Afstudeerverslag

De ontwikkeling van Ezine3 voor de opmaak van online magazines

Afstudeerder:	Tim Wassenburg 20062006
Onderwijsinstelling:	De Haagse Hogeschool Johanna Westerdijkplein 75 2521EN Den Haag
Opleiding:	Informatica
Begeleidend examinator: Expert/examinator:	J.D. Maas G.M. Tuk
Afstudeerbedrijf:	Studio Projectie Haagweg 4 G5-G6 2311AA Leiden
Opdrachtgever: Begeleider:	T. Dobbelburgh M. Kucevic

Referaat

Dit afstudeerverslag is geschreven in het kader van de studie Informatica aan de Haagse Hogeschool te Den Haag. het afstudeerproject is uitgevoerd bij het bedrijf Studio Projectie te Leiden in de periode van 12 mei 2014 tot en met 6 oktober 2014.

De opdracht betreft de ontwikkeling van Ezine3.0, een applicatie waarmee het mogelijk is om online magazines op te maken.

Afstudeerder:	Tim Wassenburg
Studentnummer:	20062006
Onderwijsinstelling:	De Haagse Hogeschool
Opleiding:	Informatica
Periode:	4 en 1, 2014
Organisatie:	Studio Projectie
Bedrijfsmentor:	Miki Kucevic
Opdrachtgever:	Tijmen Dobbelburgh
Examinatoren:	J.D. Maas G.M. Tuk
Trefwoorden:	Online magazines Afstudeerverslag Laravel framework REST API AngularJS Bootstrap Overheid

Voorwoord

In dit verslag beschrijf ik het proces dat ik heb doorlopen als afstudeerder bij het bedrijf Studio Projectie. De afstudeerperiode heeft plaats gevonden van 12 mei 2014 tot en met 6 oktober 2014 en is ter afronding van mijn opleiding: HBO informatica op De Haagse Hogeschool in Den Haag. Dit verslag heb ik geschreven om inzicht te geven in het proces dat ik heb doorlopen bij de uitvoering van mijn project en de keuzes die ik daarbij heb gemaakt.

Na mijn stage in 2011 bij Microweb Edu heb ik 3 jaar full-time gewerkt als software ontwikkelaar en werkervaring opgedaan, maar ik heb besloten om verder te gaan met mijn opleiding en het af te maken. Ik vond het een prettige en leerzame ervaring om bij Studio Projectie in een nieuwe omgeving te werken. De opdracht was de ontwikkeling van een webapplicatie waarmee online magazines worden opgemaakt. Dit heb ik ervaren als een grote uitdaging omdat het erg lastig is om technische aspecten die komen kijken bij de opmaak van een magazine te vertalen naar een gebruiksvriendelijke interface voor de gebruiker. Dit in combinatie met nieuwe technieken en het toepassen van de richtlijnen die de overheid als potentiële klant heeft opgesteld. Dit alles heeft geleid tot een aantal oplossingen welke aan bod komen in de loop van dit afstudeerverslag.

Daarbij neem ik hierbij de gelegenheid om een aantal mensen te bedanken voor de plezierige afstudeerperiode, het meedenken en de begeleiding:

Tijmen Dobbelburgh	Opdrachtgever
Miki Kucevic	Bedrijfsbegeleider
J.D. Maas	Begeleidend examinator
G.M. Tuk	Expert/examinator

Tim Wassenburg,
Leiden, 7 oktober 2014

Inhoudsopgave

1 Inleiding

Onderwerp van dit afstudeerverslag is de ontwikkeling van Ezine3.0 bij het bedrijf Studio Projectie te Leiden. De opdracht wordt uitgevoerd in het kader van het afstuderen op De Haagse Hogeschool ter afronding van de opleiding Informatica.

De opdracht is om software te ontwikkelen waarmee klanten op een gebruiksvriendelijke manier een eigen online magazine op kunnen maken. De software zal de naam Ezine3.0 gaan dragen. Zoals het versienummer doet vermoeden zijn er twee eerdere versies gemaakt van Ezine, Ezine1.0 was een prototype. Ezine2.0 is in gebruik maar kent diverse beperkingen en problemen. De beperkingen en problemen van Ezine2.0 zijn aanleiding om Ezine3.0 te gaan ontwikkelen. Ezine3.0 gaat niet verder op basis van Ezine2.0 maar wordt opnieuw ontworpen en gebouwd in een modern PHP framework. Ezine3.0 moet zich gaan onderscheiden van de concurrentie door gebruiksvriendelijkheid en flexibiliteit zodat gebruikers zonder technische kennis een online magazine kunnen opmaken. Dit zal worden gemeten door willekeurige mensen zonder technische kennis de opdracht te geven om enkele magazine pagina's te laten opmaken en te kijken welke moeilijkheden er hierbij voorkomen.

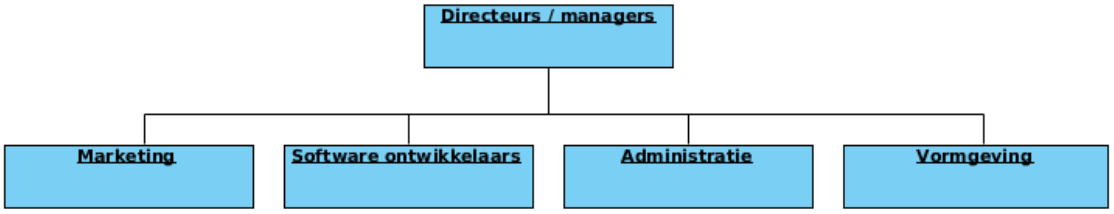
Dit verslag begint met een kennismaking met het afstudeerbedrijf Studio Projectie en Ezine3 als afstudeeropdracht voor de opmaak van online magazines. Daarna volgt de beschrijving van de voorbereiding, hierbij heb ik gekeken naar het oude Ezine2, zijn er gesprekken gevoerd om de wensen vast te stellen en zijn vervolgens de eerste wensen geformuleerd naar user stories op basis van een variant op de Scrum methodiek. Na de opstart beschrijf ik het proces van de ontwikkeling van Ezine3. Dit begint met de ontwikkeling van een eenvoudige basis gevolgd door twee spikes om oplossingen te vinden voor de onzekerheden die op dat moment heersten. Na de uitvoering van de spikes zijn de sprints uitgevoerd om de magazine editor te ontwikkelen op basis van de nieuwe bevindingen. Het verslag word afgesloten met de evaluatie, er wordt hierbij geëvalueerd op het product, het proces en de beroepstaken.

2 Studio Projectie

Studio Projectie is het bedrijf waar ik mijn afstudeeropdracht uitvoer. Het is gevestigd in Leiden in het kunstenaars en bedrijven centrum aan de Haagweg. In dit hoofdstuk ga ik in op het bedrijf Studio Projectie om de lezer context te bieden in de omgeving waar de afstudeeropdracht is uitgevoerd. Onderwerpen die in dit hoofdstuk aan bod komen zijn de organisatie, diensten, producten en mijn plek daarin als afstudeerder.

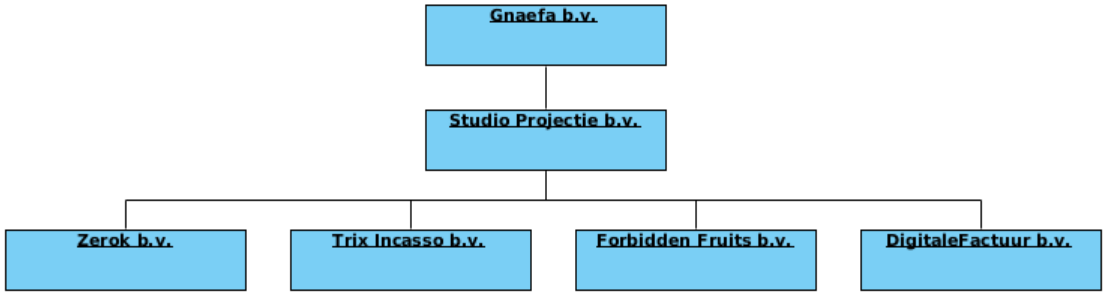
2.1 Organisatie

Studio Projectie is in 2000 opgezet door Tijmen Dobbelburgh en Philip Niewold. De core business was toen het zelfgemaakte Content Management System (CMS) Prowriter. Inmiddels is de organisatie gegroeid naar 25 medewerkers waarvan 18 software-ontwikkelaars. Er heerst een informele bedrijfscultuur met een platte hiërarchie (zie afbeelding 1). Software projecten worden voornamelijk ontwikkeld volgens een variant van de Scrum methodiek. Op het moment van schrijven zijn er drie stagiaires en twee afstudeerders waarvan ik er één ben.



Afbeelding 1: de hiërarchie van Studio Projectie.

Studio Projectie is de overkoepelende organisatie voor de bedrijven: Zerok, Trix, Forbidden Fruits en DigitaleFactuur (zie afbeelding 2). Deze bedrijven beheren ieder een eigen product. Studio Projectie heeft ook zijn eigen producten en diensten en wordt weer overkoepeld door Gnaefa, maar Gnaefa. bestaat alleen op papier om belastingtechnische redenen.



Afbeelding 2: de bedrijven die onderdeel zijn van Studio Projectie.

2.2 Producten

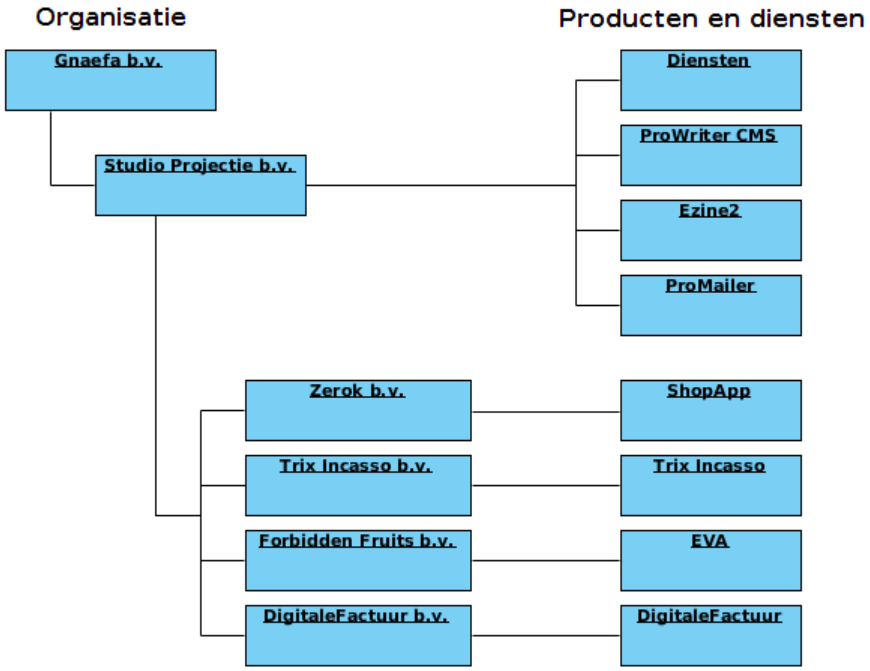
Na de start met het Prowriter CMS is later DigitaleFactuur erbij gekomen. DigitaleFactuur heeft een stabiele groei doorgemaakt en is inmiddels gegroeid tot de nieuwe core business van Studio Projectie en wordt beheerd door DigitaleFactuur b.v als onderdeel van Studio Projectie. Met DigitaleFactuur kunnen MKB, ZZP'ers en multinationals eenvoudig facturen en offertes opstellen in eigen huisstijl en versturen naar hun klanten. In opdracht van de overheid heeft Studio Projectie, op basis van DigitaleFactuur, het E-factuurportaal gebouwd. Via dit portaal kunnen ondernemers eenvoudig elektronische facturen naar de rijksoverheid sturen.

Een ander product waar de overheid interesse in heeft is Ezine. Met het huidige Ezine2 is het mogelijk om online magazines op te maken en te publiceren op internet.

Naast het DigitaleFactuur en Ezine2 biedt Studio Projectie nog een aantal SaaS (Software as a Service) producten, zoals:

- EVA: een evaluatie omgeving voor het onderwijs (er lopen op dit moment twee pilots op De Haagse Hogeschool)
- Trix Incasso: voor het starten van een incasso traject.
- ShopApp: een app dat kortingacties verzamelt in de omgeving van de gebruiker.

Deze producten worden beheerd door de bedrijven die onder Studio Projectie vallen (zie afbeelding 3).



Afbeelding 3: de organisatiestructuur en de producten die daaraan zijn gekoppeld.

2.3 Diensten

Studio Projectie zelf biedt daarnaast diverse diensten op het gebied van mobile- en webdevelopment, ontwerp en marketing (zie afbeelding 4). Klanten kunnen een huisstijl laten ontwerpen, een website of een app laten ontwikkelen en deze in de bekendheid brengen door middel van online marketing op social media en search engine optimalisatie (SEO). Op die manier worden de technische aspecten overgenomen van de klant en kan de klant zich richten op inhoudelijke zaken. Op dit moment wordt er bijvoorbeeld een nieuwe website ontwikkeld voor de studentenvereniging Minerva in Leiden en krijgt de website van Akzo Sikkens een update.

Afbeelding 4: diensten van Studio Projectie.

Het huidige Ezine2 valt ook voor een groot deel onder de diensten. De mogelijkheden voor de opmaak zijn beperkt waardoor veel wensen van de klant als dienst moeten worden gerealiseerd. Dit betekent dat Studio Projectie het magazine opmaakt en invult volgens de wensen van de klant. Dit is bijvoorbeeld gedaan voor de overheid met het magazine[1] in Ezine2. De bedoeling is dat het nieuwe Ezine3 minder een dienst wordt en dat de klant zijn eigen online magazine kan opmaken. Daarnaast is het de bedoeling dat de magazines voldoen aan de richtlijnen[2] van de overheid, om deze als klant aan te trekken.

[1] http://tappan.e-zine.eu
[2] http://www.rijkshuisstijl.nl/prototype-ezines

3 De opdracht: Ezine 3.0

De opdracht is om Ezine3 opnieuw te ontwikkelen in een modern framework met de wensen van de overheid.

Het huidige Ezine2 is ontwikkeld op Prowriter. Prowriter is het content management system (CMS) van Studio Projectie dat in 2000 is ontwikkeld met het doel om op een relatief eenvoudige manier een website te beheren. Dat doel is bereikt en daarom wordt het nog steeds gebruikt voor eenvoudige websites. Het idee was om dezelfde eenvoud aan te bieden voor de opmaak van online magazines in Ezine2, waardoor Prowriter als basis een logische keuze leek. In de praktijk blijkt echter dat ProWriter niet de flexibiliteit biedt die bij de opmaak van online magazines gewenst is. Het is bijvoorbeeld niet uit te breiden met nieuwe functionaliteit met als gevolg dat de structuur vanuit technisch oogpunt niet naar wens is. Voor complexe websites maakt Studio Projectie daarom tegenwoordig gebruik van de geavanceerde frameworks Yii en Laravel. Deze frameworks bieden oplossingen voor veel voorkomende functionaliteiten in de vorm van plugins en zijn veel flexibeler dan een CMS waar de structuur al grotendeels is vastgesteld.

Met de overheid als potentiële klant is besloten om Ezine opnieuw te ontwikkelen in een modern framework. Studio Projectie wil de ontwikkeling van Ezine combineren met kennisverbreding van het Laravel framework; een opkomend framework dat gericht is op de toekomst. Studio Projectie is bekend met het framework en weet inmiddels welke mogelijkheden het biedt wat voordelig is bij de ontwikkeling van Ezine3.

Het doel is dat Ezine3:

- beschikt over basis functionaliteit voor de opmaak van online magazines.
- wordt ontwikkeld in het Laravel framework.
- een nieuw technisch ontwerp heeft met de focus op uitbreidbaarheid.
- magazines opstelt die voldoen aan de richtlijnen van de overheid.

Het eindresultaat is een robuust systeem voor het maken van online magazines, dat zich onderscheidt door gebruiksvriendelijkheid naar de gebruiker.

4 Opstellen Plan van Aanpak

Naar aanleiding van de opdracht om Ezine3 te ontwikkelen, ben ik begonnen met het opstellen van het Plan van Aanpak (zie bijlage). In dit hoofdstuk licht ik de belangrijkste keuzes toe die tijdens dit proces zijn gemaakt. Het gaat hierbij om keuzes die betrekking hebben op de ontwikkelmethode, het framework, de huidige situatie, het ontwikkelteam en de planning.

4.1 Ontwikkelmethode

Bij het opstellen van het afstudeerplan gaf mijn begeleider aan dat het bedrijf met Scrum werkt en dat ze de wens hebben dat dit project ook volgens de Scrum methodiek wordt uitgevoerd. Omdat ik bij verschillende ICT bedrijven heb gewerkt, welke ook Scrum gebruikte, weet ik uit ervaring dat er over het algemeen een verschil zit tussen Scrum in de theorie en Scrum in de praktijk. Het vermoeden dat het werken met Scrum problemen met zich zal meebrengen, werd versterkt omdat ik (voornamelijk) alleen werk aan het project, terwijl Scrum een team methodiek is. Meestal worden in de praktijk alleen bepaalde elementen gebruikt van Scrum, afhankelijk van de situatie. Hier is, naar mijn mening, niks mis mee. In kleinere bedrijven heeft een aangepaste versie van scrum zelfs voordelen, omdat je anders teveel tijd kwijt bent aan het klassieke scrummen zonder dat het meetbare voordelen oplevert. Het belangrijkste is dat er afspraken worden gemaakt over de manier van werken, waar de betrokkenen tevreden mee zijn, en dat het gewenste resultaat wordt bereikt.

Ik ben in gesprek gegaan om te kijken welke wensen er zijn bij de ontwikkeling van Ezine, zodat we hier rekening mee kunnen houden. Het is belangrijk dat de werkwijze aansluit bij de bedrijfscultuur. We kwamen daarbij uit op de volgende wensen.

- Omdat er nog enkele onzekerheden zijn moet er de mogelijkheid zijn om requirements bij te stellen.
- Periodiek overleg met de opdrachtgever en begeleider.
- Regelmatig overleg met medeontwikkelaars.
- De mogelijkheid om extra mensen in te zetten op het project.
- Tussentijds producten opleveren.

Op basis hiervan is afgesproken om te werken met korte sprints van 2 weken, user stories op te stellen en deze te prioriteren. Aan het einde van een sprint wordt samen met de opdrachtgever en de begeleider het huidige product doorgenomen en worden de user stories opnieuw bekeken om de inhoud van de volgende sprint te bepalen. Er worden dus, zoals verwacht, niet alle elementen gebruikt uit het klassieke Scrum. Daily scrums worden bijvoorbeeld niet toegepast en ik ben de enige ontwikkelaar in het project.

4.2 Laravel framework

Tijdens het kennismakingsgesprek werd aangegeven dat het bedrijf met twee PHP frameworks werkt, namelijk met Yii en Laravel. Later, bij het bespreken van het

afstudeerplan, werd duidelijk dat er werd verwacht dat Ezine3.0 in Laravel wordt ontwikkeld. Ik heb beperkte ervaring met beide frameworks en ik vroeg mij daarom af waarom Laravel de voorkeur krijgt bij dit project. Als daar geen duidelijke motivatie voor is, kan het nuttig zijn om kort onderzoek te doen om te kijken of Laravel daadwerkelijk geschikter is voor Ezine3.0 dan Yii.

Tijdens de gesprekken werd duidelijk dat de keuze voor Laravel is gemaakt omdat Laravel eenvoudiger te gebruiken is voor de front-end ontwikkelaars. De widgets van Yii worden vaak als lastig ervaren omdat hier meer technische kennis bij vereist is. De kans is groot dat er in de levensloop van Ezine3 frontend ontwikkelaars aan zullen werken, het is daarom nuttig om hier rekening mee te houden.

4.3 Inventarisatie Ezine2.0

Voordat ik het plan van aanpak heb opgesteld, heb ik eerst een korte inventarisatie gedaan van Ezine2.0 (de huidige versie), om een idee te krijgen wat er mis gaat, wat de knelpunten zijn en wat er beter kan. Ik heb dit eerst gedaan met de opdrachtgever en later met mijn begeleider.

Tijdens het opstellen van de afstudeeropdracht werd duidelijk dat Studio Projectie niet tevreden is met de architectuur van Ezine2.0 en dat het concept te complex is voor ProWriter. Door de functionaliteit en source code van Ezine2.0 te bekijken werd mij duidelijk wat de oorzaak van deze problemen is. ProWriter is geen framework zoals werd gesuggereerd tijdens het opstellen van het originele afstudeerplan (zie bijlage), Prowriter is een content management system (CMS) (ik heb later de opdrachtsomschrijving in dit verslag aangepast op deze nieuwe informatie). Omdat het een CMS betreft, en er dus een CMS structuur aanwezig is, wordt er een werkwijze afgedwongen en is er geen volledige vrijheid om een nieuwe architectuur te implementeren. Er moesten regelmatig omwegen worden gebouwd om op maat gemaakte functionaliteit te kunnen implementeren. Hierdoor is het moeilijk te beheren en is de ontwikkeling uiteindelijk vastgelopen. Het resultaat is dat er nog veel fouten in zitten. Ezine3.0 herontwikkelen in een modern framework is daarom een verstandige keuze. Een modern framework geeft meer vrijheid op het gebied van architectuur en zal naar verwachting een beter eindproduct opleveren.

4.4 Samenstellen expertiseteam

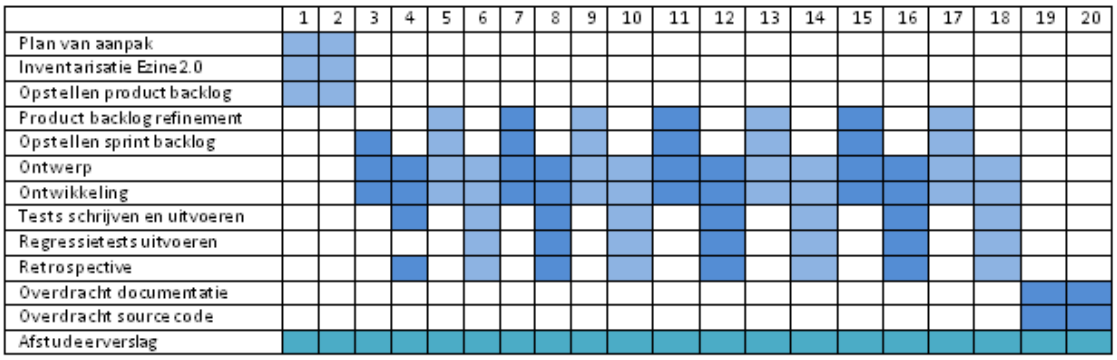
Op aanraden van mijn begeleider heb ik een expertiseteam samengesteld. De bedoeling is dat ik bij deze collega's terecht kan met vragen en feedback. Ik ben begonnen met bepalen welke disciplines ik nodig heb in het team. Op basis van het afstudeerplan kwam ik uit op de volgende disciplines: ontwikkelaar, grafisch ontwerper, architect en een tester. Ik ben daarna in gesprek gegaan met collega's om ze te leren kennen en er achter te komen wat hun werkzaamheden zijn. Ook heb ik mijn begeleider gevraagd voor advies om te kijken wie ik zou kunnen vragen. Vervolgens ben ik weer in gesprek gegaan, heb ik uitleg gegeven over de opdracht en gevraagd of ze tijd/interesse hadden om deel uit te maken van het team. Op die manier heb ik uiteindelijk het volgende expertiseteam samengesteld:

- Ontwikkelaar: Tjeerd Ritsma

- Designer: Miki Kucevic en Marcel van Werkhoven
- Architect: Martijn Zuijderduijn
- Tester: Yoran Meijer

4.5 Planning

De planning is gebaseerd op sprints. Aan het begin van iedere sprint wordt de sprint backlog opgesteld op basis van de prioritering van de product backlog. De sprint backlog bevat de taken die worden uitgevoerd in de betreffende sprint. De lengte van de sprints is vastgesteld op 2 weken zodat we korte sprints hebben die regelmatig worden besproken en de opdrachtgever regelmatig de mogelijkheid heeft tot bijsturing. Per sprint worden de requirements uitgevoerd volgens de verschillende disciplines zoals in onderstaande planning is te zien.



Afbeelding 5: de planning.

5 Opstellen van requirements

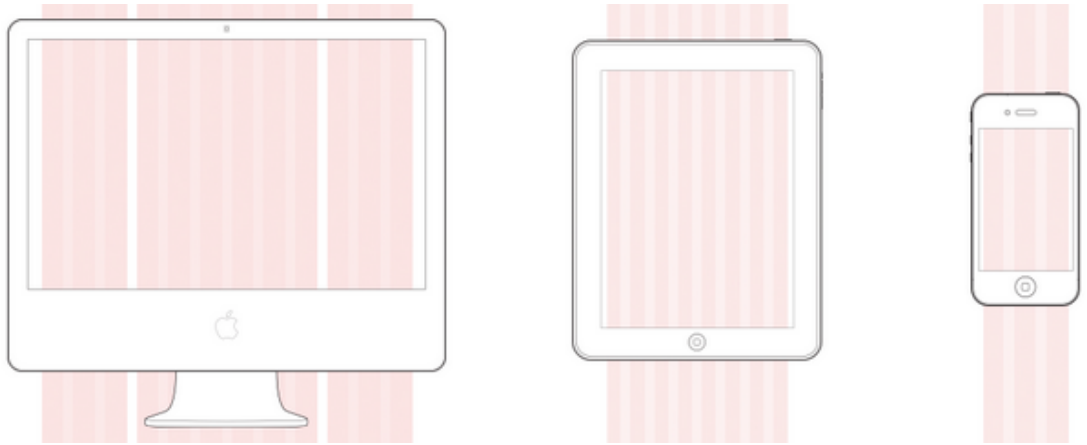
Omdat er gebruik wordt gemaakt van een Scrum variant, worden de requirements opgesteld in de vorm van user stories. Aan de hand van de prioritering kan worden bepaald welke user stories als eerste worden uitgevoerd. De story points geven een indicatie van de hoeveelheid werk om een user story te realiseren.

5.1 Bronmateriaal

Voor het opstellen van de user stories heb ik gebruik gemaakt van de volgende bronnen:

- De opdrachtbeschrijving
- Gesprekken met de opdrachtgever
- Gesprekken met mijn begeleider
- Inventarisatie van Ezine2.0
- Richtlijnen van de overheid

De eerste wensen werden duidelijk gemaakt bij het opstellen van de opdrachtbeschrijving. Om deze wensen verder te detailleren, ben ik in gesprek gegaan met mijn opdrachtgever. Ook heb ik samen met de opdrachtgever het oude Ezine2.0 doorgenomen om een idee te krijgen van de knelpunten en functionaliteiten. Vervolgens heb ik hetzelfde gedaan met mijn begeleider om zijn kant van het verhaal te horen. Uiteindelijk heb ik het oude Ezine2.0 op mijn eigen computer geïnstalleerd om te kijken of ik er nog een aantal user stories uit kon halen (deze zijn later nog een keer besproken met de opdrachtgever om te kijken of het wel wenselijk is om deze over te nemen). Als laatste heb ik de richtlijnen[3] bekeken die de overheid heeft opgesteld. Een richtlijn die de overheid stelt is bijvoorbeeld het gebruik van een 12 punts grid systeem dat op computer, tablet en smartphone werkt (zie afbeelding 6), zodat een magazine onafhankelijk is van het platform, en over een swipe mogelijkheid beschikt op touch screens.



Afbeelding 6: Het grid systeem volgens de richtlijnen van de overheid.

[3] http://www.rijkshuisstijl.nl/prototype-ezines

5.2 Prioritering

De prioritering heb ik samen met de opdrachtgever gedaan. De opdrachtgever legde uit dat dit gebeurt door de user stories te beoordelen op basis van: afhankelijkheden(A), minimum marketable features(MMF), business value(BV) en risico(R). Uiteindelijk komt hier een eindbeoordeling uit met een lage, middel of hoge prioriteit. Bij deze eindbeoordeling komt geen wiskundige formule kijken, maar is het van belang dat er een beoordeling uitkomt waar ontwikkelaar en opdrachtgever zich goed in kunnen vinden. In het begin heb ik bijvoorbeeld aangegeven dat de afhankelijkheid voor mij zwaar weegt, omdat het anders een blokkade kan vormen voor de realisering van andere user stories.

De bepaling van de afhankelijkheid was lastig omdat er nog niks op papier stond. Bij de bepaling van de afhankelijkheid heb ik daarom naar de benodigde klassen bekeken van een functionaliteit. Op die manier werd het duidelijk dat bijvoorbeeld de pagina's van een magazine afhankelijk zijn van het magazine zelf, wat leidt tot een hogere prioriteit voor user stories voor het magazine. Bij de bepaling van het risico heb ik een schatting gemaakt van hoe groot de kans is op uitloop: een makkelijke user story heeft weinig kans op uitloop terwijl een moeilijkere meer risico geeft op uitloop. Door de moeilijke user stories eerst uit te voeren is het makkelijker om de voortgang van het project onder controle te houden. Als er namelijk nog veel moeilijke user stories open staan tegen het einde van deze afstudeerperiode, is het risico namelijk groot dat het niet meer gaat lukken om het project af te ronden. De schatting van de marketable features en business value heb ik vooral door de opdrachtgever laten, doen omdat hij daar beter zicht op heeft.

Uiteindelijk is op deze manier de prioritering aangepakt van de user stories, zoals hieronder is weergegeven. De beoordelingen zijn als richtlijn gebruikt voor de eindbeoordeling, wat in dit geval Hoog is, en dus wordt opgenomen in de eerstvolgende sprint.

User story	**A**	**MMF**	**BV**	**R**
Als klant wil ik een magazine kunnen maken zodat ik deze online kan aanbieden.	H	H	H	**L**

5.3 Toekennen van story points

Naast de prioritering heb ik story points toegekend aan de user stories. De story points zijn niet gebaseerd op tijd, maar geven een indicatie van inspanning, risico en complexiteit. Als ik verwacht dat een user story hoog scoort op deze punten zal dat resulteren in een hoog aantal story points op een schaal van 1 tot 5.

De user stories met een hoge prioritering en een hoog aantal story points ben ik verder gaan opdelen. Door ze verder op te delen in kleinere user stories daalt de inspanning, risico en complexiteit per user story en wordt de kans dat een user story succesvol wordt uitgevoerd tijdens een sprint vergroot.

De user stories met een lage prioriteit maar een hoog aantal story points laat ik nog even zoals ze zijn. Deze keuze heb ik mede genomen op basis van het advies van Nicole de Swart[4] "Add details at the last responsible moment". Op die manier blijft de product backlog overzichtelijk. Als de sprints vorderen en de prioriteit hoog genoeg is om het op te nemen in de sprint backlog, zal ik de user story verder uitwerken in kleinere user stories. Samengevat:

Prioriteit	**Story Points**	**Actie**
Hoog	Laag	Opnemen in sprint backlog
Hoog	Hoog	Verder detailleren
Laag	Laag	Geen
Laag	Hoog	Geen

5.4 Resultaat

Het resultaat is de eerste versie van de product backlog (zie voorbeeld, een klein deel van de product backlog. Zie het requirements document in de bijlage voor de complete product backlog).

User story	**Prioriteit**	**Story points**
Als gebruiker wil ik een nieuw magazine kunnen aanmaken.	H	2
Als gebruiker wil een magazine beschikbaar kunnen stellen voor mijn publiek.	H	1
Als gebruiker wil ik een overzicht kunnen opvragen van mijn magazines zodat ik mijn magazines kan beheren.	H	2
Als gebruiker wil ik pagina's toe kunnen voegen aan mijn magazine.	H	3
Als gebruiker wil ik pagina's toe kunnen verwijderen.	H	2

De product backlog is een product van een iteratief systeemontwikkeltraject. Dit betekent dat het niet, zoals bij de watervalmethode, meteen klaar is en vast staat. Het proces dat in dit hoofdstuk is beschreven van bronmateriaal, prioritering en story points, is bij de start van iedere nieuwe sprint opnieuw doorlopen.

[4]Nicole de Swart - Handboek requirements

6 Sprint 1: de basis

Omdat er nog geen basis was kwamen de user stories met een hoge afhankelijkheid voor andere onderdelen snel bovenaan te staan in de product backlog. De user stories worden in de sprint opgenomen in de sprint backlog en op basis hiervan heb ik een ontwerp gemaakt en de software ontwikkeld. In deze sprint worden ook enkele elementen van Laravel framework toegelicht om de lezer meer context te bieden in het proces van de ontwikkeling. In latere sprints zal hier minder aandacht aan worden besteed. De sprint eindigt met het testen van de applicatie.

6.1 Opstellen sprint backlog

Bij het opstellen van de sprint backlog heb ik de user stories gekozen met de hoogste prioritering. Aan het einde van de sprint worden op die manier de user stories opgeleverd die de opdrachtgever het belangrijkst vindt of nodig zijn voor het verdere verloop van de ontwikkeling.

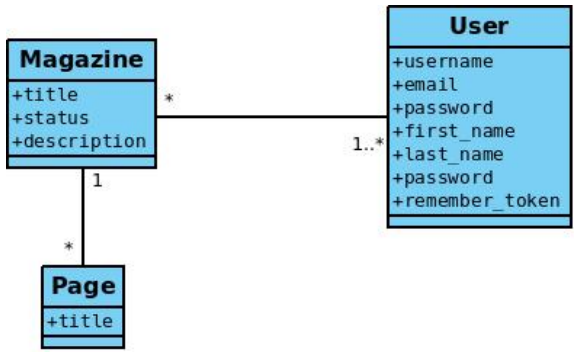
Uiteindelijk heb ik de volgende sprint backlog opgesteld voor sprint 1:

User story	**Prioriteit**	**Story points**
Als gebruiker wil ik een nieuw magazine kunnen aanmaken.	H	2
Als gebruiker wil een magazine beschikbaar kunnen stellen voor mijn publiek.	H	1
Als gebruiker wil ik een overzicht kunnen opvragen van mijn magazines zodat ik mijn magazines kan beheren.	H	2
Als gebruiker wil ik pagina's toe kunnen voegen aan mijn magazine.	H	3
Als gebruiker wil ik pagina's toe kunnen verwijderen.	H	2
Als gebruiker wil ik in kunnen loggen op mijn account om mijn magazines te kunnen beheren.	H	1
Als gebruiker wil ik een magazine kunnen opslaan als schets zodat ik kan overleggen voordat het magazine word gepubliseerd.	M	2
Als gebruiker wil ik voor een magazine een begin en einddatum kunnen instellen zodat de magazine alleen in een bepaalde periode online is te bekijken.	M	2

In de sprint backlog wordt duidelijk dat in deze sprint focus ligt op het begin van het aanmaken, beheren en laten zien van de magazines aan de bezoeker. Hieruit kan worden worden geconcludeerd dat er een beheer gedeelte moet worden gemaakt voor de magazines en een publieke voorkant waarop bezoekers de magazines kunnen bekijken. Dit vormt een belangrijke basis voor de rest van de applicatie.

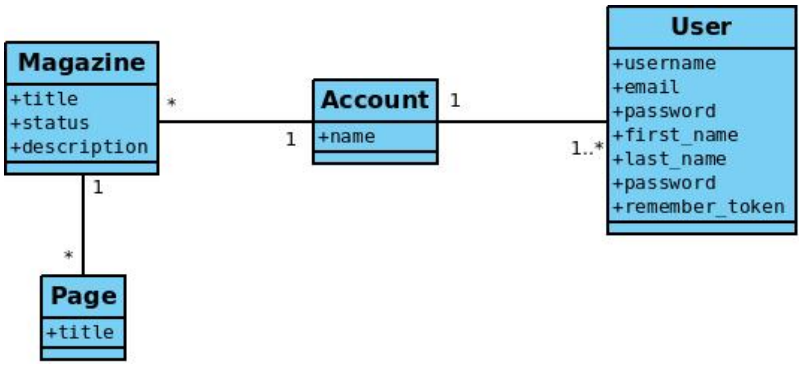
6.2 Klassendiagram

Bij het ontwerp van het klassendiagram kwam ik er achter dat ik tijdens het opstellen van de sprint backlog iets over het hoofd heb gezien. De user story in de product backlog dat een account meerdere gebruikers kan hebben hadden we beoordeeld met een lage afhankelijkheid en daarom nog niet opgenomen in de sprint backlog. Bij het bepalen van de afhankelijkheid had ik al naar de benodigde klassen gekeken die horen bij de functionaliteit. Maar toen het klassendiagram op papier stond kwam ik er achter dat de afhankelijkheid van de accounts hoger blijkt te zijn. Uit de user stories in de sprint backlog kan worden opgemaakt dat een klant gekoppeld is aan magazines. Dit zou zich vertalen naar een class User die een 1 op meer relatie heeft met de class Magazine (zie afbeelding 7).



Afbeelding 7: eenvoudig begin van het klassendiagram.

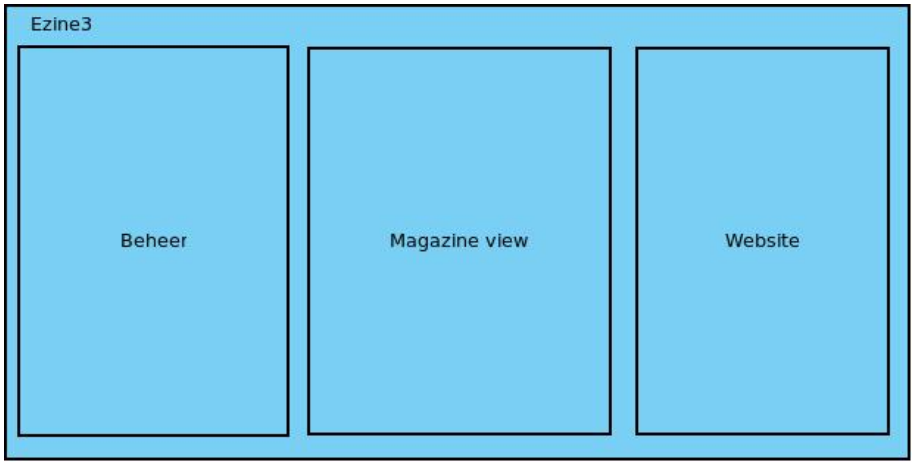
Dit leidt ertoe dat ik in een latere sprint bij het implementeren dat een account meerdere gebruikers kan hebben de relaties moet omzetten van een User met Magazines naar Magazines gekoppeld aan een Account, en een Account een relatie aan een user. Om dit voor te zijn heb ik alvast besloten om de accounts alvast op te nemen in het klassendiagram (zie afbeelding 8) zodat ik bij een komende sprint niks hoef om te zetten bij de user story dat een account meerdere gebruikers kan hebben.



Afbeelding 8: het klassendiagram uitgebreid met accounts.

6.3 Project onderdelen

Uit user story *Als klant wil een magazine beschikbaar kunnen stellen voor mijn publiek* blijkt dat er een publiekelijk pagina moet komen waar bezoekers een magazine kunnen lezen (magazine view in afbeelding 50). En uit user story *Als klant wil ik in kunnen loggen op mijn account om mijn magazines te kunnen beheren* een beheer gedeelte om de magazines in te beheren(beheer in afbeelding 50). Daarnaast moet de service uiteindelijk worden verkocht via een website(website in afbeelding 50). Dit was mondeling besproken dus ik heb alvast het initiatief genomen om daar ruimte voor te maken binnen de applicatie.

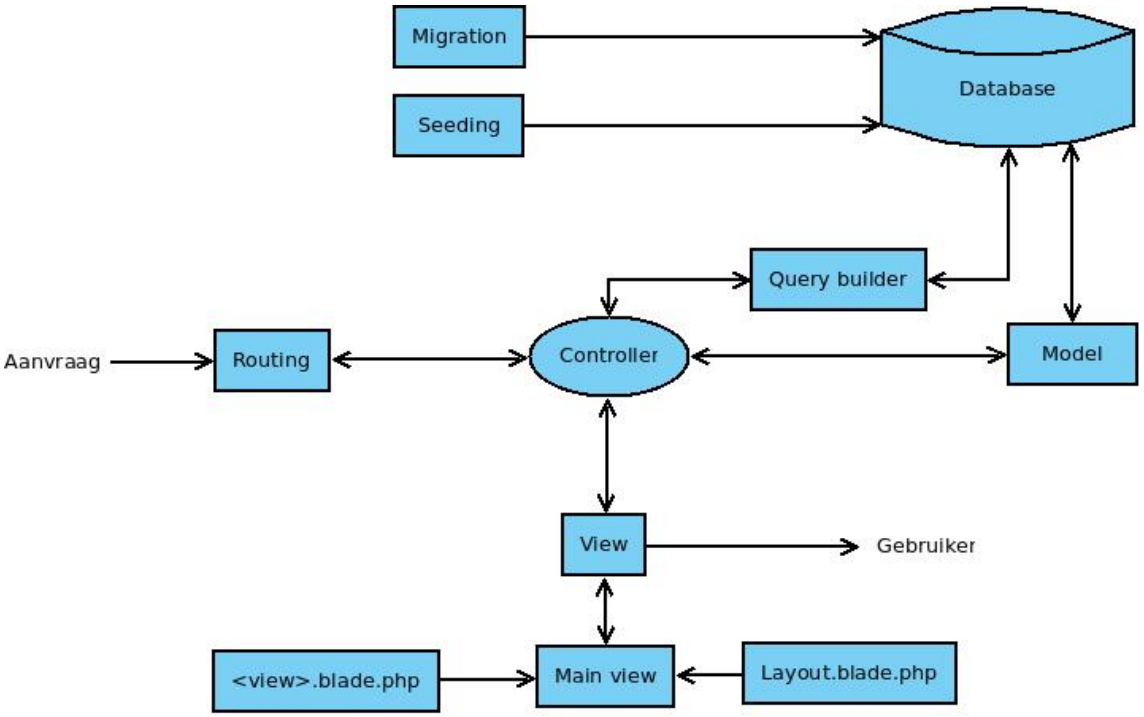


Afbeelding 9: De drie onderdelen waar het project uit bestaat.

De magazine view en het beheer gedeelte zijn eigenlijk twee aparte applicaties die de dezelfde data delen. In het beheer gedeelte krijgt een gebruiker de mogelijkheid om een magazine te maken en te publiceren. Het magazine kan vervolgens via een specifieke url worden bekeken in de magazine view door de bezoekers. De website bevat uitleg en marketing om de verkoop van het product te bevorderen. De bepaling van de inhoud van de website wordt gedaan door collega's met marketing en design als expertise. Het technische deel neem ik voor mijn rekening.

6.4 Het Laravel framework

Om de toekomstige onderdelen te beschrijven is het nuttig om eerst de structuur van het Laravel framework te bekijken. Het Laravel framework is een MVC framework (zie afbeelding 10) met een aantal speciale eigenschappen die de uitwerking van de software beïnvloeden. Daarom wil ik deze eerst in het kort toelichten.



Afbeelding 10: Architectuur van het Laravel framework

6.4.1 Model, View en RESTful controllers

Laravel gebruikt zoals de meeste PHP frameworks de Model-View-Controller(MVC) architectuur om de data met de interface te communiceren. Er wordt gebruik gemaakt van object-relational mapping (ORM) via Eloquent, tot nu toe niks bijzonders vergeleken met andere frameworks. Wat wel interessant is, is het gebruik van RESTful resource controllers. Een resource controller bevat een standaard set van RESTful actions om CRUD bewerkingen uit te voeren. Een software ontwikkelaar weet dus bij het zien van een resource controller precies wat hij kan verwachten. Voor een verder begrip van resource controllers is het nuttig om verder naar de routing te kijken.

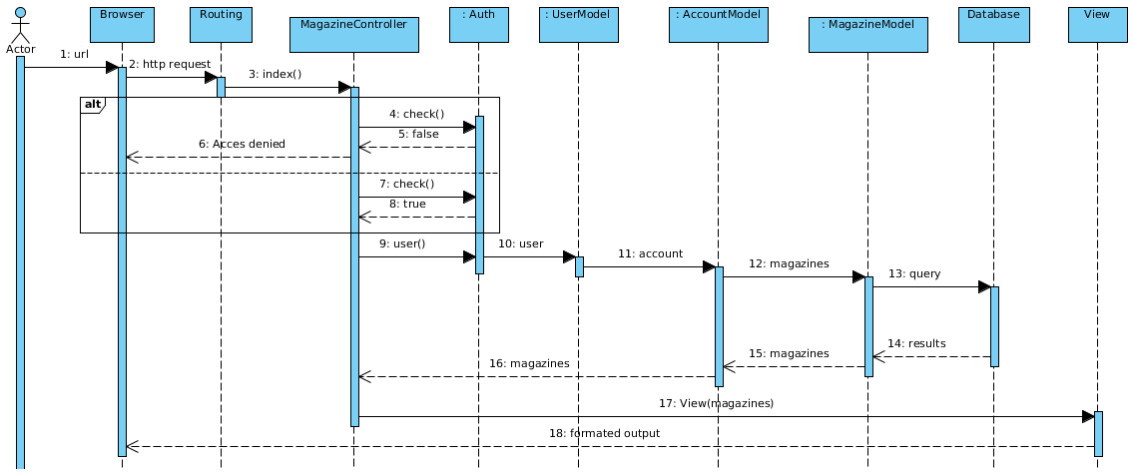
6.4.2 Routing

De routing van Laravel bepaald waar de gebruiker heen gaat bij de invoer van een url in de browser (zie afbeelding 11).

Verb	Path	Action	Route Name
GET	/resource	index	resource.index
GET	/resource/create	create	resource.create
POST	/resource	store	resource.store
GET	/resource/{resource}	show	resource.show
GET	/resource/{resource}/edit	edit	resource.edit
PUT/PATCH	/resource/{resource}	update	resource.update
DELETE	/resource/{resource}	destroy	resource.destroy

Afbeelding 11: REST API.

Als ik bijvoorbeeld de magazine controller declareer als resource in de routing kan een gebruiker met de url ezine.nl/magazines een overzicht opvragen van magazines omdat op basis van de url de index() in de controller wordt aangeroepen (zie afbeelding 52).



Afbeelding 12: Magazine overzicht aanvragen

Maar voordat dit mogelijk is moet ik een database aanmaken met een magazines table. Laravel heeft hier ook zijn eigen manier voor doormiddel van het migration script (zie afbeelding 10).

6.4.3 Migration en seeders

Vaak is het gebruikelijk om een SQL bestand toe te voegen aan een project met de structuur van de database opgesteld in SQL code. Laravel heeft hier een betere oplossing voor doormiddel van migration en seeders. De migration bevat de structuur van de database welke is beschreven in php code. De seeders zijn bedoelt om dummy data in de database te plaatsen, dit is handig bij het testen. Maar ze kunnen ook worden uitgevoerd via de command line, op die manier kunnen SQL wijzigingen snel en efficiënt worden doorgevoerd en worden aanpassingen minder snel vergeten met versiebeheer.

6.5 Model en relatie

Ik ben begonnen met het aanmaken van de models op basis van het klassendiagram (zie afbeelding 8). Omdat het Laravel ORM Eloquent automatisch de attributen uit de database tabellen gebruikt als attributen in de models was het niet nodig om deze te declareren in de models. Wat ik wel heb gedeclareerd waren de relaties tussen de models (zie afbeelding 13).

```php
<?php

class Account extends Eloquent {

    public function users()
    {
        return $this->hasMany('User');
    }

    public function magazines()
    {
        return $this->hasMany('Magazine');
    }

}
```

```php
<?php

class Magazine extends Eloquent {

    public function account()
    {
        return $this->belongsTo('Account');
    }

    public function pages()
    {
        return $this->hasMany('Page');
    }

}
```

Afbeelding 13: de één op meer relatie tussen Account en Magazine.

Doormiddel van het Eloquent ORM kan ik op basis van een model een object aanmaken. Het object erft de functies over van het Eloquent ORM waardoor ik bijvoorbeeld de functie save() kan aanroepen om een object direct weg te schrijven naar de database. Om hier gebruik van te maken heb ik de database aangemaakt doormiddel van de eerder besproken migrations.

6.6 Database opzetten

Na het vastleggen van het datamodel in de models ben ik de database op gaan zetten in het migration script. Dit script bevat slechts twee functies per database tabel, namelijk up() en down(). Down() doet niks anders dan de tabel verwijderen, up() maakt de tabel aan. Via de documentatie leerde ik dat Laravel ook in dit script hulp biedt doormiddel van de schema builder. De schema builder is een class met functies voor het aanmaken van een database tabel. De functie create() wordt via een static call aangeroepen en heeft als eerste de parameter de tabel naam, als tweede parameter bevat het een anonieme functie met de tabel eigenschappen zoals attributen en relaties (zie afbeelding 14).

```php
public function up()
{
    Schema::create('accounts', function($table)
    {
        $table->increments('id');
        $table->string('name');
        $table->timestamps();
    });
}
```

Afbeelding 14: de aanmaak van de accounts table in het Laravel migratie script.

Via dit migratiescript heb ik de database tabellen: pages, magazines, accounts en users aangemaakt. De attributen die ik heb opgegeven in de migratie kan ik nu in de controllers aanroepen aanroepen in objecten gemaakt op basis van het model, bijvoorbeeld de name van een account object.

6.7 Resource controllers, views en routes

Toen de data aanspreekbaar was doormiddel van de models en MySQL database heb ik de resource controllers aangemaakt. Dit heb ik gedaan volgens de Laravel documentatie[5] met het *controller:make* commando in de Command Line Interface (CLI), het voordeel van deze manier is dat Laravel een controller template neerzet met de RESTful functies zoals beschreven in afbeelding 11. Het enige wat ik nu hoef te doen is functies functioneel maken, want ze bevatten nog geen code. Maar eerst heb ik de routes ingesteld om de controllers bereikbaar te maken, anders kan ik niet zien of de code in de controller werkt (zie afbeelding 15).

```php
Route::resource('login', 'AuthenticationController', array('only' => array('create', 'store', 'destroy')) );
Route::resource('registration', 'RegistrationController', array('only' => array('create', 'store'));

Route::group(array('prefix' => 'admin', 'before' => 'auth'), function()
{
    Route::resource('magazines', 'MagazineController' );
    Route::resource('magazines.pages', 'PageController');
}
```

Afbeelding 15: De routes

Op de routes van de authenticatie controller heb ik restricties ingesteld zodat alleen de benodigde functies aangesproken kunnen worden. De routes van magazines en pages heb ik in een groep gezet met de ”admin” prefix zodat het duidelijk is dat het om het beheer gedeelte gaat van de magazines. Om bijvoorbeeld de magazines te kunnen zien wordt nu de url: *http://ezine.dev/admin/magazines* gebruikt. Daarnaast heb ik op de groep een authenticatie filter ingesteld zodat de magazine- en page controllers alleen aanspreekbaar zijn na authenticatie, dit is een standaard filter van het Laravel framework.

Toen controllers bereikbaar waren via de routes ben ik aan de CRUD functionaliteit gaan werken in de resource controllers. Als voorbeeld neem ik de aanmaak van een nieuw magazine via de Magazine resource controller (zie afbeelding 16). Ten eerste wordt er gecontroleerd of de gebruiker wel ingelogd is. Vervolgens geef ik in de validator aan dat bij het aanmaken van een magazine op zijn minst de titel moet zijn ingevuld. Als dat het geval is wordt er een nieuw magazine object aangemaakt, worden de gegevens ingevuld

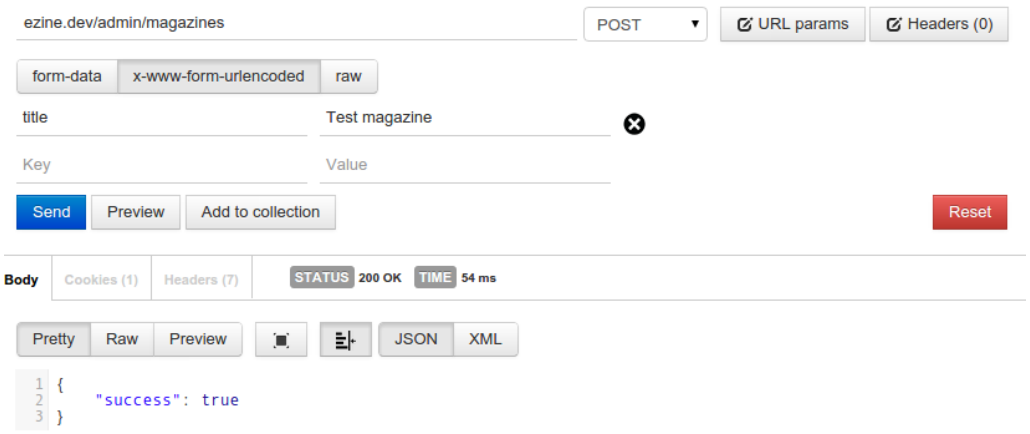
[5] http://laravel.com/docs/controllers

en opgeslagen in de database via het Eloquent ORM. Omdat er nog geen grafische interface was heb ik gebruik gemaakt van de Google Chrome extentie: Postman. Postman is een REST client waarmee REST requests verstuurd kunnen worden. Op die manier kan ik snel controleren of de functie doet wat het moet doen (zie afbeelding 17).

Op deze manier heb ik CRUD functionaliteit opgezet voor magazines en pages, daarnaast kunnen gebruikers inloggen via de authentication controller. Het technische gedeelte van de basis is functioneel en klaar om te communiceren met een grafische interface.

```php
/**
 * Store a newly created magazine in storage.
 *
 * @return Response
 */
public function store()
{
    // check if user is authenticated
    if (!Auth::check()) {
        return Response::json(array('message' => 'Permission denied'));
    }

    // validate rules
    $rules = array(
        'title' => 'required'
    );
    $validator = Validator::make(Input::all(), $rules);

    // validate values
    if ($validator->fails()) {
        return Response::json(array('message' => 'Validation error'));
    } else {
        // store
        $magazine = new Magazine;
        $magazine->title = ucfirst(Input::get('title'));
        $magazine->status = 'Online';
        $magazine->description = Input::get('description');
        $magazine->account_id = Auth::user()->account->id;
        $magazine->save();

        return Response::json(array('success' => true));
    }
}
```

Afbeelding 16: De Store() functie voor het opslaan van nieuwe magazines



Afbeelding 17: De aanroep van de store functie via een POST in de magazineController

6.8 Interface

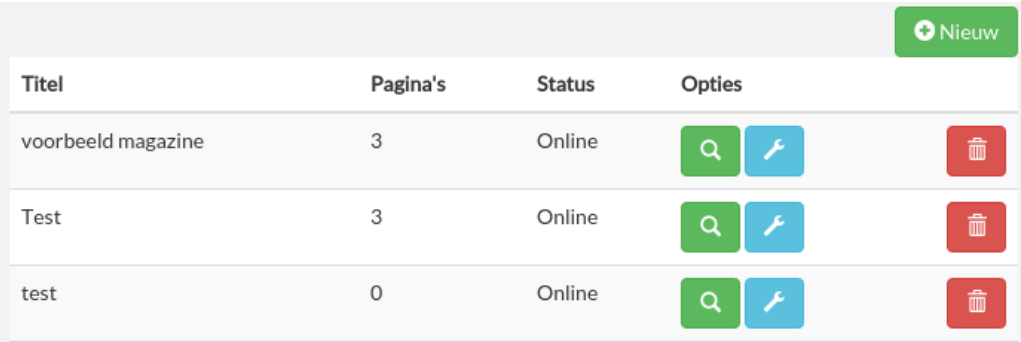
In sommige gevallen kan het handig zijn om juist te beginnen met de interface zodat het gecommuniceerd kan worden met de opdrachtgever. Maar in deze fase is die noodzaak er nog niet omdat het voornamelijk draait om het realiseren van de afhankelijkheden, oftewel, de basis van de applicatie. Als de vrijheid aanwezig is heeft het mijn (persoonlijke) voorkeur om eerst de techniek uit te denken en te realiseren, zodat ik op het laatst in de interface alleen nog maar bestaande functies hoef aan te roepen.

Na het maken van de controllers worden er alleen nog maar JSON strings gecommuniceerd naar de browser. De bedoeling is dat deze gegevens op een gebruiksvriendelijke manier worden gecommuniceerd met de gebruiker doormiddel van een interface. Laravel biedt hiervoor Blade templates. De template wordt aangeboden door de controller (zie afbeelding 18).

```php
/**
 * Magazines overview
 *
 * @return Response
 */
public function index()
{
    $account = Auth::user()->account;
    $someMagazines = Magazine::where('account_id', '=', $account->id)->paginate(12);

    return View::make('admin.magazines.index', array('magazines' => $someMagazines));
}
```

Afbeelding 18: Magazines ophalen van de gebruiker en meegeven in de view.

In het geval van afbeelding 18 worden de persoonlijke magazines van de gebruiker meegegeven in de view en in een overzicht geplaatst (zie afbeelding 19).
Na het realiseren van de interface ben ik gaan testen of alles nog naar behoren werkt. Eerder vermelde ik al dat ik Postman gebruikte om tijdens de ontwikkeling te testen of een functie werkt naar behoren via RESTful requests. Maar Postman is niet geschikt voor regressietests. Daarom heb ik die opgesteld voor later gebruik.



Afbeelding 19: Het overzicht van magazines van de huidige gebruiker.

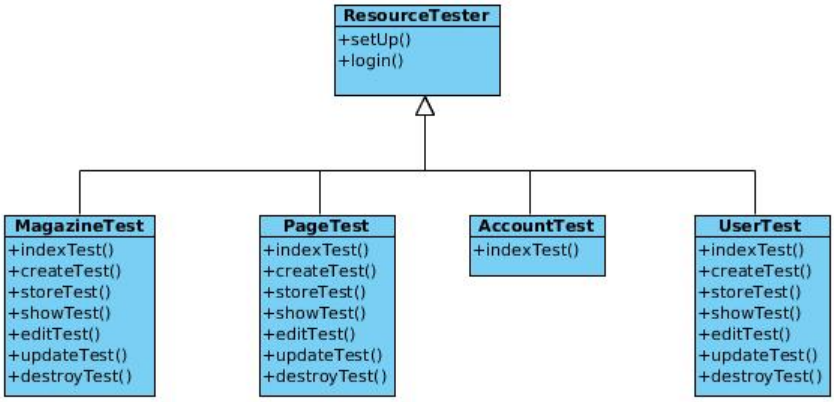
6.9 Testen

De functionaliteiten in deze sprint zijn nog beperkt en komen voornamelijk neer op CRUD functionaliteit. Ik heb daarom unit tests geschreven om te testen of alles naar verwachting werkt. De unit tests worden later gebruikt als regressietests om er zeker van te zijn dat de basis die deze sprint is neergezet naar behoren blijft werken. Laravel komt standaard met PHPUnit met enkele aanpassingen waardoor de unit tests makkelijk op het framework kunnen worden toegepast.

6.9.1 Creatie en uitvoer van unit tests

Een unit test wordt normaal gebruikt om een methode te testen. Op laracasts.com (geen directe link omdat de video alleen zichtbaar is met een betaald account) leerde ik dat dit is ook het geval is in Laravel, met het verschil dat methodes worden aangeroepen vanaf de router. Op die manier worden de routes, controllers, models en database getest wat een uitgebreider testresultaat geeft. Als alleen een methode wordt getest kan het namelijk zijn dat de methode door de test komt maar dat het niet bruikbaar is door een verkeerd ingestelde route. Dit soort problemen worden wel tijdig opgemerkt als een methode wordt aangeroepen vanaf de router.

Om de CRUD functionaliteit te testen heb ik de tests gemaakt op basis van de resource controllers (zie afbeelding 20) met de gezamelijke functionaliteit in de superclass ResourceTester. In de setUp() functie wordt het migratie script uitgevoerd en een test database opgezet. Omdat de resource controllers alleen aanspreekbaar zijn na authenticatie heb ik ook een login() functie toegevoegd in de ResourceTester welke door de setUp() wordt aangeroepen. De test classes bevatten functies op basis van de RESTful acties, op die manier is het meteen duidelijk wat en waar een methode wordt getest.



Afbeelding 20: De test classes in PHPUnit / Laravel

Na het opstellen van de structuur heb ik de unit tests geschreven (zie afbeelding 21). In het geval van afbeelding 21 roep ik op basis van de RESTful met een GET de url admin/magazines aan wat de index() functie aanroept in de MagazineController, de verwachting is dat er een view wordt teruggegeven met magazines. Daarom kijk ik met assertResponseOk() of de response code 200 is teruggegeven wat betekend dat de call zonder problemen is uitgevoerd. Vervolgens kijk ik met assertViewHas() of de view ook daadwerkelijk magazines bevat.

```php
class MagazineTest extends ResourceTester {

    /**
    * magazine overzicht ophalen
    * @test
    */
    public function indexTest()
    {
        $this->call('GET', 'admin/magazines');

        $this->assertResponseOk();
        $this->assertViewHas('magazines');
    }
```

Afbeelding 21: Voorbeeld van een unit test

Op die manier heb ik de unit tests opgesteld. Ondanks dat het om een enkele testmethode gaat biedt het een redelijke testdiepte voor de functionaliteit die de user stories beschreven voor deze sprint.

```
tim@tim-pc:~/projects/ezine3/trunk$ phpunit
PHPUnit 3.7.37 by Sebastian Bergmann.

Configuration read from /home/tim/projects/ezine3/trunk/phpunit.xml

.....................

Time: 3.02 seconds, Memory: 41.25Mb

OK (21 tests, 42 assertions)
```

Afbeelding 22: De unit tests succesvol uitgevoerd.

6.10 Evaluatie

Aan het einde van de sprint heb ik met mijn begeleider en opdrachtgever gekeken naar het verloop van de eerste sprint. De opdrachtgever was tevreden met de voortgang maar maakte zich zorgen over het toepassen van de rijksoverheid richtlijnen. Bij het oude Ezine2 was dit namelijk een probleem, vooral op het gebied van responsive design op de tablet en smartphone. Om de opdrachtgever vertrouwen te geven in het project en de risico's op dit gebied te verkleinen heb ik besloten om hier vroegtijdig aandacht aan te besteden. Ook voor mezelf is het nuttig om deze kennis achter de hand te hebben voor later in het project. ik heb daarom in overleg met de opdrachtgever besloten om een spike te doen in plaats van een sprint. Een spike is net als een sprint timeboxed en heeft dezelfde lengte als een sprint. Het verschil is dat een spike gericht is op het uitzoeken en verkleinen van risico's.

7 Functional spike: richtlijnen en interface

In deze spike wordt tijd besteed aan de rijksoverheidsrichtlijnen, dit wordt gedaan aan de hand van prototyping, mockups en concepten. Het doel van deze spike is om een geschikte oplossing te vinden om te kunnen voldoen aan de overheid richtlijnen en te bepalen hoe de interactie verloopt tussen de gebruiker en het systeem.

De richtlijnen van de rijksoverheid zijn daarom meegenomen als wens van de opdrachtgever. In deze spike zal er aandacht worden besteed aan de volgende punten voordat ze in een sprint worden uitgevoerd:

- Zoeken naar een oplossing om magazine pagina's op te maken op basis van een 12 punt grid systeem volgens richtlijn van de rijksoverheid.
- Swipe mogelijkheden op apparaten met touchscreen, dit is ook een richtlijn van de rijksoverheid
- Een concept ontwikkelen met de gevonden oplossing voor het 12 punts grid systeem en de swipe mogelijkheid.
- Uitwerking van een mockup voor het edit scherm van de magazine pagina's.
- Ontwerp van een huisstijl.

7.1 Keuze grafisch framework

Het responsive 12 punts grid systeem zie ik als een van de belangrijkste richtlijnen omdat het de basis gaat vormen voor de opbouw van de magazine pagina's. Uit ervaring weet ik dat het gebruik van een grid systeem geen onbekend concept is voor website opmaak en dat een soortgelijk principe in de meeste grafische frameworks wordt gebruikt. Door de inzet van een grafisch framework kan ik daarom voldoen aan de richtlijn van het grid systeem. Daarnaast bieden grafische frameworks vaak extra compatibiliteit met browsers en bieden ze kant en klare componenten voor de opbouw van de interface. Het biedt daarom vele voordelen om het in deze situatie toe te passen.

Er zijn tientallen grafische frameworks te vinden maar ik heb samen met mijn collega Marcel van Werkhoven (frontend ontwikkelaar) besloten om ons te beperken tot de meest bekende frameworks. Studio Projectie werkt regelmatig met Twitter Bootstrap maar heeft ook ervaring met de frameworks Foundation en Skeleton. Dit zijn tegelijk ook de meest gebruikte grafische frameworks. Ik ben daarom gaan kijken welke van deze drie frameworks het beste aansluit bij de eisen van dit project.

7.1.1 Skeleton

is een lichtgewicht framework met weinig functionaliteit. Marcel had hier goede ervaringen mee en vond het een aanrader omdat het makkelijk is te customizen. Maar het viel mij al snel op dat Skeleton al sinds juni 2012 geen update meer heeft gehad en dat het gebruik maakt van een responsive 16 punts grid systeem in plaats van 12. Dit maakt de keuze voor Skeleton twijfelachtig omdat het niet zeker is of het compatibel is met de browsers die afgelopen 2 jaar uit zijn gekomen. Hierdoor vond ik het ook niet de moeite waard om verder te kijken of het 16 punt grid kan worden aangepast naar een 12 punt grid.

7.1.2 Foundation 5

heeft zijn laatste update gehad in april 2014 en maakt gebruik van een instelbaar responsive punt grid systeem, het grid systeem is instelbaar van 1 t/m 16 punten. Het maakt gebruik van de meest recente technieken zoals HTML5, CSS3 en jQuery2 en biedt een zeer uitgebreid aantal grafische componenten. Het is hiermee een goede kandidaat voor de opmaak van magazine pagina's.

7.1.3 Twitter Bootstrap 3

is voor het laatst geupdate in juni 2014 en maakt gebruik van een responsive 12 punts grid. Het biedt net als Foundation 5 een uitgebreid aantal grafische componenten gebaseerd op recente technieken. Daarnaast is Twitter Bootstrap 3 de meest populaire van de 3 wat heeft geleid tot vele plugins van third-party ontwikkelaars. Ook Twitter Bootstrap is hierdoor een interessante kandidaat voor de opmaak van magazine pagina's.

7.1.4 Conclusie

Foundation 5 en Twitter Bootstrap 3 zijn aan elkaar gewaagd. De doorslag voor een keuze was uiteindelijk de browser ondersteuning. Foundation 5 ondersteunt Internet Explorer 9 en hoger terwijl Twitter Bootstrap 3 Internet Explorer 8 nog ondersteund.

Aangezien de rijksoverheid volgens de documentatie nog ondersteuning wil bieden aan Internet Explorer 8 en Bootstrap veel extra plugins biedt is de keuze op Twitter Bootstrap gevallen. De bedoeling is om een Swiper plugin te zoeken dat goed samenwerkt met Twitter Bootstrap.

7.2 jQuery Swiper

Ik heb mijn collega's gevraagd wat hun ervaringen zijn op het gebied van het swipen op touchscreens. Hieruit bleek dat zij hier in het verleden onderzoek naar hebben gedaan. Uit dit onderzoek kwam jQuery Swiper naar voren, jQuery Swiper wordt inmiddels naar tevredenheid gebruikt in het oude Ezine2. Aangezien dit al is uitgezocht en jQuery Swiper naar tevredenheid werkt heb ik deze ook gekozen. De bedoeling is dat de gebruiker door de pagina's van een magazine kan bladeren door te vegen op het touchscreen. Dit is precies wat jQuery Swiper mogelijk maakt. Tijdens de uitwerking van de mockup moet duidelijk worden of deze combinatie naar wens werkt, om een realistisch beeld te vormen worden er een aantal templates van de rijksoverheid toegepast.

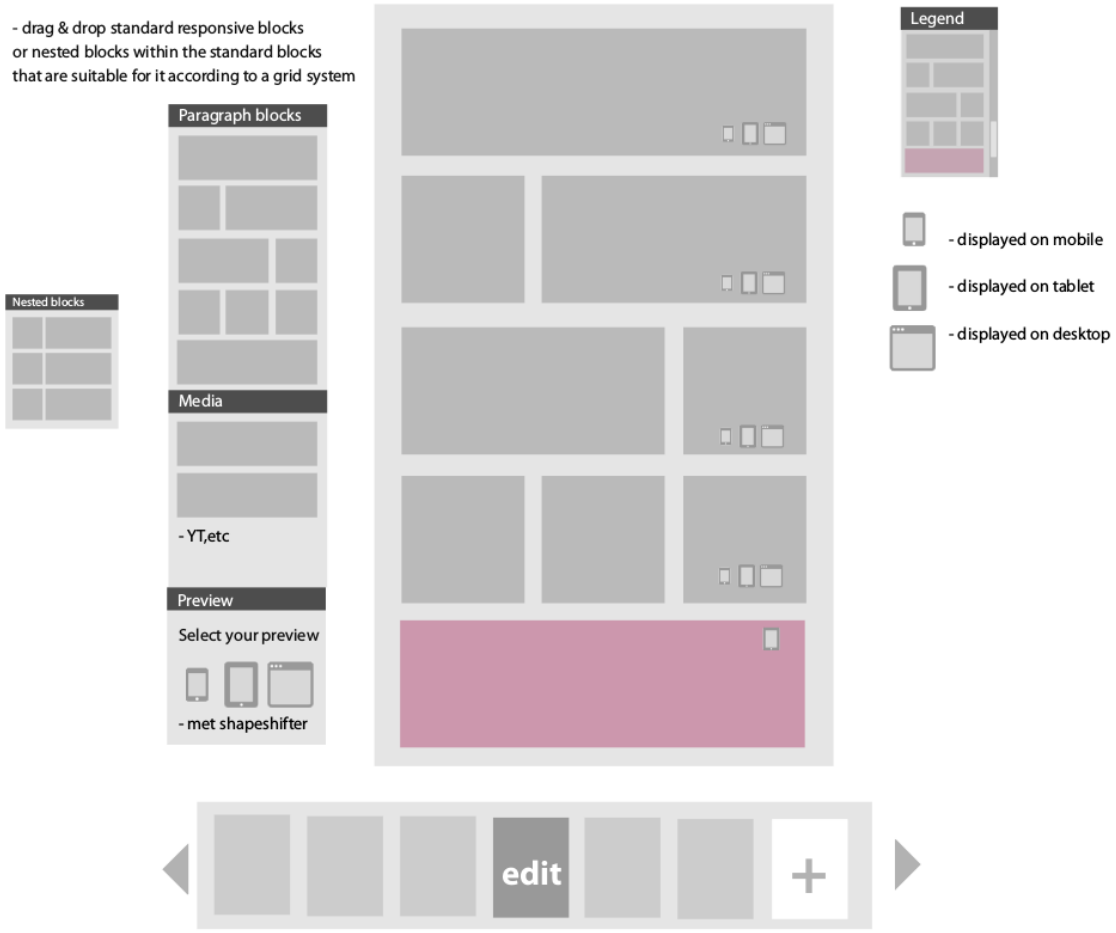
7.3 Magazine templates

De rijksoverheid heeft drie verschillende templates beschikbaar gesteld voor de opmaak van een magazine (zie afbeelding 23). Iedere template heeft een eigen stijl. De bedoeling is dat deze stijlen in het concept worden verwerkt zodat kan worden gekeken of dit goed werkt, welke eventuele problemen naar voren komen en om te kijken of de combinatie met bootstrap en swiper op computer, tablet en smartphone goed werkt. Het concept kan later worden gebruikt in het project.

Afbeelding 23: De drie overheids-templates met ieder een eigen stijl en indeling.

7.3.1 Backend mockup

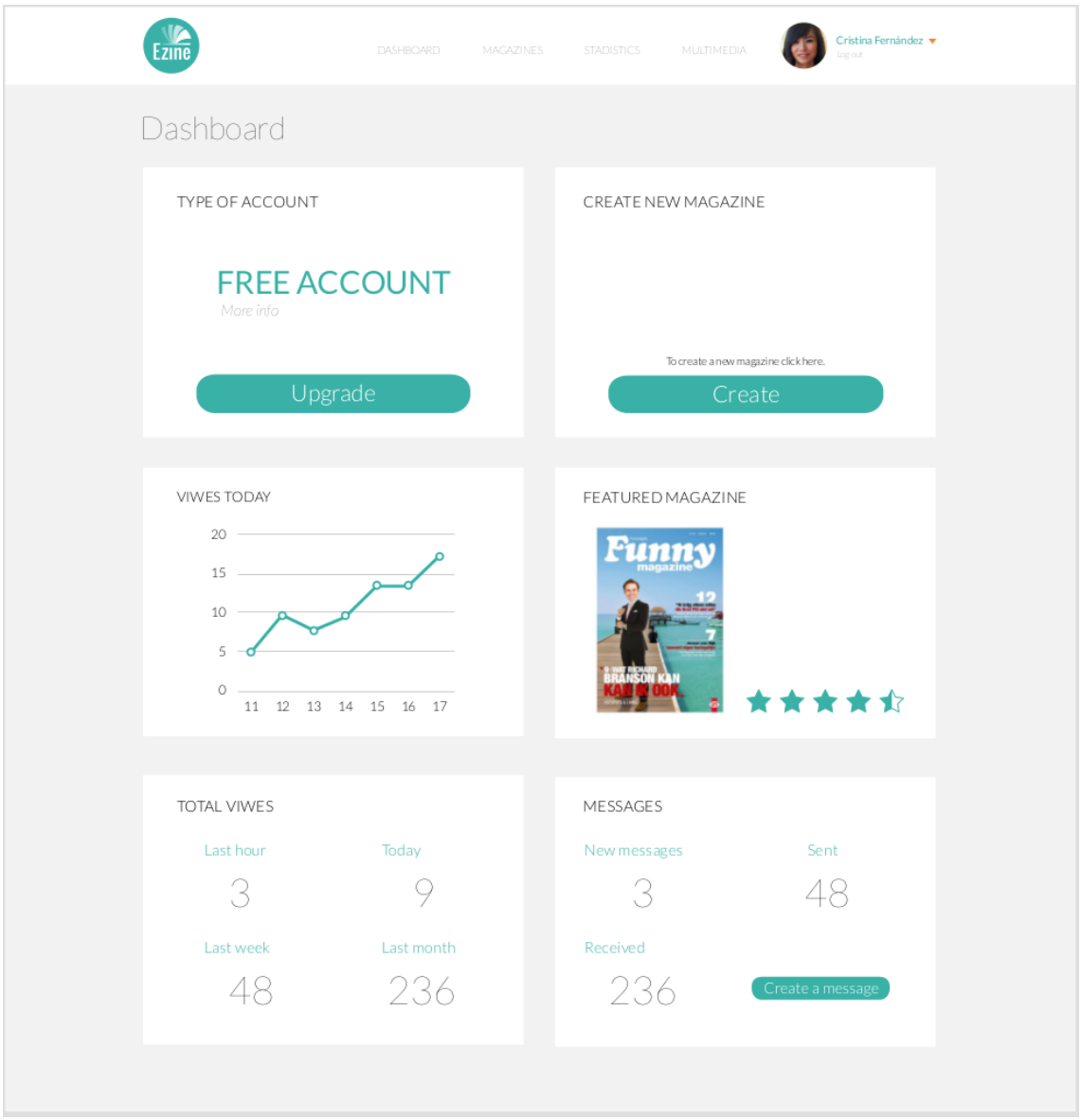
Samen met de frontend developers Marcel en Miki heb ik een mockup bedacht voor het edit scherm voor de opmaak een magazine pagina. (zie afbeelding 24). Het idee hierachter is om blokken op basis van het grid systeem op de pagina te kunnen zetten. Deze blokken kunnen op hun beurt inhoud bevatten met tekst, video of een afbeelding. Er is besloten om voor de video's gebruik te maken van externe bronnen zoals YouTube. Voor afbeeldingen is het wel de bedoeling dat de gebruiker deze kan uploaden in Ezine (hieruit zijn een aantal nieuwe user stories naar voren gekomen en in de product backlog geplaatst, zoals een upload en afbeelding beheer).



Afbeelding 24: Ezine Mockup voor de pagina opmaak.

7.3.2 Huisstijl

In samenwerking met de Spaanse stagiair Christina(grafisch designer) is er een huisstijl ontworpen voor Ezine3 (zie afbeelding 25). Cristina is onderdeel geworden van het Ezine team en gaat mij ondersteunen op het gebied van design. Omdat ze geen technische kennis heeft zal het ontwerp als afbeelding worden aangeleverd en zal ik deze toepassen in de applicatie zodat Ezine een applicatie wordt met een eigen huisstijl.



Afbeelding 25: Ezine mockup voor het dashboard.

7.4 Ontwikkeling Concept

Omdat het om een concept gaat met als doel om te kijken of de gekozen technieken goed samen werken in combinatie met de rijksoverheids stijlen wordt het alleen voor nu alleen in HTML, CSS en jQuery ontwikkeld. De content van de pagina's wordt er statisch ingezet, oftewel, de tekst wordt in de html gezet en komt niet vanuit de database.

7.4.1 Pagina's swipen

Ik ben begonnen met het swipe gedeelte van de pagina's zodat er later een frame is om de pagina inhoud in te zetten. Eerder is besloten om jQuery Swiper te gebruiken om tussen de pagina's te kunnen swipen om op die manier de gebruiker de indruk te geven dat hij naar links en naar rechts door een magazine kan bladeren. Omdat Swiper voor mij nog een onbekende library was heb ik de documentatie opgezocht op de website[6], hier vond ik een lijst met mogelijke api calls en enkele demo's als voorbeelden. Als eerste heb ik de benodigde libraries en css bestanden ge-include (zie afbeelding 26) in de index.

```
    <title>Ezine3.0</title>

    {{ HTML::script('assets/js/jquery.min.js'); }}
    {{ HTML::script('assets/js/bootstrap.min.js'); }}
    {{ HTML::script('assets/js/idangerous.swiper-2.1.min.js'); }}
    {{ HTML::script('assets/js/functions.js'); }}

    {{ HTML::style('assets/css/bootstrap.min.css'); }}
    {{ HTML::style('assets/css/idangerous.swiper.css'); }}
    {{ HTML::style('assets/css/rijksoverheid.css'); }}
</head>
```

Afbeelding 26: Benodigde libraries en css bestanden

De syntax met dubbele brackets in afbeelding 26 is anders dan gebruikelijk omdat ik gebruik maak van Laravel blade. Blade is de template engine van Laravel, ik heb ervoor gekozen om het meteen in blade te maken omdat de code later met minimale inspanning gebruikt kan worden in het project. Vervolgens heb ik de bijbehorende HTML in de view geplaatst (zie afbeelding 27), in deze HTML code worden later de magazine pagina's geplaatst. Daarnaast heb ik op een soortgelijke manier een tweede swiper bijgeplaatst welke moet gaan fungeren als navigatie tussen de magazine pagina's zoals we eerder hebben bepaald in de mockup in afbeelding 24.

[6] http://www.idangero.us/sliders/swiper/

```
<div class="swiper-container">
    <div class="swiper-wrapper">
        <!--First Slide-->
        <div class="swiper-slide">
            <!-- Any HTML content of the first slide goes here -->
        </div>

        <!--Second Slide-->
        <div class="swiper-slide">
            <!-- Any HTML content of the second slide goes here -->
        </div>

        <!--Third Slide-->
        <div class="swiper-slide">
            <!-- Any HTML content of the third slide goes here -->
        </div>

    </div>
</div>
```

Afbeelding 27: HTML voor de swiper

Na het plaatsen van de HTML kwam het interessante deel. Beide swipers moeten samen gaan werken, het doel is dat als er wordt geklikt op de navigatie swiper (.swiper-minibar) de pagina swiper (.swiper-container) automatisch scrolt naar de gewenste pagina. Maar ook als de gebruiker de pagina swiped naar de volgende pagina moet de navigatie swiper zich daarop aanpassen. Ik heb dit opgelost door gebruik te maken van onSlideClick, op die manier kon ik beide swipers automatisch laten swipen met swipeTo naar dezelfde slideIndex waardoor beide swipers na iedere click de pagina index synchroniseert (zie afbeelding 28).

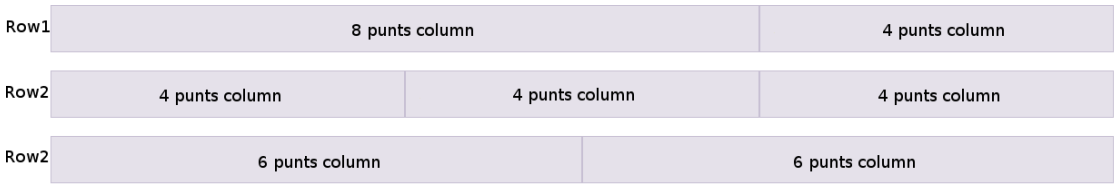
```
//swiper
$(document).ready(function() {
    var slideindex = null

    var pages = $('.swiper-container').swiper({
        paginationClickable: false,
        grabCursor: false,
        paginationClickable: false,
        calculateHeight: true,
        touchRatio: 0,
        onSlideChangeEnd : function() {
            $(window).mgMiniMap('update')
        },
        onSlideClick : function() {
            pages.swipeTo(pages.clickedSlideIndex, 600, true)
            minibar.swipeTo(pages.clickedSlideIndex, 600, true)
        }
    });

    var minibar = $('.swiper-minibar').swiper({
        paginationClickable: true,
        slidesPerView: 7,
        centeredSlides: true,
        touchRatio: 0,
        onSlideClick : function() {
            pages.swipeTo(minibar.clickedSlideIndex, 600, true)
            minibar.swipeTo(minibar.clickedSlideIndex, 600, true)
        }
    });

});
```

Afbeelding 28: HTML voor de swiper

7.4.2 Structuur Twitter Bootstrap

De structuur van Twitter Bootstrap is interessant om toe te lichten omdat het gaat bepalen hoe de pagina's worden opgebouwd. Tijdens dit concept statisch en later wordt de opbouw dynamisch.

Twitter Bootstrap werkt met rows en columns (zie afbeelding 29). Een row is een horizontale container die de volledige spanwijdte gebruikt van de pagina. Er kunnen meerdere rows onder elkaar geplaatst worden. Binnen een row kunnen columns worden geplaatst, de breedte van de columns wordt bepaald door het 12 punts grid systeem. Een column van 12 punten gebruikt de volledige breedte binnen de row, maar er kunnen bijvoorbeeld ook 2 columns van ieder 6 punten of 4 columns van ieder 3 punten. In een column kan weer tekst, video of een afbeelding worden geplaatst. Door gebruik te maken van dit grid systeem zorgt Bootstrap er automatisch voor dat de pagina responsive wordt en goed leesbaar is op pc, tablet en smartphone.



Afbeelding 29: Rows en Columns waarmee de magazine pagina's worden opgebouwd.

Deze structuur wordt normaal door een ontwikkelaar opgebouwd in HTML en CSS, de grote uitdaging van dit project is om dit technische deel beschikbaar te maken doormiddel van een interface zodat een gebruiker zonder technische kennis een pagina kan opmaken zoals een frontend ontwikkelaar dat doet. Over de technische uitwerking van dit gedeelte zijn nog veel twijfels, er is daarom besloten om ook hiervoor een Spike te houden, deze keer gaat het om een technische spike om uit te zoeken hoe ik dit het beste technisch gezien kan uitwerken.

8 Technical spike: bepaling pagina opmaak

Het volgende onderdeel is om te bepalen hoe de pagina opmaak word gedaan. Het doel is dat de klant op een eenvoudige manier de pagina's kan opmaken in zijn eigen online magazine. Hoe de interface eruit gaat zien is bepaald in de tweede sprint, maar de technische uitwerking is nog niet duidelijk. Waarschijnlijk is dit een van de grootste uitdagingen van dit project en er zijn vele manieren om dit doel te bereiken. Omdat het succes van dit project mede afhangt van dit onderdeel hebben we besloten nog een keer af te wijken van de planning en meer aandacht te geven aan dit onderwerp. Dit wordt gedaan dooreen aantal concepten uit te werken en te kijken wat hier de voor- en nadelen zijn. Het hoofdstuk eindigt met een keuze voor een concept.

8.1 Concept: XML

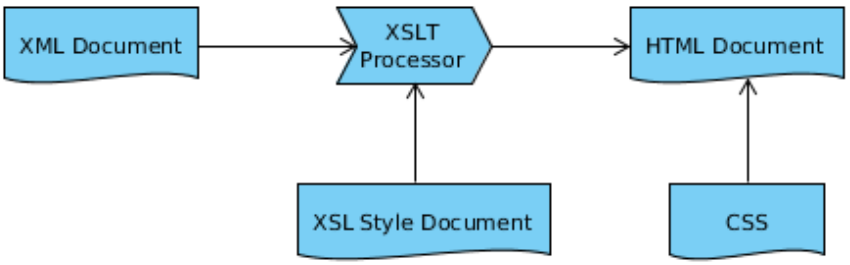
In het verleden heb ik als software ontwikkelaar gewerkt bij een uitgeverij. De DTP'ers die de opmaak van boeken deden moesten gaan leren hoe ze Epubs konden opmaken. Ik had geen ervaring met de opmaak van Epubs maar Epubs worden voor een groot deel met dezelfde techniek opgemaakt als websites, namelijk met XML, HTML en CSS. Op technisch gebied heb ik toen regelmatig geholpen met de opmaak van Epubs. Uit deze ervaring is het idee voortgekomen om deze techniek ook toe te passen op de opmaak van magazine pagina's. Een Epub en een online magazine hebben namelijk veel overeenkomsten. Daarbij is XML heel geschikt voor het opslaan van een structuur wat van pas komt bij het opslaan van de structuur van een magazine pagina.

De opdrachtgever was enthousiast over dit idee en vond het de moeite waard om dit uit te werken tot concept. Ik ben begonnen met een opmaken van een XML bestand met de eigenschappen van de pagina. Ik heb hierin de structuur van de rows en columns verwerkt van Bootstrap en daar content in geplaatst (zie afbeelding 30).

```
<?xml version="1.0" encoding="UTF-8" ?>
<page>
    <row>
        <id>12</id>
        <title>titel1</title>
        <column>
            <id>2</id>
            <type>text</type>
            <span>8</span>
            <content>
                Lorem ipsum dolor sit amet, consectetuer adipisc
            </content>
        </column>
        <column>
            <id>1</id>
            <type>text</type>
            <span>4</span>
            <content>
                orem ipsum dolor sit amet, consectetuer adipisci
            </content>
        </column>
    </row>
```

Afbeelding 30: XML magazine pagina.

De bedoeling is om het XML geautomatiseerd om te zetten naar HTML zodat het in de browser weergegeven kan worden als een magazine pagina. In eerste instantie heb ik dit gedaan met PHP, door PHP op basis van de XML attributen en parameters de HTML te laten schrijven. Toen ik aan de slag ging merkte ik dat dit een hoop moeilijk leesbare code opleverd dat in de toekomst hoogstwaarschijnlijk er niet beter op gaat worden met het toevoegen van nieuwe functionaliteiten. Meestal is er in dit soort situaties een betere oplossing, en dat bleek ook het geval toen ik verder ging zoeken hoe ik XML het beste om kan zetten naar HTML. Ik kwam na enige zoekwerk via Google uit op EXtensible Stylesheet Language (XSLT). XSLT is een stylesheet taal voor XML (zie afbeelding 31).



Afbeelding 31: XML omzetten naar HTML doormiddel van XSLT.

De XSLT processor leest het XML document, vervolgens vertelt het XLS Style Document hoe een XML element moet worden omgezet naar HTML. Op die manier was het eenvoudig om het XML om te zetten naar HTML en weer te geven als magazine pagina. Het is een geschikte techniek voor het weergeven van de pagina's maar het wijzigen van een pagina blijkt minder eenvoudig te zijn. Bij het wijzigen van bijvoorbeeld een stuk tekst moet het hele XML document worden opgehaald en opnieuw worden weggeschreven. Om deze reden ben ik verder gaan zoeken naar andere mogelijkheden.

8.1.1 Voordelen

- Gestructureerd opslaan in XML
- Eenvoudig weer te geven met XSLT
- Flexibiliteit door verschillende XSL style documents

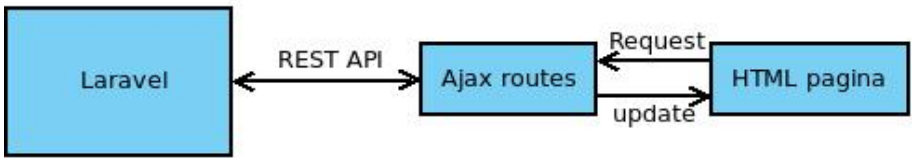
8.1.2 Nadelen

- Niet geschikt voor frequente wijzigingen.

8.2 Concept: jQuery ajax

Door de XML ervaring kwam ik toch tot de conclusie dat het beter is om de data van magazine pagina's op te delen en op te slaan in de database op de traditionele manier. Door de data op te delen in rows en columns kan het op de Twitter Bootstrap manier worden opgebouwd in de interface. Voor de gebruiker is het niet prettig om een steeds de hele pagina te moeten vernieuwen bij iedere wijzigen op een magazine pagina. Dit kan worden opgelost door jQuery Ajax. Met Ajax is bi-directionele communicatie mogelijk van bepaalde elementen zonder dat de hele pagina hoeft te worden ververst.

Tijdens de ontwikkeling van een eenvoudig concept bleek dit redelijk in de goede richting te zitten, rows en columns zijn eenvoudig te wijzigen door middel van ajax requests met de REST API in Laravel(zie afbeelding 32). jQuery Ajax biedt alleen weinig structuur en is daardoor eigenlijk alleen geschikt als het aantal ajax requests beperkt is. Bij de vele soorten transacties die vereist zijn bij de opmaak van een magazine pagina wordt de code waarschijnlijk lang en onoverzichtelijk. Het principe is dus goed maar ik zoek een meer gestructureerde aanpak.



Afbeelding 32: Ajax met jquery.

8.2.1 Voordelen

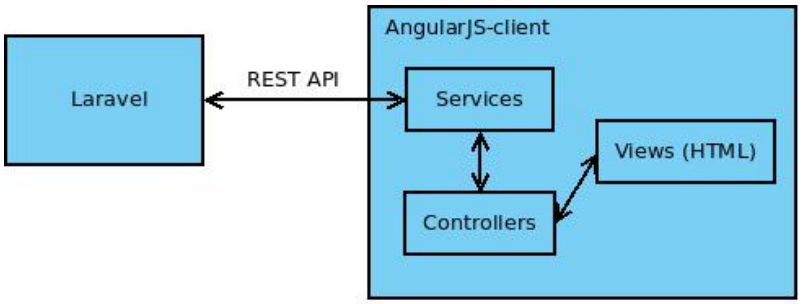
- veel documentatie
- veel plugins
- eenvoudig in gebruik
- bi-directionele communicatie via ajax

8.2.2 Nadelen

- weinig structuur

8.3 Concept: AngularJS

Vorig jaar heb ik een workshop AngularJS gevolgt bij InfoSupport, daarvan herinner ik mij dat AngularJS ook bi-directionele communicatie biedt en dat het gebruik maakt van MVC wat structuur biedt (zie afbeelding 33). AngularJS is daarom meer een framework voor een client-side applicatie terwijl jQuery slechts een library is. Het zou daarom een interessant alternatief kunnen zijn voor jQuery.



Afbeelding 33: AngularJS client.

Bij de ontwikkeling van het concept ontdekte ik dat AngularJS veel directives biedt bij het opbouwen van pagina's, het kan bijvoorbeeld op een soortgelijke manier als PHP een object ophalen en met een foreach de layout opbouwen op basis van de data. Dit is een voordelige eigenschap waar ook de frontend ontwikkelaars mee om kunnen gaan omdat dit een bekend principe is. Daarnaast is de bi-directionele communicatie eenvoudig en gestructureerd opgezet. AngularJS past ook goed binnen het bedrijf omdat het meer gestandaardiseerde technieken wil gebruiken en inmiddels voor meerdere projecten wordt gebruikt.

8.3.1 Voordelen

- Gestructureerd
- bi-directionele communicatie

8.3.2 Nadelen

- Vereist meer kennis dan jQuery

8.4 Conclusie

Het uitwerken van de concepten heeft nieuwe inzichten gegeven op de techniek achter het bewerken van magazine pagina's. De methode van XML en XLS zag ik in het begin als schot in de roos waar ik overtuigd van was dat het de ideale methode was. Voor de weergave van de pagina's was het ook geschikt. Maar tijdens de uitwerking van het concept bleek het niet heel geschikt te zijn voor het dynamische karakter dat de pagina opmaak moet krijgen.

De methode van jQuery en Ajax calls naar een API werkte goed maar de onderhoudbaarheid en uitbreidbaarheid was twijfelachtig omdat jQuery niet veel structuur biedt. Het lijkt vooral een oplossing te zijn voor eenvoudige toepassingen.

AngularJS werkt volgens hetzelfde principe als jQuery met Ajax maar is specifiek ontworpen voor toepassingen met uitgebreide bi-directionele communicatie. Het is hierdoor op een meer gestructureerde manier opgezet door gebruik van de MVC structuur wat ten goede komt op het gebied van onderhoudbaarheid en uitbreidbaarheid. De keuze is dan ook gemaakt verder te gaan met AngularJS voor de opmaak van de magazine pagina's.

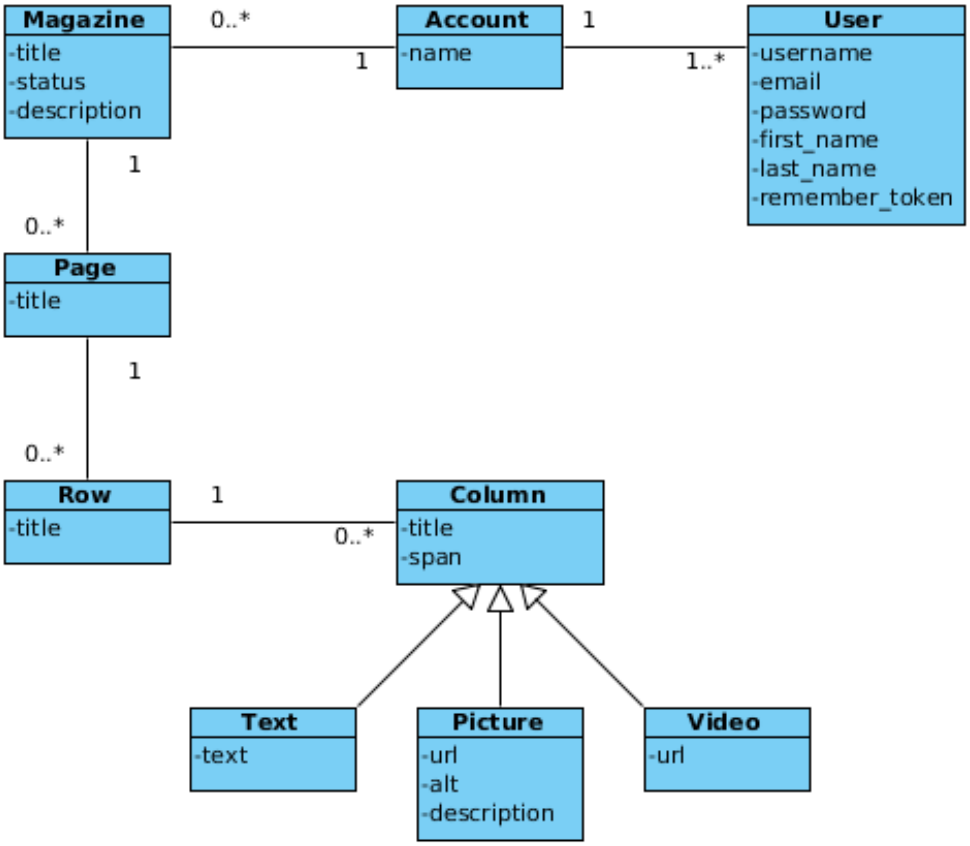
9 Sprint (2, 3, 4): opmaak van de magazine pagina's

In de eerste sprint is een eerste opzet gemaakt van de backend in het Laravel framework. In de functional spike is de wens van de opdrachtgever uitgevoerd om een concept te maken voor een magazine dat voldoet aan de rijksoverheidsrichtlijnen in het Twitter Bootstrap framework. Vervolgens is nogmaals een technische spike geweest om te kijken naar de meest geschikte techniek voor de opmaak van magazine pagina's, met als resultaat de keuze voor het AngularJS framework. In de komende sprints worden deze technieken samengevoegd zodat deze losse elementen een geheel gaan vormen tot applicatie. Voor de user stories van deze sprints verwijs ik naar het requirements document in de bijlage.

Ik ben begonnen om het Laravel framework aan te passen voor de API communicatie met AngularJS. Na deze aanpassingen heb ik het AngularJS framework ingericht voor het ophalen van de magazine pagina's en ze klaar te maken voor bewerkingen. Als laatste heb ik de leesbare versie van de online magazines geïmplementeerd met Swiper in combinatie met de richtlijnen van de rijksoverheid.

9.1 Uitbreiding database en models

In de eerste sprint heb ik een basis opgezet van de database en het Laravel framework. Deze basis heb ik verder uitgebreid. De database en het MVC bevatten magazines met pagina's maar nog geen pagina inhoudt. De pagina's moeten rows en columns gaan bevatten zodat de inhoud op basis van bootstrap grid kan worden weergegeven. Een column kan bestaan uit drie verschillende types, namelijk: tekst, video en afbeelding. Ik heb gekozen om hier gebruik te maken van overerving(zie afbeelding 34), ieder column heeft namelijk een span waarde en kan optioneel een titel hebben. Maar tekst, video of afbeeldingen hebben verschillende waarden. Door tekst, video en afbeeldingen op te splitsen en de gezamelijke attributen en relaties over te erven van column worden dubbele waardes voorkomen. Een ander voordeel van deze constructie is dat het type van een column kan worden aangepast zonder dat de column eigenschappen verloren gaan. Dit brengt extra flexibiliteit met zich mee in de opmaak van een pagina.



Afbeelding 34: Klassendiagram dat de MySQL database spiegelt.

Ik wist niet direct hoe overerving geïmplementeerd moest worden in het Eloquent ORM van Laravel. In de documentatie[7] van Eloquent leerde ik dat dit via morph werkt (zie afbeelding 35). Het voordeel van deze methode is dat ik in de interface het object van Column kan aanroepen met de relatie naar het object "content". Content kan in dat geval een afbeelding, video of text bevatten.

```php
class Column extends Eloquent {

    public function row()
    {
        return $this->belongsTo('Row');
    }

    public function content()
    {
        return $this->morphTo();
    }

}
```

```php
class Text extends Eloquent {

    public function columns()
    {
        return $this->morphMany('Column', 'content');
    }

}
```

Afbeelding 35: De overerving tussen Column en Text in het Eloquent ORM

[7] http://laravel.com/docs/eloquent

9.2 Opzetten REST API in Laravel

Omdat AngularJS een API vereist heb ik Laravel hierop aangepast. Veel hoefde ik hiervoor niet te doen omdat een controller van Laravel al standaard een REST API is. Wat ik wel heb aangepast is dat er geen view wordt aangeroepen maar dat alleen het object in JSON formaat wordt teruggegeven. Waar ik meer werk aan heb gehad is het bedenken van een efficiënte strategie voor het laden van de data, eager loading bleek hiervoor een oplossing te bieden.

9.2.1 Pagina laden met eager loading

Een pagina is dus opgedeeld in rows en columns welke op hun beurt weer van een bepaald type met content zijn. Mijn eerste gedachte was om ieder element appart op te halen met een API request. Een row wordt opgehaald en vervolgens wordt ieder column opgehaald die bij die row horen. Maar op die manier krijg je erg veel API requests bij het laden van iedere pagina wat de server kan gaan belasten met veel bezoekers tegelijk. Ik heb daarom verder gekeken of dit handiger kan door een JSON vanaf de API server te verzenden dat alle rows en columns bevat. Via de Laravel documentatie[8] kwam ik er achter dat dit mogelijk is met Eager loading. Laravel laadt met Eager loading een pagina inclusief alle geneste relaties naar rows en columns. Dit resulteert in een object, dit object kan worden omgezet naar een JSON met alle gegevens van een pagina dat via de API kan worden opgevraagd door AngularJS. Via de Chrome developers tools kon ik zien dat magazine pagina's op die manier een stuk sneller laden. Pagina's kunnen nu worden geladen, de bedoeling is dat de gebruiker er zelf content op kan plaatsen in de vorm van blokken.

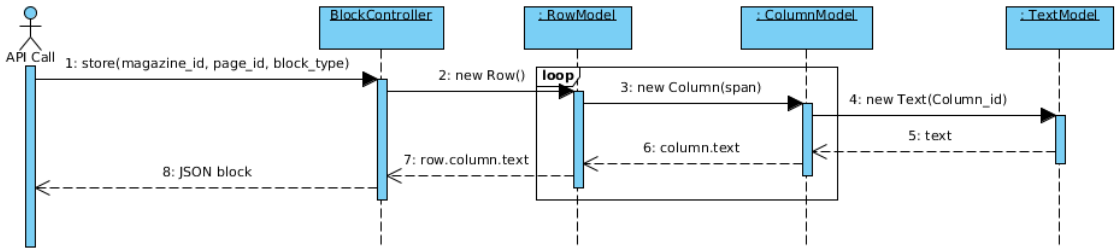
```php
/**
 * Get the magazine page
 *
 * @param  int  $id
 * @return Response
 */
public function show($id)
{
    $page = Page::find($id);
    $page->load('rows.columns.content');

    return Response::json($page);
}
```

Afbeelding 36: Eager loading

9.2.2 Blokken op een pagina plaatsen

Een blok is de verzamelnaam van een row, column en content maar om het eenvoudig en gebruiksvriendelijk te houden communiceren we dit in de interface naar de gebruiker als een blok. De bedoeling is dat de gebruiker de keuze krijgt uit verschillende blokken met ieder een eigen structuur. Als de gebruiker een blok aanklikt wordt deze in Laravel opgebouwd, opgeslagen en teruggegeven als JSON (zie afbeelding 53).

[8] http://laravel.com/docs/eloquent/eager-loading



Afbeelding 37: Opbouw van een blok in Laravel

In de BlockController wordt er een nieuwe Row aangemaakt en gekoppeld aan de betreffende magazine pagina. Een Row bevat altijd 1 of meerdere columns, de hoeveelheid en de breedte van de column vorm de structuur van een blok (zie afbeelding 24). Standaard zijn de columns van het type text, de bedoeling is dat de gebruiker dit later kan aanpassen naar een afbeelding of video.

9.2.3 Beveiliging

Bij het gebruik van een API moet er extra op beveiliging worden gelet. Op dit moment is het bijvoorbeeld mogelijk de API server aan te spreken via de browser of met curl zonder tussenkomst van de frontend interface. Wat je dan terug krijgt is een kale JSON string. Een gebruiker met logingegevens kan data ophalen en wegschrijven op de API via een externe applicatie. Dit wordt ook wel cross-site request forgery (CSRF) genoemd.

Om dit voorkomen heb ik gebruik gemaakt van Laravel's CSRF Protection Filter. Dit heb ik toegepast door een tutorial[9] te volgen. Met dit filter stuurt de server een geheime encrypted token naar de client, bij iedere API request wordt er gekeken of deze token overeenkomt met de gegenereerde code op de server. Op die manier kan een gebruiker niet zomaar zonder tussenkomst van de frontend interface de API interface aanspreken.

[9] http://angularjs4u.com/security/inject-laravel-4-csrf-token-angularjs/

9.3 Opzetten AngularJS

Na het aanpassen van Laravel heb ik AngularJS opgezet. Belangrijk om hierbij te vermelden dat ik AngularJS alleen voor de pagina opmaak (de editor) van de magazines gebruik. De rest van de applicatie maakt nog steeds gebruik van de blade templates van het Laravel framework. De hele frontend in AngularJS schrijven is een mogelijkheid maar de investering van de tijd die dat zou kosten wegen niet op tegen de voordelen bij de andere interface elementen dan de editor.

9.3.1 Realisatie

De opzet van AngularJS heb ik gedaan aan de hand van een tutorial[10], deze tutorial had ik eerder al gebruikt bij het concept dat ik eerder had gemaakt. Als eerste heb ik de service opgezet voor het ophalen van de magazines (zie afbeelding 38). Ik heb hierbij bewust dezelfde namen (get, show, save, etc) gehanteerd als in de controllers van Laraval zodat het duidelijk is welke componenten met elkaar communiceren.

```
.factory('Page', function($http, $stateParams) {

    return {

        get : function() {
            return $http.get("/admin/magazines/$stateParams.magazine_id/page");
        },

        show : function() {
            return $http.get("/admin/magazines/$stateParams.magazine_id/page/1");
        },

        save : function(data) {
            data["magazine_id"] = $stateParams.magazine_id;
            return $http({
                method: 'POST',
                url: '/service/pages',
                headers: { 'Content-Type' : 'application/x-www-form-urlencoded' },
                params: data
            });
        },

        update : function(data) {
            return $http({
                method: 'PUT',
                url: '/service/pages/'+data.id,
                headers: { 'Content-Type' : 'application/x-www-form-urlencoded' },
                params: data
            });
        },

        destroy : function(id) {
            return $http.delete("/service/pages/" + id);
        }

    }

})
```

Afbeelding 38: De service in AngularJS met REST API calls voor de pagina's.

Na het opzetten van de services heb ik de controllers aangemaakt. Iedere controller heeft zijn eigen scope, een scope bevat alle variabelen en functies (zie afbeelding 39) welke later binnen de view kunnen worden aangesproken. In de controller worden de services aangesproken, een service haalt de data op via de Laravel API en communiceert het met de view (notitie: zie seq diagram). In afbeelding afbeelding 39 is bijvoorbeeld te zien hoe een pagina wordt verwijdert in de deletePage functie. De destroy functie wordt aangeroepen

[10] http://scotch.io/tutorials/php/create-a-laravel-and-angular-single-page-comment-application

in de Page service met het id van de pagina als parameter. Als de API transactie succesvol wordt uitgevoerd wordt de functie getPages() aangeroepen welke de huidige pagina's ophaalt en in de scope plaatst. De gegevens worden op die manier geupdate met twee API requests zonder dat de gebruiker zijn pagina hoeft te verversen.

```
$scope.getPages = function()
{
    Page.get()
        .success(function(data) {
            $scope.pages = data;
        });
}

$scope.deletePage = function(id)
{
    Page.destroy(id)
        .success(function(data) {
            $scope.getPages();
        });
}
```

Afbeelding 39: Een pagina verwijderen in de controller.

Het verschil tussen MVC in Laravel en AngularJS is dat Laravel vanaf de controller de data meegeeft aan de view. in AngularJS is dit andersom, de view roept de data aan vanuit de controller. In afbeelding 40 wordt met data-ng-init() de functie getPage() aangeroepen in de PageController. Vervolgens wordt het page object ingeladen met de benodigde HTML om het aan de gebruiker te presenteren. Hier is meteen te zien hoe de pagina wordt opgebouwd met rows en columns en de switch tussen de verschillende type content in een column.

```
<!-- get magazine page -->
<div data-ng-controller="pageController" data-ng-init="getPage()" class="container page {{page.style}}">

    <!-- page title -->
    <h1>{{ page.title }}</h1>

    <!-- print page.rows -->
    <div ng-model="rows" ng-repeat="rows in page.rows">

        <!-- print page.rows.columns -->
        <div ng-model="row.columns" ng-repeat="column in row.columns" class="col-md-{{column.span}}">

            <!-- check content type (page.rows.columns.content) -->
            <div ng-switch="column.content_type">
                <div ng-switch-when="Video">
                    <div data-column-title></div>
                    <iframe class="youtube_video" src="//www.youtube.com/embed/{{column.content.url}}" allowfullscreen></iframe>
                </div>
                <div ng-switch-when="Picture">
                    <div data-column-title></div>
                    <img src="{{column.content.url}}" alt="{{column.content.alt}}" class="image_header">
                    <p>{{ column.content.description }}</p>
                </div>
                <div ng-switch-when="Text">
                    <div data-column-title></div>
                    <p>{{column.content.text}}</p>
                </div>
            </div>

        </div>

    </div>

</div>
```

Afbeelding 40: Een magazine pagina weergeven in de view.

De edit opties heb ik in afbeelding 40 even weggehaald voor de afbeelding om de pagina opbouw te laten zien. De edit opties worden namelijk weergegeven in sliders wat veel HTML code meegeeft. De moeilijkste edit optie vond ik de sortering van de rows (blokken) op een pagina. Vandaar dat het interessant is om toe te lichten.

9.3.2 Sortering van blokken

De bedoeling is dat de klant zijn pagina's en de rijen op de pagina kan sorteren. Ik gaan Googlen om te kijken hoe dit het beste kan worden gerealiseerd in combinatie met AngularJS. Ik ontdekte hierbij dat bij de meeste websites Node.js en MongoDB werden geadviseerd als backend in combinatie met AngularJS. Node.js is javascript backend speciaal ontwikkeld voor API communicatie. MongoDB is een noSQL database dat JSON direct kan accepteren en wegschrijven. Het grote voordeel hiervan is dat de volgorde in de JSON behouden blijft in de database. Maar dat is niet bruikbaar in dit geval.

Ik heb daarom een collega gevraagd wat hier de beste oplossing voor is. Met AngularJS blijkt het mogelijk om de huidige elementen zoals de rows up te vragen als array waarbij meteen de volgorde zichtbaar is. Deze kan met een API request naar Laravel worden gestuurd waar vervolgens de nieuwe volgorde van de rows kan worden gesorteerd en opgeslagen (zie afbeelding 41).

```php
/**
 * Reorder the rows according to the given order
 *
 * @param type $page_id
 */
public function updateRowOrder($page_id)
{
    $page = Page::find($page_id);
    if(isset($_POST['order'])){
        $order = $_POST['order'];

        // Check if order is array and populated
        if(is_array($order) && !empty($order)){
            $offset = 0;
            // Loop through id's retrieving the row
            foreach($order as $position => $row_id){
                $row_id = substr($row_id, 3);
                $row = Row::find($row_id);
                // if row exists update position
                if(!empty($row)){
                    $row->sortby = $position + 1 - $offset;
                    $row->save();
                } else { // if row doesn't exist add 1 to offset to ensure sort order
                    $offset++;
                }
            }
        }
    }

}
```

Afbeelding 41: Rijen sorteren en opslaan

9.4 Uitwerking van de magazine editor

Uiteindelijk is de structuur van de mockup op afbeelding 24 toegepast, hierbij is in samenwerking met de designer een ontwerp voor gemaakt welke ik heb toegepast. Afbeelding 42 is het resultaat. Aan de linkerkant kunnen de gewenste blokken worden geselecteerd welke vervolgens op de pagina worden geplaatst.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa.

Afbeelding 42: de uitwerking van de magazine pagina editor

De pagina kan nu word de view mode worden geopend. De opbouw van de pagina in de view is technische gezien hetzelfde als in de editor alleen dan zonder de edit opties en de pagina's worden in swiper geplaatst waardoor de gebruiker door de pagina's kan bladeren op een touchscreen.

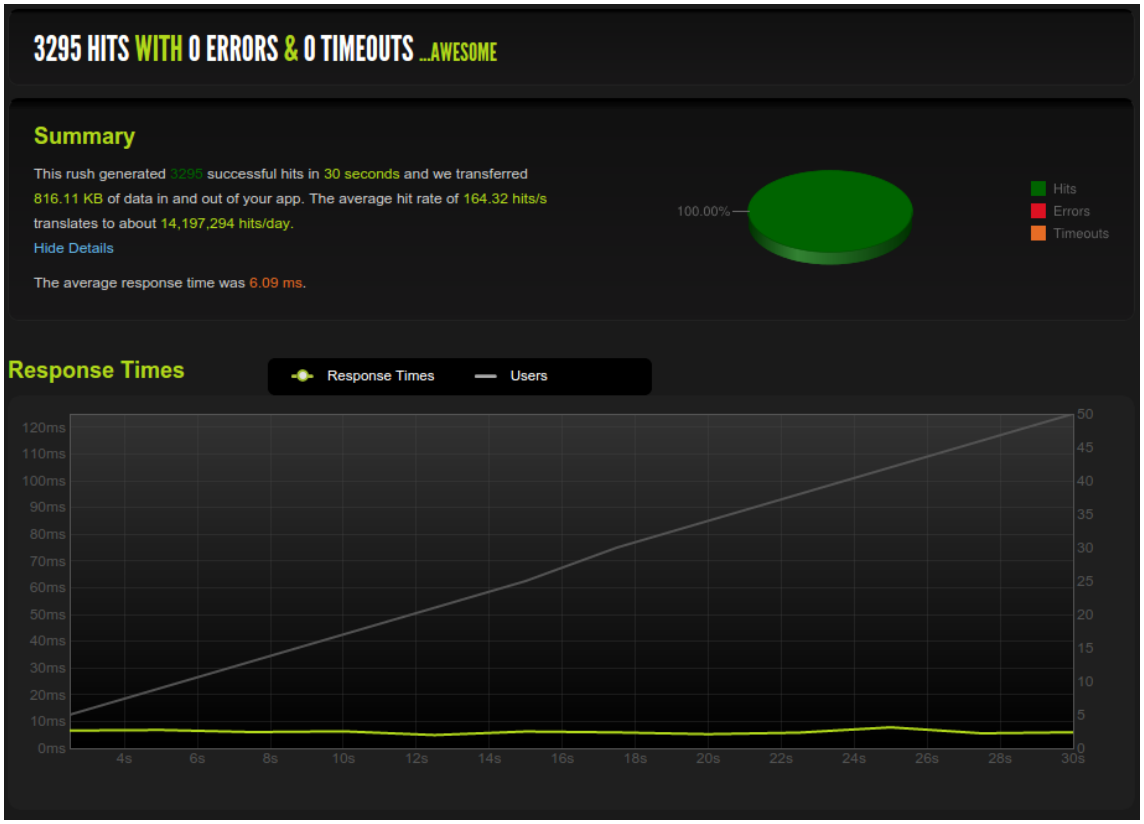
9.5 Testen

9.5.1 Acceptatie test

Met de begeleider en opdrachtgever is er bij iedere sprint een acceptatie test gedaan waarbij de functionaliteiten van de user stories worden bekeken en doorlopen. Er wordt hierbij gekeken of de kwaliteit acceptabel is qua werking, of er bugs in zitten en of er nog functionaliteit compleet is of dat er nog iets aan ontbreekt.

9.5.2 Stress test

Als een klant een magazine opmaakt en naar een grote groep klanten mailt kan er piekbelasting ontstaan als veel mensen het magazine tegelijk gaan lezen. Op aanraden van een collega heb ik dit gedaan met blitz.io, blitz simuleert piekbelasting in een korte tijd om te kijken hoe de website reageert. Dit bleek weinig problemen op te leveren met het bekijken van magazines (zie afbeelding 43). Waarschijnlijk mede door het gebruik van eager loading lijkt piekbelasting weinig problemen op te leveren.



Afbeelding 43: Blitz stress test

9.5.3 Gebruikersacceptatietests

Ik heb gekozen om ook een gebruikersacceptatie test te doen. Dit heb ik gedaan door met iemand af te spreken en op basis van de user stories instructies te geven, bijvoorbeeld het aanmaken van een magazine. Door dit samen te doen kan ik zien wat een gebruiker verwacht van het systeem, of de functionaliteit goed werkt en welke usability problemen er kunnen voorkomen. De instructies worden mondeling gegeven, dit is een bewuste keuze omdat de interactie met de gebruiker waardevol kan zijn door samen over de functionaliteit te praten en tot nieuwe inzichten en ideeën te komen, deze kunnen later worden besproken of geadviseerd aan de opdrachtgever.

9.5.4 Unit tests

Als laatste heb ik de unit tests uitgebreid en uitgevoerd. Het viel mij hierbij op dat door de resource controllers de tests makkelijk zijn uit te breiden omdat iedere controller dezelfde soort functies bevat voor CRUD.

10 Sprint 5: stijlen en afbeelding bibliotheek

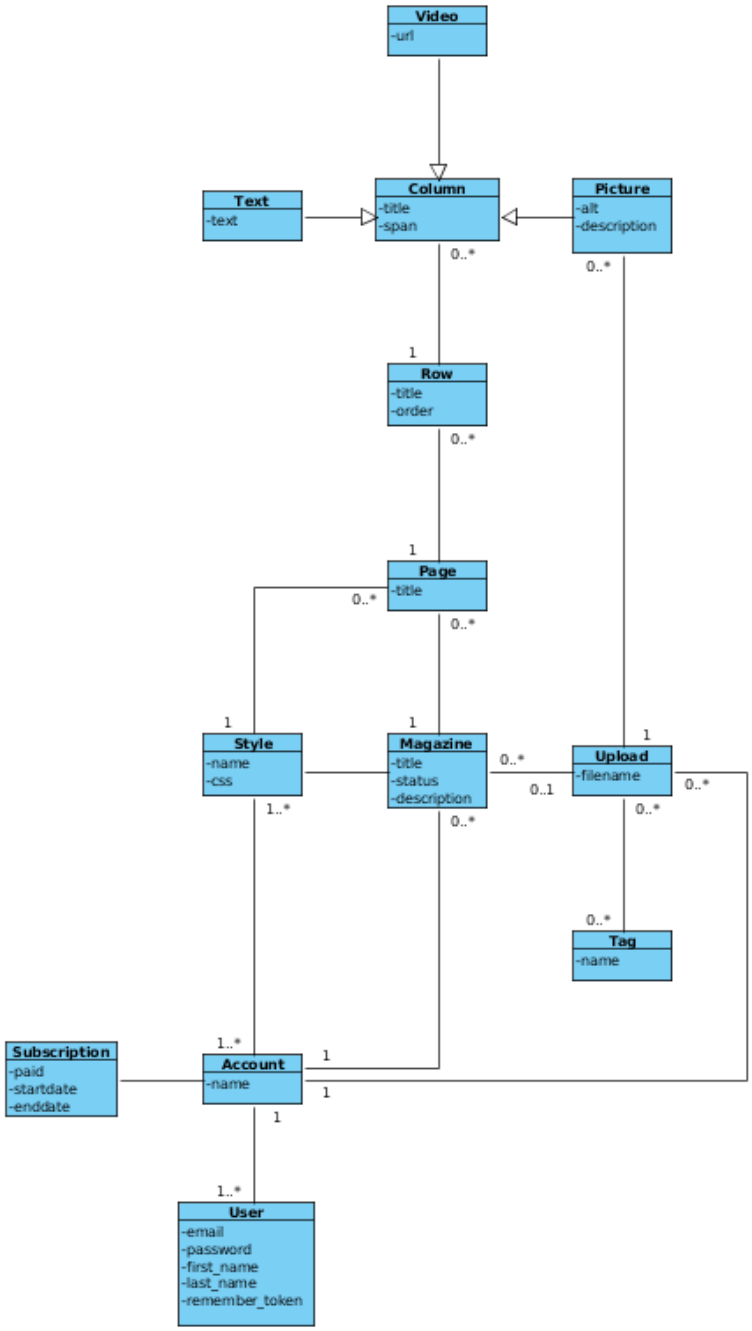
Sprint 5 van het project en tevens de laatste sprint in deze afstudeerperiode richt zich erop om de eerste versie van het eindproduct met basis functionaliteit volgens afspraak neer te zetten (zie ”3 De opdracht: Ezine 3.0”). Gezien de hoeveelheid user stories zal ik alleen de interessante toelichten(mondeling kan ik de rest ook toelichten). De volgende user stories zijn daarbij gerealiseerd tijdens deze sprint:

10.1 Sprint backlog

User story	**Prioriteit**	**Story points**
Als gebruiker wil ik een account kunnen aanmaken.	H	2
Als ontwikkelaar wil ik een stijl kunnen toewijzen aan het account van de klant zodat we op maat gemaakte stijlen kunnen maken voor bepaalde klanten.	H	3
Als gebruiker wil ik een stijl kunnen toevoegen aan een pagina.	H	3
Als gebruiker wil ik afbeeldingen kunnen uploaden.	H	3
Als gebruiker wil ik mijn afbeeldingen kunnen beheren in een overzicht.	H	2
Als gebruiker wil ik afbeeldingen kunnen toevoegen aan de pagina’s.	H	2
Als gebruiker wil ik een cover kunnen toevoegen aan een magazine.	H	2
Als gebruiker wil ik de mogelijkheid om de column type aan te kunnen passen.	H	4
Als opdrachtgever wil ik de titel van de pagina niet in magazine zelf.	H	1
Als gebruiker wil ik in redactor de volgende opties hebben: bold, italic, link, underlined, unordered list, ordered list, tabel, html, dropdown voor h2 en h3 koppen.	H	1
Als gebruiker wil ik bij meer dan 10 magazines pagination zien zodat er overzicht blijft.	H	1
Als gebruiker wil ik bij het toevoegen van een blok dat er automatisch naar toe wordt gescrolt zodat het duidelijk is welk blok er nieuw bij is gekomen.	H	1
Als gebruiker wil ik afbeeldingen kunnen verwijderen in afbeelding bibliotheek.	H	1
Als gebruiker wil ik kunnen klikken op de afbeelding bij het selecteren van een cover afbeelding (niet alleen op radiobutton).	H	1
Als gebruiker wil ik dat de interface gebruiksvriendelijk is.	H	3

10.2 Afbeelding bibliotheek

De opdrachtgever heeft de wens om een afbeelding bibliotheek in de applicatie te implementeren. Gebruikers kunnen hier eigen afbeeldingen uploaden, de afbeeldingen beheren en gebruiken in een magazine als cover- of paginaafbeelding. De bedoeling hierbij is dat een afbeelding hergebruikt kan worden op meerdere plekken. Ik ben begonnen met het uitbreiden van het klassendiagram (zie afbeelding 51). Om het afbeeldingsbestand te kunnen koppelen aan andere modellen heb ik een Upload class gecreëerd dat de naam bevat van de afbeelding in de vorm van een bestand.



Afbeelding 44: Pagina stijl en het toevoegen van afbeeldingen.

10.2.1 Upload realisatie

Als eerste heb ik de upload functionaliteit gemaakt. Ook hierbij biedt Laravel hulpmiddelen om dit op een efficiënte manier te realiseren (zie afbeelding 51). Laravel maakt een object van de image, dit object krijgt een aantal methodes mee die helpen met het verwerken van de afbeelding. Om te voorkomen dat er een bestaande afbeelding wordt overschreven met dezelfde bestandsnaam geef ik de bestandsnaam een prefix van zes random karakters met de str-random() functie. Vervolgens maak ik een nieuw Upload object aan, sla ik de upload path met filename op en koppel ik de afbeelding aan het account van de huidige gebruiker. Als laatste verplaats ik het bestand naar de upload map en return ik een bericht dat de afbeelding is toegevoegd. Na het uploaden is het de bedoeling dat de gebruiker zijn afbeeldingen kan bezichten en beheren in de afbeelding bibliotheek.

```php
/**
 * Store a newly created resource in storage.
 *
 * @return Response
 */
public function store()
{
    $file = Input::file('image');
    $destinationPath = 'upload/';
    $filename = str_random(6) . '_' . $file->getClientOriginalName();

    $upload = new Upload;
    $upload->filename = $destinationPath.$filename;
    $upload->account_id = Auth::user()->account->id;
    $upload->save();

    $file->move($destinationPath, $filename);

    Session::flash('flash_message', "Afbeelding toegevoegd");
    return Redirect::to('admin/multimedia');
}
```

Afbeelding 45: De store functie in de upload resource controller

10.2.2 Afbeelding bibliotheek

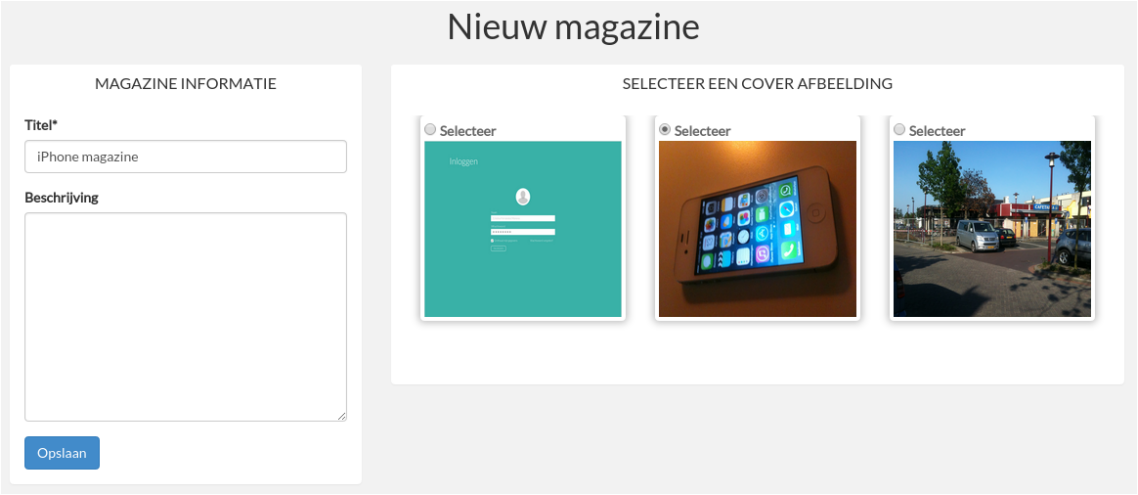
Zoals in afbeelding 51 is te zien had ik al een redirect gemaakt naar admin/multimedia. Ik heb daarom een nieuwe (resource) MultimediaController aangemaakt voor de CRUD bewerkingen van bestaande afbeeldingen. Voor het overzicht haal ik de uploads op van de huidige gebruiker en geef ik deze weer in de view.

```php
public function index()
{
    $uploads = Auth::user()->account->uploads;

    return View::make('admin.multimedia.index', array('uploads' => $uploads));
}
```

Afbeelding 46: Het ophalen van de uploads van de gebruiker

10.2.3 Magazine covers en pagina afbeeldingen

Door de uploads beschikbaar te maken als optie in de interface (zie afbeelding 47) kan het upload id worden meegegeven aan een nieuw magazine of een pagina afbeelding. Op die manier kan er een relatie worden aangemaakt tussen een magazine en een picture column waardoor de juiste afbeelding kan worden weergegeven.



Afbeelding 47: Aanmaken van een magazine met cover keuze

10.3 Pagina stijlen

Pagina stijlen zijn een belangrijk onderdeel van de paginaopmaak. Niet alleen de rijksoverheid stelt de eis om zijn eigen huisstijl te kunnen toepassen in een magazine maar het kan daarnaast ook een interessant verdienmodel vormen. De verwachting is dat andere klanten ook een eigen huisstijl willen toepassen in hun online magazines, Studio Projectie kan hierop inspelen om tegen betaling een stijl te ontwikkelen naar de wensen van de klant welke dan ook exclusief bruikbaar is voor die klant.

De oplossing die ik hiervoor heb bedacht komt voort uit de ervaring die ik heb opgedaan bij mijn vorige baan. We boden daar geautomatiseerd webwinkels aan, iedere webwinkel had dezelfde HTML structuur en de huisstijl van de klant werd toegepast door alleen de CSS aan te passen. Een zelfde strategie is perfect bruikbaar in deze situatie, de HTML structuur is namelijk altijd hetzelfde in magazine pagina's door het gebruik van het bootstrap grid systeem. Door gebruik te maken van CSS combinators[11] kan met het toewijzen van een enkele CSS class aan een magazine pagina alle pagina elementen worden aangesproken. Het toepassen van een huisstijl was een zwak punt in het oude Ezine2, de klacht van de frontend-ontwikkelaars was dat het teveel tijd koste. Omdat een frontend-ontwikkelaar zich met zich met deze werkwijze alleen hoeft te beperken tot de CSS opmaak, kan een huisstijl binnen een paar minuten worden toegepast waardoor een op maat gemaakte stijl ontwikkelen meer winst oplevert dan in de oude situatie.

[11] http://www.w3schools.com/css/css-combinators.asp

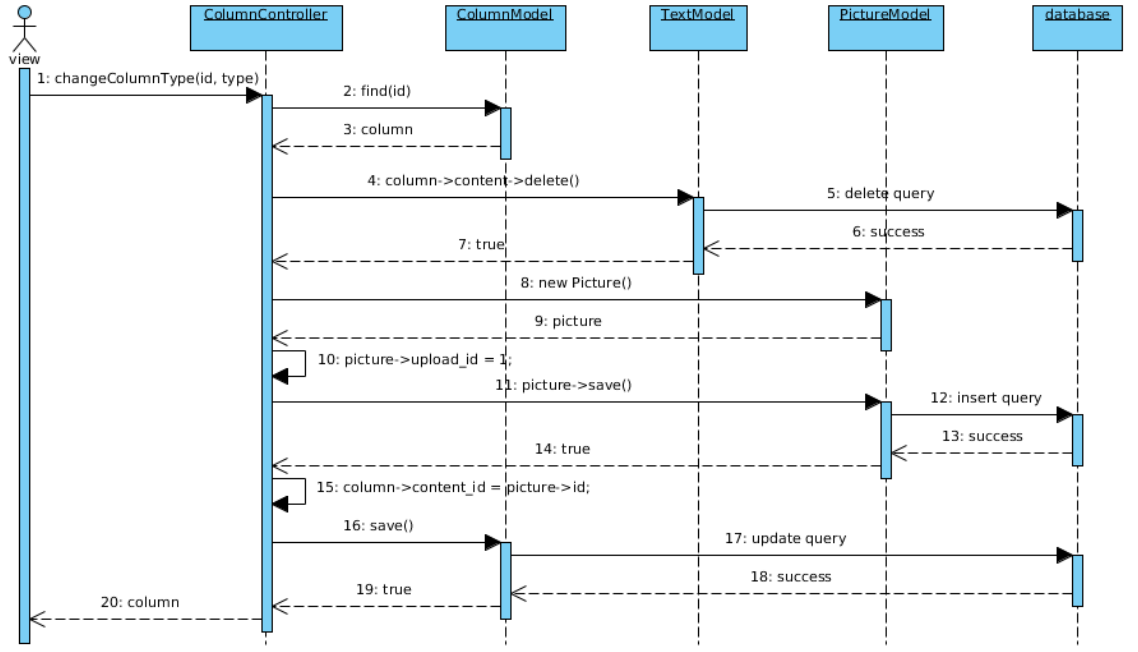
10.3.1 Toepassing

In het eerdere klassendiagram (afbeelding 51) was de Style class al te zien met een relatie naar Account, Magazine en Page. De Style class werkt op een soortgelijke manier als de Upload class. De Style class is gekoppeld aan een bestand, in dit geval een CSS bestand. Op die manier kan er een CSS met stijl eigenschappen worden gekoppeld aan het account van een klant. De gebruikers van dat account kunnen vervolgens de style toepassen op een magazine of alleen een magazine pagina om de magazine style te overschrijven.

10.4 Column type aanpassen

De opdrachtgever had de wens dat de klant bij het plaatsen van een block het type van het column kan aanpassen. Een tekst column moet bijvoorbeeld kunnen worden aangepast naar een afbeelding column. Op die manier hoeft in de editor alleen de structuur worden aangeboden aan de gebruiker en kan de gebruiker vervolgens de inhoud bepalen.

Het aanpassen het column type is met de huidige constructie al mogelijk zonder grote aanpassingen, wat ook meteen de flexibiliteit aantoont. In het klassendiagram in afbeelding 51 is te zien hoe ik eerder al de Column class als superclass had opgesteld met de subclasses Text, Picture en Video. Om het type van een Column te wijzigen kan het huidige column object worden behouden maar wijzig ik de subclass naar het gewenste type. Op die manier blijven de column eigenschappen behouden maar krijgt het de eigenschappen van een ander type (zie afbeelding 54).



Afbeelding 48: Het aanpassen van een tekst column naar een afbeelding column

10.5 Testen

Omdat dit de laatste sprint is voor deze afstudeerperiode wil de opdrachtgever nog graag een aantal bedrijfseigen tests uitvoeren is acceptabel genoeg is voor de overdracht, het gaat hierbij om een code review en een bug test.

10.5.1 Beheeracceptatietests

De code review en bug test zijn voor de algehele applicatie, beide tests hebben het doel om te kijken of de code logisch is opgebouwd, gedocumenteerd is met commentaar, er duidelijke naamconventies zijn gebruikt en de structuur goed is. Dit is voor de opdrachtgever belangrijk zodat hij in te schatten of de applicatie beheerbaar is in productie.

De code review De opdracht hierbij is dat ik de structuur en code laat zien aan twee ervaren programmeurs om te kijken of de applicatie ook qua structuur en code netheid aan de verwachtingen voldoet.

Tijdens de code review heb ik uitleg gegeven over de werking van de software, ik heb eerst het klassendiagram laten zien om een idee te geven van de structuur en uitleg gegeven over de werking van het Laravel framework. Vervolgens heb ik de functionaliteit laten zien en de technische uitwerking achter de functionaliteit. De algemene feedback die ik kreeg op de code en structuur was positief met enkele kleine verbeterpunten zoals het verder opslitsen van de templates.

Uitvoer van een ”bug test” De ”bug test” is geen officiële test maar wordt in het bedrijf gebruikt om te toetsen of de code ook in de praktijk logisch in elkaar zit. Bij de test worden er bewust een aantal bugs aangebracht in de software en wordt er gekeken hoe snel een ontwikkelaar dat niet bekend is met de software de fout kan achterhalen. De theorie hierachter is dat als de code logisch in elkaar zit dat dit weinig tijd in beslag zou moeten nemen.

Ik heb daarom bewust een aantal fouten in de software gemaakt, deze heb ik verdeelt over de routes, controllers, views en database. Vervolgens heb ik twee ontwikkelaars gevraagd of ze het kunnen oplossen om te kijken of ze alles makkelijk kunnen vinden. Dit bleek bij beide gevallen geen problemen op te leveren, iedere bug werd binnen 5 minuten opgelost wat volgens de opdrachtgever een positief resultaat waardoor de test als geslaagd kan worden beschouwd.

10.5.2 Gebruikersacceptatietests

Een niet functionele user story beschrijft dat de interface gebruiksvriendelijk moet zijn. Dit is net als in de vorige sprint gedaan door een willekeurige gebruiker instructies te geven op basis van de user stories die deze sprint zijn uitgewerkt, bijvoorbeeld: plaats een nieuwe afbeelding in de afbeeldingbibliotheek of maak een magazine pagina met een bepaalde layout. Op die manier kan er worden gekeken welke problemen de gebruiker ondervindt bij de uitvoer van een instructie. Op die manier kwamen er een aantal interessante bevindingen naar boven, gebruikers willen bijvoorbeeld graag ook de mogelijkheid om afbeeldingen te uploaden in het magazine en niet alleen in de afbeeldingbibliotheek. Deze nieuwe bevindingen zijn daarom opgenomen in de product backlog zodat deze later kunnen worden opgenomen in een sprint.

10.5.3 Regressie tests

Omdat er weer een aantal nieuwe CRUD methoden zijn toegevoegd heb ik de unit tests uitgebreid voor de nieuwe afbeelding opties en de stijlen. De meeste aanpassingen gingen in dit geval om het toevoegen van de relaties in de bestaande models. Daarnaast zijn de bestaande tests nogmaals uitgevoerd om te kijken of bestaande methoden nog naar behoren werkte met de nieuwe aanpassingen. Dit bleek geen probleem te zijn.

11 Evaluatie

11.1 Procesevaluatie

Het project is volgens een Studio Projectie variant van Scrum uitgevoerd, een werkwijze waarbij bepaalde elementen van Scrum worden gebruikt. Ik vond het daarbij prettig dat ik veel verantwoordelijkheid en vrijheid kreeg bij de uitvoering van het project. Er is regelmatig communicatie geweest met de opdrachtgever, dit heeft invloed gehad op de inhoud van de sprints. Door het ontwikkelen van concepten in een vroeg stadium, bijvoorbeeld met de rijksoverheid richtlijnen en de opmaak van de magazine. Kon ik op die manier zorgen wegnemen bij de opdrachtgever, deze werkwijze gaf de opdrachtgever meer vertrouwen omdat hij een beeld krijgt van de uitwerking.

Omdat er een aantal uitdagende beslissingen moesten worden genomen heb ik mijzelf tijdens het proces meer tijd gegund om tot een weloverwogen keuze te komen en te adviseren aan de opdrachtgever. Om hier extra tijd voor te nemen was ook te rechtvaardigen omdat versie 1 en 2 van Ezine geen succes waren door een gebrek aan flexibliteit en uitbreidbaarheid. Het was daarom van belang dat het deze keer wel goed doordacht zou zijn en ik denk dat dit is geslaagd door mij vooral te focussen op een goed uitgedachte basis dat aansluit op de wensen.

11.2 Productevaluatie

Ik ben tevreden over het eindresultaat, het zit solide in elkaar. Ik heb naar mijn idee de technieken en architectuele beslissingen genomen die goed aansluiten bij de wensen en eisen van de opdrachtgever, welke nauw betrokken is geweest bij de ontwikkeling van het product. Als ik terug kijk naar de opdrachtbeschrijving heb ik ook voldaan aan de vooraf opgestelde doelen: het product is ontwikkeld in het Laravel framework, het beschikt over de basisfunctionaliteit om een magazine op te maken en te publisheren, het voldoet aan de richtlijnen van de rijksoverheid en het technisch ontwerp is uitbreidbaar. Ik heb het product ontwikkeld voor mijn begeleider en opdrachtgever, ik vond het daarom ook belangrijk om hun mening te vragen. Ze waren tevreden met de huidige staat en de kwaliteit van het product.

Als ik kritisch terug kijk met de kennis van nu zou ik een andere backend kiezen. Niet omdat Laravel slecht is, Laravel is een van de beste PHP frameworks waar ik mee heb gewerkt en heeft aan alle wensen voldaan. Maar ik denk dat een javascript backend als Node.js en MongoDB(database) in combinatie met AngularJS beter op elkaar aansluiten. Met de huidige opstelling communiceert AngularJS via JSON via de REST API met Laravel, Laravel zet de JSON om naar een object en slaat het via het ORM op in de MySQL database. Op zich is hier niks mis mee maar Node.js en MongoDB hebben als voordeel dat beide native met JSON werken, de communicatie kan in dat geval volledig van database tot frontend worden gedaan met JSON. Dit vormt architectureel gezien een betere homogone structuur dan met PHP en MySQL. Een aantal interessante bevindingen die ik tijdens dit project heb opgedaan en waar ik in mijn vrije tijd nog steeds mee ben met experimenteren op dat gebied.

Als ik de technische aspecten even opzij zet en het product vanuit de ogen van de opdrachtgever bekijk denk ik dat het gewoon een goed product is dat voldoet aan de

afspraken en potentie heeft als commercieel product. Opmaak van online magazines is een onbekend concept, er zijn geen standaarden voor en er zijn weinig tot geen webapplicaties die dit op een gebruiksvriendelijke manier aanbieden. Wat misschien ook wel iets zegt over de moeilijkheidsgraad. Pagina opmaak zonder technische kennis op basis van de Twitter Bootstrap structuur is voor zover ik weet ook nog niet eerder in een CMS gebruikt, maar zou ook interessant zijn voor een CMS. Door dit unieke concept is het gelukt om de eisen te realiseren met als eindresultaat een webapplicatie waarmee, zonder technische kennis, online magazines kunnen worden opgemaakt.

11.3 Beroepstaken evaluatie

In dit hoofdstuk evalueer ik de beroepstaken die ik voor aanvang van afstuderen heb opgesteld.

11.3.1 1.4 Uitvoeren analyse door definitie van requirements, niveau 3

Op eigen inititatief heb ik regelmatig bijeenkomsten ingepland met opdrachtgever, begeleider, frontend ontwikkelaars en ontwerpers. De opdrachtgever wist door Ezine1 en 2 goed wat zijn algemene wensen waren voor Ezine3. Daarnaast heb ik gesprekken gevoerd met de eerdere ontwikkelaars van Ezine 1 en 2. Op die manier kon ik een beeld vormen van het project en de wensen opstellen in de vorm van user stories. Door verschillende expertises te betrekken bij de periodieke bijeenkomsten werden de eisen meer verfijnt wat ten goede kwam in de kwaliteit van de user stories en de uitwerking van de user stories.

Door de onzekerheden die er in het begin waren zijn de user stories regelmatig naar voortschrijdend inzicht aangepast. Zo heb ik bijvoorbeeld in overleg besloten met opdrachtgever en begeleider om na de eerste sprint twee spikes uit te voeren om meer inzicht te krijgen op de realisering van responsive magazines op basis van een grid systeem, en de techniek voor de opmaak van de magazines. Achteraf gezien ben ik tevreden met deze keuze omdat het voor mijzelf meer inzicht gaf en mijn opdrachtgever meer vertrouwen kreeg in het project. Daarbij leverde het concepten en kennis op dat ik later kon verwerken in het project.

11.3.2 3.2 Ontwerpen systeemdeel, niveau 4

Ezine3 was op het gebied van ontwerp een uitdaging omdat het veel dynamische eisen had op het gebied van paginaopmaak en dynamische oplossingen zijn vaak de moeilijkste. Een oplossing waar ik zelf erg tevreden mee ben is om de paginaopmaak op basis van bootstrap grid op te bouwen en deze componenten (row, column, content) als individuele objecten te gebruiken en op te slaan in de database. Deze oplossing maakte het niet alleen mogelijk om magazinepagina's responsive op te bouwen maar ook om de gebruiker met slechts enkele interface knoppen een magazinepagina op laten te bouwen wat ten goede kwam in de gebruiksvriendelijkheid. Dankzij de overerving tussen de superclass Column en de subclasses Text, Video en Picture werd het ook mogelijk om van een bestaande column de inhoudt aan te passen van bijvoorbeeld tekst naar een afbeelding. Door de individuele componenten was het ook mogelijk om ze individueel een dynamisch eigen stijl te geven met CSS. Het ontwerp biedt alle opties die nodig waren en is het in de toekomst eenvoudig uit te breiden met nieuwe mogelijkheden. Achteraf gezien denk ik

nog steeds dat dit een van de best mogelijke oplossingen was in deze situatie en dat dit ontwerp ook niet zou mistaan in een modern content management system.

De keuze voor het AngularJS framework als frontend maakte het mogelijk om de pagina asynchroon op te bouwen doormiddel van API calls met het Laravel framework. AngularJS biedt op zijn beurt net als Laravel een MVC structuur en is in tegenstelling tot bijvoorbeeld jQuery gestructureerd opgebouwd. Ondanks dat het goed werkt is dit tegelijk ook een punt dat achteraf gezien misschien beter zou kunnen. Beide frameworks bieden een schone structuur maar de combinatie van beide frameworks maakt de architectuur soms in zijn geheel complexer dan noodzakelijk is. Een actie van de gebruiker moet namelijk twee MVC modellen door voordat het de database bereikt, vervolgens moet de response weer door twee MVC modellen heen om de interface te updaten. Beide frameworks geven flexibiliteit omdat ze individueel van elkaar functioneren maar maken de structuur ook gevoeliger voor fouten, het heeft zo zijn voor- en nadelen. Maar dit is wel een punt waar ik in de toekomst extra rekening mee zal houden. Een eenvoudige applicatie is namelijk eenvoudiger te beheren, en als het beheer eenvoudig is zijn de kosten lager. Maar omdat AngularJS alleen wordt gebruiker in de magazine editor is het in deze situatie beheersbaar.

11.3.3 3.3 Bouwen applicatie, niveau 3

De beroepstaak ”Bouwen applicatie” beschrijft dat er wordt voldaan aan niveau 3 als er geavanceerde eigenschappen van een taal worden gebruikt en er rekening wordt gehouden met toekomstige wijzigingen, testbaarheid en hergebruik. Mijn keuze voor bijvoorbeeld AngularJS in plaats van jQuery is een belangrijke aanwijzing dat ik veel waarde heb gehecht aan deze punten. jQuery was de makkelijke keuze, AngularJS de betere omdat het meer structuur biedt met MVC waardoor de AngularJS-client een applicatie is op zich, dat uitbreidbaar en testbaar is. Daarnaast wordt de backend gebruikt als API wat een hoge herbruikbaarheid met zich meebrengt. Zo is het bijvoorbeeld mogelijk om in de toekomst een alternatieve client te ontwikkelen, voor een nieuwe klant met afwijkende eisen, dat op dezelfde API kan worden ontwikkeld.

De beroepstaak beschrijft daarnaast dat er wordt voldaan aan niveau 4 als er gebruik is gemaakt van een framework in een geavanceerde ontwikkelomgeving. Ik ben van mening dat ik ook aan niveau 4 heb voldoen. Omdat ik gebruik heb gemaakt van een het Laravel framework(PHP), Twitter Bootstrap framework(jQuery, CSS, HTML), en AngularJS framework(Javascript) binnen hetzelfde project in een geavanceerde Linux ontwikkelomgeving met Composer, Apache, MySQL met versiebeheer in Subversion.

Persoonlijk ben ik het niet helemaal eens met de criteria voor niveau 4. Een framework maakt de ontwikkeling van een applicatie niet complexer, het maakt het juist efficienter en makkelijker. Kennis van minimaal 1 framework is meestal ook een vereiste als basiskennis je als software ontwikkelaar gaat solliciteren. Eigenlijk zou het gebruik van een (MVC) framework onderdeel moeten zijn van de opleiding.

11.3.4 3.5 Uitvoeren van en rapporteren over het testproces, niveau 3

De nadruk bij het project lag voornamelijk bij het bedenken en uitvoeren van oplossingen op de problemen waar het oude Ezine2 mee kampte. Dit is ook de

reden waarom de tests vaak uitkwamen op verschillende soorten acceptatie tests zoals de gebruikersacceptatietest, functionele acceptatietest, beheersacceptatietest en een stress test. Op die manier kon namelijk worden gekeken of de bedachte oplossingen ook voldeden aan de verwachtingen. Door de interactie met verschillende mensen met verschillende expertises werd er steeds vanuit een ander oogpunt naar de software gekeken waardoor de kwaliteit door voortschrijdend inzicht verbeterde. Op die manier kon er ook goed rekening worden gehouden met de niet functionele user stories voor usability en flexiliteit van de software.

Daarnaast heb ik gebruik gemaakt van unit tests om de technische kant van de software te testen. De unit tests zijn dankzij de resource controllers eenvoudig uit te breiden en bruikbaar voor regressietests. Dit was voor mezelf nuttig bij de ontwikkeling om te kijken of alles nog naar verwachting werkt na een wijzing, maar ook voor de overdracht omdat andere ontwikkelaars na het installeren van het project op een server meteen kunnen zien of de software naar behoren werkt.

Bijlage A: Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok	2014-1.2 (start uiterlijk 12 mei 2014)
Startdatum uitvoering afstudeeropdracht	12 mei 2014
Inleverdatum afstudeerdossier volgens jaarrooster	6 oktober 2014
Studentnummer	20062006
Achternaam	Wassenburg
Voorletters	T.A.N.
Roepnaam	Tim
Adres	Wederikstraat 218
Postcode	2403CL
Woonplaats	Alphen a/d Rijn
Telefoonnummer	-
Mobiel nummer	0639469516
Privé emailadres	t.a.n.wassenburg@student.hhs.nl
Opleiding	HBO Informatica
Locatie	Den Haag
Variant	Voltijd
Naam studieloopbaanbegeleider	G.A. Mijnarends
Naam begeleidend examinator	J.D. Maas
Naam tweede examinator	G.M. Tuk
Naam bedrijf	Studio Projectie
Afdeling bedrijf	ICT
Bezoekadres bedrijf	Haagweg 4 G5-G6
Postcode bezoekadres	2311AA
Postbusnummer	-
Postcode postbusnummer	-
Plaats	Leiden
Telefoon bedrijf	071 513 89 85
Telefax bedrijf	-
Internetsite bedrijf	projectie.com
Achternaam opdrachtgever	Dhr. T. van Dobbenburgh
Voorletters opdrachtgever	Tijmen
Titulatuur opdrachtgever	
Functie opdrachtgever	Management
Doorkiesnummer opdrachtgever	-
Email opdrachtgever	tijmen@projectie.com

Achternaam bedrijfsmentor	Dhr. M. Kucevic
Voorletters bedrijfsmentor	Miljko
Titulatuur bedrijfsmentor	
Functie bedrijfsmentor	Software-ontwikkelaar
Doorkiesnummer bedrijfsmentor	-
Email bedrijfsmentor	miki@projectie.com
Doorkiesnummer afstudeerder	-
Functie afstudeerder (deeltijd/duaal)	Software-ontwikkelaar

Titel afstudeeropdracht

De herontwikkeling van Ezine, een applicatie voor de creatie van online magazines.

Opdrachtomschrijving

Bedrijf

Studio Projectie is in 2000 begonnen met 6 mensen en is inmiddels gegroeid naar 25 mensen. Het bedrijf heeft 18 ontwikkelaars in dienst, de ontwikkelaars hebben ervaring in het bouwen van creatieve en betaalbare websites en het bieden van innovatieve en gebruiksvriendelijke internetoplossingen. De overige mensen zijn werkzaam in het management, marketing en administratie.

De klanten zijn divers, van hogescholen, universiteiten, gemeenten tot verzekeringsmaatschappijen. Het belangrijkste product van Studio Projectie is de DigitaleFactuur, dit is een online facturatiesysteem met het keurmerk voor E-facturen. DigitaleFactuur heeft duizenden gebruikers en richt zich op gebruiksvriendelijkheid. De bedrijfscultuur is informeel maar professioneel.

Probleemstelling

Op dit moment werkt Studio Projectie met de webapplicatie Ezine. Met Ezine is het mogelijk om een eigen onlinemagazine te maken zoals bijvoorbeeld een dynamisch jaarverslag, een nieuwbouwtraject of evenement met beeld en geluid. De huidige versie is verouderd en beperkt in zijn mogelijkheden. Dat komt omdat het is gemaakt in het zelfbouw framework Prowriter, waarvan de ontwikkeling stil ligt.

Prowriter is het framework dat 13 jaar geleden opgezet is. In de 13 jaar is het doorontwikkeld en kan worden ingezet voor het maken van eenvoudige websites. In praktijk blijkt het framework niet geschikt te zijn voor dit complexe project omdat het een zelfbouw framework en alle functionaliteiten ook zelf ontwikkeld moeten worden. Dit is erg tijdrovend en risicovol omdat het gebaseerd is op inmiddels verouderde technieken. Voor complexe sites wordt dit systeem ook niet meer gebruikt en wordt er gekozen voor een ander framework (Yii of Laravel) waar heel veel dingen al in geregeld zijn. Veiligheid, goede structuur en ondersteuning voor plugins zijn belangrijke punten die ontbreken in Prowriter.

Het is wenselijk om Ezine opnieuw te ontwerpen en op het Laravel framework weer op te bouwen. Studio Projectie heeft gekozen voor Laravel omdat het een opkomend framework is dat gericht is op de toekomst. Daarnaast is het geschikt voor ieder type project en heeft het in tegenstelling tot het Prowriter framework oplossingen voor veel voorkomende functionaliteit. Studio Projectie is op dit moment bezig met kennisverbreding over Laravel en heeft meerdere projecten draaien in het framework. Het bedrijf is bekend met het framework en weet inmiddels welke mogelijkheden het biedt wat voordelig is bij de ontwikkeling van Ezine.

Naast het opzetten van de basisfunctionaliteiten is het wenselijk om ook nieuwe functionaliteiten bij te bouwen. De bedoeling is dat het naast de computer ook geschikt is voor de tablet, dit zal in samenwerking met ontwerpers worden gerealiseerd. Het uiteindelijke doel is een robuust systeem dat commercieel is te verkopen, waarbij rekening is gehouden met toekomstige uitbreidingen.

Doelstelling van de afstudeeropdracht

Een robuust systeem voor het maken en beheren van online magazines.

Resultaat

Een actuele gebruikersvriendelijke applicatie voor het maken en beheren van online magazines.

Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

Vooronderzoek (2 weken)

- Inventariseren wensen/nieuwe functionaliteiten
- Opstellen product backlog
- Plan van Aanpak schrijven

Iteratief

- Sprint backlog opstellen
- Technisch ontwerp
- Bouwen/ontwikkelen
- Testen
- Retrospectief

Afsluiting (2 weken)

- Implementeren

Op te leveren (tussen)producten

- Plan van Aanpak
- Requirements document
- Technisch ontwerp
- Source code
- Tests

Te demonstreren competenties en wijze waarop

- 1.4 Uitvoeren analyse door definitie van requirements, niveau 3
  Opstellen requirements/user stores
- 3.2 Ontwerpen systeemdeel, niveau 4
  Technisch ontwerp maken
- 3.3 Bouwen applicatie, niveau 3
  Bouwen/Ontwikkelen van de applicatie
- 3.5 Uitvoeren van en rapporteren over het testproces, niveau 3
  Testen

Bijlage B: Plan van Aanpak

Opdrachtsomschrijving

Aanleiding

Op dit moment werkt Studio Projectie met de webapplicatie Ezine. Met Ezine is het mogelijk om een eigen onlinemagazine te maken zoals bijvoorbeeld een dynamisch jaarverslag, een nieuwbouwtraject of evenement met beeld en geluid. De huidige versie is verouderd en beperkt in zijn mogelijkheden. Dat komt omdat het is gemaakt in het zelfbouw framework Prowriter, waarvan de ontwikkeling stil ligt.

Probleemstelling

Prowriter is het cms dat 13 jaar geleden opgezet is. In de 13 jaar is het doorontwikkeld en kan worden ingezet voor het maken van eenvoudige websites. In praktijk blijkt het cms niet geschikt te zijn voor dit complexe project omdat het een zelfbouw cms is en alle functionaliteiten ook zelf ontwikkeld moeten worden. Dit is erg tijdrovend en risicovol omdat het gebaseerd is op inmiddels verouderde technieken. Voor complexe sites wordt dit systeem ook niet meer gebruikt en wordt er gekozen voor een geavanceerd framework (Yii of Laravel) waar heel veel dingen al in geregeld zijn. Veiligheid, goede structuur en ondersteuning voor plugins zijn belangrijke punten die ontbreken in Prowriter. Ook het huidige ontwerp van Ezine in Prowriter is niet goed omdat er steeds rekening moest worden gehouden met de werking van Prowriter en hierdoor omwegen in het ontwerp moesten worden gebouwd die niet altijd handig zijn voor uitbreidbaarheid en beheer.

Doelstelling

Het is daarom wenselijk om Ezine opnieuw te ontwerpen en op het Laravel framework weer op te bouwen. Studio Projectie heeft gekozen voor Laravel omdat het een opkomend framework is dat gericht is op de toekomst, waarbij ook font-end ontwikkelaars (in tegenstelling tot bij het Yii framework) goed mee om kunnen gaan. Daarnaast is het geschikt voor ieder type project en heeft het in tegenstelling tot het Prowriter framework oplossingen voor veel voorkomende functionaliteit. Studio Projectie is op dit moment bezig met kennisverbreding over Laravel en heeft meerdere projecten draaien in het framework. Het bedrijf is bekend met het framework en weet inmiddels welke mogelijkheden het biedt wat voordelig is bij de ontwikkeling van Ezine.

Naast het opzetten van de basisfunctionaliteiten is het wenselijk om ook nieuwe functionaliteiten bij te bouwen. De bedoeling is dat het naast de computer ook geschikt is voor de tablet, dit zal in samenwerking met ontwerpers worden gerealiseerd.

Resultaat

Het uiteindelijke resultaat is een robuust systeem dat zicht onderscheid door gebruiksvriendelijkheid naar de gebruiker en commercieel is te verkopen, waarbij rekening is gehouden met toekomstige uitbreidingen.

Afbakening

In de oude versie van Ezine is de opmaak van magazines gecombineerd met een cms voor een website. De bedoeling is dat alleen de opmaak van de magazines in Ezine3.0 wordt geïmplementeerd, dus niet het CMS.

Randvoorwaarden

Voor een goede verloop van de ontwikkeling van het project zijn de volgende voorwaarden van belang.

- Toegang tot een repository voor versiebeheer.
- Toegang tot de oude versie van Ezine.
- Feedback en samenwerking met belanghebbenden voor Ezine3.0.
- Een geschikte werkplek met computer en internet.

Risicofactoren

Om risicos voor te zijn is er een analyse gemaakt van mogelijke risicos die kunnen voorkomen tijdens het project. Door te anticiperen op de risicos is het mogelijk om voortijdig oplossingen te bedenken. De kans dat het eindresultaat in gevaar komt wordt hierdoor kleiner.

Risico: Uitloop van de planning

Uitloop van de planning is een reeel risico bij de ontwikkeling van software. Dit kan diverse redenen hebben, het belangrijkste is hoe hiermee word omgegaan zodat de invloed op het eindresultaat kan worden geminimaliseerd.

Risico: Miscommunicatie

Door miscommunicatie is het mogelijk dat het eindresultaat verschilt van de wensen die de opdrachtgever of belanghebbende hebben. Om dit voor te zijn zal de softwareontwikkelaar ervoor zorgen dat het project word opgedeeld in kleine delen welke periodiek worden besproken met de opdrachtgever. Op die manier is er regelmatig communicatie en kunnen afwijkingen op tijd worden bijgestuurd.

Risico: Ziekte

Een risico is dat de ontwikkelaar word getroffen door ziekte en afwezig is, dit zou het project aanzienlijk vertragen. Het is in dit geval van belang om met de betrokken personen te communiceren welke maatregelen er worden genomen.

Risico: Dataverlies

Als dataverlies voorkomt kan dit een grote invloed hebben op het verloop en het eindresultaat van het project. Om dit risico te verkleinen word er gebruik gemaakt van versiebeheer met Subversion(SVN) voor de software en Google Drive voor documentatie. Als bijvoorbeeld een computer of laptop stuk gaat heeft dit geen invloed op de data omdat het niet lokaal staat opgeslagen, en kan er met minimaal tijdverlies op een vervangende computer worden verder gewerkt.

Risico: Benodigde personen niet beschikbaar

Om diverse redenen kan het voorkomen dat er personen die betrokken zijn bij de ontwikkeling van het project niet beschikbaar zijn. Omdat de consequenties per persoon zullen verschillen zal de oplossing afhankelijk zijn van de situatie. Bij eventuele langdurige afwezigheid van iemand is het van belang snel om in gesprek te gaan met de andere betrokkenen en te kijken naar een passende oplossing.

Risico: Expertise

Tijdens de ontwikkeling van EzOm risicos voor te zijn is er een analyse gemaakt van mogelijke risicos die kunnen voorkomen tijdens het project. Door te anticiperen op de risicos is het mogelijk om voortijdig oplossingen te bedenken. De kans dat het eindresultaat in gevaar komt wordt hierdoor kleiner.

Risico: Uitloop van de planning

Uitloop van de planning is een reeel risico bij de ontwikkeling van software. Dit kan diverse redenen hebben, het belangrijkste is hoe hiermee word omgegaan zodat de invloed op het eindresultaat kan worden geminimaliseerd.

Risico: Miscommunicatie

Door miscommunicatie is het mogelijk dat het eindresultaat verschilt van de wensen die de opdrachtgever of belanghebbende hebben. Om dit voor te zijn zal de softwareontwikkelaar ervoor zorgen dat het project word opgedeeld in kleine delen welke periodiek worden besproken met de opdrachtgever. Op die manier is er regelmatig communicatie en kunnen afwijkingen op tijd worden bijgestuurd.

Risico: Ziekte

Een risico is dat de ontwikkelaar word getroffen door ziekte en afwezig is, dit zou het project aanzienlijk vertragen. Het is in dit geval van belang om met de betrokken personen te communiceren welke maatregelen er worden genomen.

Risico: Dataverlies

Als dataverlies voorkomt kan dit een grote invloed hebben op het verloop en het eindresultaat van het project. Om dit risico te verkleinen word er gebruik gemaakt van versiebeheer met Subversion(SVN) voor de software en Google Drive voor documentatie. Als bijvoorbeeld een computer of laptop stuk gaat heeft dit geen invloed op de data omdat

het niet lokaal staat opgeslagen, en kan er met minimaal tijdverlies op een vervangende computer worden verder gewerkt.

Risico: Benodigde personen niet beschikbaar

Om diverse redenen kan het voorkomen dat er personen die betrokken zijn bij de ontwikkeling van het project niet beschikbaar zijn. ine3.0 word er gewerkt met het Laravel framework. De ervaring van Studio Projectie met Laravel is nog beperkt. Er bestaat dus een kans dat de ontwikkelaar vast komt te zitten door een probleem waar ook collega's niet direct een oplossing voor weten. In zo'n situatie loopt het project vertraging op. Om de vertraging te minimaliseren is het van belang dat er snel een oplossing word gevonden, dit kan door de Laravel documentatie te raadplegen of om te kijken of er een oplossing te vinden is op Laracasts (Laravel tutorial video's). Als dit ook geen oplossing biedt word er overleg gepleegd om te kijken of het mogelijk is om deze taak te laten liggen en andere taken te ontwikkelen.

Risico: Omvang

De verwachting is dat het een omvangrijk project is. Het risico bestaat dat het te groot is voor de geplande 20 weken. Daarom is het gebruik van Scrum nuttig, hierbij kan per sprint de belangrijkste user stories worden bepaald om de belangrijkste functionaliteiten in het eindresultaat te hebben. Daarnaast is het mogelijk om collega's in te schakelen door een team samen te stellen, ook hierbij is het gebruik van Scrum nuttig omdat de taken makkelijk kunnen worden verdeeld.

Projectorganisatie

Na een kort onderzoek naar een geschikte ontwikkelmethode is besloten om de SCRUM methodiek te gebruiken voor de ontwikkeling van het project. Scrum sluit goed aan op de situatie door zijn flexibelen karakter, structuur en de nauwe samenwerking met de opdrachtgevers.

Product backlog

De product backlog bevat een geprioriteerde lijst van de functionele eisen. De prioritering zal gebeuren in samenwerking met de opdrachtgevers.

Sprint planning

Aan het begin van een sprint vind er een sprintplanning plaats. Tijdens de planning worden de user stories in de product backlog opgenomen in de sprint backlog waarbij rekening wordt gehouden met de haalbaarheid.

Sprint backlog

Op basis van de prioritering word een sprint backlog opgesteld, de sprint backlog bevat de functionele eisen die worden uitgevoerd tijdens de sprint.

Sprints

Een sprint is een korte periode van 2 weken waarin de functionele eisen uit de sprint backlog worden uitgevoerd.

Retrospective

Na iedere sprint vind er een evaluatie plaats met de teamleden, de evaluatie heeft als doel om te kijken wat er goed ging en wat er eventueel de volgende sprint beter kan.

Rollenverdeling

Ontwikkelteam

Tim Wassenburg

Product owner

Philip Niewold
Tijmen Dobbenburgh

Wijze van rapporteren

Documentatie

Aan het einde van de sprint word er een retrospective worden gehouden, hierbij word ook de nieuwe documentatie opgeleverd en word er feedback gegeven op het resultaat. Op die manier word tijdens het proces de kwaliteit in de gaten gehouden en verbeterd.

Overleg

Naast de retrospective word er ook dagelijks kort overleg gehouden (daily scrum), na de lunch, zodat het team op de hoogte blijft van de status en voortgang. Ook word tijdens de sprintplanning overlegt over de prioritering van de user stories.

Benodigde mensen en middelen

Benodigde mensen

- Tijmen Dobbenburgh (opdrachtgever)
- Miki Kucevic (begeleider)
- Tim Wassenburg (ontwikkelaar)

Benodigde middelen

- Werkplek met computer
- Internet
- Software:

  Besturingssysteem

  IDE

  Office pakket

  Versiebeheer (Subversion)

  UML pakket

  Apache, MySQL en PHP.

  Laravel Framework

Planning

Opstart

In de eerste 2 weken word de opstart van het project voorbereid. Tijdens de voorbereiding worden de volgende werkzaamheden uitgevoerd:

- Plan van Aanpak
- Inventarisatie Ezine2.0
- Opstellen product backlog

Sprints (iteratief)

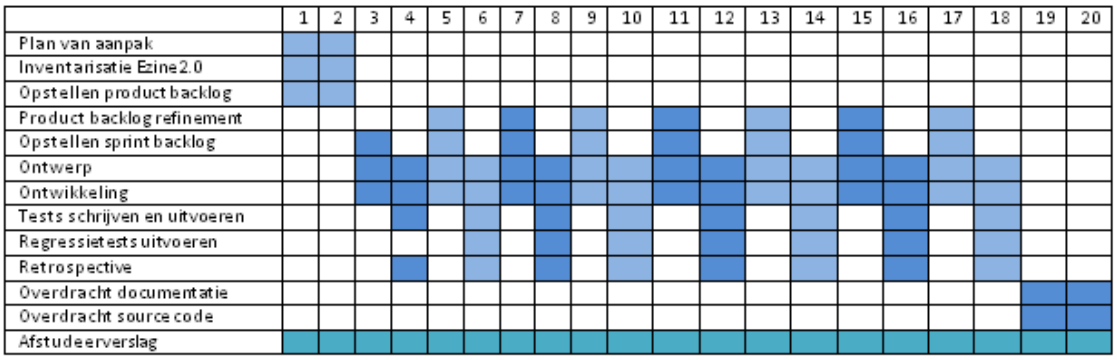
De inhoud van een sprint is van te voren niet vastgesteld maar word duidelijk na de product backlog refinement en het opstellen van de sprint backlog. Wel word tijdens iedere sprint een vast aantal werkzaamheden uitgevoerd:

- Product backlog refinement
- Opstellen sprint backlog
- Ontwerp maken
- Ontwikkeling
- Tests schrijven en uitvoeren
- Regressietests uitvoeren
- Retrospective

Oplevering

- Overdracht documentatie
- Overdracht source code

Overzicht



Afbeelding 49: de planning.

Beschrijving van mijlpaalproducten

Plan van Aanpak

Het plan van aanpak beschijft het plan voor de uitvoering van het project.

Requirements document

Het requirements document bevat de product backlog en de sprint backlogs. Het document word na iedere sprint incrementeel uitgebreid en aangepast zodat aan het einde van het project precies zichtbaar is welke taken er iedere sprint zijn uitgevoerd.

Design document

Het design document bevat UML diagrammen van het project. De diagrammen worden per sprint incrementeel uitgebreid zodat er altijd een document beschikbaar is met een up-to-date ontwerp van de software. Het type UML diagram is afhankelijk van de situatie.

Source code

De software zelf word als source code opgeleverd. De source code is het resultaat van de eerder bepaalde taken en ontwerpen. Ook bevat de source code de tests.

Test document

Om te kijken of de source code voldoet aan de beschrijving van de taken die uit de sprint backlog zijn voortgekomen worden er tests geschreven en uitgevoerd. De tests en de resultaten worden gedocumenteerd in het test document.

Bijlage C: Requirements document

Inleiding

Na aanleiding van de ontwikkeling van Ezine3 worden er requirements opgesteld in de vorm van user stories. Er is gekozen voor user stories omdat deze meer flexibiliteit bieden dan traditionele requirements met use cases. Daarnaast worden user stories opgesteld in de taal van de opdrachtgever wat bevordelijk is voor de communicatie.

In dit document wordt in het laatste hoofdstuk de meest recente product backlog bijgehouden tijdens de ontwikkeling van het project. De bedoeling is dat als een andere ontwikkelaar dit document bekijkt hij/zij kan zien welke user stories zijn voltooid per sprint en wat de status is van de huidige product backlog. Op die manier is het voor iedere ontwikkelaar eenvoudig om nieuwe toepassingen toe te voegen aan de product backlog, te prioriteren en een nieuwe sprint in te plannen om Ezine3 verder te ontwikkelen.

Eerste opzet product Backlog

De product backlog beschrijft de wensen van de opdrachtgever en gebruikers, ze worden daarbij in de taal van de opdrachtgever beschreven. De reden hiervoor is dat het dan begrijpelijk is voor de opdrachtgever wat sneller leidt tot feedback en de verwachtingen en ideeen van de ontwikkelaar en de opdrachtgever zoveel mogelijk overeenkomen.

De prioritering is gedaan samen met de opdrachtgever. De user stories zijn ieder beoordeeld op afhankelijkheid, risico, business value en verkoopbaarheid op basis van laag (L), middel(M) en hoog (H). Vervolgens nemen we het gemiddelde van deze waarden wat de uiteindelijke prioriteitsbeoordeling vormt van de user story.

Als laatste zijn de user stories op volgorde gezet op basis van de prioriteit, op die manier is het direct duidelijk welke user stories de hoogste prioriteit hebben voor de komende sprint.

Beschrijving	**Prioriteit**	**Story points**
Als gebruiker wil ik een nieuw magazine kunnen aanmaken.	H	2
Als gebruiker wil een magazine beschikbaar kunnen stellen voor mijn publiek.	H	1
Als gebruiker wil ik een overzicht kunnen opvragen van mijn magazines zodat ik mijn magazines kan beheren.	H	2
Als gebruiker wil ik pagina's toe kunnen voegen aan mijn magazine.	H	3
Als gebruiker wil ik pagina's toe kunnen verwijderen.	H	2
Als gebruiker wil ik in kunnen loggen op mijn account om mijn magazines te kunnen beheren.	H	1
Als gebruiker wil ik een magazine kunnen opslaan als schets zodat ik kan overleggen voordat het magazine word gepubliseerd.	M	2
Als gebruiker wil ik voor een magazine een begin en einddatum kunnen instellen zodat de magazine alleen in een bepaalde periode online is te bekijken.	M	2
Als gebruiker wil ik met meerdere gebruikers gebruik kunnen maken van mijn account zodat ik samen met collegas aan een magazine kan werken.	L	2
Als gebruiker wil ik gebruik kunnen maken van een voorbeeld dat ik kan aanpassen aan mijn wensen zodat het makkelijker is om te beginnen aan een nieuw magazine.	L	4
Als gebruiker wil ik een sjabloonset kunnen toewijzen aan een magazine zodat paginas automatisch worden opgemaakt in de gewenste huisstijl.	L	4
Als gebruiker wil ik inhoud kunnen plaatsen op de paginas in een magazine.	L	5
Als gebruiker wil ik een wachtwoord reset kunnen doen zodat ik zelf mijn wachtwoord kan wijzigen als ik de oude ben vergeten.	L	3
Als gebruiker wil ik een domainnaam laten verwijzen naar mijn account zodat bezoekers makkelijk mijn magazines kunnen bereiken.	L	5
Als product owner wil ik dat het gratis account beperkingen heeft om gebruikeren te motiveren een betaald abonnement af te sluiten.	L	3
Als gebruiker wil ik een account aan kunnen maken zodat ik mijn online magazines kan beheren.	L	2
Als gebruiker wil ik een maillijst bijhouden zodat ik geinteresseerde een mail kan sturen bij een nieuw magazine.	L	5
Als rijksoverheid wil ik dat magazines responsive worden zodat de pagina leesbaar is op ieder formaat scherm.	L	4
Als rijksoverheid wil ik dat de lezer door de bladzijdes kan swipen op een touchscreen.	L	4

Sprint 1: de basis

De eerste sprint is voornamelijk gericht op het opstellen van de basis functionaleit waarop later verder kan worden gewerkt. Deze sprint is ook om kennis te maken met de nieuwe technieken zoals het Laravel framework.

Sprint backlog

User story	**Prioriteit**	**Story points**
Als gebruiker wil ik een nieuw magazine kunnen aanmaken.	H	2
Als gebruiker wil een magazine beschikbaar kunnen stellen voor mijn publiek.	H	1
Als gebruiker wil ik een overzicht kunnen opvragen van mijn magazines zodat ik mijn magazines kan beheren.	H	2
Als gebruiker wil ik pagina's toe kunnen voegen aan mijn magazine.	H	3
Als gebruiker wil ik pagina's toe kunnen verwijderen.	H	2
Als gebruiker wil ik in kunnen loggen op mijn account om mijn magazines te kunnen beheren.	H	1
Als gebruiker wil ik een magazine kunnen opslaan als schets zodat ik kan overleggen voordat het magazine word gepubliseerd.	M	2
Als gebruiker wil ik voor een magazine een begin en einddatum kunnen instellen zodat de magazine alleen in een bepaalde periode online is te bekijken.	M	2

Sprint 2: opmaak van magazine paginas(1)

In sprint 2 zijn voor een deel de resultaten uit de spikes verwerkt. Er word hierbij een eerste opzet gemaakt van de magazine opmaakt editor.

Sprint backlog

User story	Prioriteit	Story points
Als gebruiker wil ik blokken kunnen toevoegen aan een pagina.	H	4
Als gebruiker wil ik de structuur kunnen bepalen van een blok.	H	4
Als gebruiker wil ik tekst kunnen plaatsen in een column.	H	3
Als gebruiker wil ik afbeeldingen kunnen plaatsen in een column.	H	3
Als gebruiker wil ik video kunnen plaatsen in een column.	H	3
Als gebruiker wil ik een column leeg kunnen maken zodat ik extra structuur krijg in een blok.	H	2
Als gebruiker wil ik een titel kunnen geven aan een column.	H	2
Als opdrachtgever wil ik dat de ontworpen huisstijl wordt toegepast.	H	4
Als opdrachtgever wil ik dat het opmaken van magazines gebruiksvriendelijk is.	H	3
Als gebruiker wil ik een wachtwoord reset kunnen doen zodat ik zelf mijn wachtwoord kan wijzigen als ik de oude ben vergeten.	M	2

Sprint 3: opmaak van magazine paginas(2)

In sprint 3 wordt de functionaliteit van de magazine editor verder uitgebreid.

Sprint backlog

User story	Prioriteit	Story points
Als gebruiker wil ik een tekst kunnen wijzigen.	H	3
Als gebruiker wil ik een video kunnen wijzigen.	H	3
Als gebruiker wil ik een afbeelding kunnen wijzigen.	H	3
Als gebruiker wil ik een blok kunnen verwijderen.	H	2
Als gebruiker wil ik een ondertitel kunnen geven aan een afbeelding.	H	2
Als gebruiker wil ik een alt tekst kunnen geven aan een afbeelding	H	2
Als gebruiker wil ik dat er een standaard afbeelding wordt ingesteld als ik zelf geen afbeelding kies.	H	3
Als gebruiker wil ik de pagina titel kunnen wijzigen.	H	2
Als gebruiker wil ik de magazine titel kunnen wijzigen.	H	2
Als gebruiker wil ik de volgorde van de blokken kunnen bepalen.	H	4
Als gebruiker vind ik het niet meer nodig om een titel aan een blok te geven omdat de columns al titels hebben.	M	1

Sprint 4: opmaak van magazine paginas(3)

In deze sprint is de magazine view gemaakt waarin de magazine die wordt gemaakt in de magazine editor beschikbaar is voor het publiek.

Sprint backlog

User story	Prioriteit	Story points
Als gebruiker wil ik een magazine aan mijn lezers doorgeven met een url.	H	1
Als rijksoverheid wil ik dat de lezer door de bladzijdes kan swipen op een touchscreen.	H	4
Als rijksoverheid wil ik dat magazines responsive op basis van een 12 punts grid worden zodat de pagina leesbaar is op ieder formaat scherm.	H	4
Als gebruiker wil ik een overzicht van de pagina's kunnen zien als ik een magazine lees.	H	3
Als gebruiker wil ik met een links en rechts knop tussen de pagina's kunnen bladeren.	H	3
Als gebruiker wil ik dat de volgende pagina voor een klein deel zichtbaar is zodat het duidelijk is dat ik naar de volgende pagina kan navigeren.	H	3
Als gebruiker wil ik een column titel links, rechts of gecentreerd uit kunnen lijnen.	H	1
Als gebruiker wil ik een aantal voorbeeld pagina's bij de aanmaak van een nieuw magazine.	H	4
Als gebruiker wil ik een link kunnen toevoegen aan een tekst.	H	2

Sprint 5: stijlen en afbeelding bibliotheek

Ezine3 heeft in sprint 5 de mogelijkheid voor stijlen en een afbeelding bibliotheek erbij gekregen. Gebruikers kunnen nu eigen afbeeldingen uploaden en een custom style toepassen op een magazine.

Sprint backlog

User story	Prioriteit	Story points
Als gebruiker wil ik een account kunnen aanmaken.	H	2
Als ontwikkelaar wil ik een stijl kunnen toewijzen aan het account van de klant zodat we op maat gemaakte stijlen kunnen maken voor bepaalde klanten.	H	3
Als gebruiker wil ik een stijl kunnen toevoegen aan een pagina.	H	3
Als gebruiker wil ik afbeeldingen kunnen uploaden.	H	3
Als gebruiker wil ik mijn afbeeldingen kunnen beheren in een overzicht.	H	2
Als gebruiker wil ik afbeeldingen kunnen toevoegen aan de pagina's.	H	2
Als gebruiker wil ik een cover kunnen toevoegen aan een magazine.	H	2
Als gebruiker wil ik de mogelijkheid om de column type aan te kunnen passen.	H	4
Als opdrachtgever wil ik de titel van de pagina niet in magazine zelf.	H	1
Als gebruiker wil ik in redactor de volgende opties hebben: bold, italic, link, underlined, unordered list, ordered list, tabel, html, dropdown voor h2 en h3 koppen.	H	1
Als gebruiker wil ik bij meer dan 10 magazines pagination zien zodat er overzicht blijft.	H	1
Als gebruiker wil ik bij het toevoegen van een blok dat er automatisch naar toe wordt gescrolt zodat het duidelijk is welk blok er nieuw bij is gekomen.	H	1
Als gebruiker wil ik afbeeldingen kunnen verwijderen in afbeelding bibliotheek.	H	1
Als gebruiker wil ik kunnen klikken op de afbeelding bij het selecteren van een cover afbeelding (niet alleen op radiobutton).	H	1
Als gebruiker wil ik dat de interface gebruiksvriendelijk is.	H	3

Huidige product backlog

De de prioritering is bewust opengelaten voor de overdracht, zodat de ontwikkelaar en opdrachtgever op dat moment kunnen bepalen wat het belangrijkst is.

User story	**Prioriteit**	**Story points**
Als gebruiker wil ik standaard beschikken over een standaard magazine sjabloon.		5
Als opdrachtgever wil ik een voorbeeld afbeelding hebben in de afbeelding bibliotheek.		2
Als opdrachtgever wil ik dat het boekenkast ontwerp van Cristina wordt toegepast.		2
Als gebruiker wil ik de mogelijkheid om een sjabloon van magazine op te slaan zodat ik deze kan gebruiken bij het aanmaken van een nieuw magazine.		4
Als gebruiker wil ik bij het maken van een nieuwe pagina keuze uit verschillende layouts.		3
Als gebruiker wil ik een afbeelding kunnen wijzigen zonder weg te navigeren.		2
Als gebruiker wil ik de structuur van een blok achteraf kunnen wijzigen.		3
Als gebruiker wil ik op de voorpagina een voorbeeld zodat ik deze alleen hoef te wijzigen.		2
Als gebruiker wil ik een inhoudsopgave kunnen toevoegen aan mijn magazine.		3
Als gebruiker wil ik een pagina header kunnen instellen per pagina.		2
Als gebruiker wil ik een stijl kunnen toepassen per block.		2
Als gebruiker wil ik een stijl kunnen toepassen per column.		2
Als gebruiker wil ik een cover afbeelding van een magazine achteraf kunnen wijzigen.		2
Als opdrachtgever wil ik dat de afbeeldingen worden verkleint als ze te groot zijn.		3
Als gebruiker wil ik ook afbeeldingen kunnen uploaden als ik in een magazine zit zodat niet steeds heen en weer hoef te klikken tussen een magazine en de afbeelding bibluitheek.		2
Als gebruiker wil ik een pagina header kunnen instellen per pagina.		3
Als opdrachtgever wil ik dat de toolbar kleur van redactor wordt ingesteld.		1
Als gebruiker wil ik een domainnaam laten verwijzen naar mijn account zodat bezoekers makkelijk mijn magazines kunnen bereiken.		4
Als opdrachtgever wil ik dat het gratis account beperkingen heeft om gebruikeren te motiveren een betaald abonnement af te sluiten.		2
Als gebruiker wil ik een maillijst bijhouden zodat ik geinteresseerde een mail kan sturen bij een nieuw magazine.		5

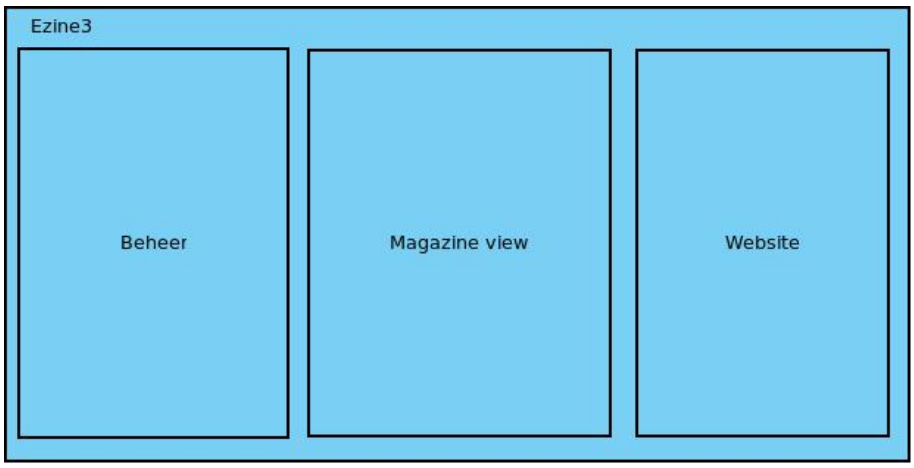
Bijlage D: Design document

Inleiding

Het design document bevat de diagrammen van de software. De afspraak was om alleen het hoognodige te documenteren.

Project onderdelen

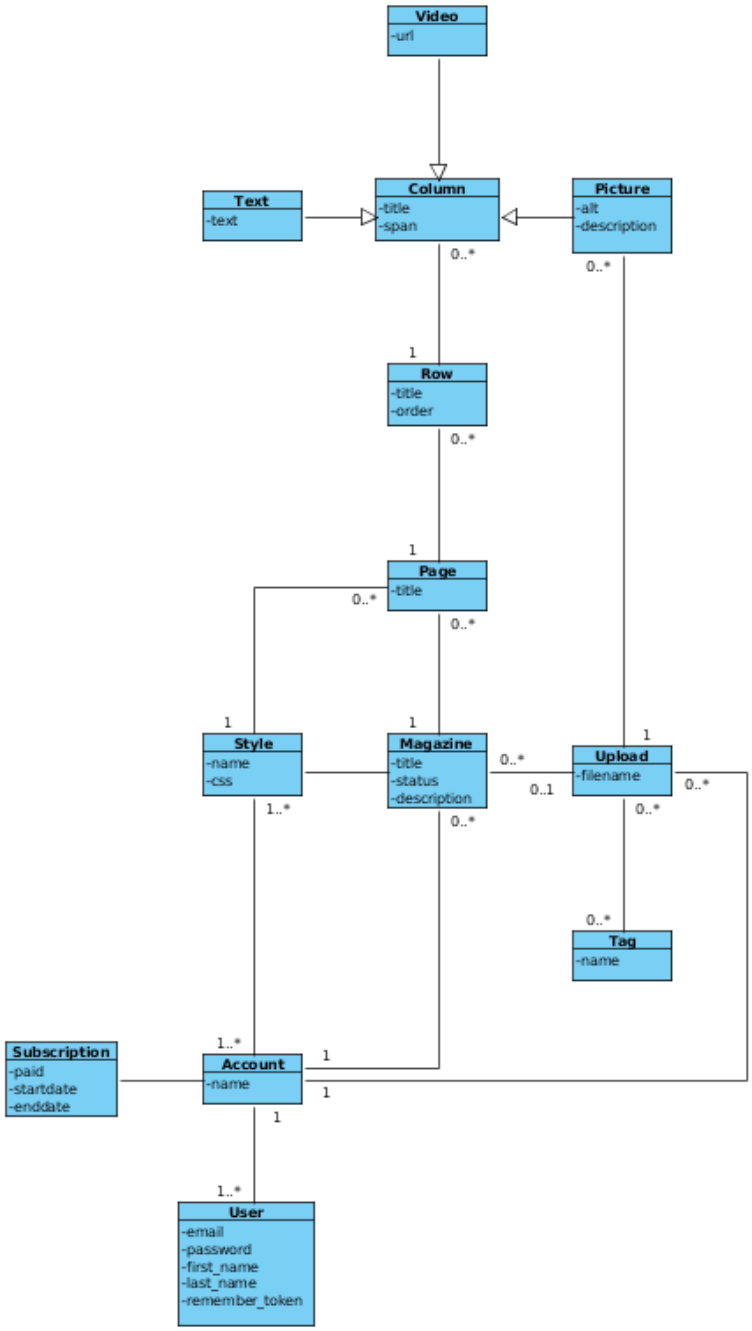
De software bestaat globaal uit drie onderdelen, beheer waar magazines worden opgemaakt en beheert, de magazine view waar lezers de magazine kunnen openen en lezen via een url, en de website die het product moet verkopen. De verdeling van de onderdelen vindt vooral plaats in de controllers en views. Alleen de website bestaat uit statische content en bevat dus alleen views.



Afbeelding 50: De drie onderdelen waar het project uit bestaat.

Klassendiagram

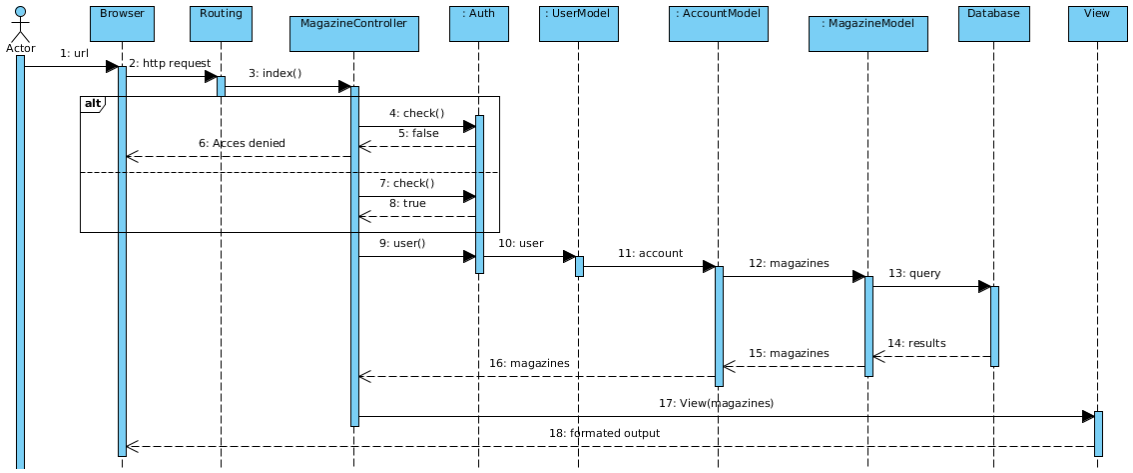
Het klassendiagram is opgebouwd in de models van het MVC maar is dankzij het Eloquent ORM ook een weerspiegeling van de database. Diagrammen van de database zijn daarom achterwege gelaten.



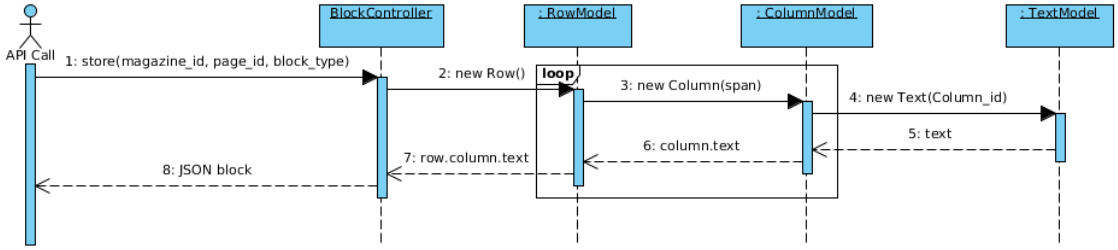
Afbeelding 51: Klassendiagram Ezine3.

Sequence diagrams

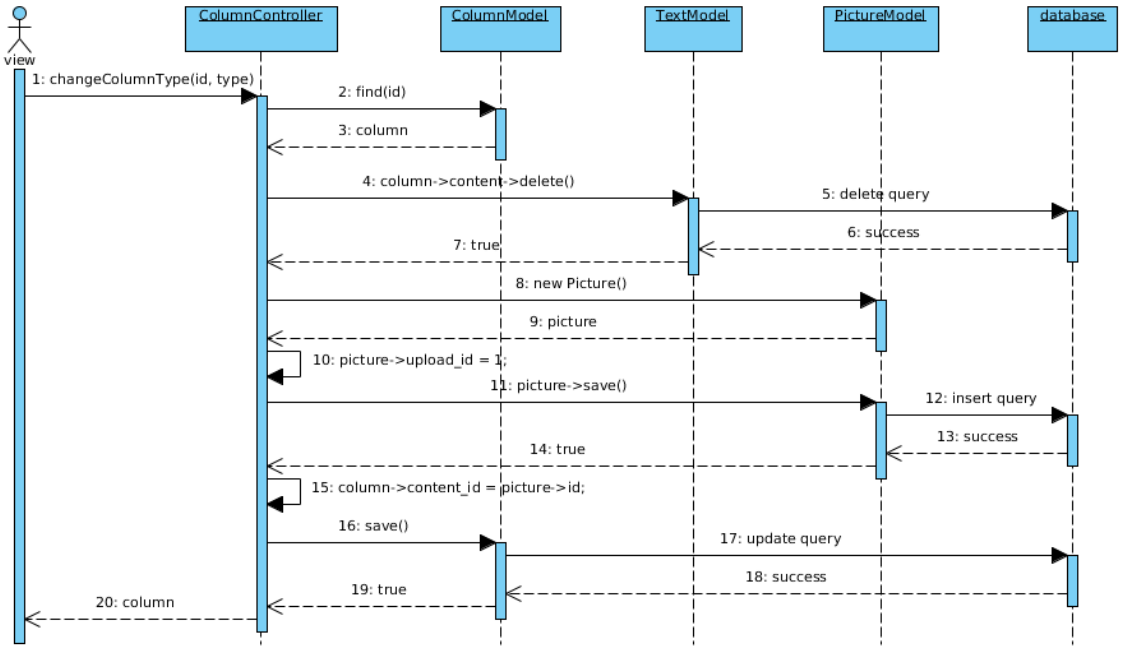
De sequence diagrams geven de interacties weer bij een bepaalde handeling. Omdat de resource controllers meestal de standaard CRUD acties hebben en daarom consequent werken is het voor de meeste ontwikkelaars wel duidelijk hoe dat werkt. De meest interessante onderdelen was de opbouw van een blok en het wijzigen van een column, hier zijn daarom sequence diagrams van gemaakt om de werking te beschrijven.



Afbeelding 52: Magazine overzicht aanvragen



Afbeelding 53: Opbouw van een blok in Laravel



Afbeelding 54: Het aanpassen van een tekst column naar een afbeelding column

Bijlage E: Test document

Inleiding

In dit document zijn de testresultaten bijgehouden. De tests zijn verdeeld per sprint en gekoppeld aan de user stories. Per user story is te zien welke tests er zijn gebruikt en of het resultaat naar verwachting is.

Sprint 1: testresultaten

User story	**Test type**	**Resultaat**
Als gebruiker wil ik een nieuw magazine kunnen aanmaken.	Unit test	Geslaagd
Als gebruiker wil een magazine beschikbaar kunnen stellen voor mijn publiek.	Unit test	Geslaagd
Als gebruiker wil ik een overzicht kunnen opvragen van mijn magazines zodat ik mijn magazines kan beheren.	Unit test	Geslaagd
Als gebruiker wil ik pagina's toe kunnen voegen aan mijn magazine.	Unit test	Geslaagd
Als gebruiker wil ik pagina's toe kunnen verwijderen.	Unit test	Geslaagd
Als gebruiker wil ik in kunnen loggen op mijn account om mijn magazines te kunnen beheren.	Unit test	Geslaagd
Als gebruiker wil ik een magazine kunnen opslaan als schets zodat ik kan overleggen voordat het magazine word gepubliseerd.	Unit test	Geslaagd
Als gebruiker wil ik voor een magazine een begin en einddatum kunnen instellen zodat de magazine alleen in een bepaalde periode online is te bekijken.	Unit test	Geslaagd

Sprint 2: testresultaten

In sprint 2 zijn voor een deel de resultaten uit de spikes verwerkt. Er word hierbij een eerste opzet gemaakt van de magazine opmaakt editor.

Sprint backlog

User story	**Test type**	**Resultaat**
Als gebruiker wil ik blokken kunnen toevoegen aan een pagina.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik de structuur kunnen bepalen van een blok.	acceptatietest	Geslaagd
Als gebruiker wil ik tekst kunnen plaatsen in een column.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik afbeeldingen kunnen plaatsen in een column.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik video kunnen plaatsen in een column.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik een column leeg kunnen maken zodat ik extra structuur krijg in een blok.	acceptatietest	Geslaagd
Als gebruiker wil ik een titel kunnen geven aan een column.	Unittest, acceptatietest	Geslaagd
Als opdrachtgever wil ik dat de ontworpen huisstijl wordt toegepast.	acceptatietest	Geslaagd
Als opdrachtgever wil ik dat het opmaken van magazines gebruiksvriendelijk is.	acceptatietest	Geslaagd
Als gebruiker wil ik een wachtwoord reset kunnen doen zodat ik zelf mijn wachtwoord kan wijzigen als ik de oude ben vergeten.	acceptatietest	Geslaagd

Sprint 3: testresultaten

In sprint 3 wordt de functionaliteit van de magazine editor verder uitgebreid.

Sprint backlog

User story	**Test type**	**Resultaat**
Als gebruiker wil ik een tekst kunnen wijzigen.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik een video kunnen wijzigen.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik een afbeelding kunnen wijzigen.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik een blok kunnen verwijderen.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik een ondertitel kunnen geven aan een afbeelding.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik een alt tekst kunnen geven aan een afbeelding	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik dat er een standaard afbeelding wordt ingesteld als ik zelf geen afbeelding kies.	acceptatietest	Geslaagd
Als gebruiker wil ik de pagina titel kunnen wijzigen.	acceptatietest	Geslaagd
Als gebruiker wil ik de magazine titel kunnen wijzigen.	Unittest, acceptatietest	Geslaagd
Als gebruiker wil ik de volgorde van de blokken kunnen bepalen.	acceptatietest	Geslaagd
Als gebruiker vind ik het niet meer nodig om een titel aan een blok te geven omdat de columns al titels hebben.	acceptatietest	Geslaagd

Sprint 4: testresultaten

In deze sprint is de magazine view gemaakt waarin de magazine die wordt gemaakt in de magazine editor beschikbaar is voor het publiek.

Sprint backlog

User story	Test type	Resultaat
Als gebruiker wil ik een magazine aan mijn lezers doorgeven met een url.	Stresstest	Geslaagd
Als rijksoverheid wil ik dat de lezer door de bladzijdes kan swipen op een touchscreen.	Acceptatietest	Geslaagd
Als rijksoverheid wil ik dat magazines responsive op basis van een 12 punts grid worden zodat de pagina leesbaar is op ieder formaat scherm.	Acceptatietest	Geslaagd
Als gebruiker wil ik een overzicht van de pagina's kunnen zien als ik een magazine lees.	Unittest	Geslaagd
Als gebruiker wil ik met een links en rechts knop tussen de pagina's kunnen bladeren.	Acceptatietest	Geslaagd
Als gebruiker wil ik dat de volgende pagina voor een klein deel zichtbaar is zodat het duidelijk is dat ik naar de volgende pagina kan navigeren.	Acceptatietest	Geslaagd
Als gebruiker wil ik een column titel links, rechts of gecentreerd uit kunnen lijnen.	Acceptatietest	Geslaagd
Als gebruiker wil ik een aantal voorbeeld pagina's bij de aanmaak van een nieuw magazine.	Acceptatietest	Geslaagd
Als gebruiker wil ik een link kunnen toevoegen aan een tekst.	Acceptatietest	Geslaagd

Sprint 5: testresultaten

User story	**Test type**	**Resultaat**
Als gebruiker wil ik een account kunnen aanmaken.	Unittest	Geslaagd
Als ontwikkelaar wil ik een stijl kunnen toewijzen aan het account van de klant zodat we op maat gemaakte stijlen kunnen maken voor bepaalde klanten.	Acceptatietest	Geslaagd
Als gebruiker wil ik een stijl kunnen toevoegen aan een pagina.	Acceptatietest	Geslaagd
Als gebruiker wil ik afbeeldingen kunnen uploaden.	Acceptatietest	Geslaagd
Als gebruiker wil ik mijn afbeeldingen kunnen beheren in een overzicht.	Unittest	Geslaagd
Als gebruiker wil ik afbeeldingen kunnen toevoegen aan de pagina's.	Unittest	Geslaagd
Als gebruiker wil ik een cover kunnen toevoegen aan een magazine.	Unittest, Acceptatietest	Geslaagd
Als gebruiker wil ik de mogelijkheid om de column type aan te kunnen passen.	Unittest, Acceptatietest	Geslaagd
Als opdrachtgever wil ik de titel van de pagina niet in magazine zelf.	Acceptatietest	Geslaagd
Als gebruiker wil ik in redactor de volgende opties hebben: bold, italic, link, underlined, unordered list, ordered list, tabel, html, dropdown voor h2 en h3 koppen.	Acceptatietest	Geslaagd
Als gebruiker wil ik bij meer dan 10 magazines pagination zien zodat er overzicht blijft.	Acceptatietest	Geslaagd
Als gebruiker wil ik bij het toevoegen van een blok dat er automatisch naar toe wordt gescrolt zodat het duidelijk is welk blok er nieuw bij is gekomen.	Acceptatietest	Geslaagd
Als gebruiker wil ik afbeeldingen kunnen verwijderen in afbeelding bibliotheek.	Acceptatietest	Geslaagd
Als gebruiker wil ik kunnen klikken op de afbeelding bij het selecteren van een cover afbeelding (niet alleen op radiobutton).	Unittest, Acceptatietest	Geslaagd
Als gebruiker wil ik dat de interface gebruiksvriendelijk is.	Acceptatietest	Geslaagd

Bijlage G: Formulier tussentijds assessment

Formulier tussentijds assessment

Student:	Tim Wassenburg	**Studentnummer:**	20062006
Datum:	9 september 2014	**eerste / tweede TTA:**	eerste

	Tijdens het tussentijds assessment is het volgende geconstateerd:	**ja**	**nee**
a	*Het voortgangsverslag is ontvangen*	X	
b	*Het afstudeerdossier is digitaal beschikbaar*	X	
c	*Het afstudeerdossier is opgebouwd conform de richtlijnen*	X	
d	*Het goedgekeurde afstudeerplan is aanwezig*	X	
e	*Het plan van aanpak is aanwezig*	X	
f	*Reeds geleverd commentaar is aanwezig*	X	
g	*Het afstudeerdossier geeft voldoende inzicht in de stand van zaken*	X	
h	*De afstudeeropdracht is tot nu toe naar behoren uitgevoerd*	X	

Aanpak	**O**	**T**	**V**	**G**
Passend			X	
Theoretisch verantwoord		X		
Samenhang uitvoering beroepstaken			X	

	Beroepstaken op afgesproken niveau uitgevoerd?	**O**	**T**	**V**	**G**
1	*1.4 Uitvoeren analyse door definitie van requirements, niveau 3*			X	
2	*3.2 Ontwerpen systeemdeel, niveau 4*		X		
3	*3.3 Bouwen applicatie, niveau 3*		X		
4	*3.5 Uitvoeren van en rapporteren over het testproces, niveau 3*	X			

Producten	**O**	**T**	**V**	**G**
Tussenproducten		X		
Eindproducten		X		

Effectief communiceren	**O**	**T**	**V**	**G**
Binnen afstudeerbedrijf			X	
Afstudeerdossier			X	

Reflectie	**O**	**T**	**V**	**G**
Inzicht in eigen functioneren			X	
Inzicht in eigen leerproces			X	

Toelichting per beoordelingscriterium

Aanpak
Er is gekozen voor een ontwikkelmethode die gebruikelijk is binnen het bedrijf. Die wordt in het verslag Scrum genoemd, maar wijkt op veel punten daarvan af. Met de aanpak zelf is binnen de gegeven omstandigheden niet veel mis, maar de theoretische onderbouwing hapert als dit Scrum wordt genoemd.

Beroepstaken op afgesproken niveau uitgevoerd?
Het ontwerp is niet slecht, maar de omvang is nogal gering. Het testen wordt zo goed als niet beschreven, al is in het TTA-gesprek wel benoemd dat er met unit-testing is gewerkt.

Producten
Voor producten wordt twijfel als beoordeling gekozen, omdat het verslag op dit moment onvoldoende handvatten heeft om daar duidelijkheid over te krijgen.

Effectief communiceren
Het verslag is goed leesbaar maar bevat wel veel taalfouten. Vooral werkwoordfouten komen veel te vaak voor.

Reflectie
De student concludeert in hoofdstuk 11 dat hij met de kennis van nu een andere backend zou kiezen. Deze een soortgelijke afwegingen in het verslag maken duidelijk dat zijn reflectie voldoende is.

Advies

X	**Inleveren** (bindend advies)
	Verlengen (vrijblijvend advies)
	Stoppen (vrijblijvend advies)

Besluit student
Aankruisen welke beslissing de student heeft genomen (alleen na vrijblijvend advies)

	Afstudeerdossier wordt op afgesproken datum ingeleverd Inleverdatum:
	Afstudeerperiode wordt verlengd Inleverdatum:
	Student stopt met afstudeeropdracht

Naam begeleidend examinator: J.D. Maas

Naam tweede examinator: G.M. Tuk

Datum: 9 september 2014

Dit formulier wordt door de tweede examinator digitaal ingevuld, waarna de begeleidend examinator het per email verstuurt naar de student met een cc naar de cordinator van ICT en Media @ Work (A.M.Schipper@hhs.nl). Het formulier dient door de student te worden opgenomen in het afstudeerdossier.