BEHEER VAN HTML OP ÉÉN CENTRALE LOCATIE

HTML management at one central location

BEN VAN DE WAL

15 JANUARI 2015

Colours

BEHEER VAN HTML OP ÉÉN CENTRALE LOCATIE

HTML MANAGEMENT AT ONE CENTRAL LOCATION

BEN VAN DE WAL
2015

COLOFON

ALGEMEEN

Titel afstudeerverslag	:	Beheer van HTML op één centrale locatie
Afstudeerperiode	:	1 september 2014 t/m 6 februari 2015
Datum van uitgifte	:	15 januari 2015

AUTEUR

Naam	:	Wal, Ben B.J. van de
Studentnummer	:	2164044
E-mail	:	bjvandewal@gmail.com
Opleiding	:	Fontys Hogeschool ICT, Eindhoven
Afstudeerrichting	:	ICT & Media Design, voltijd

BEDRIJF

Naam	:	Colours
Afdeling	:	Development
Adres	:	Prins Bernhardstraat 14
Postcode, Plaats	:	5211 HE, 's-Hertogenbosch
Website	:	www.colours.nl

BEDRIJFSBEGELEIDER

Naam	:	Habraken, Rob R.
Functie	:	Technical Lead
E-mail	:	r.habraken@colours.nl

DOCENTBEGELEIDER

Naam	:	Graaumans, Joris J.P.M.
E-mail	:	j.graaumans@fontys.nl

GETEKEND VOOR GEZIEN DOOR BEDRIJFSBEGELEIDER

Datum	:	15 januari 2015
Naam	:	Habraken, Rob R.

_ _

VOORWOORD

Dit is de scriptie 'Beheer van HTML op één centrale locatie'. Deze scriptie is geschreven aan de hand van het onderzoek dat ik gedaan heb in opdracht van Colours, een full service internet bureau in 's-Hertogenbosch. In het kader van mijn afstuderen aan de opleiding ICT & Media Design aan Fontys Hogescholen ICT (FHICT) te Eindhoven is deze scriptie geschreven en is bedoeld voor mijn bedrijfsbegeleider en docentbegeleider.

Tijdens de ontwikkeling van een website lopen de front-end en back-end developers van Colours tegen het probleem dat ze vaak dubbel werk moeten verrichten. Dit treedt op bij het overzetten van HTML. Daarom heb ik onderzoek gedaan naar de mogelijkheden om HTML op één centrale locatie te beheren.

Ik heb voor deze afstudeeropdracht gekozen omdat het raakvlakken heeft met front-end en back-end development. Beide aspecten van webdevelopment vind ik interessant. Daarnaast wilde ik graag een technische opdracht aangaan in plaats van een designgerichte opdracht.

Dit onderzoek had ik nooit uit kunnen voeren zonder hulp. Daarom wil ik deze mogelijkheid gebruiken om de volgende mensen te bedanken:

ROB HABRAKEN

Voor zijn goede algemene en technische begeleiding, uitgebreide feedback en enthousiasme.

BERTUS GROENEWEGEN

Voor het delen van zijn kennis op het gebied van front-end en de energizers op momenten dat ik het nodig had.

BEN VERHEES

Voor het duwtje in de rug tijdens de keuze van de 'vertaalslag' en het beantwoorden van mijn technische vragen.

JORIS GRAAUMANS

Voor zijn begeleiding vanuit school, toezicht op mijn proces, het beantwoorden van mijn vragen en het feit dat ik altijd bij hem terecht kon.

Daarnaast wil ik alle Colours collega's bedanken voor hun behulpzaamheid en belangstelling. Ik heb een leuke en leerzame periode gehad bij Colours waar ik met veel plezier op terugkijk.

BEN VAN DE WAL

Oirschot, 11 januari 2015.

INHOUDSOPGAVE

INHOUDSOPGAVE

SAMENVATTING

Van 1 september 2014 tot en met 6 februari 2015 heb ik een afstudeeropdracht uitgevoerd binnen Colours. Colours is een fullservice internetbureau in 's-Hertogenbosch. Het ontwikkelen van succesvolle digitale concepten en daarbij websites is de specialiteit van Colours. Deze websites worden door middel van online marketing gelanceerd. Colours bestaat uit een gemixt team van designers, developers, projectmanagers en marketeers.

Voor elke website wordt een style guide opgezet. Aan de hand van deze style guide wordt de website opgebouwd, de .NET Solution. Hierin wordt HTML uit de style guide gekopieerd. Zo ontstaan twee HTML versies welke apart beheerd worden. Aanpassingen moeten dus twee keer gedaan worden. Dit kost onnodig veel tijd. Ook kunnen er verschillen in de HTML ontstaan doordat Colours werkt met versiebeheer. Er kunnen conflicten ontstaan bij het samenvoegen van lokale branches.

Om deze problemen te voorkomen is er onderzoek gedaan naar mogelijkheden om HTML op een efficiënte manier te gebruiken binnen een project, met als uitgangspunt dat developers geen of minder dubbel werk hoeven te doen en front-end aanpassingen sneller zijn door te voeren in de .NET Solution.

De afstudeeropdracht heb ik aangepakt volgens het Tien Stappen Plan in combinatie met Scrum. Tijdens fase één, de oriëntatiefase, is gekozen voor een aanpak met bijbehorende planning. Dit is opgenomen in het Project Initiatie Document (PID). Hierop volgend is het onderzoeksplan geschreven. Tijdens de onderzoeks- en oplossingsfase is het probleem onderzocht door desk- en fieldresearch. Hierbij is eerst onderzocht wat de beste manier is om HTML te beheren. Vervolgens zijn de tools onderzocht en gebruikt die de front-end developers bij Colours gebruiken. Daarna is het .NET-based Content Management Systeem (CMS) Sitecore onderzocht en gebruikt. Door middel van het toevoegen van MVC5 is hier nog dieper op in gegaan. Vervolgens is een beslissing genomen over de te ontwikkelen oplossing. Aan de hand van deze conclusies heb ik een proof of concept ontwikkeld. Dit proof of concept is een custom view engine die aantoont dat het mogelijk is Mustache files als HTML te renderen in de browser binnen een .NET Sitecore solution. De laatste fase is de invoeringsfase. Hier wordt aangetoond dat HTML beheren op één centrale plaats technisch mogelijk is. De custom view engine wordt gepresenteerd aan Colours en Fontys en wordt vervolgens beoordeeld.

Uit het onderzoek is gebleken dat het beter is om de front-end code binnen het CMS te genereren. Dit brengt front-end en back-end development dichter bij elkaar. Ook is gebleken dat Pattern Lab gebruikt moet blijven worden. De gestructureerde, generieke opzet en het genereren van statische style guides maakt Pattern Lab een krachtige tool. Door de back-end op te zetten volgens het MVC5-model is de vertaalslag tussen front-end en back-end kleiner. Om deze vertaalslag efficiënt te maken is gekozen voor een custom view engine. Dit scheelt een transformatieslag waardoor het minder foutgevoelig is. Daarnaast heeft een view engine meer waarde voor Colours gezien de uitbreidingsmogelijkheden.

SUMMARY

From September 1, 2014 till February 6, 2015 I conducted a graduation assignment for Colours. Colours is a full service internet agency in 's-Hertogenbosch. Developing successful digital concepts and associated websites is the specialty of Colours. These websites are launched through online marketing. Colours has a mixed team of designers, developers, project managers and marketeers.

For each website there is a style guide. Based on this style guide the website can be built, this is the .NET Solution. The HTML is copied from the style guide. This creates two HTML versions which are managed separately. Adjustments should be done twice. This takes unnecessary time. There may also arise differences in the HTML because Colours works with version control. Conflicts can arise when merging local branches.

To avoid these problems, there has been done research to find opportunities to use HTML in an efficient manner within a project, based on the starting point that developers don't have to do duplication work. Also front-end adjustments can be implemented faster to the .NET Solution.

I addressed the graduation project according to the Ten Steps Plan in conjunction with Scrum. During phase one, the orientation phase, an approach with corresponding planning is chosen. This is included in the Project Initiation Document (PID). During the investigation and solution phase (phase two) the problem is investigated by desk and field research. First, the best way to manage HTML is researched. Then the tools which are used by the Colours front-end developers are examined and used.

After that the .NET-based Content Management System (CMS) Sitecore is investigated and used. By adding MVC5, Sitecore was even investigated deeper. Subsequently, a decision is taken on the solution to be developed. Based on these findings I developed a proof of concept. This proof of concept is a custom view engine that demonstrates the possibility to use Mustache files to render HTML in the browser within a .NET Sitecore solution. The last phase is the implementation phase. Here is proven that HTML management in one place is possible. The custom view engine is presented to Colours and Fontys, to judge it.

The research has shown that it is better to generate front-end code within a CMS. This brings front-end and back-end development closer together. Also, it was found that Pattern Lab must be continued to use. The structured, generic design and generating static style guides make Pattern Lab a powerful tool. By setting up the back-end according to the MVC5 model, the translation between front-end and back-end is smaller. To make this translation more efficient there is chosen to develop a custom view engine. This saves one transformation step, and therefore less chance on errors. In addition, a view engine is more valuable for Colours considering the expansion possibilities.

WOORDENLIJST

B

Back-end
Het deel van een website dat op de server draait. De ingevoerde informatie wordt hier verwerkt en teruggekoppeld naar de gebruiker.

Bower
Een tool die automatisch Git repositories ophaalt en toevoegt aan een project.

F

Front-end
De presentatielaag van een website. Het gedeelte dat de gebruiker ziet en waar interactie mee mogelijk is.

G

Git
Een versiebeheersysteem.

Grunt
Een tool die andere tools aanstuurt aan de hand van taken die de gebruiker mee kan geven.

M

Model-view-controller (MVC)
Een ontwerppatroon waarbij de applicatie in drie delen wordt opgedeeld: datamodel, datapresentatie en applicatielogica.

P

Placeholder
Een term of vorm die gebruikt wordt als tijdelijke invulling totdat de definitieve data ingevuld wordt.

Project Initiatie Document (PID)
Het PID is het document dat in de beginfase van het project wordt opgesteld. Hierin worden onder andere de opdracht, randvoorwaarden en afspraken vastgelegd.

R

Renderen
Het genereren van data in de browser aan de hand van een datamodel.

Repository
Een opslagplaats om broncode op te slaan.

S

SASS
SASS is een uitbreiding op CSS. SASS maakt het gebruik van variabelen, nesten en mixins mogelijk.

Scrum
Een projectmethodiek waarbij een iteratieve, flexibele manier van werken wordt gehanteerd in een multidisciplinair team.

V

View engine
Zorgt ervoor dat de mix van HTML en logica in views gerenderd wordt in de browser.

W

Wireframe
De bouwtekening van een website. Een wireframe wordt tijdens de ontwerpfase gemaakt om aan te geven waar elementen geplaatst worden.

1.1 HUIDIGE SITUATIE

Dit onderzoek voer ik uit binnen het bedrijf Colours. Colours is een full service internetbureau. Hier wordt gewerkt volgens de Scrum methodiek.

Bij de start van een project zet de front-end developer een style guide op in HTML/CSS. Dit is een stand alone webpagina waarin alle elementen, kleuren, headings, et cetera voor die website worden opgenomen. Dit is opgebouwd uit losse modules volgens het Atomic Design principe waardoor de herbruikbaarheid groot is. Door deze style guide op te stellen ontstaat een vast uitgangspunt van de website voor de developers.

1.2 HET PROBLEEM

CSS en JavaScript wordt automatisch geïntegreerd in de .NET Solution. De .NET Solution is de daadwerkelijke website. De HTML wordt echter handmatig gekopieerd door een back-end developer vanuit de bestanden die de front-end developer eerder al gemaakt heeft. Er ontstaan op deze manier twee HTML versies welke apart beheerd worden. Het nadeel hiervan is dat veranderingen in de style guide niet doorgevoerd worden naar de .NET Solution en andersom. Dit moet twee keer met de hand gedaan worden wat veel tijd kost. Hierdoor wordt ook de foutgevoeligheid groter. Niet doorgevoerde updates in de .NET Solution zorgen ervoor dat de style guide achterhaald wordt en zijn meerwaarde verliest.

Ook kunnen er verschillen in de HTML ontstaan doordat er bij Colours met versiebeheer gewerkt wordt. Er is één centrale codebase van een project. Elk projectlid maakt hier een lokale kopie van (branch). In deze branch gaan zij dingen aanpassen. Later worden deze branches weer samengevoegd (mergen). Hierbij kunnen conflicten ontstaan wanneer er in de branches dingen verschillend zijn aangepast op dezelfde plaats. Voorbeeld: bij de ene branch is een stuk HTML aangepast. Bij de andere is dezelfde HTML code aangepast, maar dan anders. Bij het mergen moet dan gekozen worden welke versie er naar het samengevoegde project op de testserver gaat en wat doorgevoerd wordt naar de style guide.

Door dit onderzoek uit te voeren ga ik op zoek naar een oplossing om de workflow te verbeteren. De front-end en back-end developers kunnen dan efficiënter samenwerken doordat er geen of minder dubbel werk gedaan hoeft te worden. Daarnaast zijn aanpassingen in de front-end sneller door te voeren in de .NET Solution. Dit is nodig aangezien Colours de Scrum methodiek hanteert.

1.3 LEESWIJZER

In hoofdstuk twee wordt het bedrijf Colours beschreven. Hierin wordt onder andere de werkwijze beschreven. In hoofdstuk 3 wordt de opdracht gedefinieerd. Hieruit volgt het onderzoeksplan, hoofdstuk vier, waarin de hoofdvraag en deelvragen worden toegelicht. In hoofdstuk vijf worden deze hoofdvraag en deelvragen beantwoord. De huidige manier van front-end en back-end ontwikkelen wordt hier onder andere uitgelegd. De conclusies en aanbevelingen die hieruit volgen worden uitgelicht in hoofdstuk zes. Tot slot volgen hoofdstuk zeven, het oplossingsplan en hoofdstuk acht, het implementatieplan. Hierin wordt de uitwerking van de oplossing beschreven en hoe deze geïmplementeerd kan worden.

2.1 ORGANISATIE

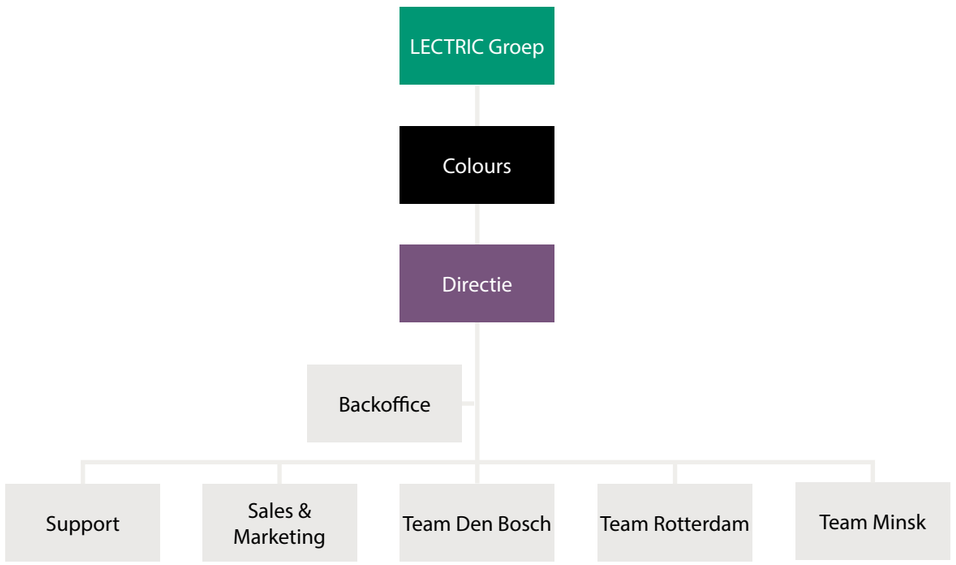
Colours is een vooraanstaand full service internetbureau in 's-Hertogenbosch. Samen met acht andere bedrijven vormen zij LECTRIC Groep. Een schematische weergave hiervan wordt weergegeven in figuur 1. Het ontwikkelen van succesvolle digitale concepten en daarbij websites is de specialiteit van Colours. Deze websites worden door middel van online marketing gelanceerd. Colours bestaat uit een gemixt team van designers, developers, projectmanagers en marketeers. Doordat Colours is aangesloten bij LECTRIC Groep kunnen ze ook de specialiteiten van de zusterbedrijven gebruiken. De afgelopen jaren hebben ze een mooi klantenbestand opgebouwd (figuur 2).

2.2 MISSIE

Colours is een full service online internetbureau dat de optimale balans tussen design en conversie biedt. Colours wil dan ook actief en concreet bijdragen aan het online succes van haar klanten, dit doen zij door het realiseren van succesvolle websites (Colours, z.d.).

2.3 VISIE

Colours is ervan overtuigd dat organisaties nog onvoldoende profiteren van de mogelijkheden uit het domein internet en technologie. De link tussen het effect van online interactie en de bijdrage daarvan aan business doelstellingen wordt nog nauwelijks gemaakt. In die transitie staat niet alleen de directe klant centraal maar moet er ook goed geluisterd worden naar de gebruikers. Colours ziet websites als levende 'dingen' die nooit af zijn en alsmaar veranderen, zij streven dan ook naar een proces van continu meten en verbeteren (Colours, z.d.).



Figuur 1: Bedrijfsstructuur van Colours.

2.4 WERKWIJZE

Het ontwikkelen van een website gebeurt bij Colours volgens de Scrum methodiek. In korte sprints worden werkende producten opgeleverd. Bij het ontwikkelen van een website staat de klant centraal. Voordat er ontwerpen gemaakt kunnen worden, wordt samen met de klant het doel en de strategie bepaald. Dit wordt gedaan aan de hand van de doelgroep, verwachtingen en de wensen. Vervolgens worden deze bevindingen omgezet in een slim concept. Dit concept wordt omgezet in ontwerpen volgens een duidelijke Fact Based aanpak door middel van Interaction design, visual design en functional design. Aan de hand van de gemaakte ontwerpen wordt de website gebouwd. Achter elke Colours website zit een Content Management Systeem (CMS). Dit kan Sitecore, Umbraco, Drupal, Smartsite of EPiServer zijn. Door middel van een CMS kan de klant zelf de website met content vullen maar ook Colours kan hier voor zorgen.

Nadat een website is gelanceerd biedt Colours nog ondersteuning. Bij incidenten kan altijd contact opgenomen worden en worden de problemen zo snel mogelijk verholpen. De website optimaliseren voor zoekmachines (SEO) en bijhouden en analyseren van het websitegebruik horen bij de krachten van Colours (Colours, z.d.).

Figuur 2: Een selectie van klanten.

2. BEDRIJF

In dit hoofdstuk wordt de projectdoelstelling besproken en de gekozen aanpak om deze doelstelling te kunnen halen. Als laatste worden de kaders van het project bepaald; de scope.

3.1 PROJECTDOELSTELLING

Zoals eerder beschreven is het dubbele beheer van HTML het probleem. De Colours developers zouden graag zien dat de HTML op één plek beheerd wordt. De back-end developers merken dat ze veel tijd kwijt zijn aan het kopiëren van geschreven HTML. Zij willen graag een geautomatiseerde oplossing maar hebben door het vele werk niet de tijd om dit te ontwikkelen.

Door dit project uit te voeren wordt er een efficiëntere samenwerking tussen de front-end en back-end developers mogelijk. De gehele workflow tussen de front-end en back-end developers moet vloeiender verlopen zonder dat er dubbel werk wordt gedaan. Bij de afronding van het project moet daarom een werkend proof of concept gepresenteerd kunnen worden waarmee aangetoond wordt dat HTML op één centrale locatie beheerd kan worden.

3.2 AANPAK

Ik heb ervoor gekozen om het Tien Stappen Plan (TSP) te gebruiken in combinatie met Scrum. Dit heb ik gekozen omdat TSP een beproefde manier van werken is binnen een afstudeerproject. Voor de student ontstaat een helder beeld doordat het project gefaseerd aangepakt wordt en voor de begeleider is het mogelijk om effectiever te begeleiden. De combinatie met Scrum is gekozen omdat op deze manier in korte tijd werkende (deel)producten opgeleverd worden waardoor bijsturing door de opdrachtgever makkelijker is. Daarnaast is de kans groter dat er aan het einde van de rit een voor Colours bruikbare oplossing ligt, blijven de developers betrokken en is Scrum de werkwijze die Colours zelf hanteert.

In de oriëntatiefase ga ik het probleem verder onderzoeken. Dit wordt gedaan door gesprekken te voeren met front-end en back-end developers. Hier wordt geen gebruik gemaakt van een vooraf opgestelde vragenlijst om de gesprekken spontaan te houden. Daarnaast wordt er gekeken naar het globale proces dat een website bij Colours doorloopt in de development fase en welke tools hierbij gebruikt worden. Dit wordt gedaan door middel van desk research en demo's van de tools. Deze bevindingen vormen uiteindelijk het PID (bijlage I).

Op de oriëntatiefase volgt fase twee: de onderzoeks- en oplossingsfase. De informatie die ik verzameld heb tijdens de oriëntatiefase analyseer en verwerk ik. Aan de hand van het PID kan het onderzoeksplan opgesteld worden. Hierin worden de hoofdvraag en deelvragen beschreven en op welke manier het onderzoek aangepakt wordt. Dit wordt gedaan tijdens sprint 0, de voorbereidingssprint. Dan volgt de uitvoer van het onderzoek. Dit begint in sprint 1. De focus van dit onderzoek ligt op de huidige workflow aan de developers kant bij Colours en het beheren van geschreven HTML code. Om antwoorden te vinden onderzoek ik eerst de beste manier om code te beheren. Vervolgens worden de tools die front-end developers bij Colours al gebruiken geleerd, gebruikt en beschreven. Hierna wordt ook het .NET-based Content Management Systeem (CMS) Sitecore geleerd en

Figuur 3: Schematische weergave van de aanpak.

gebruikt. Dit gebeurt met de hulp van Colours developers in sprint 2. Tijdens sprint 3 wordt er verdiept in Sitecore. Ik ga hier een website opzetten volgens het MVC5-model. Deze drie sprints vormen het onderzoek. De resultaten hiervan zijn te lezen in hoofdstuk 5. Daarna volgt sprint 4. Hier ligt de focus op de stap tussen de front-end developing (sprint 1) en de back-end developing (sprint 2 en 3). Hier ga ik de softwarematige oplossing ontwikkelen die ervoor zorgt dat de HTML code op één plaats beheerd kan worden. De conclusies en functies die zijn vastgesteld worden hierin verwerkt. Deze oplossing wordt opgeleverd in de vorm van een proof of concept. Om het bruikbaar te houden voor de toekomst wordt er een implementatieplan geschreven waarin staat hoe het gebruikt kan worden en waar op gelet moet worden.

Tijdens een sprint worden telkens dezelfde stappen doorlopen. Er wordt eerst een stukje onderzoek gedaan. Met deze verkregen kennis wordt een afgebakend deel gerealiseerd dat toewerkt naar het eindproduct. Dit wordt 'on the fly' getest en vervolgens gepresenteerd door middel van een sprintdemo. Deze demo en de documentatie voor de scriptie zorgen voor de evaluatie.

De laatste fase is de invoeringsfase. In deze fase wordt door middel van een eindpresentatie en demo het proof of concept gepresenteerd en kan beslist worden wat Colours ermee doet. Het proof of concept kan vanaf dat moment verder ontwikkeld worden. Een schematische weergave is te zien in figuur 3.

3.3 SCOPE VAN HET PROJECT

Dit project draait om het beheren van de HTML op één centrale plaats. Om dit te bereiken doe ik onderzoek naar een mogelijke oplossing. Tijdens de oriëntatie is onderzocht of een softwarematige of procesmatige oplossing op zijn plaats is. Een procesmatige oplossing zou kunnen zijn dat alleen front-end developers aanpassingen maken in de HTML of front-end developers gaan zelf de HTML omzetten in de back-end. Hier heb ik echter niet voor gekozen omdat Colours met de oplossing back-end en front-end development dichter naar elkaar toe wil brengen. Bij de genoemde aanpakken wordt de scheiding nog groter. Ook is het ontwikkelen van een werkend ICT proof of concept een randvoorwaarde. Dit is bij genoemde oplossingen niet mogelijk. Daarom wordt een softwarematige oplossing onderzocht. Hiervoor zijn in eerste instantie twee oplossingen mogelijk: de HTML bron wordt binnen of buiten het CMS gegenereerd. Vervolgens worden de huidige tools die Colours gebruikt onderzocht. Dit is onder andere Pattern Lab, de bron van de ontwikkeling. Daarnaast zijn de Content Management Systemen belangrijk. Het zwaartepunt ligt op de conversie tussen Pattern Lab en CMS aangezien dit de meeste business value heeft voor Colours. Door tijdens het onderzoek ook rekening te houden met de complexiteit van het visual design is het van meerwaarde dat ik als ICT & Media Design (IMD) afstudeerstagiair dit onderzoek uitvoer. Ook mijn kennis van front-end én back-end is van meerwaarde aangezien een IMD'er met de sterktes en beperkingen van beide disciplines rekening kan houden.

Om een duidelijk beeld te krijgen van het te onderzoeken probleem heb ik een hoofdvraag met bijbehorende deelvragen opgesteld. Om de hoofdvraag te kunnen beantwoorden dienen eerst de deelvragen beantwoord te worden. Wat deze hoofdvraag en deelvragen inhouden wordt in dit hoofdstuk kort beschreven met bijbehorende onderzoeksmethode.

4.1 HOOFDVRAAG

"Op welke manier kan HTML code op één centrale plaats beheerd worden binnen Colours?"
Het antwoord op deze vraag moet leiden tot een oplossing waarmee het probleem van dubbel beheer aangepakt kan worden.

4.2 DEELVRAGEN

"Wat is de meest geschikte plaats om de front-end code van een website te beheren?"
Door deze vraag te beantwoorden moet blijken of het beter is om HTML binnen of buiten het CMS te beheren.

"Op welke manier wordt de front-end van een website ontwikkeld binnen Colours?"
Om deze vraag te beantwoorden ga ik de tools installeren en gebruiken die Colours al gebruikt. Hierbij let ik op de voor- en nadelen en zet ik een testomgeving op. Dit is het beginpunt van het probleem.

"Op welke manier wordt de back-end van een website ontwikkeld binnen Colours?"
Om dit te onderzoeken installeer ik Sitecore en maak ik een Sitecore website volgens het MVC5-model. Dit is het eindpunt van het probleem.

"Wat is de beste manier om de vertaalslag van front-end naar back-end te automatiseren?"
Deze vraag beantwoord ik door de voor- en nadelen van de mogelijke oplossingen af te wegen waaruit de beste keuze voor de oplossing naar voren komt.

4.3 ONDERZOEKSMETHODES

Ik heb gekozen voor de onderzoeksmethodes werkplaats, bieb en veld. Om globale informatie te verzamelen gebruik ik de methode 'bieb'. Dit houdt in dat er desk research gedaan wordt naar aanleiding van het vastgestelde probleem. Om Colours specifieke informatie te verkrijgen worden er gesprekken met developers gehouden. Deze gesprekken vallen onder de methode 'veld'. Onder de term 'werkplaats' valt het opzetten van de huidige tools die Colours gebruikt, Sitecore en (test)uitvoeringen van de oplossing. Hier heb ik voor gekozen omdat deelvraag twee en drie het best te beantwoorden zijn door te doen. Door veel te proberen, testen en weer verder te ontwikkelen kan ik veel kennis opdoen.

In dit hoofdstuk wordt antwoord gegeven op de hoofdvraag:

"Op welke manier kan HTML code op één centrale plaats beheerd worden binnen Colours?"

Dit wordt gedaan door vier deelvragen te behandelen en hieruit vervolgens conclusies te trekken.

5.1 "WAT IS DE MEEST GESCHIKTE PLAATS OM DE FRONT-END CODE VAN EEN WEBSITE TE BEHEREN?"

De ontwikkeling van een website wordt binnen webdevelopment in het algemeen onderverdeeld in front-end en back-end. Front-end wordt ook wel omschreven als de presentatielaag. Het is het gedeelte dat de gebruiker ziet en waar interactie mee mogelijk is. Dit is opgebouwd in HTML, CSS en JavaScript (Treehouse, 2012). Zonder back-end kan een gebruiker niet veel met de front-end. Wanneer er informatie ingevoerd wordt, wordt dit verwerkt door de back-end. Informatie wordt naar een database verzonden en informatie wordt uit een database gehaald. Back-end kan in meerdere talen geprogrammeerd worden. Voorbeelden hiervan zijn PHP, C# en Ruby.

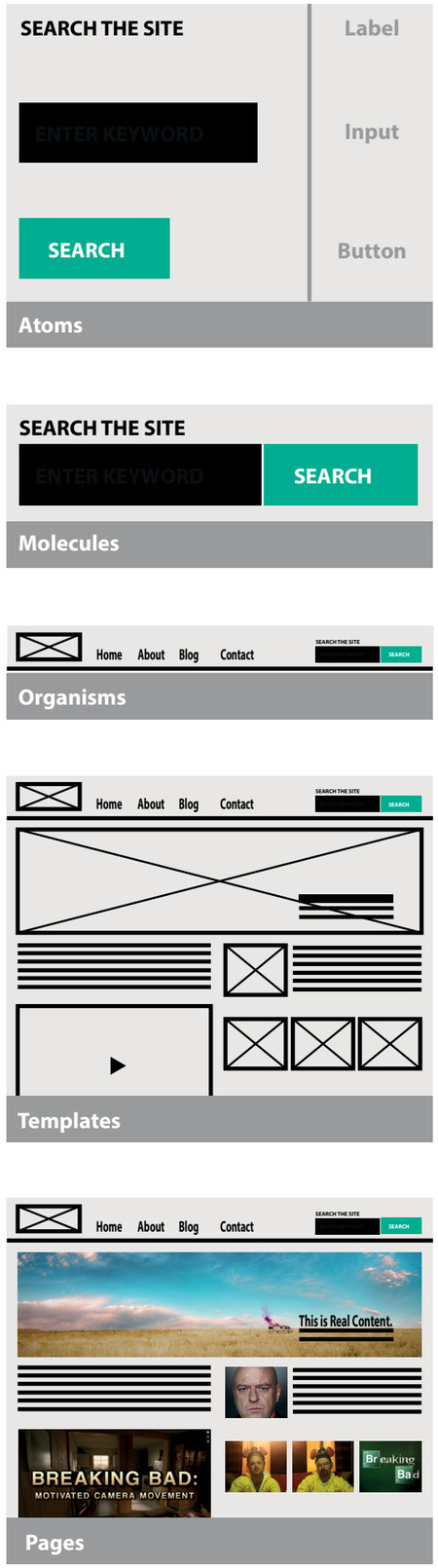
Deze twee segmenten samen vormen dus een website. Maar hier komt vaak nog een laag bij; een CMS. Door middel van een CMS kunnen content editors de website bijwerken (updaten van de content) zonder dat hier een developer aan te pas hoeft te komen. Deze kan de website dus anders indelen, foto's, video's en teksten toevoegen.

STYLE GUIDE

Om tijdens de ontwikkeling de gebruikersinterface, de front-end dus, te scheiden van de back-end is sinds 2012 Style Guide Driven Development in opkomst. Door deze scheiding zijn de front-end en back-end niet van elkaar afhankelijk waardoor snelle prototypes van de interface te ontwikkelen zijn (Lewis, J. 2014). Door zo een style guide op te zetten kunnen er interfaces ontwikkeld worden zonder dat de back-end helemaal af is. Daarnaast kunnen vaste afspraken gemaakt worden door de style guide als uitgangspunt te nemen bij het ontwikkelen van de website.

In deze style guide komen alle elementen die op de website komen met hierbij een code voorbeeld en eventueel een stukje tekst met informatie hoe en wanneer het element gebruikt dient te worden. Om dit makkelijker te maken is er een tool genaamd Pattern Lab. Pattern Lab is een tool waarmee systemen volgens het Atomic Design principe gebouwd kunnen worden. Dit principe leunt op vijf bouwstenen: atoms, molecules, organisms, templates en pages. Voorbeelden hiervan zijn te zien in figuur 4. Pattern Lab maakt gebruik van Mustache. Mustache kan HTML genereren aan de hand van meegegeven 'expressions'. Hierdoor kunnen stand alone componenten gemaakt worden die herbruikbaar zijn. Pattern Lab zorgt ervoor dat deze losse modules samengevoegd worden tot één HTML pagina. Er wordt een statische website gegenereerd.

Ook bij Colours hebben ze zich deze manier van werken toegeëigend. Developers merken echter dat ze vaak dubbel werk moeten doen door HTML code te kopiëren van de style guide naar de back-end. Daarom ontstaat de vraag: kan het efficiënter? Kan de bron, de HTML code, binnen de CMS omgeving beheerd worden? Hierdoor zou geen of minder kopieerwerk plaats hoeven vinden. Of is het toch beter om dit helemaal gescheiden te houden van het CMS?



Figuur 4: Atomic Design elementen.

Het gebruik van levende style guides en pattern libraries is in deze tijd een beproefde aanpak. Bekende front-end developers als Brad Frost, Nicole Sullivan en Nico Hagenburger zweren bij deze aanpak. Daarnaast is Pattern Lab volgens Brad Frost, de bedenker en ontwikkelaar van het Atomic Design principe en Pattern Lab, een goede tool om te helpen de front-end code in een CMS te integreren. Omdat de Mustache code in te zien is wordt duidelijk welke code dynamisch is en welke statisch. Dit is al beter dan platte HTML/CSS opleveren. Daarom, en omdat Colours al succesvol gebruik maakt van Pattern Lab, kies ik ervoor dit te gebruiken als bron van de ontwikkeling.

Een style guide wordt in veel gevallen los van het CMS gegenereerd en aangepast, wat Pattern Lab dus goed doet. Maar een style guide generen zou echter ook binnen het CMS kunnen. Hiervoor hoeft Pattern Lab niet eens te wijken.

5. ONDERZOEK

VOORDELEN
Het genereren binnen het CMS heeft een aantal voordelen ten opzichten van erbuiten.
Zo hoeft de HTML niet voor een tweede keer bewerkt te worden maar wordt de HTML op één plaats ontwikkeld en gebruikt op twee plaatsen. De HTML wordt zo één-op-één overgezet. Daarnaast worden front-end en back-end development dichter bij elkaar gebracht. Iets wat Colours graag wil zien.

NADELEN
Maar er zijn ook nadelen. Zo zijn er geen kant-en-klare tools of plug-ins beschikbaar die style guides genereren binnen een CMS. Dit zou 'from scratch' ontwikkeld moeten worden. Ook kost het maken van aanpassingen voor front-enders meer tijd. Builden van CMS code duurt aanzienlijk langer dan het builden van Pattern Lab.

KANS
Pattern Lab exporteert nog niet automatisch HTML templates naar CMS templates. Ze zijn hier wel mee bezig (Brad Frost, 2014). Maar op dit moment, november 2014, is dit nog niet gerealiseerd. Vanuit verschillende ontwikkelaars is hier ook vraag naar gezien de issues in de Pattern Lab Git repository. Zij wensen een rechtstreekse export naar een CMS of hebben hieraan gerelateerde wensen. Hier ligt dus een kans om op in te spelen.

5.2 "OP WELKE MANIER WORDT DE FRONT-END VAN EEN WEBSITE ONTWIKKELD BINNEN COLOURS?"

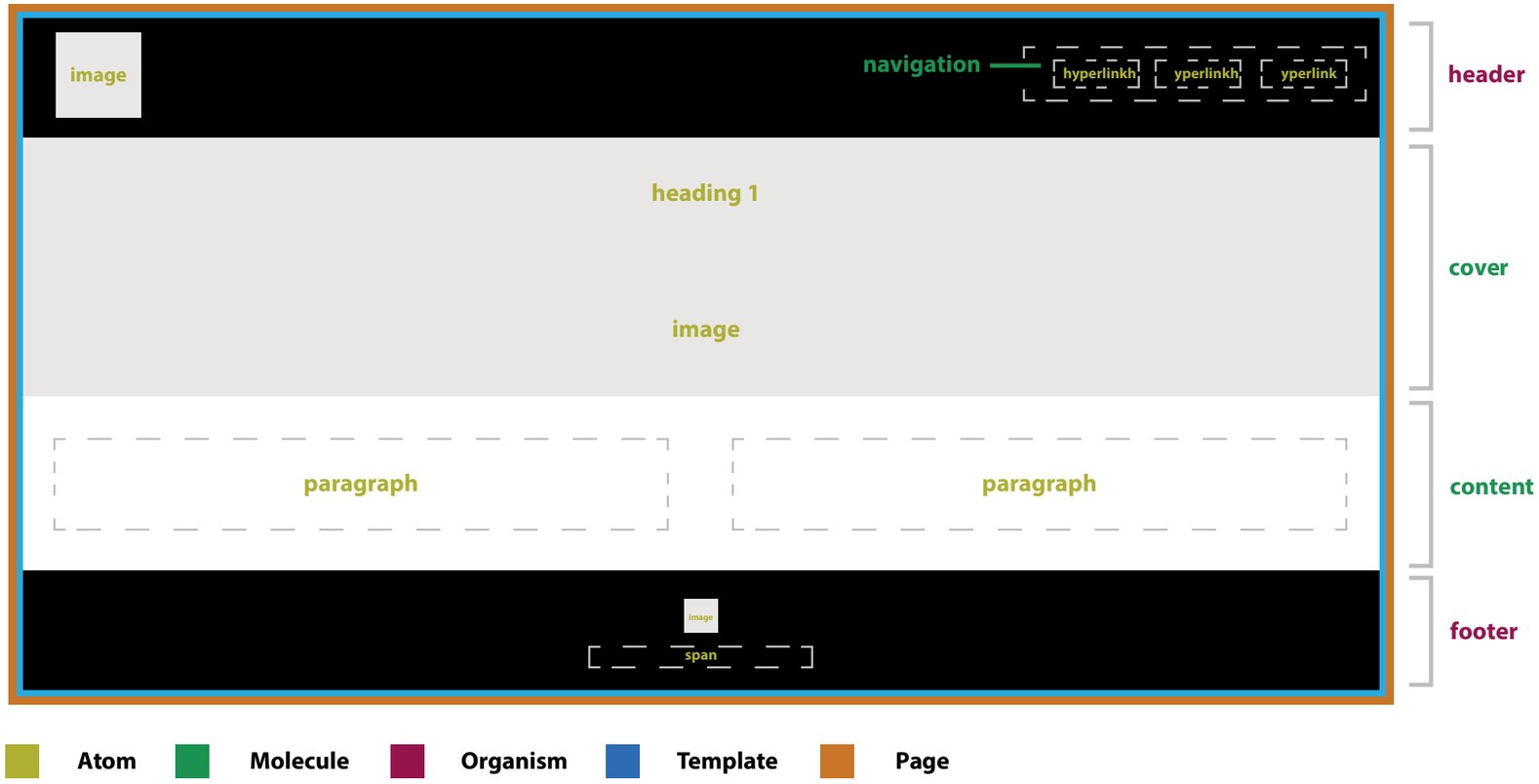
Het is dus bekend dat de HTML code die de front-enders ontwikkelen efficiënter moet worden gebruikt dan nu het geval is. Dit is het startpunt voor de oplossing die moet gaan komen. Het eindpunt is in dit geval de back-end code in het CMS. Zoals te lezen is in hoofdstuk 3.3 moet een softwarematige vertaalslag plaats gaan vinden tussen deze twee stappen. Om te weten wat er precies vertaald moet worden moet het startpunt zijn opgezet zoals dit bij Colours het geval is. De front-end developers bij Colours maken gebruik van de tools Pattern Lab, SASS, Bower en Grunt.

PATTERN LAB

Pattern Lab is beschikbaar op verschillende platformen en allemaal open source. Echter, de PHP versie wordt het meest actief doorontwikkeld en blijft dus up-to-date. Bij Colours wordt deze PHP versie gebruikt gezien de stabiliteit van het platform. Wat Pattern Lab doet en hoe dit is opgezet wordt uitgelegd in hoofdstuk 5.1.

SASS

Een tweede tool die Colours gebruikt is SASS. SASS is een CSS preprocessor en zorgt ervoor dat mogelijkheden van CSS groter worden. LESS is ook zo een CSS preprocessor. De voordelen van SASS zijn dat het robuuster is, minder gekopieerde code oplevert, er een scheiding is tussen 'gewone' en responsive code en een betere rekentechniek heeft (Coyler, C. 2014). Ook Colours maakt gebruik van SASS in plaats van LESS gezien de voordelen.



Figuur 5: Ontwerp Pattern Lab testopzet.

BOWER

Een ander belangrijk onderdeel van websites zijn de libraries en frameworks die in veel gevallen worden gebruikt. Dit is waar Bower een helpende hand biedt. Bower zorgt ervoor dat git repositories met libraries of frameworks automatisch worden opgehaald en toegevoegd aan het project. De voordelen van Bower zijn dat alle beschikbare packages zijn verzameld op de Bower website, de leercurve laag is en het makkelijk is om nieuwe packages te publiceren (Reiz, R. 2014). Aangezien het handmatig toevoegen en up-to-date houden een hele klus is bij grote corporate websites gebruikt Colours deze tool ook.

GRUNT

Om alle eerder genoemde tools aan te sturen maakt Colours gebruik van Grunt. Grunt zorgt ervoor dat bestanden worden geminified en gecompiled. Het grote voordeel van Grunt zijn de 3.638 plug-ins, wat duidt op een grote community. Gulp is eenzelfde soort tool waarvoor echter maar 779 plug-ins te downloaden zijn. Daar moet wel bij vermeld worden dat Grunt een jaar langer bestaat dan Gulp (Blanchard, N. 2014).

PATTERN LAB TESTOPZET

Om zelf te onderzoeken hoe Colours de front-end opzet heb ik bovenstaande tools geïnstalleerd. Vervolgens heb ik me ingewerkt met deze tools om daarna een eigen website op te kunnen zetten. Hiervoor is een schets gemaakt met elementen die de website bevat. Zie hiervoor figuur 5.

Deze schets ben ik vervolgens om gaan zetten naar een daadwerkelijke Pattern Lab website. De elementen heb ik opgezet volgens het Atomic Design principe. Ook de bijbehorende CSS is zo opgezet. Iedere module heeft zijn eigen stukje CSS wat samengevoegd wordt in één CSS bestand.

De data voor de atoms tot en met de templates komt uit een JSON file; _data.json. Deze data wordt gebruikt om dummy content in te kunnen laden voor de website. Hier wordt geen gebruik van gemaakt wanneer de HTML naar de back-end gaat. Voor een page, homepage in dit geval, komt de data uit 00-homepage.json. De data kan overschreven worden wanneer de pagina-specifieke JSON file dezelfde keys heeft maar met andere values. Een voorbeeld hiervan is te zien in figuur 6. Daardoor krijgt de website het uiterlijk zoals het ook ontworpen is.

De resultaten van mijn testopzet zijn te zien in figuur 7 tot en met 10.

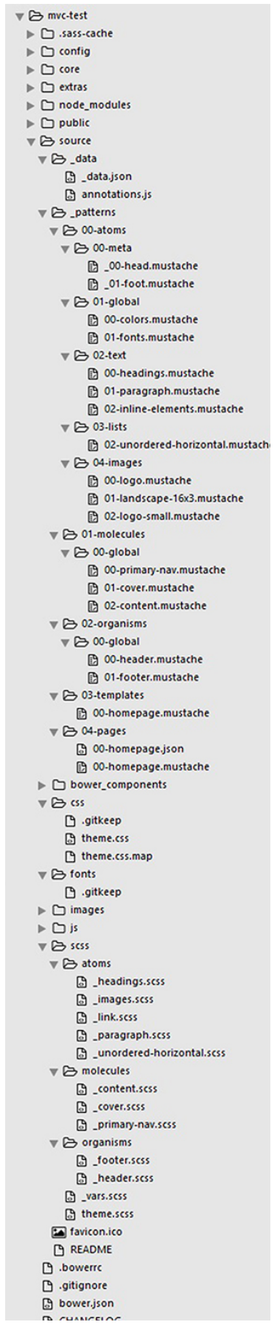
_data.json

```
{
        "menuItem" : {
                "firstItem"     :"Link 1",
                "secondtem"     :"Link 2",
                "thirdItem"     :"Link 3"
        }
}
```

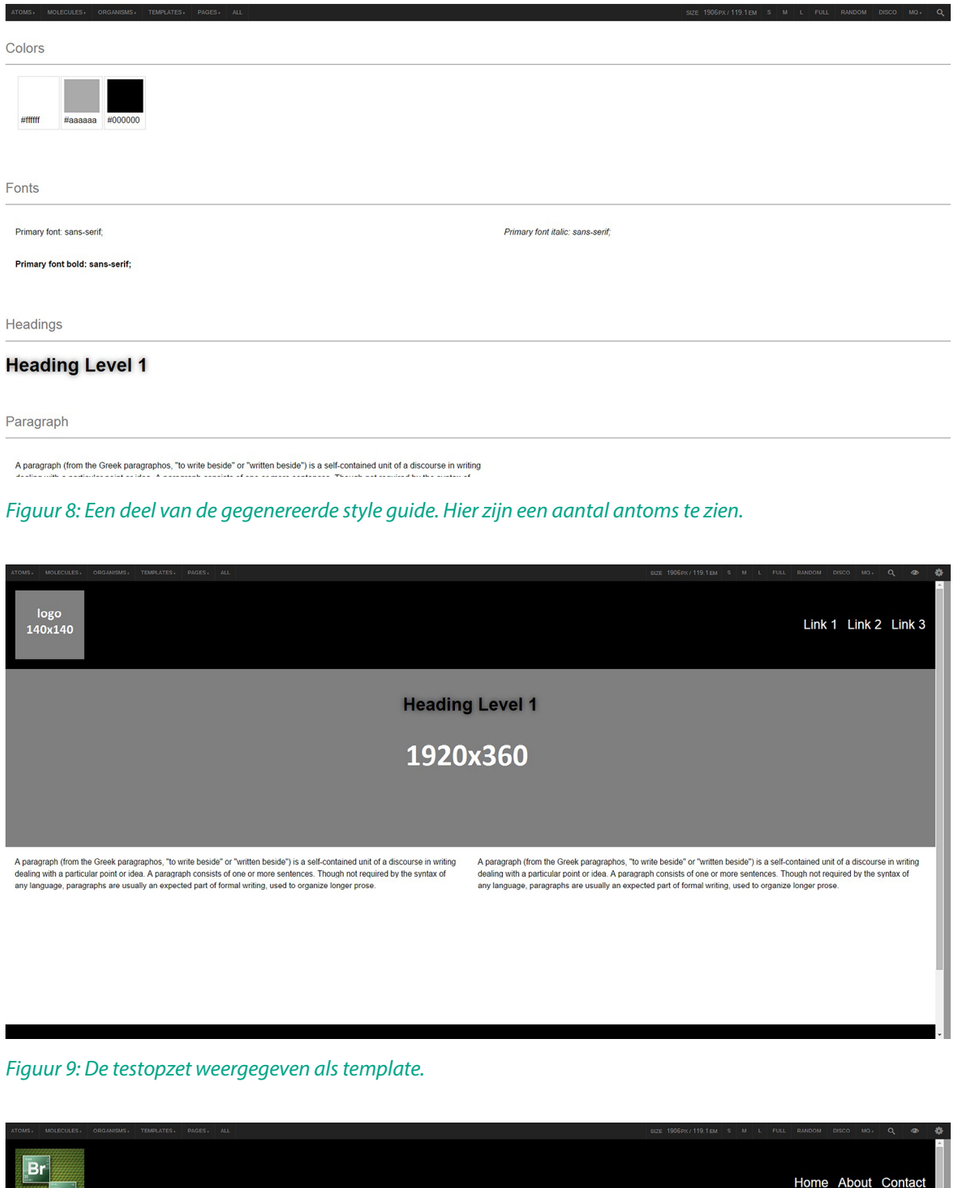
00-homepage.json

```
{
        "menuItem" : {
                "firstItem"     :"Home",
                "secondtem"     :"About",
                "thirdItem"     :"Contact"
        }
}
```

Figuur 6: Overschrijven van JSON data.

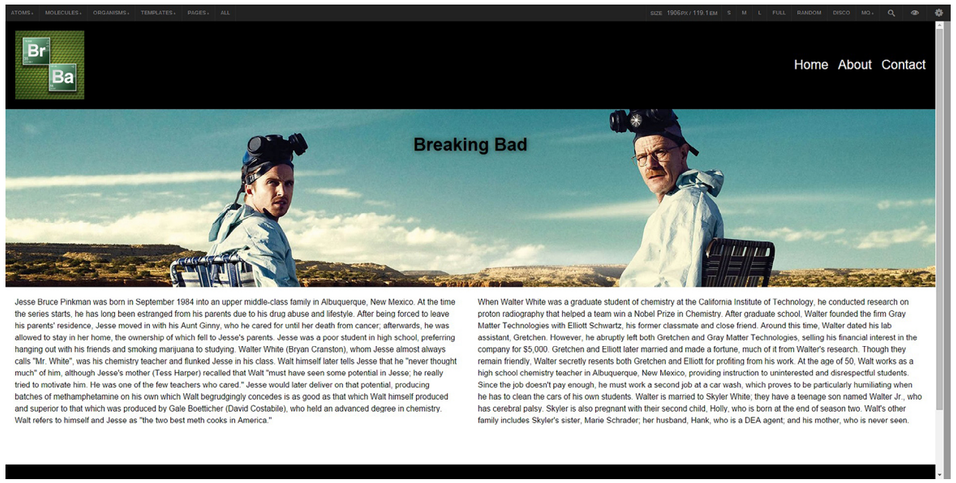


Figuur 7: De structuur van de testopzet



Figuur 8: Een deel van de gegenereerde style guide. Hier zijn een aantal antoms te zien.

Figuur 9: De testopzet weergegeven als template.



Figuur 10: De testopzet weergegeven als page.

5.3 “OP WELKE MANIER WORDT DE BACK-END VAN EEN WEBSITE ONTWIKKELD BINNEN COLOURS?”

Naast de front-end is ook de back-end een essentieel onderdeel van de te ontwikkelen oplossing. De back-end bestaat bij Colours altijd uit een met de klant bepaald CMS. Dit kan Sitecore, Umbraco, EPiServer, Smartsite of Drupal zijn. Hiervan is Sitecore het CMS waar Colours het meest mee werkt. Daarom heb ik ervoor gekozen om tijdens dit onderzoek te focussen op een oplossing voor Sitecore (.NET platform).

BENODIGDE TOOLS

Voordat een Sitecore website ontwikkeld kan worden moeten er eerst een aantal tools geïnstalleerd worden. De eerste is Internet Information Services (IIS). IIS is een serverdienst waarmee een computer gebruikt kan worden als webserver. Hierdoor kan een website gehost worden. De tweede is het .NET framework. Dit framework zorgt ervoor dat applicaties in verschillende programmeertalen met elkaar kunnen samenwerken. De derde is SQL Server. Dit is een beheersysteem voor databases. In de database worden alle componenten van Sitecore opgeslagen. Visual Studio 2012, tot slot, is de software waarmee C#.NET geprogrammeerd wordt bij Colours. Alle genoemde tools zijn ontwikkeld door Microsoft.

Dan volgt de Sitecore installatie. Het installeren van Sitecore houdt in dat er één Sitecore instantie wordt aangemaakt. Wanneer er een nieuwe Sitecore instantie aangemaakt wordt, wordt dezelfde stap doorlopen. Bij het verwijderen van een instantie wordt een de-installatie gedaan. Voor dit onderzoek is de nieuwste versie van Sitecore, versie 7.5, geïnstalleerd. Dit heb ik gedaan omdat de oplossing met nieuwe technieken zoals MVC5 moet gaan samenwerken.

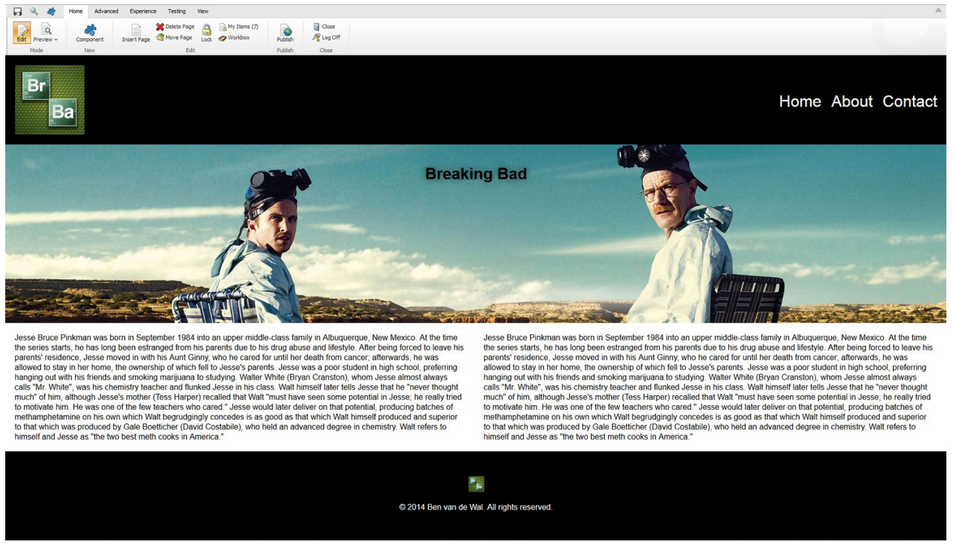
SITECORE MVC5 TESTOPZET

Om een goed inzicht te krijgen in een Sitecore MVC5 website heb ik een testopzet gemaakt. Ik heb ervoor gekozen om hier de website te gebruiken die ik tijdens het onderzoeken van Pattern Lab heb opgezet. Deze website is opgebouwd uit simpele elementen zodat het makkelijk op te zetten en te testen is.

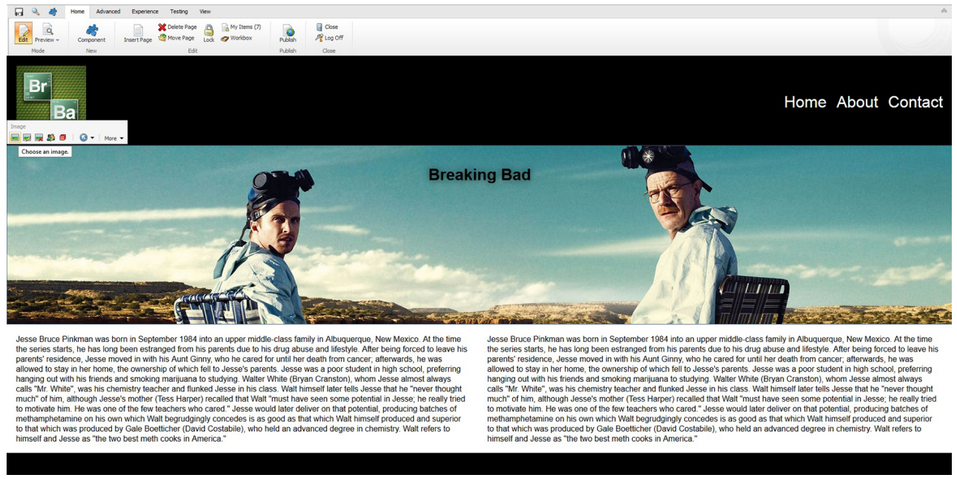
De hele website heb ik handmatig omgezet naar een Sitecore website. Dat betekent dat de content nu beheerbaar is binnen Sitecore. In principe is zo dezelfde vertaalslag gemaakt als de oplossing moet gaan doen maar dan handmatig. De Atomic Design structuur is hierbij zo goed als mogelijk overgenomen. Hier en daar hebben er kleine verschuivingen plaats gevonden.

Hiervoor is eerst een schone Sitecore instantie geïnstalleerd. Vervolgens is een MVC5 .NET Solution aangemaakt. Deze solution maakt gebruik van de Razor view engine. Een view engine zorgt ervoor dat de mix van HTML en logica in views gerenderd wordt in de browser.

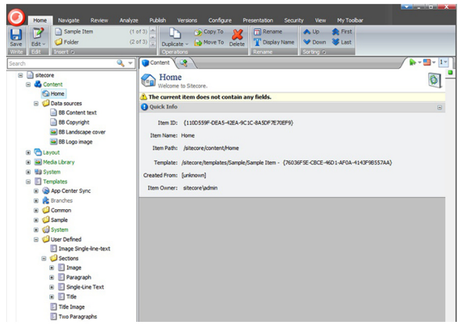
Voordat er iets gerenderd wordt moet de website goed opgebouwd zijn in Sitecore.
Door middel van figuur 11 wordt uitgelegd hoe de opbouw van pagina’s dynamisch gehouden kan worden.



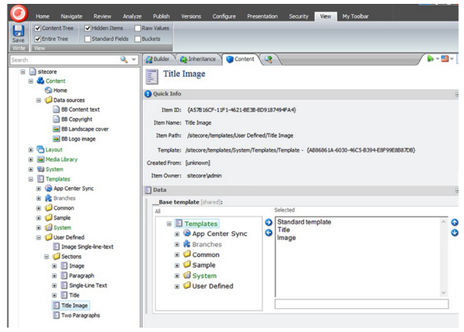
Figuur 12: Sitecore website. Er is ingelogd als content editor.



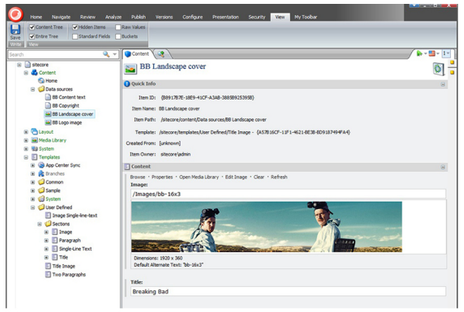
Figuur 13: Sitecore website. De content editor kan de inhoud van de ‘Cover’ moncecule aanpassen.



Stap 1.

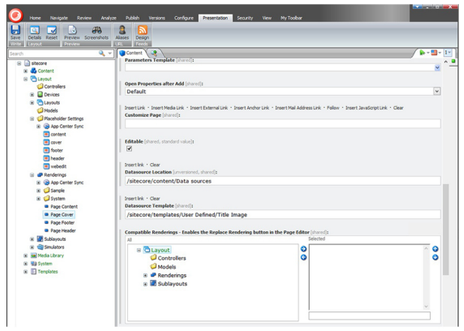


Stap 2.

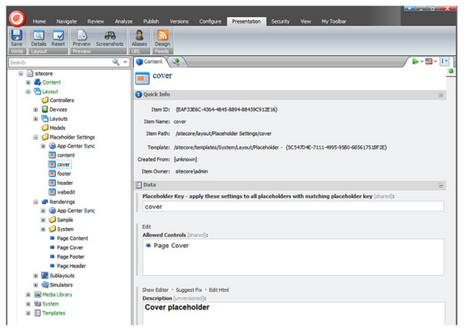


Stap 3.

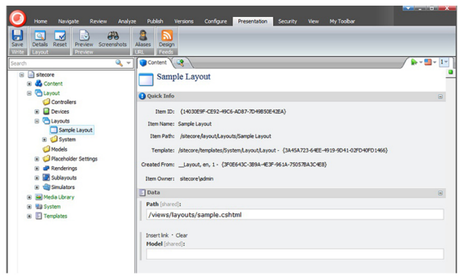
Dit stappenplan geldt voor alle componenten binnen een Sitecore website. Wanneer alles gerenderd is in de browser, en de content editor logt in, ziet de website eruit zoals in figuur 12 en 13.



Stap 4.



Stap 5.



Stap 6.

Figuur 11: Een schematische weergave van de dynamische opbouw van een Sitecore website.

Stap 1. De pagina, Home in dit geval, is gebaseerd op een lege template. Hier worden geen veldtypes meegegeven.

Stap 2. De opzet van de velden wordt dynamisch gemaakt door de componenten aan te maken in de User Defined folder onder 'Templates'. Deze zijn abstract opgezet, zoals te zien is aan de naamgeving. Aan elke template wordt meegegeven hoe ze zijn opgebouwd. In figuur 11 bevat 'Title Image' een Title veld en Image veld. Er zijn nu dus lege componenten die met eigen content gevuld kunnen worden.

Stap 3. Onder Content wordt een folder aangemaakt genaamd 'Data sources'. Hierin komt alle content voor de webpagina. Deze content wordt gebaseerd op de component templates die in stap 2 aangemaakt zijn. In het geval van Title Image wordt de url naar de afbeelding en de tekst voor de titel ingevuld.

Stap 4. De component templates moeten ook gerenderd kunnen worden. Daarom worden controller renderings aangemaakt gebaseerd op die componenten. De controls zijn de controller renderings. Hier wordt het pad naar de data sources meegegeven. Dit kan als pad of als id. De voorkeur gaat uit naar id aangezien het item zo altijd gevonden kan worden (Vidal, B. 2012).

Stap 5. Hier wordt de placeholder gedefinieerd. De wordt later in de back-end code aangeroepen. Op de plaats van de placeholder wordt het component gerenderd.

Stap 6. Het renderen wordt in de main lay-out gedaan. Het pad naar deze lay-out file wordt hier gedefinieerd.

5. ONDERZOEK

5.4 "WAT IS DE BESTE MANIER OM DE VERTAALSLAG VAN FRONT-END NAAR BACK-END TE AUTOMATISEREN?"

Nu het start- en eindpunt bekend zijn ga ik bekijken hoe de tussenliggende fase eruit moet gaan zien: de vertaalslag van front-end naar back-end. Er zijn al diverse tools die Colours gebruikt tijdens de ontwikkeling van een website. De geautomatiseerde transformatie moet goed samenwerken met deze tools. Maar in wat voor vorm wordt deze vertaalslag gegoten? Met de resultaten van de vorige deelvragen in het achterhoofd zijn er twee mogelijkheden: een transformatietool (Grunt plug-in) of een custom view engine (.NET-based).

VEREISTEN

Aan de oplossing zijn een aantal vereisten gesteld.

Past binnen huidige Colours tools
Tijdens de ontwikkelfase van een website worden bij Colours al diverse tools gebruikt om de workflow te optimaliseren. De tool voor de transformatie moet uiteraard naadloos aansluiten op deze tools anders ontstaat er een de-optimalisatie van de workflow wat natuurlijk niet de bedoeling is.

Vaak herhalende taak automatiseren
Het omzetten van de Mustache bestanden kan een vaak herhalende taak zijn. Bij elke aanpassing die gedaan wordt in de front-end code moet die verandering ook in de back-end code zichtbaar zijn en andersom. Dit moet dus geautomatiseerd gebeuren omdat het handmatig teveel moeite zou kosten.

IDE en teksteditor onafhankelijk
De oplossing heeft de meeste waarde wanneer deze aanpassingen in Visual Studio automatisch doorvoert naar de front-end en andersom. Gezien de verschillende platformen moet de oplossing integrated development environment (IDE) en teksteditor onafhankelijk zijn.

GRUNT PLUG-IN

Hieronder de voordelen van een Grunt plug-in op een rijtje. Deze voordelen heb ik geconstateerd tijdens het opzetten van de front-end tools.

Gelijk met genereren style guide
Bij elke 'save' binnen Pattern Lab wordt de style guide opnieuw gegenereerd. Grunt stuurt dit aan. Daarom is dit het meest logische moment om ook de transformatie naar back-end plaats te laten vinden. Op deze manier lopen de aanpassingen synchroon.

Taken
De transformatie is een taak die uitgevoerd moet worden. Tijdens de ontwikkeling voert Grunt al een aantal taken uit. Dit zijn onder andere het compilen en minifyen van bestanden. Het transformeren van bestanden past precies in dit rijtje thuis.

Lokaal draaien
De transformatie wordt uitgevoerd wanneer een developer in een lokale kopie (branch) werkt. Hierdoor kunnen er lokaal geen zogeheten merge conflicten ontstaan. Deze conflicten ontstaan pas wanneer de branch weer via Git gemerged wordt met het project.

Maar een Grunt plug-in heeft ook nadelen.

Tweezijdige transformatie
Wanneer een Grunt plug-in ontwikkeld wordt moet de transformatie naar beide kanten opgezet worden. Doordat er twee keer getransformeerd moet worden is de kans groter dat er fouten optreden.

JavaScript
Een Grunt plug-in wordt geschreven in JavaScript. Dit is geen elegante programmeertaal en is daarnaast lastig te debuggen.

VIEW ENGINE

Naast een transformatietool behoort ook een view engine tot de mogelijkheden. De voordelen van een view engine op een rijtje.

Geen conversie
De Mustache files kunnen rechtstreeks gebruikt worden in het project. De files hoeven niet geconverteerd te worden naar een andere extensie. Daardoor wordt de front-end code één op één overgenomen en zullen minder fouten optreden.

Goed te onderhouden
Een view engine is goed te onderhouden. De basiswerking is onderverdeeld in verschillende functies. Wanneer de view engine uitgebreid dient te worden kunnen er functies bijgeschreven worden.

Gelijk met genereren style guide
Bij elke 'save' binnen Pattern Lab wordt de style guide opnieuw gegenereerd. Aangezien dezelfde files in de .NET omgeving gebruikt worden kunnen er geen versieverschillen ontstaan.

Lokaal draaien
Voor de view engine geldt hier hetzelfde voordeel als de Grunt plug-in.

De custom view engine heeft eigenlijk maar één nadeel.

Weinig voorbeelden
Er zijn veel simpele basisopzetjes voor een custom view engine te vinden. Maar dan houdt het vaak op. Zelf een view engine bouwen vergt dus veel uitzoekwerk gezien de beperkte informatie die voorhanden is.

6. CONCLUSIES & ADVIES

Na het onderzoek uitgevoerd te hebben trek ik de volgende conclusies.

Het is beter om de front-end code binnen het CMS te genereren. De redenen hiervan zijn dat de HTML op deze manier één-op-één overgenomen wordt en front-end en back-end development dichter bij elkaar komen. Ook is er een kans om een veelgevraagde nieuwe aanvulling op Pattern Lab te ontwikkelen.

Om de front-end te ontwikkelen is Pattern Lab een geschikte tool. Het Atomic Design principe is een mooie opzet waardoor er meer gestructureerd gewerkt kan worden. Het gevolg hiervan is dat er minder dubbele code is. Door het automatisch laten genereren van de style guide zijn veranderingen in de front-end snel voor iedereen zichtbaar.

Sitecore is op het eerste gezicht een erg groot CMS. Wanneer een gebruiker zich hier in verdiept blijkt dit ook zo te zijn, maar dit is geen nadeel. Het is een erg uitgebreid en veelzijdig pakket waar veel mogelijk mee is. Dat de back-end is opgezet volgens MVC5 is een goede keuze. Dit geeft meer structuur en scheiding van functies.

Als vorm voor de oplossing is gekozen voor een custom view engine. Doordat de Mustache files worden gebruikt als bronbestand scheelt dit een transformatieslag. Daardoor is een view engine minder foutgevoelig. Immers, hoe meer er omgezet moet worden, hoe meer fouten er op kunnen treden. Daarnaast heeft een view engine meer waarde voor Colours gezien de uitbreidingsmogelijkheden.

Aangezien er een proof of concept opgeleverd wordt is het aan te bevelen verder onderzoek te doen. Zo is rr nu voor gekozen om de oplossing te richten op het Sitecore CMS. Een CMS dat draait op het .NET platform. Colours bouwt ook websites met Drupal. Dit is een PHP-based CMS. Aan de hand van dit onderzoek en verder onderzoek zou een soortgelijke view engine ontwikkeld kunnen worden om PHP-based Content Management Systemen te ondersteunen.

Daarnaast zijn de belangrijkste functies van Mustache gebruikt. Daarom is het aan te raden verder onderzoek te doen naar de Mustache tags en hoe deze uitgelezen kunnen worden. Er is nu ook geen rekening gehouden met logica in de views terwijl dit wel voor kan komen.

7. OPLOSSINGSPLAN

Naar aanleiding van het uitgevoerde onderzoek heb ik een proof of concept opgeleverd: een custom view engine. Deze heb ik 'from scratch' gebouwd. De custom view engine zorgt ervoor dat Mustache files gebruikt kunnen worden als view files. Op deze manier werken de front-end developers rechtstreeks in de .NET omgeving van de website.

Om eerst een duidelijk plan te bepalen voor de ontwikkeling van de custom view engine heb ik een compacte scope opgesteld met betrekking tot de oplossing. De custom view engine kan de meest gebruikte Mustache tags uitlezen. Daarnaast wordt door middel van Glass de juiste Sitecore rendering opgehaald aan de hand van het meegegeven model. Ten slotte worden automatisch de namespaces toegevoegd zodat de Sitecore renderings ook daadwerkelijk herkend worden. Deze scope en planning is terug te vinden als bijlage II.

BESTAANDE VIEW ENGINES

Om een goed inzicht te krijgen in de techniek van een view engine heb ik de broncode van een aantal bestaande projecten gedownload en bestudeerd.

De front-end developers bij Colours gebruiken de PHP versie van Pattern Lab. Hier is ook een .NET versie van maar deze wordt nauwelijks geüpdatet. De broncode heb ik gedownload om te kijken of een deel van de code hergebruikt kan worden. Deze code bevat ook een Mustache View Engine. Echter, deze code gebruikt wel Mustache files als input maar gebruikt deze op een andere manier dan voor mijn view engine nodig zou zijn. Dit voorbeeld heeft wel gezorgd voor een beter beeld van een view engine.

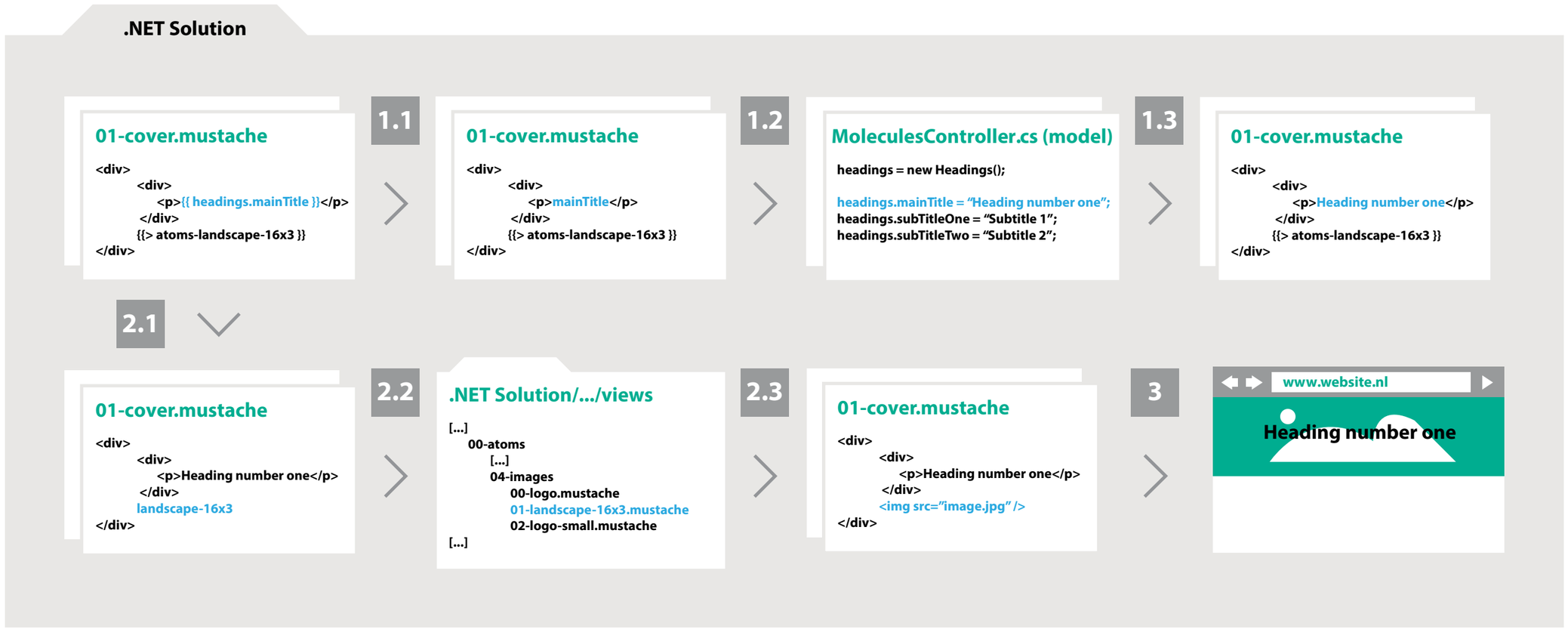
Hiernaast heb ik ook twee andere view engines onderzocht. De eerste is de Razor view engine. Dit is een bekende .NET view engine die veel gebruikt wordt door back-end developers, ook bij Colours. Dit is een uitgebreide en complexe view engine. Té uitgebreid in vergelijking met wat mijn custom view engine moet doen bleek wel.

Daarom ben ik ook naar een minder grote view engine gaan kijken: een PHP view engine. Deze view engine zorgt ervoor dat PHP bestanden gebruikt kunnen worden als view files en rendert deze als HTML. De broncode hiervan gaf mij meer duidelijkheid over de techniek achter een view engine.

Naast deze bestaande view engines ben ik ook research gaan doen naar kleine voorbeelden en tutorials van custom view engines. Aan de hand van een tutorial heb ik een simpel opzetje gemaakt waarbij een woord tussen < > symbolen gerenderd wordt als de huidige datum en tijd. De basisopzet om een custom view engine te laten werken binnen een project werd me hierbij duidelijk.

CUSTOM VIEW ENGINE

Nadat de opzet van een view engine duidelijk geworden was ben ik begonnen aan de ontwikkeling van de custom view engine. Als eerste ben ik begonnen met de routing van de engine. De view engine moet weten waar en welke soort viewfiles hij moet gebruiken om te renderen. In dit geval zijn dat de Mustache files.



Figuur 14: Het uitlezen, verwerken en renderen van de Mustache tags.

Daarna is begonnen met het uitlezen van de tags in deze files, het omzetten van deze tags en vervolgens renderen. Hoe dit werkt wordt uitgelegd in figuur 14. Hier wordt het molecule 'Cover' gebruikt als voorbeeld.

De view engine zoekt eerst naar alle Mustache tags waarbinnen data gerenderd moet worden. Dat zijn deze tags: {{ }}.

1.1 Het eerste deel vanaf de dubbele accolades tot en met de punt en de laatste spatie en afsluitende accolades worden weggehaald.

1.2 In het molecules model wordt gezocht naar de gestripte tekst uit stap 1.1.

1.3 De bijhorende value van de tekst wordt terug gegeven aan de Mustache file. De tag is nu vervangen door data.

Vervolgens zoekt de view engine naar alle Mustache tags waarbinnen een ander HTML
element gerenderd moet worden. Dit zijn deze tags: {{> }}.

2.1 Het eerste deel vanaf de dubbele accolades en pijltje tot en met het streepje en de laatste spatie en afsluitende accolades worden weggehaald.

2.2 In de views folder van de .NET Solution wordt gezocht naar een Mustache file die de term uit stap 2.1 in zich heeft.

2.3 De content van de gevonden file wordt terug gegeven aan de Mustache file. De tag is nu vervangen door een HTML element.

3 Vervolgens wordt de Mustache file met vervangen content gerenderd in de browser en is een image te zien met daar overheen een titel.

Dit was de basis van het renderen van Mustache files. Om dit te implementeren in een Sitecore website wordt Glass gebruikt. Hierdoor hoeft geen rekening gehouden te worden met het veldtype van een Sitecore rendering die in het model gedefinieerd is. Glass zet de data die vanuit Sitecore komt om naar code waar de .NET solution mee kan werken. Om dit goed te laten werken geldt er wel een randvoorwaarde, deze wordt beschreven in hoofdstuk 8.

Om ook iets te kunnen doen met de Sitecore-gerelateerde code moeten er bepaalde usings aan de viewfiles toegevoegd worden. Dit wordt automatisch gedaan tussen het moment dat alle Mustache tags zijn vervangen door data en HTML elementen en de code gerenderd wordt in de browser. Het gaat hier om de usings Sitecore.Mvc en Sitecore.Mvc.Presentation.

Deze functies zorgen voor een basisopzet van de custom view engine. Een webpagina met daarin de twee meest voorkomende Mustache tags kan gerenderd worden en heeft data in zich afkomstig uit het Sitecore CMS en bijbehorende usings.

Aangezien de ontwikkelde oplossing is opgeleverd als proof of concept wordt in dit implementatieplan beschreven hoe de oplossing ingezet kan worden en hoe deze mogelijk uitgebreid kan worden.

De custom view engine is ontwikkeld voor de front-end en back-end developers. Als front-ender moet er kennis zijn van Pattern Lab, Mustache, SASS, Grunt en Bower. Een back-ender moet kennis hebben van C#.NET, MVC5 en Sitecore.

Op welke manier kan de basisopzet van de view engine geïmplementeerd worden in een project? Als eerste moet er een .NET solution zijn die samenwerkt met een Sitecore instantie. Deze .NET solution moet volgens het MVC5-model opgezet zijn en moet de Glass.Mapper extensie bevatten.

De dummydata die wordt gebruikt bij het opzetten van Pattern Lab mag in de naamgeving geen streepjes of andere 'ongewone' karakters bevatten. Het model in de .NET solution kan op die manier niet gedefinieerd worden. Daarnaast moet de JSON data vooraf gegaan worden door 'Model.'. Dit heeft Glass.Mapper nodig om te kunnen werken. Door deze manier van noteren al te gebruiken tijdens het ontwikkelen van de Pattern Lab style guide kan de custom view engine dit één-op-één overnemen.

De usings om Sitecore te laten werken en de using die Glass.Mapper laat werken worden automatisch toegevoegd door de view engine. Als later blijkt dat er meerdere of andere usings toegevoegd moeten worden kan dit eenvoudig uitgebreid worden.

Het proof of concept behelst de basisfuncties. De twee belangrijkste tags worden uitgelezen. Dit is de tag om data in te laden en de tag om een ander HTML element in een pagina te laden. Hiernaast zijn nog enkele andere tags. Zo is er een tag waarmee door data 'geloopt' kan worden, is er een tag waarmee juist gerenderd wordt wanneer iets niet in de data voor komt en is er een tag om commentaar te plaatsen.

Het kan ooit voorkomen dat er één of meerdere parameters meegeven moeten worden met een Sitecore rendering. Met Glass kunnen deze parameters ook meegegeven worden waardoor bijvoorbeeld een class toegevoegd kan worden of de width en height van een image gezet kunnen worden. Dit zou later toe te voegen zijn.

Dit proof of concept is ontwikkeld op basis van het .NET-based CMS Sitecore. Wanneer de view engine een duidelijke meerwaarde blijkt te hebben kan overwogen worden om de view engine ook andere .NET-based CMS'en te laten ondersteunen. Bijvoorbeeld Umbraco of Smartsite. Daarnaast zou ook een PHP-versie ontwikkeld kunnen worden. Dit is niet zozeer een uitbreiding maar dit proof of concept kan hierbij wel als voorbeeld dienen.

9. REFLECTIE

Ineens is het januari. De vijf maanden zijn voorbij gevlogen, tijd om terug te blikken.

Mijn afstudeerperiode heb ik als zeer leerzaam ervaren. Onderweg heb ik veel nieuwe dingen geleerd en opgestoken. Zo is de weg naar het eindresultaat niet altijd soepel verlopen maar heb ik geleerd dat dit in het werkveld ook zeker niet altijd het geval is. Geef nooit op, maar dat credo hanteerde ik al. Ook de algemene en technische begeleiding heeft hier aan bijgedragen. Deze was uitstekend. Ik heb veel tips gekregen om naar bepaalde dingen te kijken of te onderzoeken. Ook bij vragen en het opzetten van de tools was er altijd wel iemand bereid om te helpen. Hierdoor ben ik weinig tijd verloren aan installatieproblemen.

Vooraf heb ik gesteld dat ik C# wilde gaan leren. Dit ben ik gaan doen en tijdens het ontwikkelen leer ik nog steeds iedere dag nieuwe dingen. Ik heb wel gemerkt dat het leren van deze taal mij meer moeite gekost heeft dan gedacht. Daarom moet ik jammer genoeg tot de conclusie komen dat het .NET framework en bijbehorende taal C# niet voor mij is weggelegd om als professional mee verder te gaan. Front-end is toch meer mijn ding.

Ik heb geleerd om sneller hulp te vragen als het me zelf niet lukt. Ik heb altijd het idee dat het beter is als ik zelf verder zoek omdat ik het gevoel heb dat ik daar het meeste van leer. Daar is in principe niets mis mee maar het kost op sommige momenten gewoon te veel tijd. Tijd die beter besteed had kunnen worden.

Communiceren en afstemmen met betrokkenen heb ik ook als belangrijk ervaren. Een mail sturen met besproken punten na een meeting is heel simpel maar dit kwam in de beginfase van de stage nog niet bij me op. Ook het laten weten aan anderen wat je die dag gaat doen heb ik moeten leren. De nadelen als je dit niet doet heb ik ondervonden tijdens het verdiepen in MVC5. Door niet goed af te stemmen heb ik ongeveer twee weken weggegooid. Hier heb ik tijdens de ontwikkeling van de oplossing veel spijt van gehad. Ook bleef de vertaalslag van front-end naar back-end voor mij een tijdje 'zweven'. Hierdoor was het lastig om hier vorm aan te geven.

Wat ik lastig vond was het schrijven van deze scriptie. Ik wil alles veel te uitgebreid en te technisch opschrijven. De afweging maken tussen belangrijke en overbodige informatie heeft me veel pijn en moeite gekost.

Als ik de hele afstudeerperiode in ogenschouw neem kan ik zeggen dat ik mijn kennis ruim verbreed heb. Door te werken aan zowel front-end als back-end heb ik van beide disciplines veel bijgeleerd. Bij tegenslagen doorzetten en verder gaan deed ik altijd al maar tijdens deze afstudeerperiode is dit maar weer bevestigd.

10. BRONNEN

Blanchard, N. (2014). Gulp vs. Grunt: Node.js Automation Tools Showdown.
Geraadpleegd op 21 november 2014, van http://www.oomphinc.com/blog/2014-03/gulp-vs-grunt-node-js-automation-tools-showdown/

Colours. (2014). De ingrediënten voor uw succesverhaal.
Geraadpleegd op 13 december 2014, van http://colours.nl/wat-we-doen

Coyler, C. (2012). Sass vs. LESS.
Geraadpleegd op 21 november 2014, van http://css-tricks.com/sass-vs-less/

Frost, B. (2014). Question: template export for production? #171.
Geraadpleegd op 21 november 2014, van https://github.com/pattern-lab/pattern-lab-php/issues/171

Pattern Lab. (z.d.). What it is.
Geraadpleegd op 26 november 2014, van http://patternlab.io/

Lewis, J. (2014). Styleguide Driven Development.
Geraadpleegd op 29 december 2014, van http://webuild.envato.com/blog/styleguide-driven-development/

Reiz, R. (2014). Which programming language has the best package manager?
Geraapleegd op 21 november 2014, van http://blog.versioneye.com/2014/01/15/which-programming-language-has-the-best-package-manager/

Roesink, B. (2014). Frontend development tools.
Geraadpleegd op 24 november 2014, van https://psdtowp.net/frontend-development-tools.html

San Filippo, D. (2014). Sitecore Development in the Real World - Page Template Development Process.
Geraadpleegd op 17 november 2014, van http://www.mtelligent.com/journal/2014/8/21/sitecore-development-in-the-real-world-page-template-develop.html

Shemeer, NS. (2013). ASP.NET MVC 4 - Part [1] – Introduction.
Geraadpleegd op 27 november 2014, van http://www.codeproject.com/Articles/470107/ASP-NET-MVC-4-Part-1-Introduction
20 maart 2013, Shemeer NS

Sitecore. (z.d.). Corporate Overview.
Geraadpleegd op 27 november 2014, van http://www.sitecore.net/About/Corporate-Overview.aspx

Vidal, B. (2012). Rendering Data Source Saved as Path Can Cause Trouble in Sitecore.
Geraadpleegd op 17 november 2014, van http://www.roundedcube.com/Blog/2012/rendering-data-source-saved-as-path-can-cause-trouble-in-sitecore

Welander, M. (2014). View renderings vs controller renderings.
Geraadpleegd op 17 november 2014, van http://mhwelander.net/2014/06/13/view-renderings-vs-controller-renderings/

BIJLAGEN

I : Project Initiatie Document
II : Planning Custom View Engine

BIJLAGE I

Project Initiatie Document

PROJECT INITIATIE DOCUMENT

BEHEER VAN HTML OP ÉÉN CENTRALE LOCATIE.

Auteur: Ben van de Wal
Studentnummer: 2164044
Bedrijf: Colours
Versie: 1.0
Status: Final
Datum voltooid: 6 oktober 2014
Bestandsnaam: PID – Ben van de Wal – v1.0

Revisies
Dit document heeft de volgende wijzigingen gehad:

Versie	Status	Datum	Wijzigingen
0.1	Draft	03-09-2014	Eerste versie, alles ingevuld.
0.2	Draft	23-09-2014	Planning, organigrammen en spellingsfouten aangepast.
1.0	Final	03-10-2014	Planning, aanleiding en aanpak aangepast.

Goedkeuring
Dit document behoeft de volgende goedkeuringen:

Versie	Datum goedkeuring	Naam	Functie	Paraaf

Distributie
Dit document is verstuurd aan:

Versie	Datum verzending	Naam	Functie
0.1	19-09-2014	Joris Graaumans	Docentbegeleider
0.1	19-09-2014	Rob Habraken	Bedrijfsbegeleider
0.2	25-09-2014	Bertus Groenewegen	Front-end developer
1.0	06-10-2014	Joris Graaumans	Docentbegeleider
1.0	06-10-2014	Rob Habraken	Bedrijfsbegeleider
1.0	06-10-2014	Bertus Groenewegen	Front-end developer

Doel van dit document
Het doel van dit document is het stellen van projectkaders. Hierdoor kan dit document dienen als leidraad bij de uitvoer van het project. De aanleiding van de opdracht, de gekozen aanpak en globale planning zijn in dit document opgenomen. Bij eventuele problemen dient dit document als leidraad.

Aanleiding
Bij de start van de ontwikkeling van een nieuwe website wordt een style guide opgezet door de front-end developers. Dit is een standalone HTML pagina waarop alle elementen zoals headings, borders, font en dergelijke terug te vinden zijn. Voor de website (.NET solution) geldt deze style guide als referentie. De gemaakte HTML wordt vervolgens door de back-end developers in de .NET solution gekopieerd. Hierdoor ontstaan er verschillen in de HTML. Ook door het gebruik van versiebeheer kunnen verschillen ontstaan. Er ontstaan dus eigenlijk twee HTML versies, de style guide en de .NET solution.

Het probleem hiervan is dat het inefficient werkt en dat het foutgevoelig is. Het is inefficiënt doordat de font-end developers eerst de HTML schrijven waarna de back-end developers deze weer moeten kopiëren. Daarnaast moet de style guide altijd als leidraad kunnen dienen. Deze functie vervalt wanneer er in de HTML van de .NET solution aanpassingen/updates worden gedaan die niet in de style guide doorgevoerd worden. Hierdoor kan er in het vervolg verkeerde/verouderde code worden toegepast in het project.

Globale aanpak
Door middel van een softwarematige oplossing gaat dit probleem verholpen worden. Deze software moet ervoor zorgen dat gemaakte HTML op één plaats beheerdkan worden. Wanneer er in de style guide of .NET solution aanpassingen gedaan worden, moeten deze ook automatisch bij de andere doorkomen.

Om tot een goede oplossing te komen moet er ook onderzoek gedaan worden. Er wordt gekeken naar het ontwikkelproces van een website. Hiermee wordt het programmeren bedoeld. Om hier een goed beeld van te krijgen moeten de tools onderzocht worden die Colours gebruikt. Dit wordt gedaan door deskresearch en door er zelf mee aan de slag te gaan met hulp van de developers. Om de wensen van de developers terug te laten komen in de oplossing worden er interviews gehouden. Vervolgens wordt in stappen software ontwikkeld die bij de tools aansluit. Eerst ontwikkelen, testen en vervolgens voorleggen aan de developers. Deze stappen worden herhaald tot het eindproduct voltooid is.

Het project wordt op deze manier aangepakt zodat de kans groter is dat er aan het einde van de rit een voor Colours bruikbare oplossing ligt. De developers kunnen op deze manier makkelijker op gezette tijdstippen tussentijds bijsturen en input geven. Zo blijft de betrokkenheid groter en ontstaat er iets wat de developers ook écht bruikbaar vinden.

Benodigdheden
De doorlooptijd van dit project is 20 weken. Hiervan zijn 16 weken beschikbaar om naar de oplossing te werken. Dit komt neer op 640 uur. De wenselijke deadline is week 51 (15-18 december 2014).

INHOUDSOPGAVE

1. INLEIDING

Doel van dit document
Het doel van dit document is het stellen van projectkaders. Door deze kaders te stellen kan dit document dienen als leidraad bij de uitvoer van het project. De aanleiding, aanpak en globale planning van de afstudeeropdracht zijn hierin beschreven. Hierbij worden alle randvoorwaarden en benodigdheden vermeld. Bij eventuele problemen kan altijd worden teruggekeken naar dit document. In het PID worden de volgende vragen beantwoord:

- Wat is het probleem van het bedrijf?
- Waarom is dit een probleem voor het bedrijf?
- Wat gaat er gebeuren om het probleem op te lossen?
- Hoe gaat het probleem opgelost worden?
- Waarom wordt het probleem op deze manier aangepakt?

Daarnaast wordt dit PID gebruikt voor de volgende zaken:
- Aan de hand van dit PID kunnen de assessoren van school, Fontys Hogescholen, beoordelen of de afstudeeropdracht levensvatbaar is. Oftewel, voldoet het project aan de eisen van een afstudeeropdracht.
- De bedrijfsbegeleider binnen het afstudeerbedrijf, Colours, ziet duidelijk dat de afstudeerder de opdracht goed voor ogen heeft.
- Ook wordt het PID gebruikt om de afstudeeropdracht te rechtvaardigen. Oftewel, bewijzen waarom het belangrijk is dat het project uitgevoerd gaat worden.

Opbouw van dit document
Dit document is opgedeeld in de volgende hoofdstukken:
- Achtergrond (Hoofdstuk 2)
- Projectdefinitie (Hoofdstuk 3)
- Projectorganisatiestructuur (Hoofdstuk 4)
- Planning (Bijlage A)

2. ACHTERGROND

Het bedrijf
Colours is een vooraanstaand fullservice internetbureau in Den Bosch. Samen met acht andere bedrijven vormen zij LECTRIC Groep. Een schematische weergave hiervan wordt weergegeven in figuur 1. Het ontwikkelen van succesvolle digitale concepten en daarbij websites is de specialiteit van Colours. Deze websites worden door middel van online marketing gelanceerd. Colours bestaat uit een gemixt team van designers, developers, projectmanagers en marketeers.
Het gevolg is dat er dubbel werk wordt gedaan. De HTML wordt uit de bestanden van de front-end developer gekopieerd en in de daarvoor aangemaakte .NET pagina geplakt. Hierdoor ontstaan eigenlijk twee HTML versies. Wanneer er in de HTML in het CMS iets aangepast wordt is dit niet aangepast in de styleguide van de front-end developers.

Huidige situatie
Bij Colours wordt gewerkt volgens de Scrum methodiek. Scrum is een iteratieve, flexibele manier van werken in een multidisciplinair team. In korte sprints worden werkende producten opgeleverd. Bij de ontwikkeling van een nieuwe website wordt zo'n scrumteam samengesteld. Elk scrumteam heeft een businessowner. Deze zorgt ervoor dat de wensen van de klant terugkomen in het project. Deze businessowner gaat met de interactie ontwerper de wireframes van het project maken. Deze wireframes gaan naar de visual designer die deze wireframes omzet naar een visueel design zoals de website eruit komt te zien.

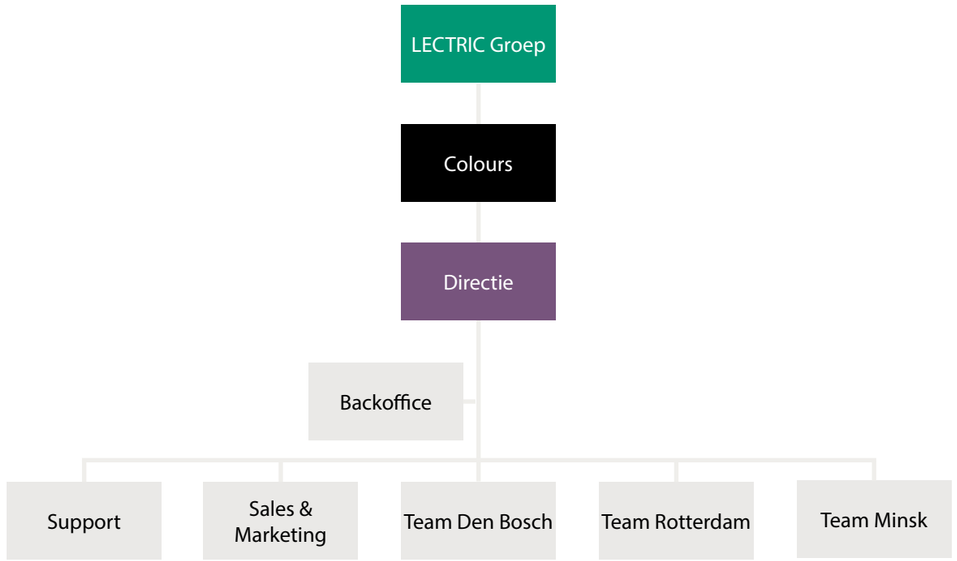
Deze ontwerpen gaan vervolgens naar de front-end developer. Hier worden de ontwerpen omgezet naar HTML/CSS. Dit wordt de style guide met alle elementen, kleuren, headings, etcetera voor die website. Op deze manier ontstaan er afspraken over de website elementen maar zouden ook alle developers aan het project kunnen werken. De style guide is nieuwe HTML/CSS en wordt in delen (losse modules) ontwikkeld volgens het Atomic Design principe. Atomic Design is een methode om websites op te bouwen. Het principe leunt op vijf bouwstenen: atoms (bijv. label), molecules (bijv. form), organisms (bijv. header met form), templates (bijv. ontwerp met placeholders) en pages (bijv. uiteindelijke website). Hierdoor kunnen stand alone componenten gemaakt worden die herbruikbaar zijn.

Om de herbruikbaarheid van HTML in de style guide te implementeren wordt er gebruik gemaakt van Handlebars. Handlebars genereert aan de hand van 'expressions' HTML. Door op laag niveau aanpassingen te doen worden deze ook doorgezet naar de hogere niveaus. Deze style guide is een stand alone webpagina welke dus apart benaderbaar is. CSS en JavaScript wordt automatisch gecompiled, minified, samengevoegd en geïntegreerd in de .NET Solution met behulp van SASS, Grunt en Bower. Deze tools draaien op het NODE.js platform. SASS is een preprocessor waardoor er meer mogelijk is met CSS. SASS zorgt ervoor dat het gebruik van variablelen, nesten en mixins mogelijk wordt. Bower zorgt ervoor dat git repositories automatisch worden opgehaald en toegevoegd aan het project. Deze repositories kunnen bijvoorbeeld libraries of frameworks bevatten. Grunt is de tool die alle andere tools aanstuurt. Grunt zorgt ervoor dat bestanden worden minified en compiled. Er zijn veel plug-ins beschikbaar waarmee functies naar eigen keuze toegevoegd kunnen worden.

Het probleem
Naast de style guide wordt de daadwerkelijke website gebouwd, de .NET Solution. Hierin wordt HTML gekopieerd die eerder al gemaakt is. De HTML wordt uit de bestanden van de front-end developer gekopieerd en in de daarvoor aangemaakte .NET pagina geplakt. Er ontstaan dus twee HTML versies welke apart beheert worden. Het nadeel hiervan is dat als er aan de ene kant iets verandert het bij de andere kant niet automatisch is aangepast. Dit moet twee keer met de hand gedaan worden indien nodig. Het is dus niet handig om de HTML op twee plaatsen te beheren aangezien dit onnodig veel tijd kost. Doordat er twee versies ontstaan wordt de foutgevoeligheid ook groter. Wanneer er in de .NET Solution HTML wordt geupdate (omdat er een element toegevoegd moet worden bijvoorbeeld), maar niet in de style guide, wordt de style guide achterhaald. De functie van een style guide is dat iedere (nieuwe) developer met het project uit de voeten zou kunnen. Op deze manier kunnen er oude, verkeerde standaarden in het project geïmplementeerd worden.

Ook kunnen er verschillen in de HTML ontstaan doordat er bij Colours met versiebeheer gewerkt wordt. Er is één centrale codebase van een project. Elk projectlid maakt hier een lokale kopie van (branch). In deze branch gaan zij dingen aanpassen. Later worden deze branches weer samengevoegd (mergen). Hierbij kunnen conflicten ontstaan wanneer er bij de branches andere dingen zijn aangepast. Voorbeeld: bij de ene branch is de HTML aangepast, maar bij de andere is er aan de back-end iets veranderd, niet de HTML. Bij het mergen moet dan gekozen worden welke versie er naar het samengevoegde project op de testserver gaat.

Door dit onderzoek uit te voeren kan er een oplossing gevonden worden om de workflow te verbeteren. De front-end en back-end developers kunnen efficiënter samenwerken doordat er minder dubbel werk gedaan hoeft te worden. Daarnaast zijn aanpassingen in de front-end sneller door te voeren in de .NET Solution (CMS). Dit is nodig doordat er bij Colours volgens de scrum methodiek gewerkt wordt. Dit houdt in dat er iedere twee weken opgeleverd wordt. Door deze korte doorlooptijd is het vereist dat aanpassingen snel doorgevoerd kunnen worden.



Figuur 1: Bedrijfsstructuur van Colours.

3. PROJECTDEFENITIE

Projectdoelstelling
Front-end developers bij Colours maken in de beginfase van een project een style guide in HTML/CSS. Deze moet dienen als leidraad/handboek voor het project. Doordat aanpassingen in de style guide of de .NET solution niet automatisch doorgevoerd worden in de style guide vervalt de leidende functie van de style guide. Zij zouden graag zien dat de HTML/CSS op één plek beheerd wordt. Daarnaast merken de back-end developers dat ze veel tijd kwijt zijn aan het kopiëren van geschreven HTML/CSS. Dit zouden zij graag meer geautomatiseerd willen zien maar hebben door het vele werk niet de tijd om dit te ontwikkelen. Door dit project uit te voeren wordt er een efficiëntere samenwerking tussen de front- en back-end developers mogelijk. De gehele workflow tussen de front-end en back-end developers moet vloeiender verlopen zonder dat er dubbel werk wordt gedaan. Over vijf maanden moeten de front-end en back-end developers bij Colours de door de front-end devleopers geschreven HTML code op één plaats kunnen beheren door middel van een softwarematige oplossing.

Gekozen oplossing of aanpak
Voor dit project is gekozen om het Tien Stappen Plan (TSP) te gebruiken in combinatie met scrum. Dit houdt in dat het project gefaseerd aangepakt wordt. De fases waarin het project is opgedeeld zijn: oriëntatiefase, onderzoeks- en oplossingsfase en de invoeringsfase. Tijdens de onderzoeks- en oplossingsfase wordt er gewerkt volgens de scrum methodiek.

In de oriëntatiefase wordt naar het probleem gekeken zoals dit is aangegeven tijdens het intakegesprek. Hier wordt verder naar gevraagd om meer informatie en inzichten te verkrijgen. Aangezien het een probleem is dat speelt bij zowel de front-end developers als de back-end developers worden er van allebei de disciplines een aantal developers geïnterviewd. Daarbij wordt er gekeken naar het proces dat een website bij Colours doorloopt in de fase van development en welke tools hierbij gebruikt worden. Deze tools worden door middel van deskresearch en demo's onderzocht. Ook wordt er een globale planning opgesteld om een goed beeld te krijgen van de doorlooptijd. Deze bevindingen vormen uiteindelijk het PID.

Op de oriëntatiefase volgt fase twee: de onderzoeks- en oplossingsfase. Tijdens deze fase wordt er gewerkt volgens de scrum methodiek. Dit wil zeggen dat er iteratief en resultaatgericht gewerkt wordt in korte sprints van één à twee weken. De informatie die verzameld is tijdens de oriëntatiefase wordt geanalyseerd en verwerkt. Naar aanleiding van het PID kan het onderzoeksplan opgesteld worden. Hierin worden de hoofdvraag en deelvragen beschreven en op welke manier het onderzoek aangepakt wordt. Ook worden de requirements opgesteld waaraan de oplossing moet gaan voldoen. Dit wordt gedaan tijdens sprint 0, de voorbereidingsspint. Dan volgt de uitvoer van het onderzoek. Dit begint in sprint 1. De focus van dit onderzoek ligt op de huidige workflow aan de developers kant bij Colours en het beheren van geschreven HTML code. Daarvoor worden de tools die front-end developers bij Colours al gebruiken geleerd, gebruikt en uitgebreid beschreven. Vervolgens worden ook de Content Management Systemen (CMS) die Colours gebruikt geleerd en gebruikt. Dat gebeurt tijdens sprint 2. Daarop volgt sprint 3. Tijdens deze sprint ligt de focus op de stap tussen de front-end developing (sprint 1) en de back-end developing (sprint 2). Hier wordt de softwarematige oplossing ontwikkeld die ervoor zorgt dat de HTML code op één plaats beheerd kan worden. Aan het eind van deze sprint wordt de oplossing gepresenteerd. De feedback die hierbij naar voren komt wordt tijdens de optimalisatiesprint verwerkt (sprint 4).

Tijdens een sprint worden telkens dezelfde stappen doorlopen. Er wordt eerst een stukje onderzoek gedaan. Met deze verkregen kennis wordt een afgebakend deel van het eindproduct gerealiseerd. Dit wordt getest en vervolgens gepresenteerd door middel van een demo. Deze demo en de documentatie voor de scriptie zorgen voor de evaluatie.

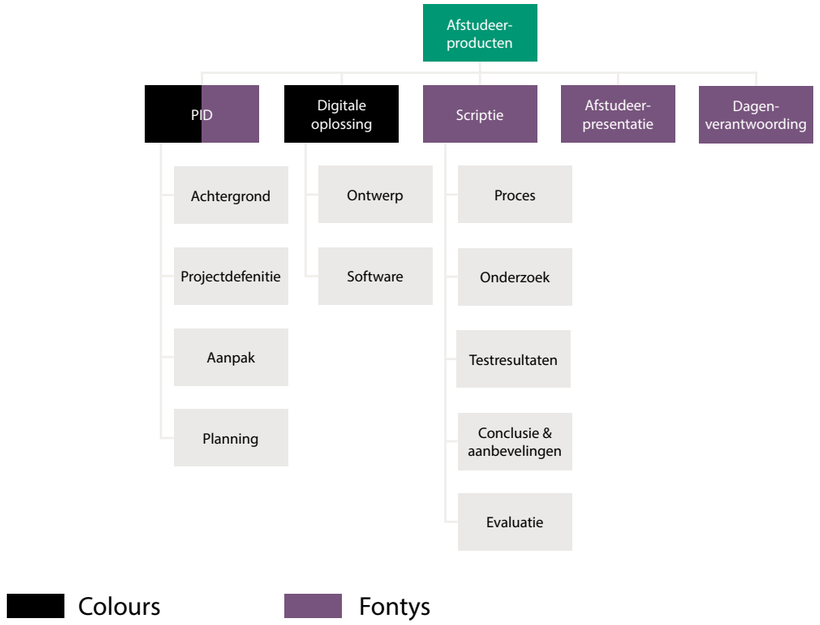
Om de juiste informatie te verkrijgen worden er interviews gehouden met front-end en back-end developers binnen Colours (onderzoeksmethode veld), wordt er deskresearch gedaan naar de tools (onderzoeksmethode bieb) en worden de tools en methodieken zelf geleerd, gebruikt en geëvalueerd met hulp van de Colours developers (onderzoeksmethode werkplaats). Alle resultaten van dit onderzoek worden verzameld en gebundeld in het onderzoeksrapport. Hieruit volgt een advies voor de beste oplossing. De functies voor de oplossing die eerder al zijn vastgesteld worden verwerkt. De oplossing is in de vorm van software. Deze software wordt ook getest tijdens de ontwikkeling. De resultaten hiervan worden opgenomen in een testrapport.

De laatste fase is de invoeringsfase. Tijdens de invoeringsfase wordt de ontwikkelde oplossing geïmplementeerd binnen Colours. Hierbij wordt door middel van een eindpresentatie en demo het product gepresenteerd en kan beslist worden wat Colours ermee doet. Het eindproduct is op dat moment voltooid en is klaar voor gebruik.

Het project wordt op deze manier aangepakt zodat de kans groter is dat er aan het einde van de rit een voor Colours bruikbare oplossing ligt. Doordat er elke twee weken een presentatie en demo plaatsvindt kunnen de developers makkelijker bijsturen en nieuwe input geven. Ook blijft op deze manier de betrokkenheid groter en ontstaat er iets wat voor de developers ook écht bruikbaar is.

Scope van het project
Bij Colours merken ze dat er tijdens het ontwikkelproces van een website meer geautomatiseerd moet worden. Het samenvoegen van CSS en JavaScript gebeurt automatisch maar de HTML wordt nog handmatig geïmplementeerd. Deze HTML wordt met de hand overgezet vanuit de style guide in het CMS wat inefficiënt werkt. Ook is het foutgevoelig doordat aanpassingen in het CMS niet automatisch in de style guide doorgevoerd worden. De style guide kan zo niet meer fungeren als handboek. Hiervoor moet een stuk software ontwikkeld worden waarmee door front-end developers HTML code op één plaats beheert kan worden waardoor aanpassingen aan beide kanten automatisch doorgevoerd worden. De oplossing moet tijdbesparend werken en minder foutgevoelig zijn. Op deze manier blijft de style guide en de daadwerkelijke website (.NET solution) hetzelfde. Het is de bedoeling dat de oplossing werkt binnen de huidige tools die Colours gebruikt. Diegenen binnen Colours die gebruik gaan maken van de oplossing zijn de front-end developers en back-end developers. Door tijdens het onderzoek ook rekening te houden met de complexiteit van het visual design is het van meerwaarde dat een ICT & Media Design afstudeerstagiair dit onderzoek uitvoert.

Projectdecompositiestructuur (PDS)



Figuur 2: Projectdecompositiestructuur afstudeerproject.

Producten c.q. eindresultaat
De producten die tijdens het afstuderen worden opgeleverd zijn te verdelen in twee categorieën: voor het bedrijf en voor school. De producten voor het bedrijf zijn:
- PID, als digitaal document.
- Digitale oplossing, als software.

De producten die voor school opgeleverd worden zijn:
- PID, als digitaal document.
- Scriptie, als digitaal document en print indien nodig.
- Afstudeerpresentatie, als digitale presentatie.
- Dagenverantwoording, als digitaal document.

Uitsluitingen
Het gaat om het beheren van HTML code. Het CSS en JavaScript wordt al automatisch beheerd en gedeeld tussen front-end en back-end developers. Dit project hoeft hier dus geen oplossing voor te bieden.

Benodigdheden
Het project heeft in principe een doorlooptijd van 20 weken. Hiervan zijn 16 weken uitgetrokken om aan de opdracht zelf te werken. Vervolgens zijn er nog vier weken waarin de scriptie geschreven wordt, het tweede bedrijfsbezoek plaats vindt, de eindpresentatie gemaakt wordt en de afstudeerzitting gehouden wordt. In figuur 3 is een schematische weergave te zien van de benodigdheden voor dit project.

Beschikbare tijd	Beschikbare resources	Gewenste opleverdata	Beschikbare doorlooptijd
640 uur *(16 weken = 80 dagen. 80x8 uur per dag = 640 uur)*	Docentbegeleider Bedrijfsbegeleider Colours developers	28 november 2014 *(eerste versie)* 18 december 2014 *(definitieve versie)*	16 weken

Figuur 3: Projectbudget.

Afhankelijkheden
Het is een interne opdracht waar alleen de afstudeerstagiair aan werkt. Hierdoor zijn er weinig afhankelijkheden. Het onderzoek en project worden zelfstandig uitgevoerd. Ter ondersteuning kunnen de bedrijfsbegeleider en developers ingeschakeld worden. Zij moeten wel tijd hebben om bijvoorbeeld geïnterviewd te worden.

Randvoorwaarden
Voor het uitvoeren van het onderzoek en de opdracht zijn maar beperkte randvoorwaarden van kracht. Voor de uitvoering van de opdracht is een Windows laptop nodig aangezien de websites bij Colours in .NET geprogrammeerd worden. Deze geeft Colours voor de gehele periode in bruikleen. Indien er een code repository of een webserver nodig is stelt Colours deze ook beschikbaar.

Vanuit Fontys geldt de randvoorwaarden dat er (minstens) een proof of concept van het eindproduct gerealiseerd wordt. De oplossing van het probleem zal dus altijd softwarematig zijn. Een eventuele procesmatige oplossing wordt wel gedocumenteerd maar niet gerealiseerd.

Aannames
Er zijn twee aannames. De eerste is dat Colours achter een methodische aanpak staat. De tweede is dat er genoeg tijd is om het onderzoek uit te voeren en vervolgens het project tot uitvoering te brengen. Dat betekent dat er naast het afstudeerproject maximaal 10% van de tijd wordt besteed aan reguliere werkzaamheden.

4. PROJECTORGANISATIE

Binnen het project zijn meerdere betrokkenen met verschillende rollen. Hoe deze verdeling is wordt in onderstaand schema weergegeven.

Opdrachtgever

Rolbeschrijving
De opdrachtgever heeft in eerste instantie het probleem geconstateerd en daarna de opdracht uitgezet. Bij Colours is hier Rob Habraken verantwoordelijk voor. Hij is 'Technical Lead' en heeft daarom veel inzicht in het ontwikkelproces van een website.

Project gerelateerde taken
Het vaststellen van het probleem en daarbij in grote lijnen de opdracht formuleren.
Het opgestelde PID inzien en goed- of afkeuren.
Feedback geven op gemaakte producten.
Het eindproduct beoordelen.

Specifieke verantwoordelijkheden
Aanspreekpunt zijn over de opdracht.
Keuren van het PID.
Aanwezig zijn bij tussentijdse demo's.
Beoordelen van het eindproduct.

Seniorgebruiker

Rolbeschrijving
De seniorgebruiker zorgt ervoor dat alle wenselijke functies van de doelgroep in het project opgenomen worden. Hier wordt toezicht op gehouden. De seniorgebruiker is in het geval van dit project de groep developers. Waarin het onderscheid kan worden gemaakt tussen front-end en back-end. Zij zijn degene die de oplossing gaan gebruiken.

Project gerelateerde taken
Toezicht houden op de door de doelgroep wenselijk gestelde functies.
Tussentijdse feedback geven aan het eind van de sprint.

Specifieke verantwoordelijkheden
De wensen van de developers duidelijk naar voren brengen.
Aanwezig zijn bij de tussentijdse demo's.

Projectborging

Rolbeschrijving
Projectborging zorgt ervoor dat de opdracht binnen afgesproken kaders blijft en dat vooraf gedefinieerde specificaties terugkomen in de oplossing. De taak van projectborging ligt in dit geval bij de projectmanager.

Project gerelateerde taken
Controleren van het project, zorgen dat de kwaliteit gewaarborgd blijft.

Specifieke verantwoordelijkheden
Zorgen dat het project binnen de kaders blijft.
Waarborgen van de specificaties.

Projectmanager

Rolbeschrijving
Zorgen voor de voortgang van het project en de planning waarborgen. Bij problemen moet de hulp ingeroepen worden van projectsupport. In eerste instantie oriënteren over de opdracht. Vervolgens deze informatie van de oriëntatie analyseren en verwerken in het onderzoek. Hierop volgend deze onderzoeksresultaten verwerken in het op te leveren product.

Project gerelateerde taken
Gemaakte planning waarborgen en zo nodig bijsturen.
Oriënteren op de opdracht (PID opstellen).
Onderzoek uitvoeren.
Onderzoeksresultaten verwerken in product.
Product testen.

Specifieke verantwoordelijkheden
Informatie vragen wanneer nodig.
Onderzoeksresultaten onderbrengen in het product.
Zorgen dat het opgeleverde product getest.

Projectsupport

Rolbeschrijving
Projectsupport wordt gedaan door de bedrijfsbegeleider binnen Colours, de docentbegeleider binnen Fontys en de developers van Colours. De bedrijfsbegeleider en docentbegeleider zijn in dit geval Rob Habraken en Joris Graaumans. Rob Habraken is binnen Colours 'Technical Lead', ofwel het aanspreekpunt onder de developers. Rob en Joris zorgen ervoor dat de kwaliteit van het project op niveau blijft. Zo niet, dan zullen zij hier melding van maken. De developers geven feedback en helpen waar dit nodig is.

Project gerelateerde taken
Controleren of de werkzaamheden volgens planning verlopen en kwaliteit in de gaten houden.
Tussentijdse feedback geven.
Hulp bieden aan de afstudeerstagiair indien nodig.

Specifieke verantwoordelijkheden
Rob: beschikbaar zijn om dagelijks of wekelijks de voortgang te checken. Joris: wekelijks de voortgang van het project checken door middel van het logboek.
Developers: tweewekelijks beschikbaar zijn om de tussentijdse demo bij te wonen en feedback te geven.

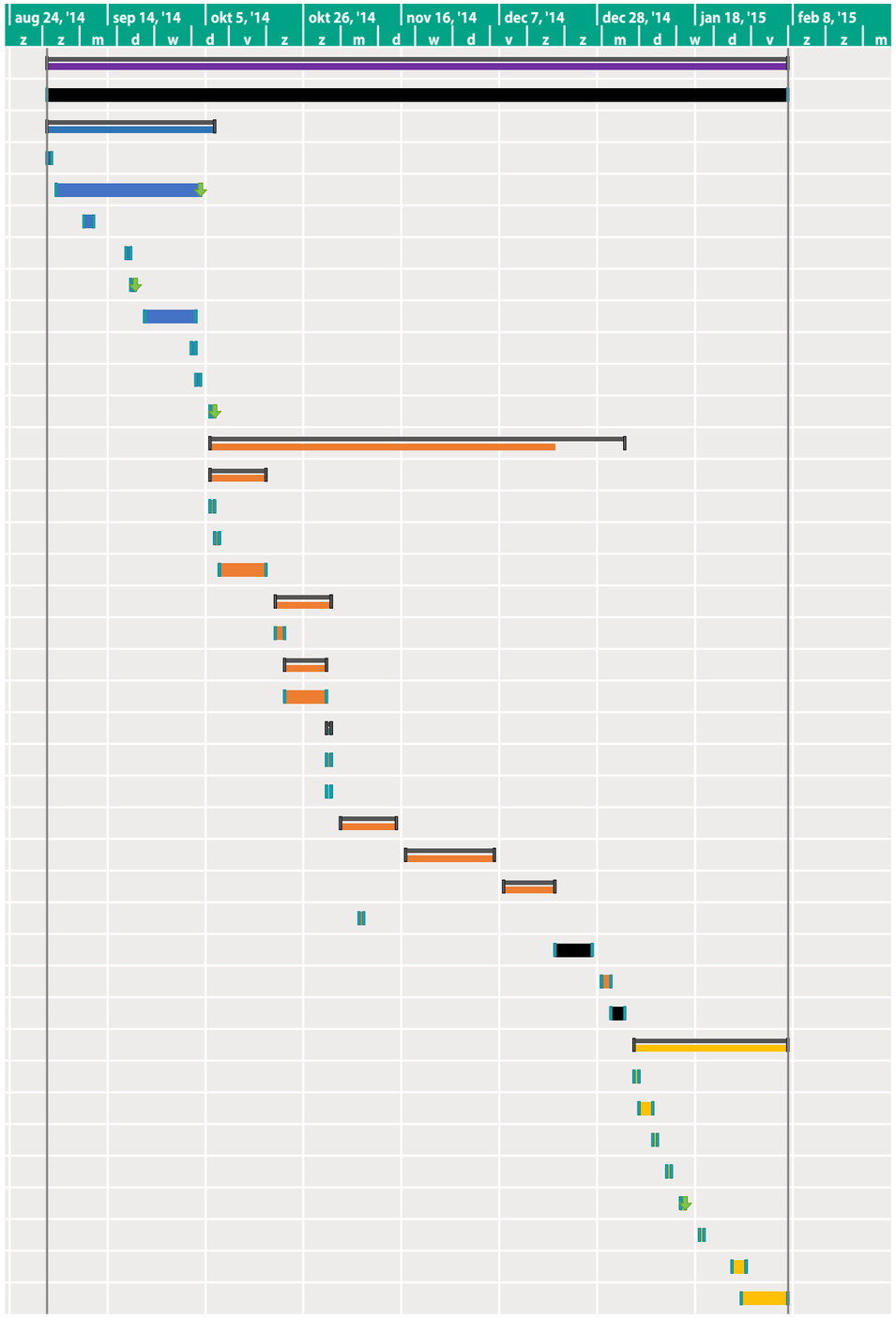
Communicatieplan
Bij het project zijn verschillende personen betrokken. In onderstaande tabel is te zien wie met wie contact heeft, hoe vaak dit contact plaats vindt en op welke manier dit contact is.

Van	Naar	Informatie	Medium	Frequentie
Ben van de Wal	Rob Habraken	Tussentijdse demo's aan eind van sprint.	Presentatie	Eens in de twee weken.
Ben van de Wal	Joris Graaumans	Logboek van werk-zaamheden.	Rapportage	Wekelijks.
Ben van de Wal	Rob Habraken Joris Graaumans Developers	Vragen of feedback.	Mondeling en/of e-mail	Indien nodig.

BIJLAGEN

A : PLANNING

Id	Taaknaam	Duur	Begindatum	Einddatum
1	**Afstudeerstage**	**115 dagen**	**maa 9/1/14**	**vri 2/6/15**
2	Logboek bijhouden	115 dagen	maa 9/1/14	vri 2/6/15
3	**Oriëntatiefase**	**25 dagen**	**maa 9/1/14**	**vri 10/3/14**
4	Stage opstart	1 dag	maa 9/1/14	maa 9/1/14
5	Project Initiatie Document	23 dagen	woe 9/3/14	vri 10/3/14
6	Interviews	2 dagen	din 9/9/14	woe 9/10/14
7	Workshop PID	1 dag	don 9/18/14	don 9/18/14
8	PID: concept	1 dag	vri 9/19/14	vri 9/19/14
9	Feedback verwerken	9 dagen	maa 9/22/14	don 10/2/14
10	Workshop Onderzoek	1 dag	don 10/2/14	don 10/2/14
11	Eerste bedrijfsbezoek	1 dag	vri 10/3/14	vri 10/3/14
12	PID: definitief	1 dag	maa 10/6/14	maa 10/6/14
13	**Onderzoeks- en oplossingsfase**	**65 dagen**	**maa 10/6/14**	**vri 1/2/15**
14	**Sprint 0 - Voorbereiding**	**10 dagen**	**maa 10/6/14**	**vri 10/17/14**
15	Onderzoeksplan	1 dag	maa 10/6/14	vri 10/6/14
16	Requirements	1 dag	din 10/7/14	din 10/7/14
17	Voorbereiding	8 dagen	woe 10/8/14	vri 10/17/14
18	**Sprint 1 - Front-end dev**	**10 dagen**	**maa 10/20/14**	**vri 10/31/14**
19	Onderzoek	2 dagen	maa 10/20/14	din 10/21/14
20	**Realisatie**	**7 dagen**	**woe 10/22/14**	**don 10/30/14**
21	Testen	7 dagen	woe 10/22/14	don 10/30/14
22	**Evaluatie**	**1 dag**	**vri 10/31/14**	**vri 10/31/14**
23	Demo	1 dag	vri 10/31/14	vri 10/31/14
24	Documentatie (scriptie)	1 dag	vri 10/31/14	vri 10/31/14
25	**Sprint 2 - CMS**	**10 dagen**	**maa 11/3/14**	**vri 11/14/14**
32	**Sprint 3 - HTML beheer**	**15 dagen**	**maa 11/17/14**	**vri 12/5/14**
39	**Sprint 4 - Optimalisatie**	**9 dagen**	**maa 12/8/14**	**don 12/18/14**
46	Terugkomdag	1 dag	vri 11/7/14	vri 11/7/14
47	VAKANTIE	6 dagen	vri 12/19/14	vri 12/26/14
48	Extra werkdagen	2 dagen	maa 12/29/14	din 12/30/14
49	VAKANTIE	3 dagen	maa 12/31/14	din 1/2/15
50	**Invoeringsfase**	**25 dagen**	**maa 1/5/15**	**vri 2/6/15**
51	Scriptie: concept	1 dag	maa 1/5/15	maa 1/5/15
52	Scriptie: vormgeving	3 dagen	din 1/6/15	don 1/8/15
53	Scriptie: grammatica	1 dag	vri 1/9/15	vri 1/9/15
54	Scriptie: oplever klaar	1 dag	maa 1/12/15	maa 1/12/15
55	Scriptie: definitief	1 dag	don 1/15/15	don 1/15/15
56	Tweede bedrijfsbezoek	1 dag	maa 1/19/15	maa 1/19/15
57	Presentatie maken	3 dagen	maa 1/26/15	woe 1/28/15
58	Afstudeerzitting	8 dagen	woe 1/28/15	vri 2/6/15



Sprint 2, 3 en 4 zijn in dezelfde fases onderverdeeld als sprint 1. Dit is in bovenstaande tabel niet weergegeven om het overzicht te behouden.

Mijlpalen

Week	Schoolweek	Datum	Activiteit
35	0	29 augustus 2014	Uitzwaaibijeenkomst
36	1	1 september 2014	Start afstudeerstage
38	3	18 september 2014 19 september 2014	Workshop PID PID: concept
40	5	2 oktober 2014	Workshop Onderzoek
41	6	6 oktober 2014, 9.00 uur	PID: definitief
42	7	17 oktober 2014	Eind sprint 0
44	8	31 oktober 2014	Demo sprint 1
45	9	7 november 2014	Terugkomdag
46	10	14 november 2014	Demo sprint 2
49	13	5 december 2014	Demo sprint 3
51	15	15 december 2014 18 december 2014	Scriptie: concept Demo sprint 4
3	17	15 januari 2015, 14.00 uur	Scriptie: definitief
4	18	23 januari 2015	Tweede bedrijfsbezoek
5	19	27 januari 2015, 10.45 uur	Afstudeerzitting CGI

BEN VAN DE WAL
2014

BIJLAGE II

Planning Custom View Engine

Taken	Prioriteit	Datum	Done
STAP 1: TAGS UITLEZEN	**MUST**	**5 december 2014**	
'Middelgroot' View Engine example downloaden en testen.	Must		
Wat kan gebruikt worden van de Pattern Lab .NET Mustache View Engine?	Must		
Renderen van Mustache tags: {{ }} variables {{> }} partials	Must		
Startpunt Pattern Lab files	Must		
Startpunt Sitecore MVC5 files	Must		
Beginpunt: renderen van ??? Mustache file met daarin {{ }}. Vervolgens een tweede file met daarin {{ }} en {{> }}.	Must		
STAP 2: SITECORE RENDERINGS / GLASS	**MUST**	**29 december 2014**	
Glass desk research, downloaden en testen.	Must		
Glass implementeren in de uitkomst van stap 1.	Must		
File uit eigen Pattern Lab website aanpassen naar Glass notatie.	Must		
Glass notatie komt terug in {{ }} variable notatie.	Must		
STAP 3: NAMESPACES TOEVOEGEN	**MUST**	**7 januari 2015**	
Onderzoeken hoe code erachterwerkt.	Must		
Met die info verder gaan. Details nog niet bekend.	Must		
STAP 4: EXTRA TAGS UITLEZEN	**COULD**	**12 januari 2015**	
Renderen van Mustache tags: {{# data-object }} {{ data-object/ }} {{^ key }} {{ key/ }} {{! }}	Could		
STAP 5: SITECORE RENDERINGS / GLASS PARAMETERS	**COULD**	**12 januari 2015**	
Verschillen in notatie onderzoeken.	Could		
Implementeren in stap 2.	Could		

BEN VAN DE WAL
2014