules te ontwikkelen. Dit is ook de manier voor Witteveen en Bos om beeldmateriaal aan content te koppelen.

4.6.1.3 DJANGO CMS

Het CMS-systeem Django CMS ziet er op het eerste gezicht niet overzichtelijk uit voor onervaren personen. Dit komt voornamelijk doordat de admin-UI is uitgebreid en niet opnieuw is geschreven.

Django CMS bestaat sinds 2007 en maakt gebruik van het Django-framework, waardoor hierover veel informatie te vinden is op internet. Django CMS ondersteunt plug-ins. Voor Witteveen en Bos is dit ook de manier om beeldmateriaal aan tekstblokken te koppelen.

4.6.1.4 WAGTAIL

Wagtail is een overzichtelijk CMS-systeem dat, net als Django CMS, ontwikkeld is met het framework Django. In tegenstelling tot Django CMS, dat het adminpaneel gebruikt van het Django-framework, is voor Wagtail een eigen admin-UI ontwikkeld. Hierdoor is er meer controle over alle elementen in de admin-UI en is Wagtail overzichtelijk en eenvoudig in gebruik. De community van Wagtail is groot en er is veel informatie te vinden op internet. Wagtail ondersteunt geen plug-ins. Mocht er uitbereidingen nodig zijn, dan kan een ‘snippet’ ontwikkeld worden. Daarnaast is de WYSIWYG-editor fraai vormgegeven en eenvoudig in gebruik.

4.6.1.5 STRAPI

Zoals eerder benoemd, is Strapi een headless CMS-systeem. De uitstraling is overzichtelijk en het systeem is eenvoudig te begrijpen. Strapi heeft een uitgebreide documentatie op zijn webpagina. Het is opgebouwd met Node.js/html/JavaScript. De meeste programmeurs kennen deze talen, waardoor wijzigingen eenvoudig zijn aan te brengen. Daarnaast kunnen plug-ins ontwikkeld worden, wat het mogelijk maakt om beeldmateriaal aan tekstuele content te koppelen.

4.6.1.6 SAMENVATTEND

Voor elke soort CMS-systeem zijn verschillende factoren bekeken. Elk onderdeel is individueel beoordeeld. De beoordelingen zijn gebaseerd op de beleving van de onderzoeker.

In Tabel 1 staan de bevindingen.

1. Wat is de uitstraling van het betreffende CMS-systeem?
2. Is het CMS-systeem overzichtelijk?
3. Is hierover informatie te vinden op internet?
4. Hoe ziet de WYSIWYG-editor eruit?
5. Hoe moeilijk is het om het CMS-systeem aan te passen?
6. Kan het adminpaneel aangepast worden zodat de functionaliteiten verwerkt kunnen worden?
7. Kan tekstuele content bijvoorbeeld gekoppeld worden aan andere content, zoals kaartmateriaal of afbeeldingen?

	WordPress	Drupal	Django CMS	Wagtail	Strapi
1.	+	+	-	+	-+
2.	+	+	-+	+	-+
3.	+	-+	+	-+	-+
4.	-+	-+	-+	+	-+
5.	-+	-	+	-	-
6.	+	+	+	-+	-
7.	-+	-+	-+	-+	-+

Tabel 1: Vergelijking

Legenda		
	-	slecht
	-+	gemiddeld
	+	goed

Zoals te zien is in Tabel 1 scoort WordPress het beste. WordPress lijkt een goed CMS-systeem te zijn waar bedrijven eenvoudig een webapplicatie mee kunnen opzetten.

Voor dit onderzoek was het van belang om te beoordelen of de requirements, waaraan het systeem moest voldoen, geïmplementeerd konden worden. Een belangrijk onderdeel van het systeem, zoals beschreven in paragraaf 3.2, is niet documentgericht maar tekstblokgericht te denken.

In het onderzoek is bekeken of het mogelijk is om met het CMS-systeem meerdere tekstblokken in één UI te realiseren, om zo het gebruikersgemak voor het onderdeel ‘rapport schrijven’ te verhogen. Dit bleek echter lastig te realiseren met de in het onderzoek geteste CMS-systemen.

4.7 CONCLUSIE

De hoofdvraag van het onderzoek luidt: ‘Hoe kunnen specialisten de content van een MER publiceren als een interactieve presentatie?’ Om deze hoofdvraag te kunnen beantwoorden, moet eerst antwoord gegeven worden op de deelvragen.

Uit het onderzoek is gebleken dat het met de onderzochte open source CMS-systemen mogelijk is om de content van een interactieve presentatie te vullen. Door veranderingen in de loop der jaren is er voor alle CMS-systemen een toepassing toegevoegd om de content op te halen door middel van een representation state transfer(RESTful) API.

Documenten moeten tekstblokgericht zijn, wat betekent dat meerdere tekstblokken aaneengesloten op één enkele pagina getoond moeten worden, zodat de gebruikers het idee hebben dat ze aan één document werken. Dit is een van de belangrijkste eisen waaraan het CMS-systeem moet voldoen. CMS-systemen zijn voornamelijk ontwikkeld om webpagina’s of blogs te schrijven. Een webpagina of blog bestaat uit een titel en een enkel tekstblok. Het is niet mogelijk om meerdere tekstblokken onder elkaar te tonen.

Dit betekent dat requirement R.07 ‘De actor kan een rapport typen’ naar verwachting niet geïmplementeerd kan worden. Doordat meerdere tekstblokken ontbreken, geldt dit ook voor requirements R.03, R.04, R.08, R.09, R.10 en R.11.

Het zou daarom beter zijn om een eigen UI te ontwikkelen voor het CMS-systeem. Door gebruik te maken van Python en specifiek van het Django-framework kunnen de functionaliteiten die Wagtail en Django CMS gebruiken, geïmplementeerd worden. Hierdoor hoeven niet alle functionaliteiten zelf geschreven te worden en hebben de ontwikkelaars alle vrijheid om onderdelen toe te voegen op elke mogelijke plaats op de pagina. Hierdoor kan ook de omvang van het project verkleind worden. Bovendien gaat op deze manier minder opslag verloren aan onderdelen van een CMS-systeem die niet gebruikt worden.

Door een eigen implementatie te ontwikkelen van de UI kan gebruikgemaakt worden van de bestaande functionaliteiten van niet alleen Django CMS, maar ook die van Wagtail. Hierdoor blijft meer tijd over voor de ontwikkeling van de eerdergenoemde requirements waaraan het systeem moet voldoen.

5 REALISATIE

5.1 AANPAK EN METHODIEKEN

Om het project in goede banen te leiden, zijn eerst de requirements opgesteld, zoals beschreven in paragraaf 4.1.1. De functionele requirements zijn verwerkt in schermontwerpen, die worden besproken in paragraaf 5.2.1.

Nadat de schermontwerpen waren goedgekeurd, zijn van alle bestaande requirements userstories gemaakt om het zo klein mogelijk te houden. Deze userstories zijn geprioriteerd volgens de MoSCoW-methode, zodat de belangrijkste functionaliteiten eerst ontwikkeld konden worden. Dit is in samenwerking met de belanghebbende geprioriteerd.

Voor de ontwikkelingsfase is gebruikgemaakt van een aangepaste variant van scrum. Hiervoor is gekozen, omdat de onderzoeker alleen aan het project werkte en er geen medescrumleden waren. Hierdoor zijn er geen ‘daily stand-up meetings’ gehouden. Wel is een werkbord aangemaakt met een product- en een sprint-backlog.

De duur van elke sprint was één week. Elke maandag aan het begin van de dag hielden de onderzoeker en de afstudeerbegeleider een sprintreview. Tijdens dit gesprek werd besproken wat de onderzoeker al had gerealiseerd, tegen welke problemen hij was aangelopen en wat hij in de aankomende sprint dacht te realiseren.

Tijdens dit project was ook een product-owner aangewezen, die meerdere malen met de onderzoeker heeft gesproken. Belangrijke ontmoetingen waren het begingesprek, de oplevering van de schermontwerpen, de tussentijdse oplevering van het project en de definitieve oplevering.

5.2 FUNCTIONEEL ONTWERP

5.2.1 USERSTORIES

De vereisten uit paragraaf 4.1.3.1 zijn onderverdeeld in userstories. Dit is overzichtelijker en minder globaal.

R.nr	Requirement	Userstory	Prioriteit
R.01	De actor moet gebruik kunnen maken van het systeem	**R.01/01:** Als gebruiker wil ik kunnen inloggen, zodat ik gebruik kan maken van het systeem	Should
R.01	De actor moet gebruik kunnen maken van het systeem	**R.01/02:** Als beheerder wil ik kunnen inloggen, zodat ik gebruik kan maken van het systeem	Should
R.01	De actor moet gebruik kunnen maken van het systeem	**R.01/03:** Als beheerder wil ik een gebruikersprofiel kunnen aanmaken, zodat de gebruiker kan inloggen	Should
R.02	De actor kan een nieuwe pagina aanmaken	**R.02/01:** Als gebruiker wil ik een basispagina aan kunnen maken	Must
R.02	De actor kan een nieuwe pagina aanmaken	**R.02/02:** Als gebruiker wil ik een verkenningsrapport aan kunnen maken	Must
R.02	De actor kan een nieuwe pagina aanmaken	**R.02/03:** Als gebruiker wil ik een kaartvariant aan kunnen maken	Must
R.03	De actor kan een basispagina samenstellen	**R.03/01:** Als gebruiker wil ik tekst die bedoeld is voor een basispagina wijzigen, zodat deze via de API getoond kan worden op een interactieve webpagina	Must
R.04	De actor kan een verkenningsrapport samenstellen	**R.04/01:** Als gebruiker wil ik tekstuele content van de verschillende rapporten samenstellen, zodat een verkenningsrapport wordt gecreëerd	Must
R.05	De actor kan een kaartvariant samenstellen	**R.05/01:** Als gebruiker wil ik tekstuele content van de verschillende rapporten samenstellen, zodat een	Must

		kaartvariant word gecreëerd	
R.06	De actor kan een nieuw rapport aanmaken	**R.06/01:** Als gebruiker wil ik een nieuw rapport creëren, zodat ik het later kan aanpassen	Must
R.07	De actor kan een rapport typen	**R.07/01:** Als gebruiker wil ik een nieuw tekstblok kunnen aanmaken, zodat ik tekst kan schrijven	Must
R.07	De actor kan een rapport typen	**R.07/02:** Als gebruiker wil ik de volgorde van de tekstblokken kunnen veranderen, zodat de tekstblokken op volgorde staan	Must
R.07	De actor kan een rapport typen	**R.07/03:** Als gebruiker wil ik een titel aan een bestaand tekstblok kunnen geven, zodat het wordt opgeslagen	Must
R.08	De actor kan een rapportbron toevoegen	**R.08/01:** Als gebruiker wil ik een rapportbron kunnen toevoegen, zodat kaartmateriaal gekoppeld is aan een tekstblok	Must
R.08	De actor kan een rapportbron toevoegen	**R.08/02:** Als gebruiker wil ik het zoomniveau van een rapportbron selecteren, zodat de eindgebruiker de juiste kaartdiepte ziet op de interactieve webpagina	Must
R.08	De actor kan een rapportbron toevoegen	**R.08/03:** Als gebruiker wil ik de getoonde ‘layers’ van een rapportbron selecteren, zodat overbodige legenda’s niet getoond zullen worden	Must
R.09	De actor kan een afbeelding toevoegen	**R.09/01:** Als gebruiker wil ik een afbeelding uploaden, zodat deze gekoppeld wordt aan een enkel tekstblok	Must
R.10	De actor kan html-tekst toevoegen	**R.10/01:** Als gebruiker wil ik html-tekst kunnen typen, zodat de geschreven	Must

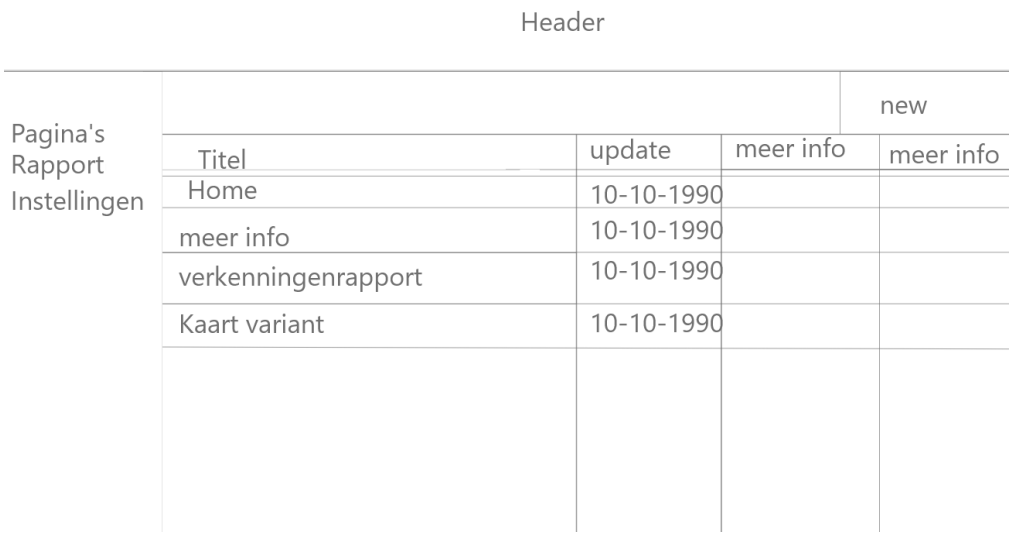
		opmaak gekoppeld wordt aan een enkel tekstblok	
R.11	De actor kan een thema selecteren	**R.11/01:** Als gebruiker wil ik een thema kunnen kiezen, zodat een tekstblok hoort bij een thema	Should
R.12	De actor kan met zijn bestaande Witteveen en Bos-account registeren	**R.12/01**Als gebruiker wil ik kunnen inloggen met mijn bestaande Witteveen en Bos-account, zodat ik niet meerdere gebruikersnamen en wachtwoorden heb	Could
R.13	De actor kan herschreven tekst terugzetten	**R.13/01**Als gebruiker wil ik herschreven tekst kunnen terugzien, zodat foutief herschreven tekst teruggeplaatst kan worden	Could
R.14	De tweede lezer kan geschreven tekst openbaar maken	**R.14/01**Als beheerder wil ik geschreven tekst kunnen accepteren, zodat de tekst getoond kan worden op een interactieve webpagina	Could

5.2.2 SCHERMSCHETSEN

Het ontwerp is opgesteld met Adobe XD. Met dit product is het mogelijk om pagina's aan elkaar te koppelen, zodat de flow van de webapplicatie zichtbaar is. Aan de hand van het design is bekeken of de opdracht duidelijk is en of alle onderdelen aanwezig zijn waaraan het CMS-systeem moet voldoen.

De applicatie bestaat uit twee onderdelen. Het eerste onderdeel is het opstellen van pagina's en het tweede onderdeel is het opstellen van documenten.

In Figuur 7 staan de pagina's van het ontwerp.

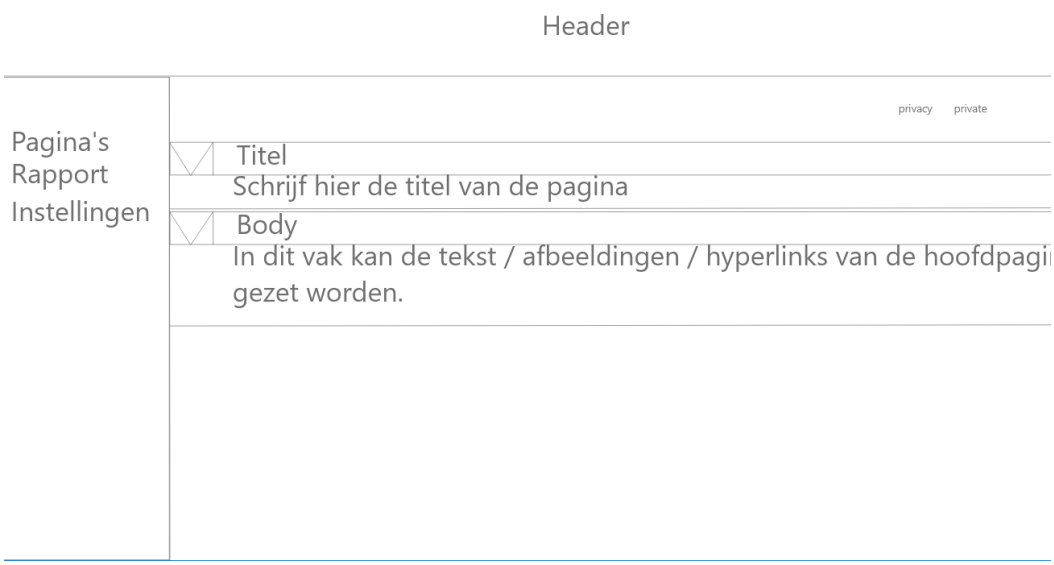


Figuur 7: Pagina's

Figuur 7 toont de pagina die te zien zijn wanneer is geklikt op de button 'Pagina's'. Op deze pagina kan de gebruiker zien welke soorten pagina's zijn aangemaakt. Alle aanwezige pagina's kunnen aangepast worden door te klikken op de titel van de te wijzigen pagina. Mocht een gebruiker een nieuwe pagina willen aanmaken, dan kan hij klikken op 'new'. Als de gebruiker op een nieuwe pagina klikt, heeft hij de mogelijkheid om te kiezen tussen drie soorten pagina's:

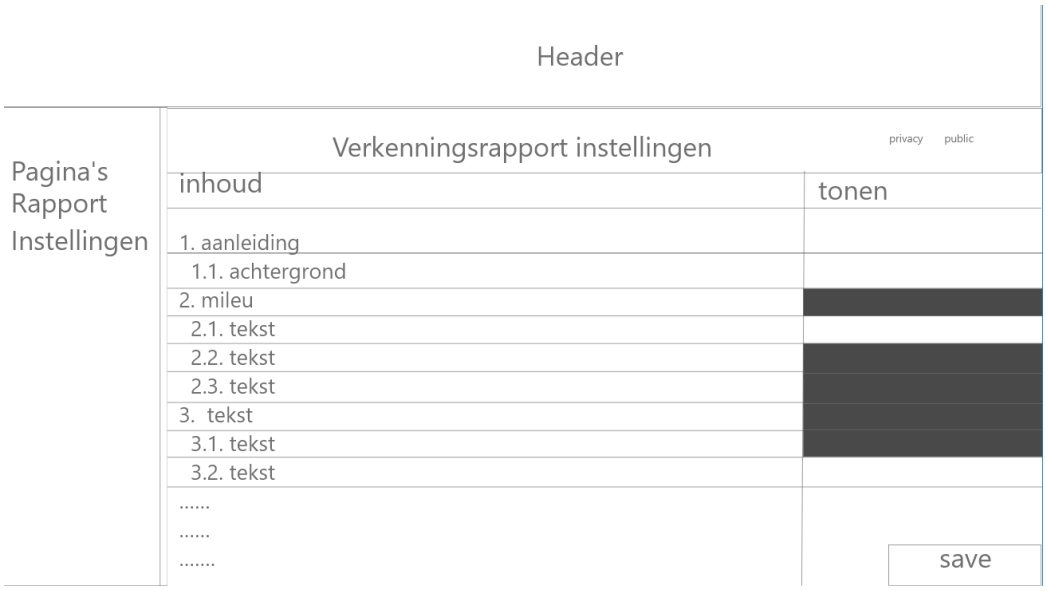
- standaardpagina
- verkenningsrapport
- kaartmateriaal

Deze verschillende pagina's worden hierna verder toegelicht.



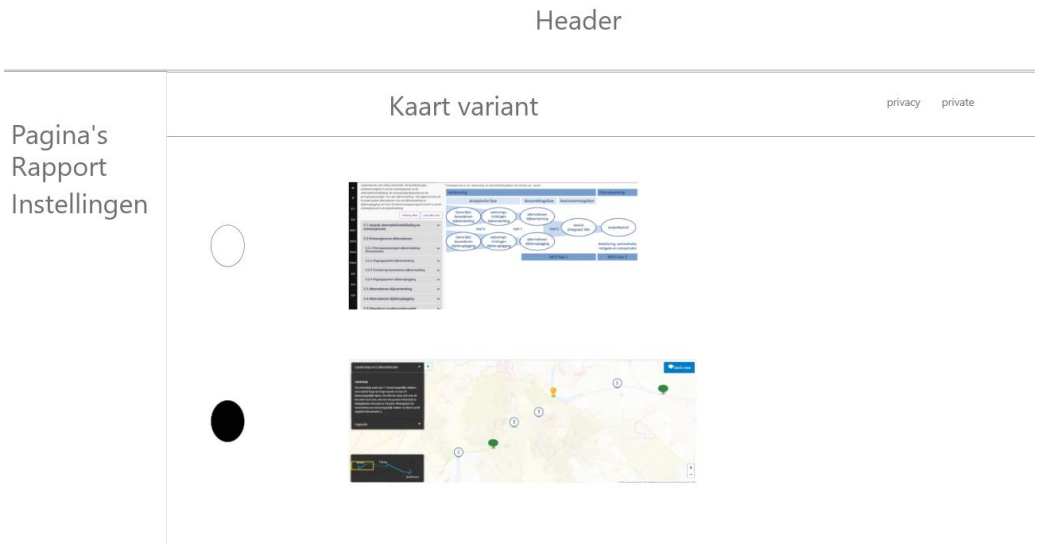
Figuur 8: Tekstuele pagina

Figuur 8 toont de pagina voor het aanmaken van de tekstuele pagina. Dit kan bijvoorbeeld content zijn op de voorpagina van een interactieve webpagina of tekstuele content op de contactpagina. Elk tekstuele pagina heeft een titel en een body. De body is de content van de pagina. Dit kan een introductietekst zijn voor bijvoorbeeld de homepage of de contactpagina. De titel wordt gebruikt om alle pagina's te tonen (Figuur 7).



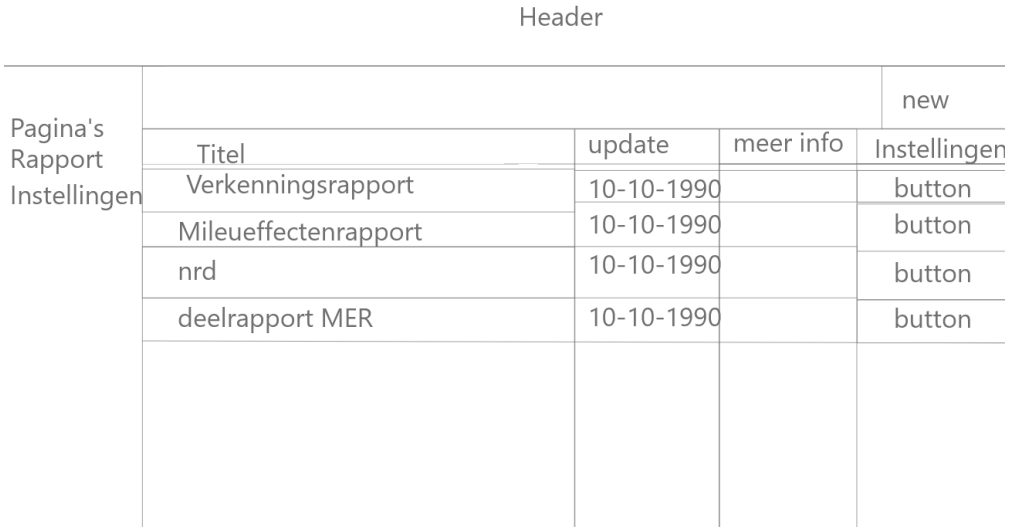
Figuur 9: Verkenningsrapport

Figuur 9 laat de pagina zien voor het samenstellen van een verkenningsrapport. Een verkenningsrapport is meestal een verzameling van tekstblokken uit meerdere rapporten. Op deze pagina kan de gebruiker enkele tekstblokken aanklikken die getoond moeten worden in een verkenningsrapport. Hierdoor is het mogelijk om uit verschillende rapporten tekstuele content te selecteren en zo een verkenningsrapport op te stellen.



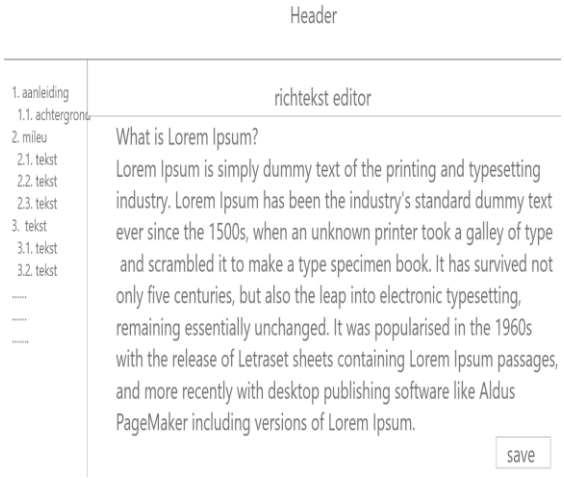
Figuur 10: Kaartvariant

Voor een interactieve webapplicatie zijn er twee kaartvarianten. Door gebruik te maken van de pagina in Figuur 10 kan de gebruiker aangeven welke variant hij wenst.



Figuur 11: Rapporten

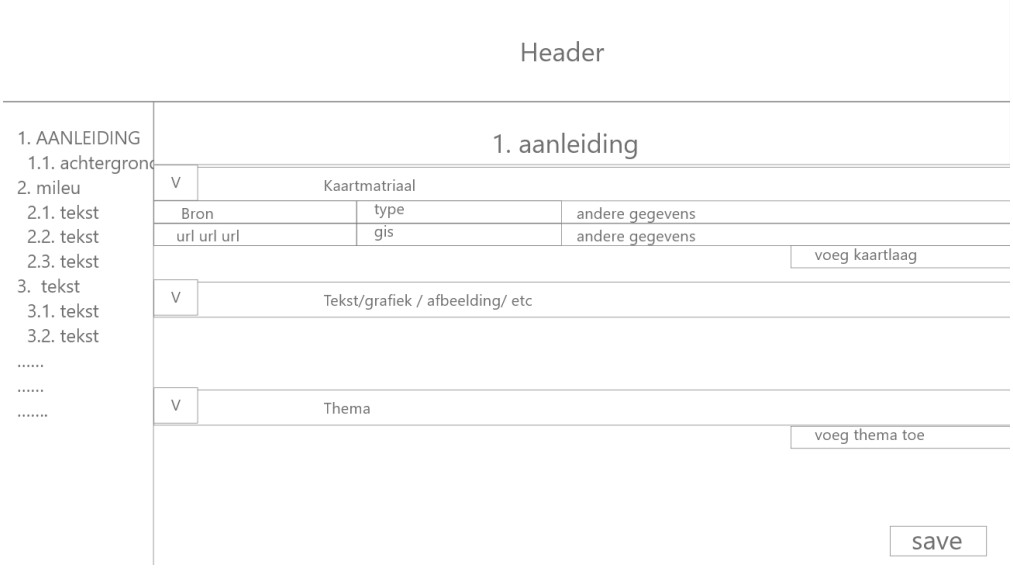
In Figuur 11 is de pagina te zien die wordt getoond nadat op de button ‘Rapport’ is geklikt. Door te klikken op de titel kan de gebruiker het rapport inzien en zo nodig het rapport wijzigen (Figuur 12). Door naar de instellingen te gaan, kan de gebruiker instellingen toevoegen aan het geselecteerde document (Figuur 14). Ook kan een nieuw rapport aangemaakt worden door te klikken op de button ‘new’.



Figuur 12: Rapportopbouw

Figuur 13: Voorbeeld van JSON

Figuur 12 toont de pagina waarop een document kan worden getypt. Zoals beschreven in de opdracht moet dit een tekstblokgericht document zijn. Dit wil zeggen dat de tekst ‘What is Lorem Ipsum?’ een ander tekstblok is dan de rest van de tekst. Hierdoor kan beeldmateriaal aan het desbetreffende tekstblok worden toegevoegd (Figuur 14). Een voorbeeld van de JSON-output staat in Figuur 13.



Figuur 14: Rapportkoppelingen

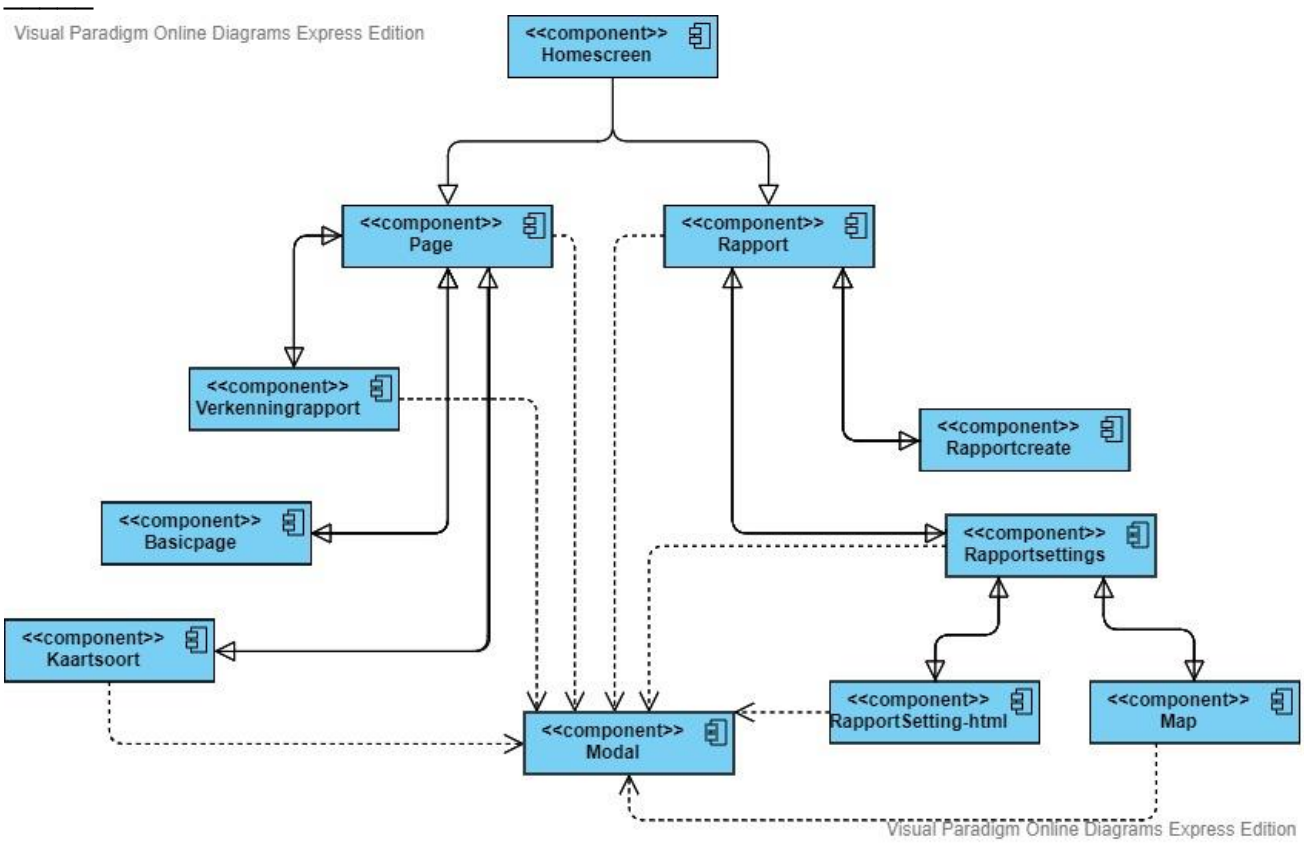
Zoals al eerder gedeeltelijk besproken, kunnen bijlages aan tekstblokken gekoppeld worden. De bijlages zijn onder te verdelen in drie categorieën:

- kaartmateriaal
- tekstuele content of afbeelding
- thema

Bij elk tekstblok mag het niet mogelijk zijn om de categorieën ‘kaartmateriaal’ en ‘tekstuele content of afbeelding’ te combineren.

Een document wordt in het algemeen geschreven door meerdere specialisten vanuit verschillende disciplines. Voor een interactieve webpresentatie is het daardoor noodzakelijk om content van elke discipline individueel te tonen. Dit wordt ook wel een thema genoemd. Door een tekstblok in de juiste themacategorie te zetten, kan gefilterd worden op thema.

5.2.3 STRUCTUUR



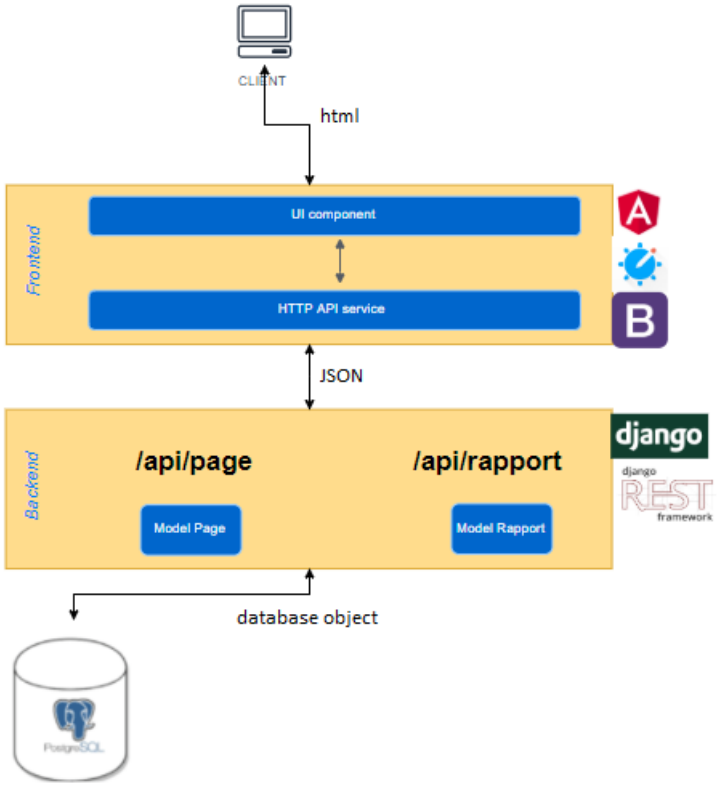
Figuur 15: Structuur diagram

In Figuur 15 is een overzicht van de structuur van de applicatie te zien. De componenten Page en Rapport zijn de twee hoofdpagina's waarmee de gebruiker de meeste functionaliteiten kan bereiken. Via de component Page kan de gebruiker alle functionaliteiten bereiken met betrekking tot het inrichten van een pagina. Via de component Rapport kan hij de functionaliteiten bereiken voor het schrijven van rapporten en het koppelen van verschillende bronnen.

5.3 TECHNISCH ONTWERP

5.3.1 DE APPLICATIE IN HOOFDLIJNEN

5.3.1.1 GRAFISCH OVERZICHT APPLICATIE



Figuur 16: Grafisch overzicht applicatie

Figuur 16 is een grafisch overzicht van hoe de communicatie verloopt tussen een cliënt en de applicatie. Door gebruik te maken van een grafisch ontwerp kan een applicatiebeheerder bijvoorbeeld in één oogopslag niet alleen zien welke prefix-endpoints van de backend aanwezig zijn, maar ook welk framework of welke bibliotheken worden gebruikt.

5.3.1.2 FRONTEND

Zoals in paragraaf 4.6 beschreven is, is de applicatie opgebouwd met een aparte frontend. Deze frontend is geschreven met het framework Angular. Witteveen en Bos ontwikkelt de meeste frontend-applicaties met het framework React.js. Vanwege de tijddruk en de kennis die de onderzoeker had van Angular is ervoor gekozen de applicatie te ontwikkelen met het framework Angular.

De applicatie is opgebouwd uit tien componenten, waarvan twee als werkelijke hoofdcomponenten gezien kunnen worden. Page en Rapport vormen samen de basis van de gehele applicatie; de meeste andere componenten functioneren als kindcomponent. Zo heeft een Page-component een verkenningsrapport-component.

Doordat de componenten opgesplitst zijn, konden veel functionaliteiten gescheiden worden gehouden. Door componenten te maken voor bijvoorbeeld een Modal-component konden deze eenvoudig worden hergebruikt op alle pagina's. De splitsing tussen componenten van deze applicatie heeft twee redenen: het hergebruik en de scheiding van complexe stukken code ('separation of concerns').

5.3.1.3 BACKEND

De backend is ontwikkeld in Python en maakt gebruikt van het Django-REST-framework. Dit is een veelgebruikt framework bij Witteveen en Bos.

De Django-backend API is onder te verdelen in twee 'applications'. De Django-terminologie 'application' beschrijft een Python-package dat een aantal 'classes' biedt. De eerste application is Page en de tweede is Rapport.

Elke application kent verschillende classes. Dit zijn de classes Model, Serializer, URL en View.

Model: De class Model biedt 'object-relational mapping' (ORM) aan de onderliggende database. ORM is een krachtige programmeertechniek die het werken met gegevens en relationele databases sterk vereenvoudigd. Dankzij Model hoeft de programmeur de onderliggende techniek niet te kennen.

Serializer: Met de class Serializer kunnen complexe gegevens, zoals querysets en modelinstanties, worden geconverteerd naar bijvoorbeeld JSON. Serializer biedt ook deserializer, waardoor JSON-gegevens weer kunnen worden omgezet in complexe types, na eerst de gegevens te valideren.

View: In de class View komen alle gegevens bij elkaar en bevindt zich de logica van de applicatie. Met View kan het model worden aangeroepen waar de gebruiker vraagt om gegevens. De gegevens kunnen vervolgens worden bewerkt of gesorteerd en door middel van Serializer class worden

‘geserializeerd’ in een JSON-object. Hierna is het mogelijk om het JSON-object te versturen.

URL: Met de class URL kan de gebruiker de URL’s van de applicatie inrichten. Voorbeelden van de uitgewerkte URL’s zijn te vinden in de bijlage.

5.3.2 BIBLIOTHEKEN

5.3.2.1 FRONTEND

Bootstrap

Voor de opmaak is gebruikgemaakt van Bootstrap. Dit is een framework voor html en CSS. Het heeft een aantal handige functies waarmee een website eenvoudig en snel kan worden ontwikkeld.

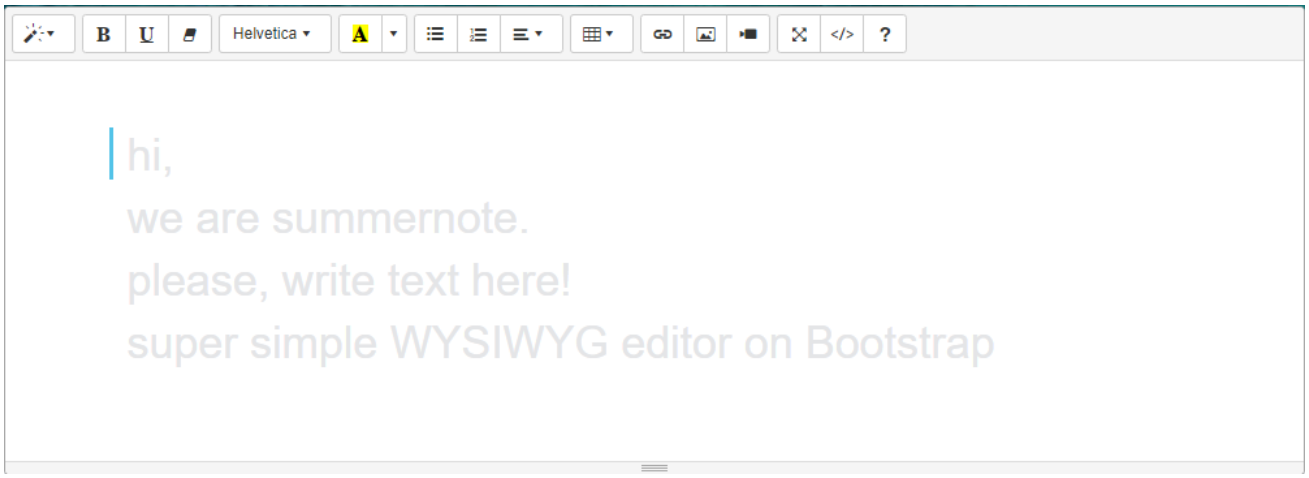
De keuze om Bootstrap te gebruiken, is omdat Summernote.js hier ook gebruik van maakt. Om de applicatie niet te overbelasten met frameworks, is voor het basisdesign gebruikgemaakt van Bootstrap.

Leaflet.js

Leaflet.js is een JavaScript-bibliotheek waarmee het mogelijk is om kaartmateriaal met verschillende kaartlayers te tonen. In de applicatie wordt Leaflet gebruikt om het zoomniveau van de opgegeven MapServer en de desbetreffende kaart te berekenen. Hierdoor kunnen deze gegevens opgeslagen worden, zodat bij de publicatie de juiste positie getoond kan worden.

Summernote.js

Summernote.js is een WYSIWYG-editor waarmee eenvoudig tekstuele content kan worden omgezet naar html-content. Figuur 17 laat de standaarduitvoering van Summernote.js zien.



Figuur 17: Summernote

Nestable.js

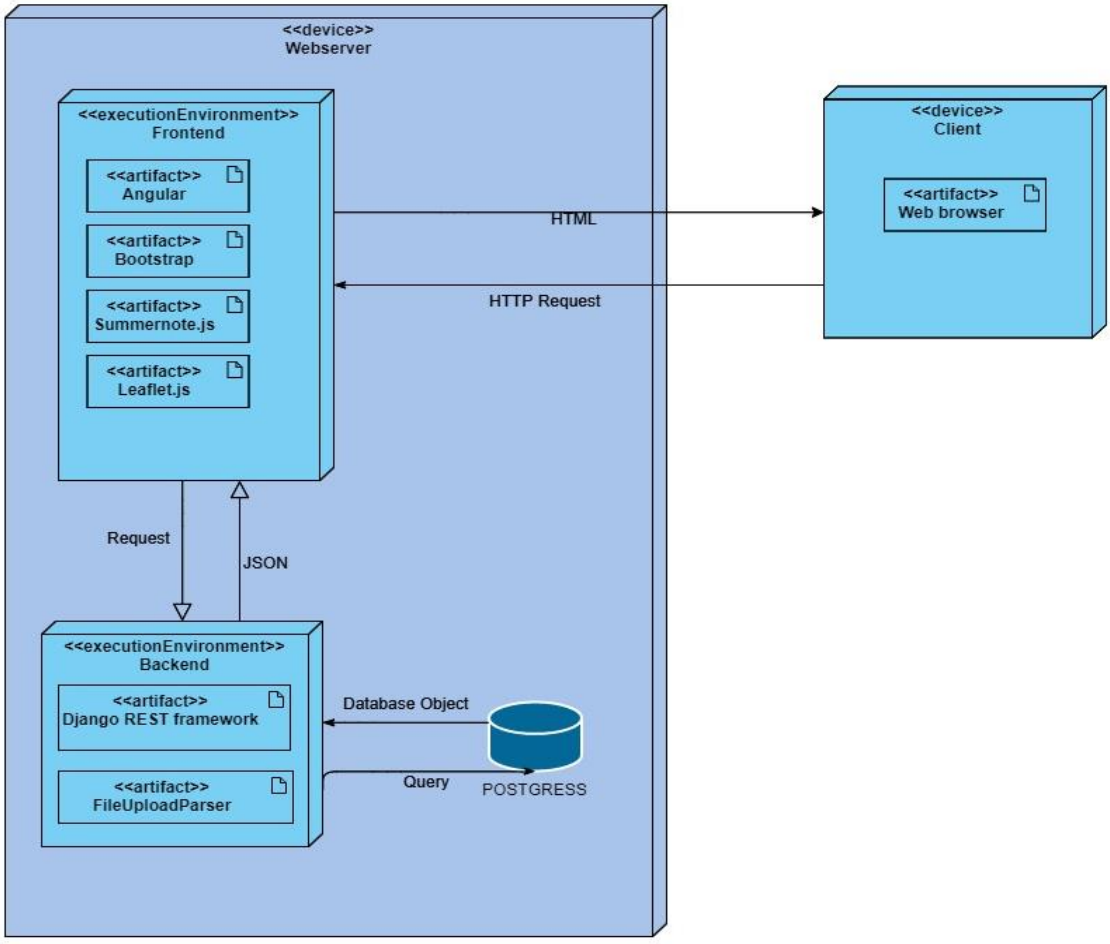
Nestable.js is een JavaScript-plug-in die drag-and-drop van de webapplicatie-onderdelen verzorgt.

5.3.2.2 BACKEND

FileUploadParser: Om afbeeldingen te uploaden, is gebruikgemaakt van een functionaliteit die ook wordt gebruikt in Wagtail en django CMS. Door deze functionaliteit aan te roepen, kan een afbeelding eenvoudig opgeslagen worden. Hierbij hoeft niet gekeken te worden of bijvoorbeeld de naam al voorkomt in de map waar de afbeelding opgeslagen gaat worden. Al deze facetten worden verzorgd door deze bibliotheek.

5.3.3 INFRASTRUCTUUR

5.3.3.1 NETWERKONTWERP



Figuur 18: Netwerkdiagram

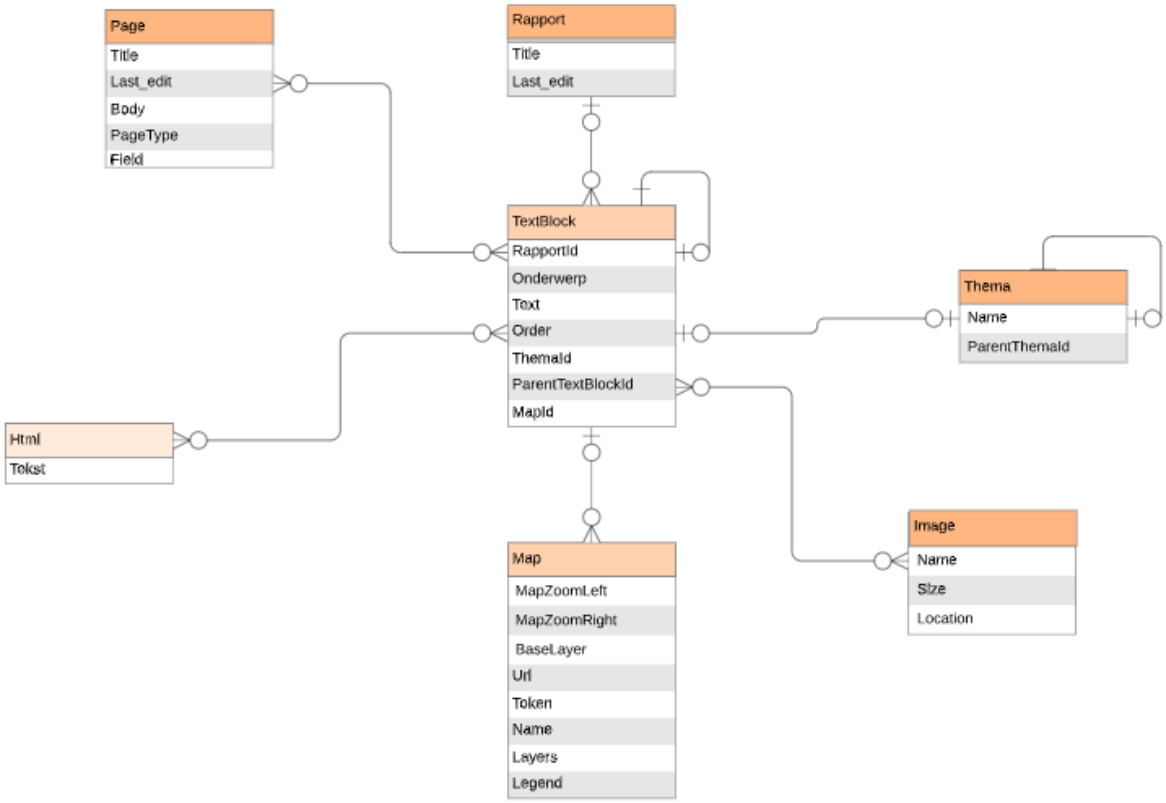
Zoals te zien is in Figuur 18, is de applicatie onder te verdelen in twee subapplicaties: de frontend en de backend. Beide kunnen op dezelfde webserver gehost worden.

5.3.4 ARCHITECTUUR APPLICATIE

Deze paragraaf beschrijft de technieken die zijn gebruikt bij de ontwikkeling van de applicatie.

5.3.4.1 OPSLAG

De data worden opgeslagen in een Postgres-database. Om de entiteiten, relaties en regels in kaart te brengen, is een entiteit-relatiediagram (ER-diagram) opgesteld (Figuur 19). Dankzij dit diagram kan in één oogopslag bekeken worden welke verbintenissen zich bevinden tussen de verschillende tabellen.



Figuur 19: Entiteit-relatiediagram

Uit het ER-diagram komt een aantal belangrijke onderdelen naar voren waar rekening mee gehouden moet worden bij het opstellen van het uiteindelijke databasemodel.

Terugkoppelde relaties

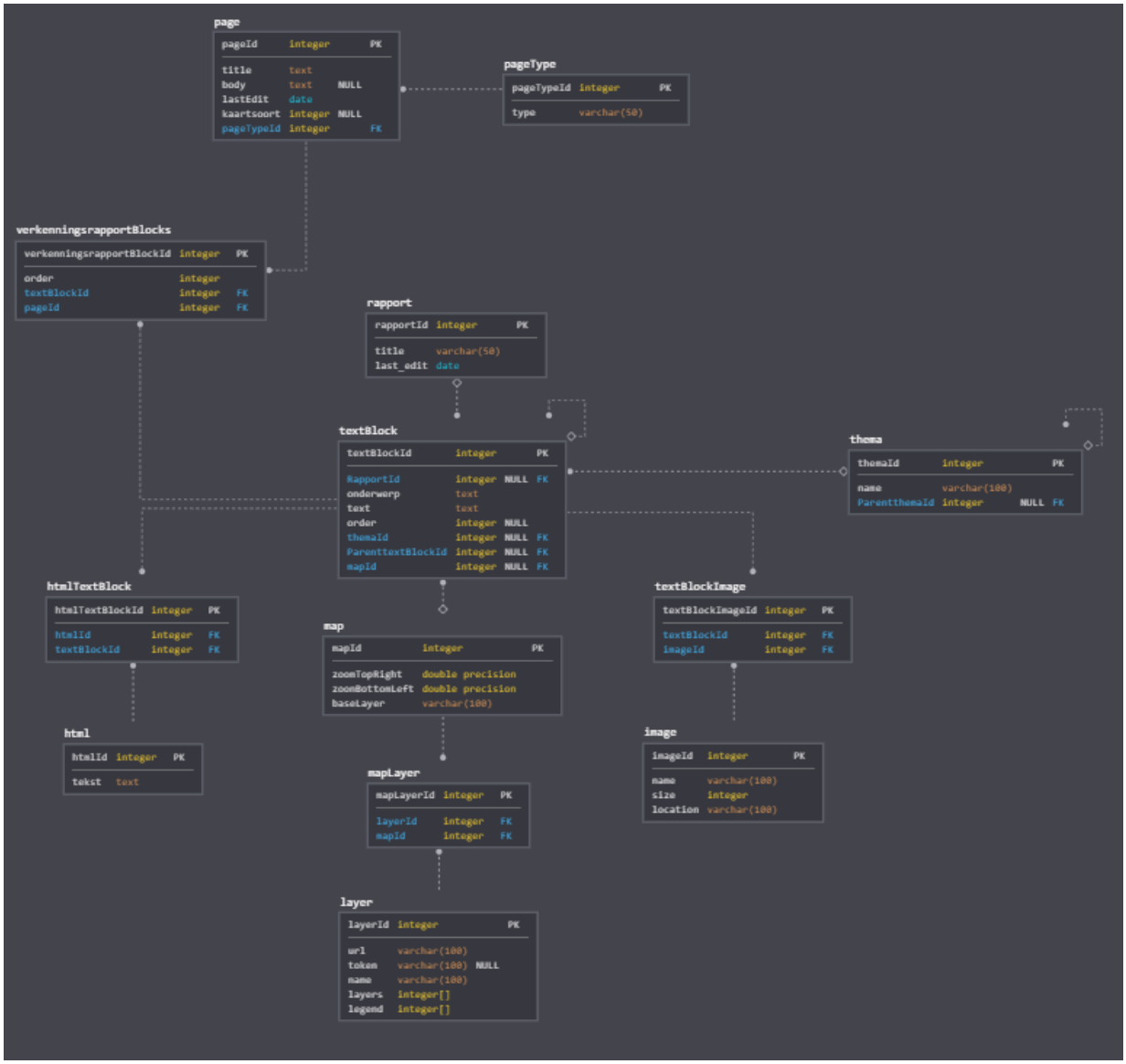
In het ER-diagram is te zien dat de tabellen TextBlock en Thema reflexieve relaties hebben. Dit komt doordat een thema bijvoorbeeld een subthema kan hebben, en een subthema weer een subthema kan hebben. Door bij het subthema het bovenliggende thema-Id op te slaan, wordt een boomstructuur verkregen die oneindig herhaald kan worden. Dit geldt ook voor de tabel TextBlock.

Many-to-manyrelaties

Ook laat het ER-diagram zien dat er een aantal many-to-manyrelaties is. Dit zijn relaties waarbij meerdere records van een tabel zijn gekoppeld aan meerdere records van een andere tabel. De many-to-manyrelaties in het ontwerp zijn:

- TextBlock – Page
- TextBlock – Image
- TextBlock – Html

Figuur 20 toont het databasemodel nadat deze onderdelen zijn verwerkt.



Figuur 20: Databasemodel

5.3.4.2 SERVER

Voor de backend-applicaties maakt Witteveen en Bos gebruik van de programmeertalen Python en PHP. Voor dit opdracht is gekozen voor Python vanwege leergierigheid van de onderzoeker, de wens kennis op te doen en de voorkeur van Witteveen en Bos.

Voor de programmeertaal Python bestaan veel frameworks waarbij het mogelijk is om een REST API op te zetten. Voor dit project is gebruikgemaakt van het Django-REST-framework. Dit framework is ook bekend bij het merendeel van de collega's.

Met Djang-REST-framework kunnen tabellen eenvoudig gedefinieerd worden en kunnen kolommen toegevoegd worden. Dit gebeurt in de file Model waar elke class een aparte tabel is. Hier is gebruik van gemaakt. Een voorbeeld van een class staat in Figuur 21.

```
class thema(models.Model):
    themaName = models.CharField(max_length=100)
    submenuId = models.ForeignKey(  'self',
                                    on_delete=models.SET_NULL,
                                    null = True,
                                    blank = True )
```

Figuur 21: Voorbeeld class Model

Aan de hand van het databasemodel, waardoor bekend was welke kolommen in welke tabel horen en met welke waarde, kon de database eenvoudig worden ingericht.

Voor de backend is een aantal endpoints ontwikkeld. Dit zijn zowel de endpoints voor de communicatie naar het CMS-systeem als die voor de communicatie naar de interactieve webapplicatie. Alle ontwikkelde endpoints zijn te vinden in bijlage 1.

5.3.4.3 CLIENT

Om de gebruikersinteractie en -dynamiek aan de applicatie toe te kunnen voegen, is gebruikgemaakt van Angular 6. Dit is volledig ontwikkeld door gebruik te maken van TypeScript, een superset van JavaScript.

Voor het template is gekozen voor HTML5. De standaardtemplates zijn wel verfraaid door gebruik te maken van Bootstrap en waar Bootstrap tekortschoot, kan dit aangevuld worden met een eigen versie van CSS.

Voor data en dynamiek die niet bij een specifieke view horen, is gebruikgemaakt van een serviceclass. In de class bevindt zich onder andere de ‘request’ naar de backend, de opslag van de opgehaalde gegevens en de doorgave van de informatie tussen verschillende componenten. Al deze design-patterns dienen om het te ontwikkelen project zo klein en overzichtelijk mogelijk te houden (‘encapsulation’).

Naast eigen ontwikkeling is gebruikgemaakt van meerdere bibliotheken. De gebruikte bibliotheken zijn vermeld in paragraaf 5.3.1.2.

5.4 KWALITEIT

De kwaliteit van de applicatie is gewaarborgd door verschillende onderdelen, die hierna worden toegelicht.

5.4.1 CODEREVIEW

De code die geschreven is, is geüpload binnen de GitLab-omgeving van Witteveen en Bos. Elk stukje code dat geschreven is, is eerst gecontroleerd door Leon Roeterdink. Hierdoor weet Witteveen en Bos dat de geschreven code van goede kwaliteit is, en de ontwikkelaar weet dat de geschreven code geaccepteerd is.

Ingewikkelde of uitgebreide functies zijn voorzien van inline-codedocumentatie. Per functie is beschreven wat de verwachte input is, en mocht een waarde teruggeven worden, wat voor soort waarde het is.

5.4.2 FRONTEND-TESTING

Om de frontend te testen, is gebruikgemaakt van Jasmin. Aan de hand van de testcases die herleid zijn van de userstories, zijn met een unittest delen van de software getest om te zien of ze goed werken en blijven werken op basis van bepaalde invoer.

In de testen zijn niet alleen de verschillende componenten getoetst, maar ook de service waar Angular gebruik van maakt. Voor ieder component uit figuur 15 wordt getoetst of het component ingeladen kan worden en juist wordt gesloten. Daarnaast worden de meest basale functionaliteiten van ieder component apart getoets in een Unit test.

Als voorbeeld wordt hier de component Modal toegelicht. De component Page overerft de Modal-component wanneer deze gebruikt gaat worden (zie Figuur 15 voor het diagram).

De Modal-component word getoond wanneer een boolean op ‘true’ komt te staan en weer verdwijnt wanneer ‘false’ verschijnt. Om dit op juistheid te testen, zijn de volgende unittesten ontwikkeld.

```
describe('check toggle modal ',function(){

  it('the newPageModal must be false', (()=>{
    expect(component.newPageModal).toBe(false)
  }))
  it('the newPageModal must change to true after open ', (()=>{
    component.toggleModal('open');
    expect(component.newPageModal).toBe(true)
  }))
  it('the newPageModal must change to false after close ', (()=>{
    component.toggleModal('close');
    expect(component.newPageModal).toBe(false)
  }))
})
```

Figuur 22: Unit-test

De testen zijn gecategoriseerd per component. Elke component is weer onderverdeeld per onderdeel zoals weergegeven in Figuur 22 (’check toggle modal’).

Figuur 22 toont drie ontwikkelde testen. Als eerste wordt gecontroleerd of de waarde daadwerkelijk op ‘false’ staat. Dit is noodzakelijk om de Modal niet te tonen. Deze wordt alleen getoond als de waarde op ‘true’ staat. De tweede en derde test zijn ontwikkeld om te controleren of de functie ‘newPageModal’ de waarde verandert.

De uitkomsten van de Modal-testen zijn weergegeven in Figuur 23.

```
check toggle modal
  the newPageModal must be false
  the newPageModal must change to true after open
  the newPageModal must change to false after close
```

Figuur 23: Uitkomst Modal-unittest

6 AANBEVELINGEN

Aangezien de afstudeerperiode een einddatum had, is tijdens de planning bekeken of het project binnen deze periode kon worden afgerond. Een aantal noodzakelijke onderdelen is daardoor nog niet ontwikkeld.

Ook is in verband met de tijd niet altijd gekozen voor de optimale technische oplossingen. Aan deze toepassingen kan in de nabije toekomst nog gewerkt worden. De voornaamste worden hierna beschreven.

API-beveiliging (requirement R.01)

De gehele API van het CMS-systeem is niet beveiligd met een gebruikersnaam en wachtwoord. Hierdoor heeft iedereen de mogelijkheid om content toe te voegen of te verwijderen. Dit is buiten de scope van het onderzoek gehouden en er is voor gekozen het project meer te richten op de belangrijkere onderdelen, zoals publicatie van beeldmateriaal.

Versiebeheer (requirement R.13)

Om documenten te schrijven, is versiebeheer een goede toepassing. Aan een verslag werken meestal meerdere personen en het kan voorkomen dat iemand het niet eens is met de tekst die een ander heeft gepubliceerd. Voor de specialisten is het relevant als zij door middel van versiebeheer de teksten kunnen vergelijken of een oudere versie kunnen terugplaatsen.

Tweede-lezerschap toetsen (requirement R.14)

Tweede-lezerschap houdt in dat content wel aangemaakt kan worden, maar dat deze alleen na een toetsing wordt vrijgegeven. Dit kan een toegevoegde waarde hebben bij bijvoorbeeld een interactieve webapplicatie voor verbreding van de snelweg A58. Hierbij kan een lid van Rijkswaterstaat de content beoordelen en mocht hij het hiermee eens zijn, de content dan pas openbaar maken. Ook kan de tweede lezer fouten op de website eerder herkennen in plaats van een ander persoon.

Single sign-on (requirement R.12)

‘Single sign-on’ biedt ook een uitkomst op de eerste aanbeveling, API-beveiliging. Aan verslagen of rapporten werken meerdere mensen, maar het zou onpraktisch zijn om voor elk project nieuwe accounts te realiseren. Door gebruik te maken van single sign-on gekoppeld aan de DHCP-server van Witteveen en Bos is het mogelijk om iedere ingelogde medewerker direct te identificeren en kan worden bekeken op welke afdeling hij werkt en of hij gebruik moet kunnen maken van het CMS-systeem. Hierdoor kan bijvoorbeeld bij versiebeheer gezien worden wie de tekst heeft aangepast of verwijderd.

7 REFLECTIE

De afstudeerperiode bij Witteveen en Bos was niet alleen leerzaam, maar vooral ook prettig. Aan het begin van de afstudeerperiode had ik geen ervaring met of kennis van veel onderdelen waarmee ik gewerkt hebt. Voorbeelden daarvan zijn MapServer en Python. Ik wist ook niet wat bijvoorbeeld het verschil is tussen een NRD en MER.

Allereerst heb ik mij verdiept in de verschillende rapporten. Ik heb ontdekt waarin de eerdergenoemde documenten verschillen. Zo weet ik nu dat een NRD een eerste stap is in de procedure van de MER. Het is een raadpleging waarover de MER moet gaan.

Daarna ben ik begonnen met het onderzoek naar bestaande CMS-systemen. Ik ontdekte dat er veel verschillende soorten CMS-systemen zijn, van open source tot dure pakketten. Aangezien Witteveen en Bos graag een open source CMS-systeem wilde, was ik in het voordeel en kon ik direct al veel CMS-systemen laten vervallen voor mijn onderzoek.

Voor de overgebleven CMS-systemen heb ik bekeken welke het beste scoorde op GitHub. Mijn idee was dat hoe meer mensen hier gebruik van maken, des te beter het systeem meestal is. Elk overgebleven CMS-systeem heb ik geïnstalleerd en er twee dagen mee gewerkt.

Tijdens mijn afstudeerperiode heb ik veel geleerd. Aan de technische kant kan ik zeggen dat ik Python beheers door gebruik te maken van de Django-framework. Ik heb geleerd hoe ik een frontend bouw met de Django-templatetaal.

Ik heb tijdens mijn afstudeerperiode niet alleen vakinhoudelijke kennis opgedaan, zoals een programmeertaal leren, maar ik heb me ook verdiept in specifieke onderdelen, zoals het gebruik van MapServer met het framework Leaflet.js. Dit vond ik interessant, al denk ik dat ik het niet vaak ga gebruiken als ik niet bij een soortgelijk bedrijf ga werken.

Daarnaast heb ik een paar stressmomenten ervaren, bijvoorbeeld toen de product-owner midden in mijn afstudeerperiode tekstblokken wilde sorteren door gebruik te maken van drag-and-drop. Dit was niet tot moeilijk te realiseren met de Django-templates die ik als eerste had gebruikt. Hiervoor moest wel de frontend geheel herschreven worden in een JavaScript-framework en de Django-applicatie in een Django-backend-applicatie. Dit heeft redelijk wat kostbare tijd gekost.

Een ander stressmoment was tijdens mijn presentatie en verdediging. Tijdens de presentatie, ging ik de applicatie demonsteren. Ik was eerder naar school gekomen om de laptop te koppelen aan het draadloos internet van school. Door de spanning was ik dat geheel vergeten. Hierdoor kon ik tijdens de demonstratie geen koppeling maken met de mapserver van Witteveen en Bos. Eenmaal toch de wifi ingesteld te hebben, bleek de token verlopen te zijn waardoor ik de gehele voorbereide tekst ook vergat.

Leermoment presentatie op de locatie al een keer te ovenem

Op persoonlijk gebied heb ik ook veel geleerd. Medewerkers van Witteveen en Bos zijn vrij om hun tijd zelf in te delen. Van hen wordt verwacht dat ze acht uur per dag werken, maar er zijn geen vaste start- en eindtijden; de medewerker moet die zelf inplannen. Dit brengt verantwoordelijkheid met zich mee. Mijn afstudeerbegeleider heeft mij goed begeleid. Ik had elke week een voortgangsgesprek en ik kon tijdens de bijeenkomsten allerlei soorten vragen stellen. Mocht hij het antwoord niet weten, dan moest ik wel zelf bij een collega aankloppen voor hulp.

Ten slotte heb ik gewerkt aan mijn professionaliteit door alle vragen en opmerkingen waar ik niet gelijk antwoord op kon geven, te noteren. De gemaakte afspraken heb ik digitaal bevestigd, zodat deze zwart-op-wit stonden.

8 BRONNENLIJST

[1] tuk, y. (2017, 24 juli). *Headless CMS'en: "Ja, een hype. Maar dat is niet slecht" - Emerce*. Geraadpleegd op 25 juli 2019, van https://www.emerce.nl/achtergrond/headless-cmsen-hype

[2] *Wat is een CMS en wat zijn de voordelen? - Hebsite*. (2017, 29 oktober). Geraadpleegd op 20 juni 2019, van https://www.hebsite.nl/uitleg/websites-cms/

[3] *Blog tool, publicatieplatform en CMS - WordPress*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://nl.wordpress.org/

[4] *WordPress/WordPress*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://github.com/WordPress/WordPress

[5] *Drupal*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://www.drupal.org/developers

[6] *drupal/drupal*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://github.com/drupal/drupal

[7] *Django CMS*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://www.django-cms.org/en/

[8] *divio/django-cms*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://github.com/divio/django-cms

[9] *Django Content Management System | Wagtail CMS*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://wagtail.io/

[10] *wagtail/wagtail*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://github.com/wagtail/wagtail

[11] *Strapi - Open source Node.js Headless CMS* 🚀. (z.d.). Geraadpleegd op 9 oktober 2019, van https://strapi.io/

[12] *strapi/strapi*. (z.d.). Geraadpleegd op 9 oktober 2019, van https://github.com/strapi/strapi

9 BIJLAGE

9.1 API ENDPOINTS

De eindpoints zijn onder te verdelen in 2 categorieën, de eerste categorie is voor het maken van pages en de andere voor het onderdeel rapport.

In de tabel 1 en 2 vind je alle apicalls die zijn gemaakt. In de kolom body staan alleen de waardes die verplicht opgegeven moet worden, optionele waardes kunnen ingezien worden in het database model.

Pages:

Request type	URL	Body	Tekstueel uitleg
GET	/page/		Vraag alle pagina's op
POST	/page/	Title:string	Maak een nieuwe pagina aan
GET	/page/single/{pageId}/		Vraag de informatie van een enkel pagina
PUT	/page/single/{pageId}/		Verander de waarde van enkel pagina
DELETE	/page/single/{pageId}/		Verwijder een enkele pagina

Rapports:

Request Type	URL	Body	Tekstueel uitleg
GET	/rapport/		Vraag alle rapporten op
POST	/rapport/	Title:string	Maak een nieuwe rapport
GET	/rapport/textblock/{rapportId}/		Vraag de informatie van een enkele pagina
PUT	/rapport/textblock/{rapportId}/	Onderwerp: string Tekst: string Order: number	Deze apicall kijkt als eerst of er een textblockId is meegestuurd. Mocht dit niet het geval zijn dan word er nieuwe textblock aangemaakt. Anders wordt de bestaande textblock geüpdatet. Het is ook mogelijk om meerdere tekstblokken te versturen.
DELETE	/kaartmateriaal/{textblockId}		Verwijder het kaartmateriaal
POST	/rapport/image/{textblockId}	Image:File	Koppelt en upload het image aan de textblock
DELETE	/rapport/image/{textblockId}		Verwijderd de koppeltabel van het image. Het image blijft nog wel bestaan.

PUT	/rapport/thema/{textblockId}	themaId:Number	Voegt een thema toe aan het textblock of update de thema
DELETE	/rapport/thema/{textblockId}		Verwijderd thema bij het textblock
POST	/rapport/htmlText/	Tekst:string textBlockId: number	Koppelt een html tekst aan het textblock en voegt de tekst toe aan de database.
GET	/rapport/htmlText/{htmltextId}		Vraag de informatie van een enkel tekstblock
PUT	/rapport/htmlText/	Tekst: string Textblockid: number htmlId: number	update de bestaande htmlText
DELETE	/rapport/htmlText/	textblockId: number htmltext: number	Verwijder de koppeltabel van de htmltext naar het textblock