Het bouwen van een online checklistsysteem

Afstudeerverslag

Afstudeerder:	Leander Dirkse (10074627)
Onderwijsinstelling:	De Haagse Hogeschool
Opleiding:	Informatica
Periode:	11-02-2013 t/m 07-06-2013
Eerste begeleider:	Dhr. E.M. van Doorn
Tweede begeleider:	Mevr. H.G.J. Bechet-Tjoonk

Versiebeheer

Versie	Datum	Omschrijving
0.1	12-02-2013	Initieel document
0.1	22-02-2013	Indeling gemaakt, begin gemaakt met hoofdstuk 3
0.2	19-03-2013	Aanpassing opmaak
0.3	19-04-2013	Plan van aanpak toegevoegd
0.4	03-05-2013	Aanpassingen aan de structuur
0.5	21-05-2013	Verbeteringen n.a.v. assessment en aanvullingen
1.0	04-06-2013	Eerste versie van het document is afgerond

Referaat

In dit verslag is het proces van mijn afstudeeropdracht "Ontwikkelen online checklistsysteem bij Wodan Brothers" bij Wodan Brothers te Den Haag beschreven.

Afstudeerder: Leander Dirkse
Onderwijsinstelling: De Haagse Hogeschool
Opleiding: Informatica
Periode: 11-02-2013 t/m 07-06-2013

Eerste begeleider: Dhr. E.M. van Doorn
Tweede begeleider: Mevr. H.G.J. Bechet-Tjoonk

Bedrijf: Wodan Brothers
Bedrijfsmentor: Thomas van den Ende
Begeleider: Joshua Angnoe
Opdrachtgever: Thomas van den Ende

Descriptoren:
- SCRUM
- TDD
- Usability
- Security
- PHP
- MySQL
- JavaScript / jQuery
- Design Patterns

Voorwoord

Voor u ligt het verslag van de zeventien weken van mijn afstuderen bij Wodan Brothers.

Wodan Brothers houdt zich bezig met een website voor het inplannen van personeel. Naast deze software leveren zij ook prikklokken die werken op basis van vingerafdrukken. Deze prikklokken kunnen samenwerken met de website waarop klanten een eigen afgeschermd systeem hebben.

U vraagt zich misschien af waarom ik gekozen heb om af te studeren bij dit bedrijf. De belangrijkste reden voor mijn keuze is de opdracht, namelijk het maken van een online checklistsysteem. Toen ik de opdracht tegenkwam leek het mij interessant en uitdagend. Vooral de koppeling met andere systemen en het valideren van bestanden vond ik interessant. Dit soort uitdagingen heeft ervoor gezorgd dat ik contact heb opgenomen met Wodan Brothers. Een andere reden dat ik voor hen heb gekozen is dat het een jong bedrijf is. In de toekomst zou ik graag een eigen bedrijf willen oprichten en op deze manier kan ik ook nog veel leren over het hebben van een eigen bedrijf in opbouw.

Graag wil ik iedereen van Wodan Brothers bedanken voor de tijd die ik bij hen heb mogen werken en in het bijzonder Thomas van den Ende en Joshua Angnoe voor de begeleiding die ik heb gehad.

Daarnaast wil ik graag de heer Van Doorn bedanken voor de goede tips voor aanvang van mijn afstuderen en de begeleiding ervan. Mevrouw Bechet-Tjoonk wil ik graag bedanken voor de feedback die ik heb gekregen.

Voor nu hoop ik dat u net zoveel plezier beleeft aan het lezen van dit verslag als ik heb gehad met het maken ervan.

Den Haag, 7 mei

Leander Dirkse

Inhoudsopgave

VERSIEBEHEER III

REFERAAT IV

VOORWOORD V

1. INLEIDING 1

2. DE ORGANISATIE 2

2.1 Organisatiestructuur 2

2.2 Probleemstelling 3

3. DE OPDRACHT 5

3.1 De opdrachtomschrijving 5

3.2 Doelstelling van de opdracht 5

4. PLAN VAN AANPAK 7

5. SITUATIE BIJ AANVANG 13

6. USABILITY 15

6.1 Wat is usability 15

6.2 Waarom ik usability belangrijk vind 15

7. VOOR DE EERSTE SPRINT 16

7.1 De voorbereidingen 16

7.2 Design Patterns 18

8. DE DATABASE 21

9. DE EERSTE SPRINT: 'VAN PAPIER NAAR DIGITAAL' 29

9.1 Een digitale checklist 29

9.2 Front-end validatie 30

9.3 Beveiliging 34

9.4 Feedback 36

10. DE TWEEDE SPRINT: 'MEER OVERZICHT' 38

10.1 Van whiteboard naar website 38

10.2 Een overzicht met tijdlijn 40

10.3 Commentaar 41

10.4 Meerdere vestigingen per opdracht 43

10.5 Rex-O-Matic 45

10.6 Feedback 46

11. De derde sprint: 'de laatste fase' 48

11.1 Excel-file validator 48
11.2 Overzicht van de configuratie 50
11.3 Koppeling Dyflexis 52
11.4 Instellingen 53
11.5 Feedback 53

12. Evaluatie gebruikte aanpak 55

12.1 SCRUM 55
12.2 Test Driven Development 58
12.3 Project of verslag 58

13. Evaluatie 60

13.1 De verwachtingen 60
13.2 Het product 60
13.3 De doelstellingen 61
13.4 De praktijktest 63
13.5 De beroepstaken 64

Geraadpleegde literatuur 66

Afkortingenlijst 67

1. Inleiding

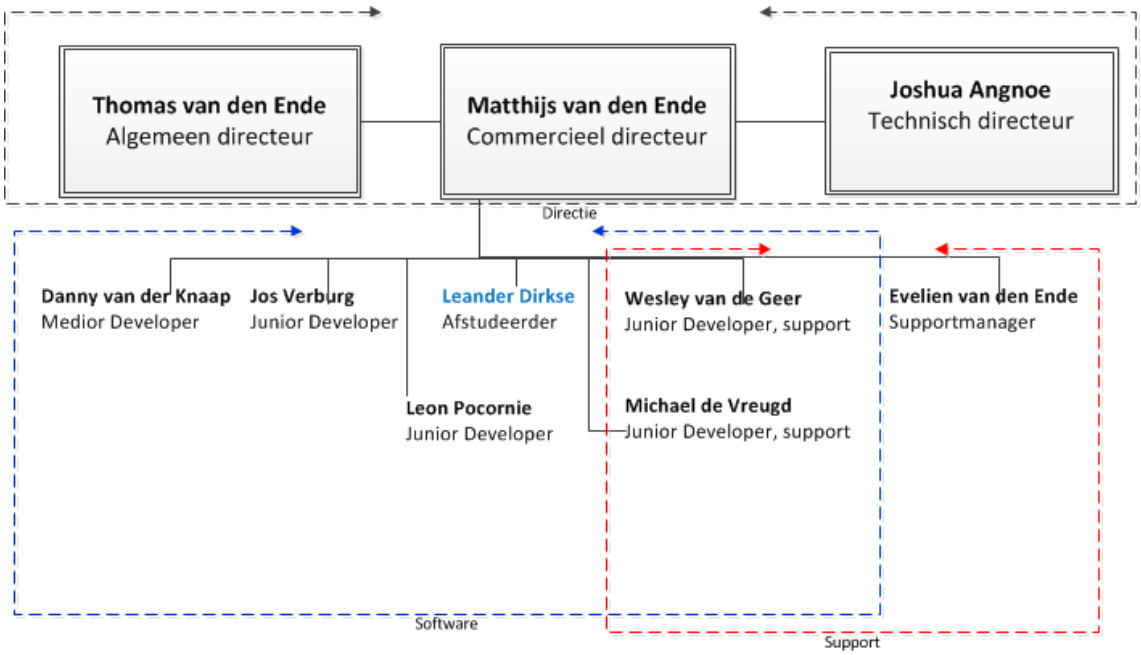
In dit verslag bespreek ik het proces van mijn afstuderen. Voordat u kunt lezen over mijn werkzaamheden bespreek ik het bedrijf Wodan Brothers en het probleem waar zij mee zaten. Vervolgens bespreek ik mijn afstudeeropdracht en de doelstellingen ervan. Om aan de opdracht te kunnen beginnen had ik een plan van aanpak nodig en dat wordt besproken in hoofdstuk 4. De hoofdstukken daarna gaan over de situatie zoals die was bij aanvang van mijn afstuderen en over usability en waarom ik dit belangrijk vind. Alle werkzaamheden die ik heb uitgevoerd en de keuzes die ik heb gemaakt bespreek ik in drie hoofdstukken. Eén hoofdstuk per uitgevoerde *sprint*. Tot slot twee hoofdstukken waarin ik terugblik op mijn afstudeerproject en de aanpak daarvan.

2. De organisatie

Wodan Brothers is een bedrijf opgericht door Thomas van den Ende, Matthijs van den Ende en Joshua Angnoe. Het bedrijf ontwikkelt het planningssysteem Dyflexis voor het inplannen van personeel. Zo wordt het personeel van alle horeca bedrijven op de Grote Markt in Den Haag ingeroosterd met software van Wodan Brothers. Het systeem wordt zo gebruiksvriendelijk mogelijk gemaakt, zodat bedrijven eenvoudig en efficiënt personeel kunnen inplannen. Van Dyflexis is er ook een variant voor de uitzendbranche. Dit product heet FlexMaster.

Bij Wodan Brothers werken momenteel negen personen – mezelf niet meegerekend – aan de ontwikkeling en de verkoop van de producten, maar ook wordt ondersteuning geboden via de helpdesk. Er wordt niet door iedereen aan producten voor klanten gewerkt. Er wordt ook gewerkt aan producten die alleen intern worden gebruikt om bepaalde processen te vereenvoudigen. Hierbij moet u denken aan het maken van offertes of het verwerken van betalingen.

2.1 Organisatiestructuur



Figuur 1: Het organigram van Wodan Brothers

2.2 Probleemstelling

Verschillende klanten maken gebruik van het planningssysteem Dyflexis van Wodan Brothers. Iedere klant heeft zijn eigen systeem met eigen instellingen. Deze instellingen zijn bijvoorbeeld of de klant de medewerkers op voor- of achternaam gesorteerd wil hebben, wat de medewerkers wel en niet mogen zien binnen het systeem of voor welke locatie het weer bijgehouden moeten worden – voor de horeca kan het weer voor de omzet erg belangrijk zijn. De gewenste instellingen worden door resellers ingevuld op papieren *checklists* die vervolgens bij Wodan Brothers terecht komen.

Medewerkers van Wodan Brothers die de taak hebben om systemen voor klanten op te leveren, zijn veel tijd kwijt aan het overtypen van deze papieren checklists. Deze checklists zijn niet altijd volledig ingevuld door de resellers. Als dat het geval is dan moet de desbetreffende reseller gebeld worden om de lijst verder in te vullen, zodat alle opties van de klant bekend zijn en het systeem klaar gezet kan worden. Het is goed mogelijk dat de reseller de klant moet bellen om zelf de lijst aan te kunnen vullen. Het overtypen van een checklist kost veel tijd, maar het kan ook een tijd duren voordat de lijst compleet is. Daarnaast is het overtypen foutgevoelig. Een medewerker kan eenvoudig fouten maken bij het overzetten van de bestelde configuratie.

Volledig ingevulde checklists worden in een daarvoor bestemde bak bewaard om er pas weer uitgehaald te worden op de dag van de oplevering. Dat is ook het moment dat een medewerker de door de klant gekozen opties gaat overtypen in het planningssysteem. In de tijd tussen het aanleveren van de checklist door de reseller en het opleveren van het systeem kan er van alles gebeuren met een checklist. In het verleden is het voorgekomen dat een checklist is kwijtgeraakt of werd vergeten. In dat laatste geval heeft een klant zijn systeem niet op de dag van de oplevering.

Omdat een klant zijn wensen telefonisch doorgeeft en geen overzicht krijgt van de door hem gekozen opties, heeft de klant geen duidelijkheid over het bestelde systeem. Het komt ook voor dat er buiten de checklist om aanpassingen aan de instellingen gedaan worden, wat ook voor onduidelijkheid kan zorgen. Omdat het om papieren checklists gaat, hebben verschillende resellers verschillende versies van de checklist in bezit.

Bij het opleveren van een besteld systeem is het mogelijk dat klanten een Excel-

file aanleveren met gegevens van hun medewerkers. Mits mogelijk worden de medewerkers in het systeem geïmporteerd. Echter, het komt vaak voor dat de aangeleverde checklists niet correct zijn ingevuld en daardoor niet geïmporteerd kunnen worden.
De meest voorkomende problemen van de oude situatie zijn:

- Checklists worden niet altijd volledig ingevuld
- Checklists kunnen kwijtraken tussen aanlevering en oplevering
- Er worden aanpassingen buiten de checklist om gedaan
- Resellers hebben verschillende versies van de checklist
- Het overtypen van checklists kost veel tijd en is foutgevoelig
- Medewerkers van Wodan Brothers moeten opleveringen zelf in de gaten houden
- Medewerkersbestanden moeten met de hand op fouten gecontroleerd worden
- Klanten hebben geen overzicht van de door hen bestelde configuratie

3. De opdracht

3.1 De opdrachtomschrijving

Wodan Brothers zou graag een systeem willen hebben dat het opleveren van systemen eenvoudiger maakt. Daardoor zouden medewerkers minder tijd kwijt moeten zijn aan de opleveringen. Ook voor de resellers van het Dyflexis planningsysteem moet het systeem eenvoudig in gebruik zijn. Het moet mogelijk zijn om checklists in te vullen en daarmee de configuratie van het bestelde systeem vast te leggen. Daarbij zal het systeem aangeven wat de gebruiker moet doen of als er fouten gemaakt zijn. In het te bouwen systeem zullen ook verschillende overzichten beschikbaar moeten zijn die erbij helpen dat systemen altijd op tijd opgeleverd worden en altijd duidelijk is wat er met een opdracht moet gebeuren. Alle informatie die nodig is om een systeem op te kunnen leveren zal beschikbaar moeten zijn. Bestanden met gegevens van medewerkers zullen niet meer via e-mail verstuurd worden, maar geüpload worden in het online checklistsysteem. Het systeem zal deze bestanden controleren en geeft aan als er fouten gemaakt zijn.

3.2 Doelstelling van de opdracht

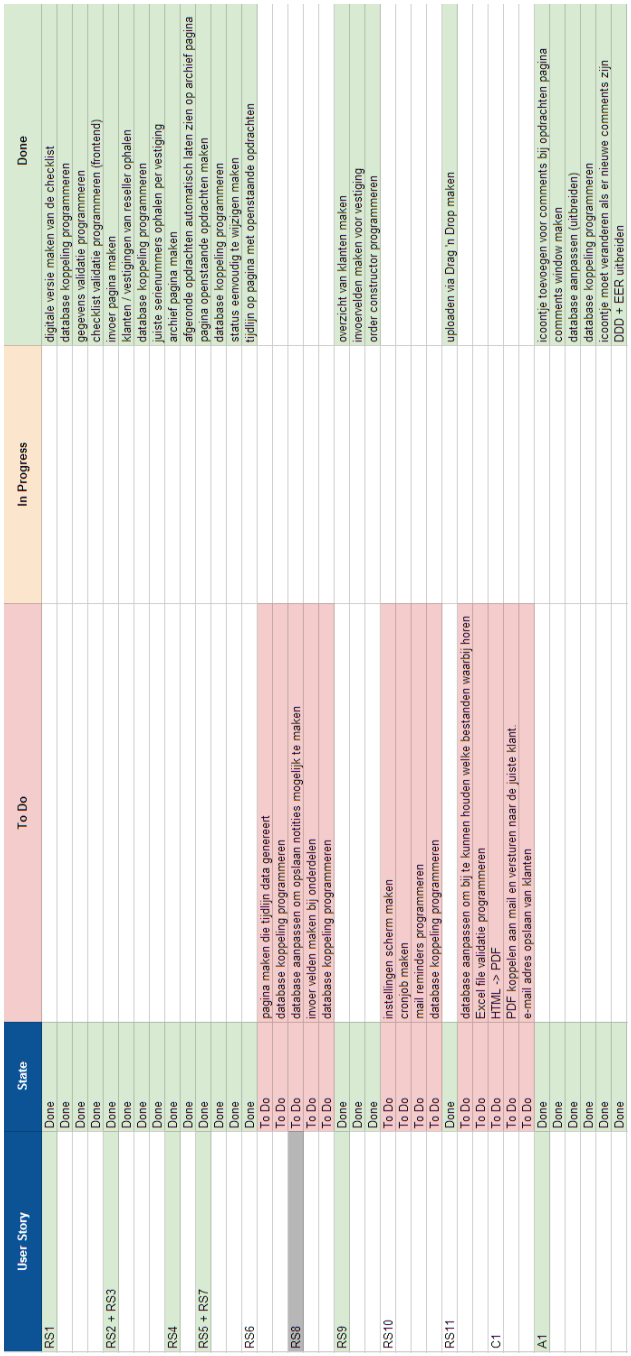
Het online checklist systeem zal ervoor zorgen dat resellers de checklists eenvoudiger kunnen invullen en zelf kunnen zien als zij een fout hebben gemaakt. Dit zorgt ervoor dat medewerkers van Wodan Brothers minder tijd kwijt zijn aan het controleren van de checklists. Daarnaast zal het systeem de gekozen opties over kunnen zetten naar het planningsysteem Dyflexis, zodat dit niet meer met de hand overgetypt hoeft te worden. Ook zal het systeem de gebruikers waarschuwen als er een oplevering nadert en de gebruiker actie dient te ondernemen.
Bij het online checklistsysteem zal het niet meer nodig zijn de Excel-files met de gegevens van medewerkers handmatig te controleren. Zodra een bestand wordt geüpload zal het door het systeem gecontroleerd worden op eventuele fouten. Een reseller kan zien waarom het bestand niet door de validatie is gekomen en kan de klant vragen deze fouten op te lossen. Het zal dus eenvoudiger zijn om medewerkers in het systeem te importeren.

Als de checklist volledig ingevuld is ontvangt de klant een PDF-bestand met alle gekozen instellingen. Zo is er voor de klant meer duidelijkheid over het bestelde systeem.
Een gebruiker krijgt met het online checklistsysteem een beter overzicht van alle openstaande opdrachten en wat er per opdracht nog gedaan dient te worden.
De doelstellingen van de opdracht zijn:

- Het invullen van de checklists moet eenvoudiger zijn voor de resellers
- Het mag niet meer voorkomen dat verplichte onderdelen van de checklist leeg blijven
- Aanpassingen aan de configuratie zullen altijd via de checklist moeten gebeuren
- Het systeem moet het maken van fouten zoveel mogelijk voorkomen
- Het systeem moet de bestelde configuratie over kunnen zetten naar het Dyflexis planningsysteem
- Het systeem zal de gebruiker waarschuwen als er een oplevering aankomt en de gebruiker actie dient te ondernemen
- Het systeem moet Excel-files met gegevens van medewerkers kunnen controleren
- De klant dient een overzicht te ontvangen van de configuratie van het bestelde systeem

4. Plan van aanpak

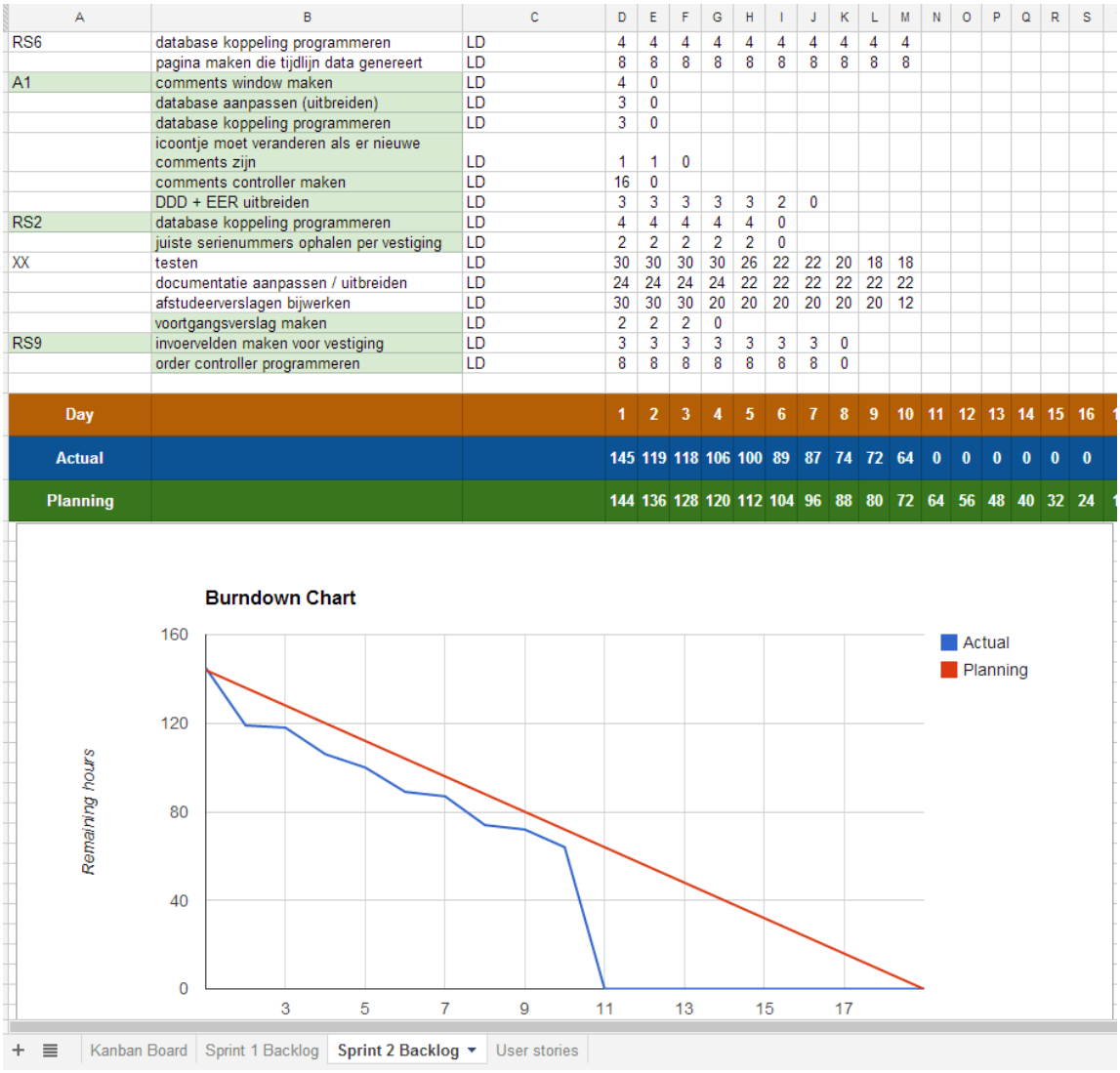
Om het online checklistsysteem te realiseren heb ik gebruik gemaakt van een aangepaste versie van *SCRUM*. Dit heb ik gedaan omdat ik de enige ben die aan het project heb gewerkt en het dus niet nuttig was om iedere ochtend een stand-up meeting te houden. Voor dit project heb ik wel een *product backlog* gemaakt. Dit product backlog bevat een *Kanban Board*, *user stories* en *Sprint Backlogs* met *Burndown Charts*.



User Story	State	To Do	In Progress	Done
RS1	Done			digitale versie maken van de checklist
	Done			database koppeling programmeren
	Done			gegevens validatie programmeren
	Done			checklist validatie programmeren (frontend)
RS2 + RS3	Done			invoer pagina maken
	Done			klanten / vestigingen van reseller ophalen
	Done			database koppeling programmeren
	Done			juiste serienummers ophalen per vestiging
RS4	Done			archief pagina maken
	Done			afgeronde opdrachten automatisch laten zien op archief pagina
RS5 + RS7	Done			pagina openstaande opdrachten maken
	Done			database koppeling programmeren
	Done			status eenvoudig te wijzigen maken
RS6	Done			tijdlijn op pagina met openstaande opdrachten
	To Do	pagina maken die tijdlijn data genereert		
	To Do	database koppeling programmeren		
RS8	To Do	database aanpassen om opslaan notities mogelijk te maken		
	To Do	invoer velden maken bij onderdelen		
	To Do	database koppeling programmeren		
RS9	Done			overzicht van klanten maken
	Done			invoervelden maken voor vestiging
	Done			order constructor programmeren
RS10	To Do	instellingen scherm maken		
	To Do	cronjob maken		
	To Do	mail reminders programmeren		
	To Do	database koppeling programmeren		
RS11	Done			uploaden via Drag 'n Drop maken
	To Do	database aanpassen om bij te kunnen houden welke bestanden waarbij horen		
	To Do	Excel file validatie programmeren		
C1	To Do	HTML -> PDF		
	To Do	PDF koppelen aan mail en versturen naar de juiste klant		
	To Do	e-mail adres opslaan van klanten		
A1	Done			icoontje toevoegen voor comments bij opdrachten pagina
	Done			comments window maken
	Done			database aanpassen (uitbreiden)
	Done			database koppeling programmeren
	Done			icoontje moet veranderen als er nieuwe comments zijn
	Done			DDD + EER uitbreiden

Figuur 2: Het Kanban Board

Op het Kanban Board staan alle user stories uitgewerkt in de taken die uitgevoerd dienen te worden om deze wensen van de opdrachtgever te realiseren. De taken staan in het begin in de kolom 'To Do', tijdens het uitvoeren van de taak staan ze in de kolom 'In Progress' en als de taken zijn uitgevoerd staan ze in de kolom 'Done'. Het verschilt per taak wanneer hij de status 'done' krijgt. Als een taak eenvoudig is, dan kan hij klaar zijn als hij geïmplementeerd is. Het testen zal bij eenvoudige taken met de hand gedaan worden. Als een taak ingewikkelder is, dan is hij pas klaar als hij volgens *Test Driven Development (TDD)* is geprogrammeerd en alle testen geslaagd zijn. Voor elke taak geldt dat hij pas klaar kan zijn als eventuele documentatie is gemaakt of bijgewerkt. Verderop in dit hoofdstuk kunt u meer lezen over TDD.



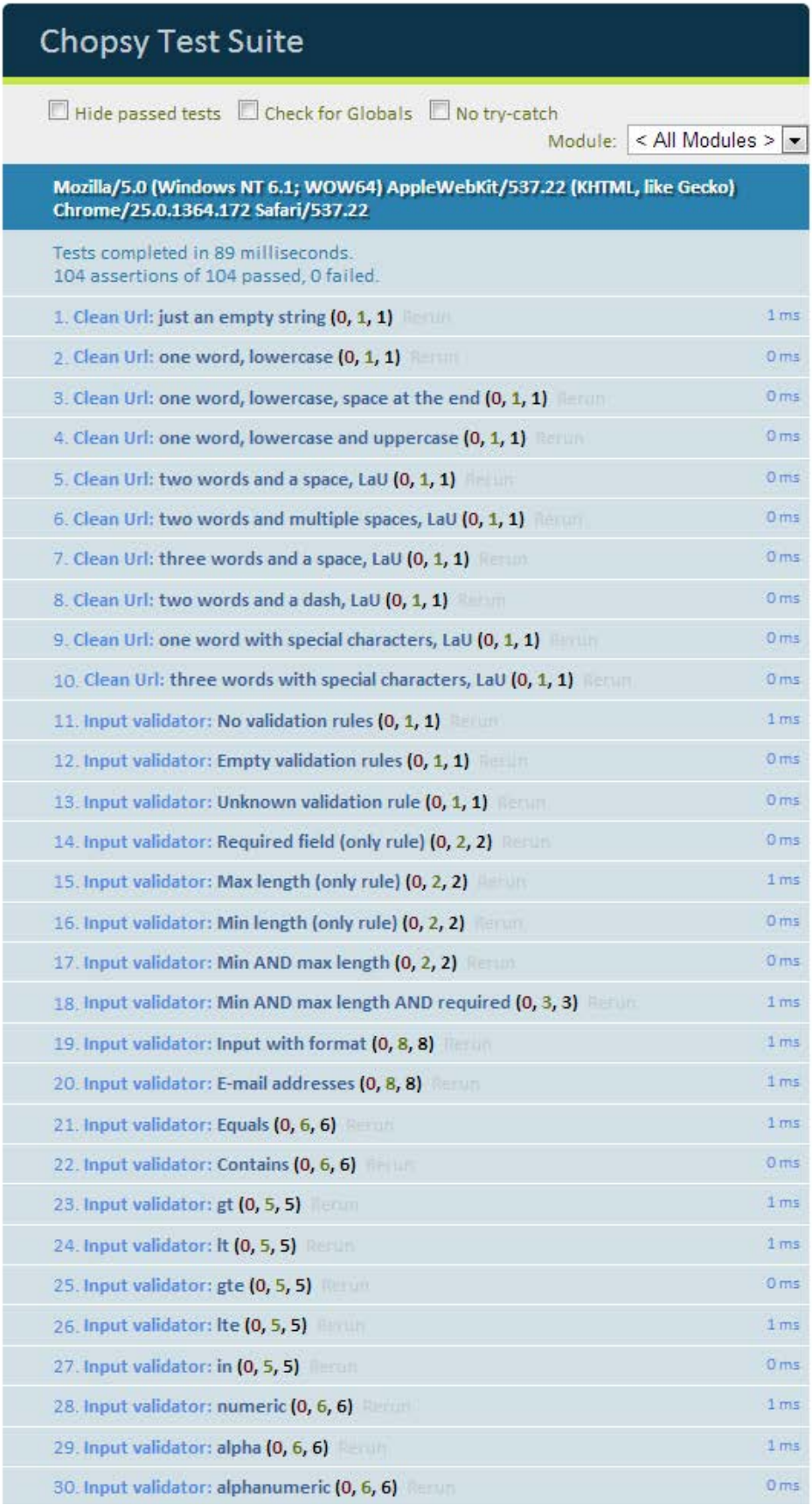
RS6	database koppeling programmeren	LD	4	4	4	4	4	4	4	4	4						
	pagina maken die tijdlijn data genereert	LD	8	8	8	8	8	8	8	8	8						
A1	comments window maken	LD	4	0													
	database aanpassen (uitbreiden)	LD	3	0													
	database koppeling programmeren	LD	3	0													
	icoontje moet veranderen als er nieuwe comments zijn	LD	1	1	0												
	comments controller maken	LD	16	0													
	DDD + EER uitbreiden	LD	3	3	3	3	3	2	0								
RS2	database koppeling programmeren	LD	4	4	4	4	4	0									
	juiste serienummers ophalen per vestiging	LD	2	2	2	2	2	0									
XX	testen	LD	30	30	30	30	26	22	22	20	18	18					
	documentatie aanpassen / uitbreiden	LD	24	24	24	24	22	22	22	22	22	22					
	afstudeerverslagen bijwerken	LD	30	30	30	20	20	20	20	20	20	12					
	voortgangsverslag maken	LD	2	2	2	0											
RS9	invoervelden maken voor vestiging	LD	3	3	3	3	3	3	3	0							
	order controller programmeren	LD	8	8	8	8	8	8	8	0							
Day			**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Actual			**145**	**119**	**118**	**106**	**100**	**89**	**87**	**74**	**72**	**64**	**0**	**0**	**0**	**0**	**0**	**0**
Planning			**144**	**136**	**128**	**120**	**112**	**104**	**96**	**88**	**80**	**72**	**64**	**56**	**48**	**40**	**32**	**24**

Figuur 3: Een Sprint Backlog met Burndown Chart

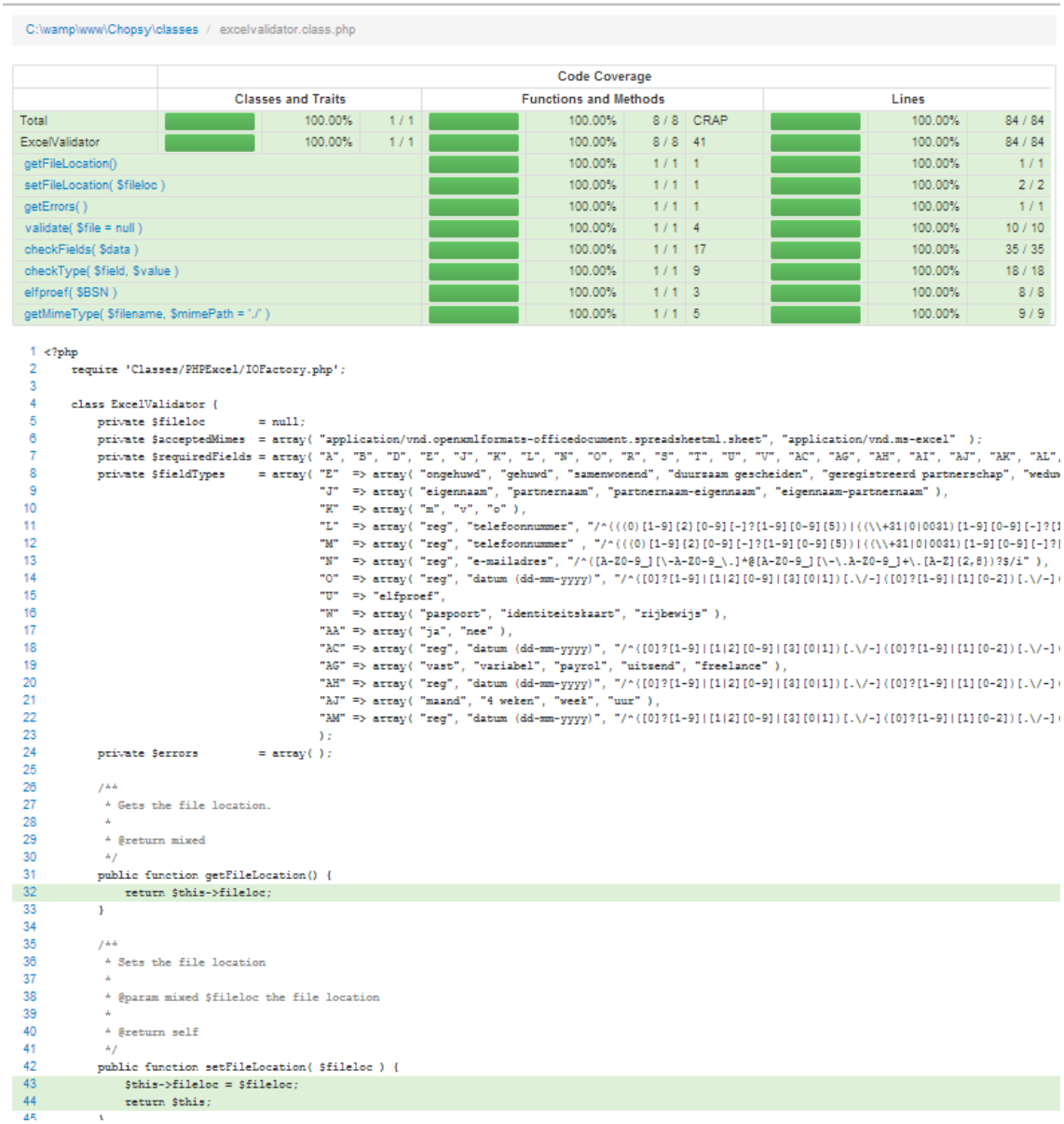
Er is om meerdere redenen gekozen voor SCRUM. Bij SCRUM wordt er minder documentatie gemaakt dan bij traditionele softwareontwikkelingsmethoden als bijvoorbeeld *RUP*. Bij SCRUM wordt alleen ontworpen wat nodig is. Als SCRUM gedaan wordt in een team, dan is het mogelijk om taken tegelijkertijd uit te voeren. Ik ben de enige die aan dit project werkt en kan daardoor niet verschillende taken tegelijkertijd uitvoeren. Door gebruik te maken van SCRUM was het voor mij mogelijk om steeds een klein gedeelte van het systeem te ontwerpen, te bouwen en te testen. Mocht het achteraf toch nodig zijn iets aan te passen, dan kon dit vaak direct gedaan worden of meegenomen worden in een volgende sprint. Het werken op deze manier is veel flexibeler dan de traditionele software ontwikkelingsmethoden. Positieve ervaring met het uitvoeren van verschillende taken op deze manier was voor mij ook een reden om voor SCRUM te kiezen. Daarnaast is er voor het afstuderen beperkte tijd beschikbaar. Bij SCRUM is het toch mogelijk om een opdrachtgever, of *product owner*, tegemoet te komen als hij zich tijdens de bouw van het systeem bedenkt. Dat kan zelfs gedaan worden zonder daarbij de planning in gevaar te brengen. Uiteraard zijn er wel grenzen aan wat er nog aangepast kan worden. Aan het einde van het project zal er altijd een werkend product zijn. Dit komt doordat elke iteratie, bij SCRUM sprint genaamd, een werkend product oplevert.

Bij het maken van het afstudeerplan is ervoor gekozen om SCRUM te combineren met TDD. De reden hiervoor is dat er bij Wodan Brothers volgens TDD ontwikkeld wordt en het mij een prettige manier van ontwikkelen leek.

Het maken van de testen voordat er code wordt geschreven zorgt ervoor dat ik op een andere manier over mijn code nadenk. Ook wordt alleen die code geschreven die nodig is om de test te laten slagen. Mocht de code eerst geschreven worden, dan is de kans groter dat er overbodige code geschreven wordt. Daarnaast is het prettig om op ieder moment de testen uit te kunnen voeren om te kijken of alles nog naar behoren werkt. De TDD manier van ontwikkelen is gebruikt waar mogelijk. Voor het uitvoeren van testen binnen PHP is gebruik gemaakt van *PHPUnit* en voor JavaScript is gebruik gemaakt van *QUnit*.



Figuur 4: Unit tests voor JavaScript met QUnit



C:\wamp\www\Chopsy\classes / excelvalidator.class.php

	Code Coverage								
	Classes and Traits			Functions and Methods				Lines		
Total		100.00%	1 / 1		100.00%	8 / 8	CRAP		100.00%	84 / 84
ExcelValidator		100.00%	1 / 1		100.00%	8 / 8	41		100.00%	84 / 84
getFileLocation()					100.00%	1 / 1	1		100.00%	1 / 1
setFileLocation($fileloc)					100.00%	1 / 1	1		100.00%	2 / 2
getErrors( )					100.00%	1 / 1	1		100.00%	1 / 1
validate($file = null)					100.00%	1 / 1	4		100.00%	10 / 10
checkFields($data)					100.00%	1 / 1	17		100.00%	35 / 35
checkType($field, $value)					100.00%	1 / 1	9		100.00%	18 / 18
elfproef($BSN)					100.00%	1 / 1	3		100.00%	8 / 8
getMimeType($filename, $mimePath = './')					100.00%	1 / 1	5		100.00%	9 / 9

```php
<?php
    require 'Classes/PHPExcel/IOFactory.php';

    class ExcelValidator {
        private $fileloc        = null;
        private $acceptedMimes  = array( "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", "application/vnd.ms-excel"  );
        private $requiredFields = array( "A", "B", "D", "E", "J", "K", "L", "N", "O", "R", "S", "T", "U", "V", "AC", "AG", "AH", "AI", "AJ", "AK", "AL",
        private $fieldTypes     = array( "E"  => array( "ongehuwd", "gehuwd", "samenwonend", "duurzaam gescheiden", "geregistreerd partnerschap", "wedu
                                         "J"  => array( "eigennaam", "partnernaam", "partnernaam-eigennaam", "eigennaam-partnernaam" ),
                                         "K"  => array( "m", "v", "o" ),
                                         "L"  => array( "reg", "telefoonnummer", "/^((((0)[1-9]{2}[0-9][-]?[1-9][0-9]{5})|(((\\+31|0|0031)[1-9][0-9][-]?[1
                                         "M"  => array( "reg", "telefoonnummer" , "/^((((0)[1-9]{2}[0-9][-]?[1-9][0-9]{5})|(((\\+31|0|0031)[1-9][0-9][-]?|
                                         "N"  => array( "reg", "e-mailadres", "/^([A-Z0-9_][\-A-Z0-9_\.]*@[A-Z0-9_][\-\.A-Z0-9_]+\.[A-Z]{2,8})?$/i" ),
                                         "O"  => array( "reg", "datum (dd-mm-yyyy)", "/^([0]?[1-9]|[1|2][0-9]|[3][0|1])[.\/-]([0]?[1-9]|[1][0-2])[.\/-]
                                         "U"  => "elfproef",
                                         "W"  => array( "paspoort", "identiteitskaart", "rijbewijs" ),
                                         "AA" => array( "ja", "nee" ),
                                         "AC" => array( "reg", "datum (dd-mm-yyyy)", "/^([0]?[1-9]|[1|2][0-9]|[3][0|1])[.\/-]([0]?[1-9]|[1][0-2])[.\/-]
                                         "AG" => array( "vast", "variabel", "payrol", "uitzend", "freelance" ),
                                         "AH" => array( "reg", "datum (dd-mm-yyyy)", "/^([0]?[1-9]|[1|2][0-9]|[3][0|1])[.\/-]([0]?[1-9]|[1][0-2])[.\/-]
                                         "AJ" => array( "maand", "4 weken", "week", "uur" ),
                                         "AM" => array( "reg", "datum (dd-mm-yyyy)", "/^([0]?[1-9]|[1|2][0-9]|[3][0|1])[.\/-]([0]?[1-9]|[1][0-2])[.\/-]
                                         );
        private $errors         = array( );

        /**
         * Gets the file location.
         *
         * @return mixed
         */
        public function getFileLocation() {
            return $this->fileloc;
        }

        /**
         * Sets the file location
         *
         * @param mixed $fileloc the file location
         *
         * @return self
         */
        public function setFileLocation( $fileloc ) {
            $this->fileloc = $fileloc;
            return $this;
        }
```

Figuur 5: Unit tests in PHP met Code Coverage verslag

PHPUnit maakt ook gebruik van Code Coverage. Code Coverage is een systeem dat bijhoudt welke code getest is en welke code nog niet. In bovenstaande afbeelding is ook te zien dat 100% van de code die uitgevoerd wordt getest is. Mocht ik om wat voor reden dan ook met mijn tests bepaalde delen van de code niet testen, dan geeft Code Coverage aan dat de code niet getest is en dan zou ik voor die code alsnog een test kunnen schrijven.

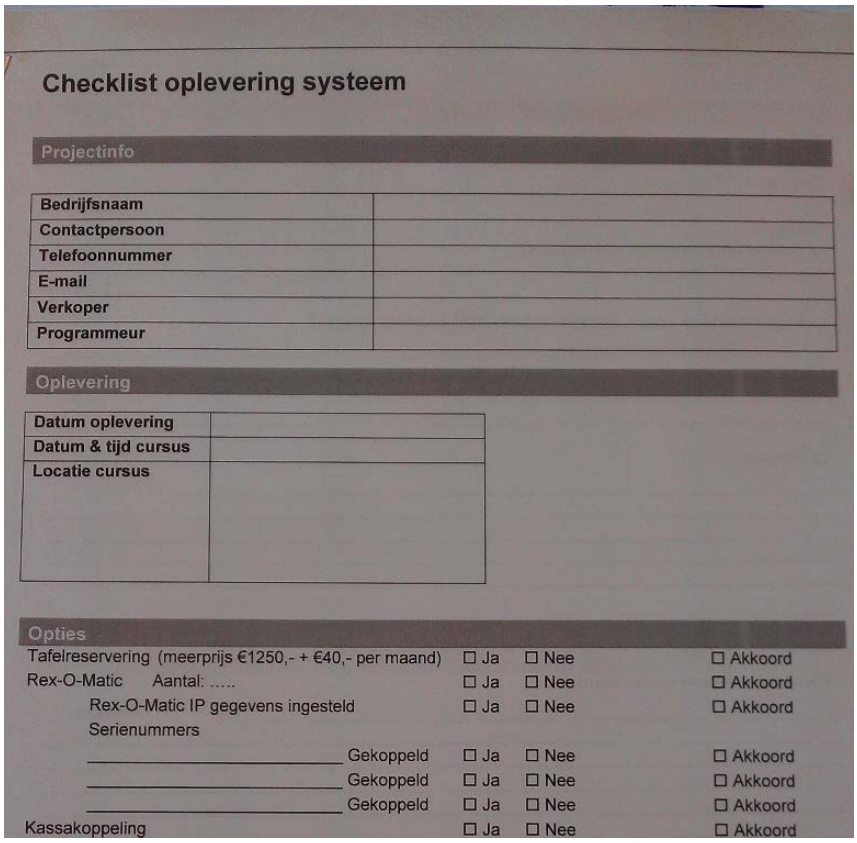
Zoals eerder besproken heb ik gebruik gemaakt van SCRUM. Bij SCRUM worden taken uitgevoerd in verschillende sprints. Ik heb ervoor gekozen om het project in drie sprints te doen. Dit om toch voldoende tijd per sprint te hebben, maar ook

om tijdens het project een paar momenten te hebben om te reflecteren, feedback te krijgen en deze feedback mee te nemen in een volgende sprint. Het opdelen in drie sprint heb ik gedaan omdat dat het project voor mij ook drie fases had. De eerste fase was het opzetten van het systeem. De tweede fase was de bouw van het systeem en de derde fase was het afronden van het systeem en toevoegen van de laatste functionaliteit.

5. Situatie bij aanvang

Bij Wodan Brothers wordt gebruik gemaakt van verschillende systemen. Het belangrijkste systeem hiervan is: 'Dyflexis', het planningsysteem. Het klaarzetten van de door klanten bestelde systemen, met stuk voor stuk een eigen configuratie, werd met de hand gedaan.

Het online checklistsysteem, genaamd 'Chopsy', is een volledig nieuw systeem. De eerder besproken papieren checklists heb ik kunnen gebruiken bij het ontwerpen van het systeem. Bij Wodan Brothers werd gebruik gemaakt van een whiteboard om op te leveren systemen bij te houden. Ook dit whiteboard heb ik kunnen gebruiken als input voor het ontwerp van het te bouwen systeem.
Qua software was er nauwelijks iets beschikbaar. Toen ik begon aan mijn opdracht, was er voor het systeem nog niets gemaakt. Wel heb ik gebruik kunnen maken van een MVC framework dat gemaakt is door de hoofdprogrammeur van Wodan Brothers. Dit framework is gebaseerd op ideeën achter het CakePHP framework. Meer over het framework kunt u lezen in hoofdstuk 7.



Checklist oplevering systeem

Projectinfo

Bedrijfsnaam	
Contactpersoon	
Telefoonnummer	
E-mail	
Verkoper	
Programmeur	

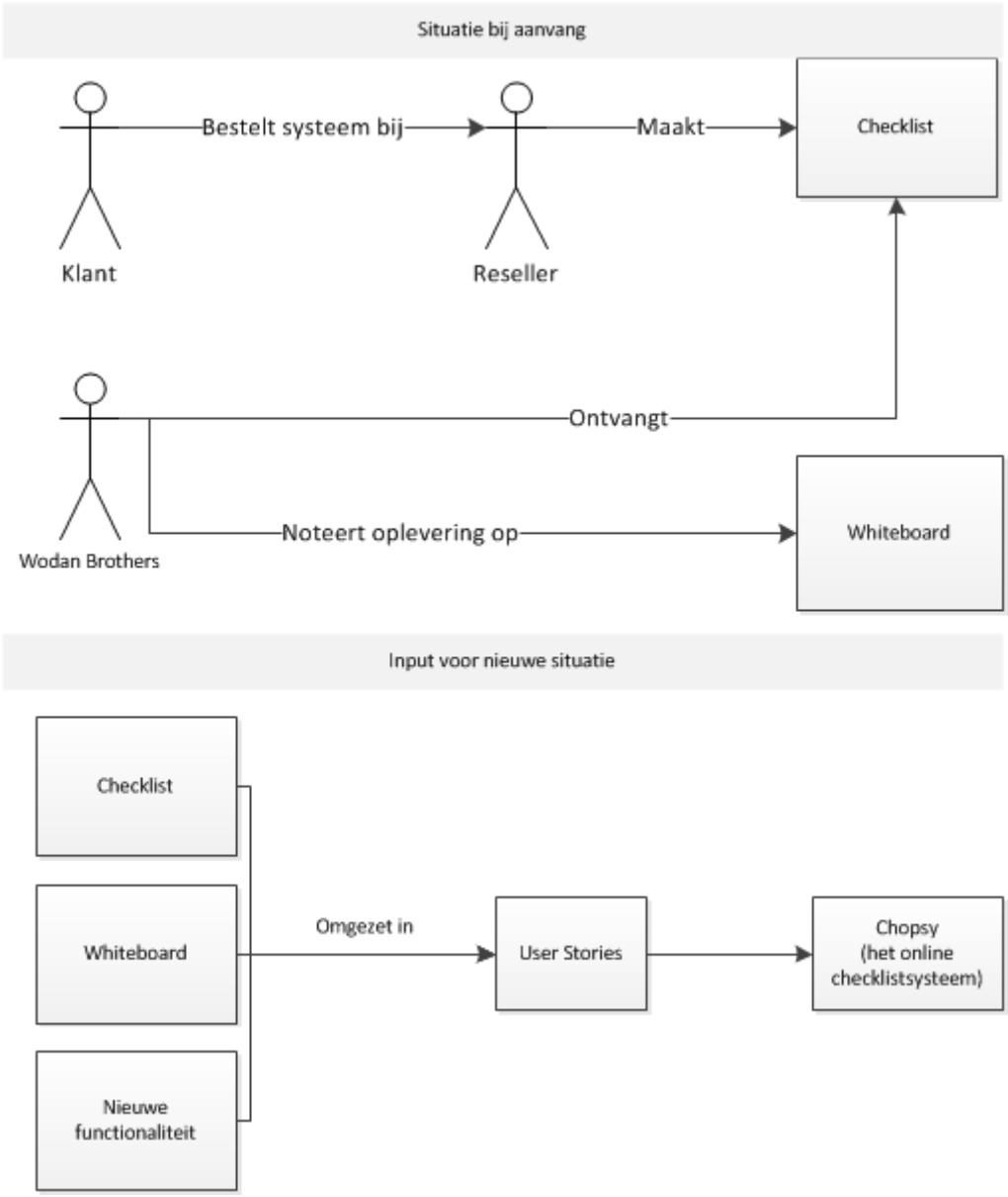
Oplevering

Datum oplevering	
Datum & tijd cursus	
Locatie cursus	

Opties

Tafelreservering (meerprijs €1250,- + €40,- per maand)	☐ Ja	☐ Nee	☐ Akkoord
Rex-O-Matic Aantal:	☐ Ja	☐ Nee	☐ Akkoord
Rex-O-Matic IP gegevens ingesteld	☐ Ja	☐ Nee	☐ Akkoord
Serienummers			
_____ Gekoppeld	☐ Ja	☐ Nee	☐ Akkoord
_____ Gekoppeld	☐ Ja	☐ Nee	☐ Akkoord
_____ Gekoppeld	☐ Ja	☐ Nee	☐ Akkoord
Kassakoppeling	☐ Ja	☐ Nee	☐ Akkoord

Figuur 6: Een van de papieren checklists



Figuur 7: De huidige situatie en de input voor het systeem

In bovenstaand figuur kunt u de situatie bij aanvang zien. Een klant bestelt een Dyflexis systeem bij een reseller (Wodan Brothers valt ook onder de resellers). De reseller vult een papieren checklist in (zie **Figuur 6** voor een lege versie). Deze checklist bevat alle opties die de klant heeft besteld. Dit is de configuratie voor het door de klant bestelde systeem. Als Wodan Brothers niet de verkopende partij is, zal de reseller de checklist naar Wodan Brothers sturen. Bij Wodan Brothers zal de oplevering op een whiteboard geschreven worden. Deze moet door de medewerkers in de gaten gehouden worden, om op tijd de bestelde systemen klaar te zetten.

6. Usability

6.1 Wat is usability

Bij usability gaat het om gebruiksvriendelijkheid. Dit kan gaan om het gebruiksgemak van een product, softwareapplicatie of een website / webapplicatie. In het geval van Chopsy gaat het om een webapplicatie.

6.2 Waarom ik usability belangrijk vind

Elk product, of dat een website is of een fysiek product, moet in mijn ogen gebruiksvriendelijk zijn. Vaak worden sites ontworpen voor het mooie uiterlijk, maar de usability wordt regelmatig vergeten. Als een gebruiker zijn weg niet kan vinden op een website zal hij snel de site verlaten. Bij Chopsy zal dat geen probleem zijn omdat de gebruikers ervan dat voor hun werk doen. Toch is het erg belangrijk dat de gebruikers eenvoudig hun weg kunnen vinden en weten wat zij moeten doen. Een gebruiker moet snel en prettig met het product kunnen werken. Voor een maker van een website is het niet moeilijk om alles te vinden, hij heeft het immers zelf gemaakt. Voor hem kan het dan ook allemaal heel logisch zijn, maar dat hoeft voor gebruikers helemaal niet het geval te zijn. Uit ervaring weet ik dat dit probleem vaak optreedt als backenddevelopers een frontend gaan ontwikkelen. Voor hen is het dan allemaal prima werkbaar, maar een leek heeft dan vaak geen idee wat hij moet doen. Omdat ik hier een aantal keer mee te maken heb gehad ben ik mij veel bezig gaan houden met usability. Een betere gebruiksvriendelijkheid zal ervoor zorgen dat gebruikers minder fouten maken.

Tijdens de ontwikkeling van het online checklistsysteem ‘Chopsy’ zal ik mij ook erg bezighouden met de gebruiksvriendelijkheid van het product. Omdat de gebruiksvriendelijkheid gedurende het hele project belangrijk is, worden de toepassingen van usability niet in een apart hoofdstuk besproken. De usability wordt besproken waar dat van toepassing is.

7. Voor de eerste sprint

Voor ik kon beginnen met de eerste sprint had ik een aantal andere taken te doen. Ik had de wensen van de product owner nodig, om te weten hoe hij het systeem voor zich zag. Daarnaast zou ik een aantal ontwerpen nodig hebben voor de gebruikersinterface en voor de database.

7.1 De voorbereidingen

Om de wensen van de product owner te achterhalen heb ik hem gevraagd hoe hij het systeem voor zich zag en wat hij wilde dat het systeem zou kunnen. Zijn wensen zijn als user stories beschreven in het product backlog. Deze user stories staan in het formaat:

> *'Als een <type gebruiker>, wil ik <een doel> zodat ik <een reden>'.*

Om een voorbeeld te geven een user story die gebruikt is tijdens dit project:

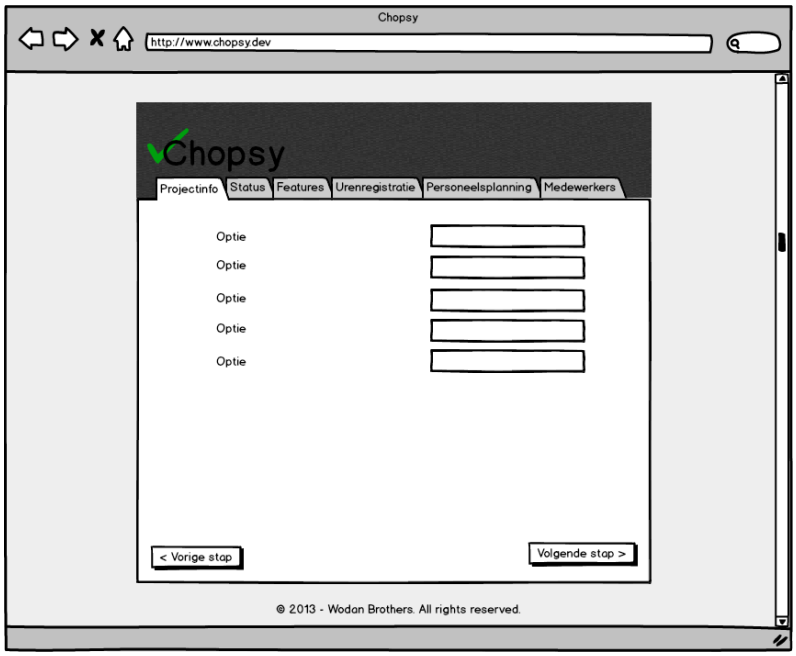
> *Als (re)seller wil ik een checklist kunnen invullen, zodat ik aan kan geven welke opties een klant heeft besteld.*

Een reden om de *requirements* op deze manier op te stellen is omdat het voor veel mensen makkelijker is om zo te beschrijven wat ze willen voor een systeem. Zij kunnen zich dan beter inleven in de rol van het type gebruiker, welke doelen ze daarvoor hebben en wat de reden is voor de wens. Daarnaast kan het voor de product owner eenvoudiger zijn de eisen te prioriteren doordat deze allemaal dezelfde structuur hebben. Hij zal ook beter zijn best moeten doen om te bedenken wat een bepaald onderdeel van het systeem is en wat de waarde daarvan is. (Cohn, 2008)

ID	Als een ...	Wil ik ...	Zodat ik ...
RS1	(re)seller	een checklist kunnen invullen	aan kan geven welke opties een klant heeft besteld
RS2	(re)seller	serienummers kunnen toevoegen van rex-o-matics	kan zien welke apparaten ik heb verkocht en aan welke klant
RS3	(re)seller	een status kunnen toevoegen aan een rex-o-matic	van alle apparaten kan zien wat er mee gebeurd is of mee moet gebeuren
RS4	(re)seller	een archief van opdrachten	het overzicht kan bewaren van openstaande opdrachten
RS5	(re)seller	een status bij opdrachten	kan zien wat er met welke opdracht moet gebeuren
RS6	(re)seller	een tijdlijn met opdrachten	een duidelijk overzicht heb van de opleveringen van systemen
RS7	(re)seller	een filter bij de openstaande opdrachten	een overzicht kan hebben van bepaalde opdrachten
RS8	(re)seller	notities kunnen maken bij onderdelen	extra informatie voor mezelf kan toevoegen
RS9	(re)seller	meerdere vestigingen van een klant kunnen toevoegen	precies weet waar welke klok terecht is gekomen
RS10	(re)seller	mail reminders ontvangen	weet wanneer een oplevering aankomt
RS11	(re)seller	een Excel file kunnen uploaden	medewerkers kan toevoegen aan een systeem
C1	klant	een PDF kunnen ontvangen	weet welke opties ik besteld heb
A1	admin	comments kunnen toevoegen aan een checklist / klant	ik kan communiceren met de reseller van die klant

Figuur 8: De user stories pagina van de product backlog

Na het opstellen van de user stories ben ik aan de slag gegaan met het opstellen van een 'Plan van Aanpak'. In dit plan heb ik de doelstelling van het project beschreven en wat mijn werkwijze zou zijn. Daarnaast heb ik ook de afbakening van het project beschreven. Dit is niet alleen voor mij prettig, maar ook de opdrachtgever weet waar hij aan toe is. Verder staat in het Plan van Aanpak een planning voor het hele project en staat er beschreven welke mensen en middelen ik nodig zou hebben. Als laatste geef ik aan wat de producten zijn die ik tijdens het project op zou leveren. Het Plan van Aanpak is zelf één van die producten. Ook de ontwerpdocumentatie en een 'Algemeen Testplan' zijn onderdeel van de mijlpaalproducten. Tevens zijn dit producten die ik in de fase voor de eerste sprint heb gemaakt, zodat ik ze kon gebruiken tijdens de verschillende sprints. De ontwerpdocumentatie betreft een aantal eenvoudige ontwerpen van de verschillende schermen, om een idee te krijgen van hoe het systeem er uit zou komen te zien, maar ook de documentatie van de database. Denk hierbij aan een *Enhanced Entity Relationship* diagram, een Database Dictionary, een *Relational Representation Model* en een *Relational Implementation Model*. De database wordt besproken in hoofdstuk 8. Daarnaast behoren ook het ontwerp van de Excel-file Validator en de Use Cases tot de ontwerpdocumentatie.



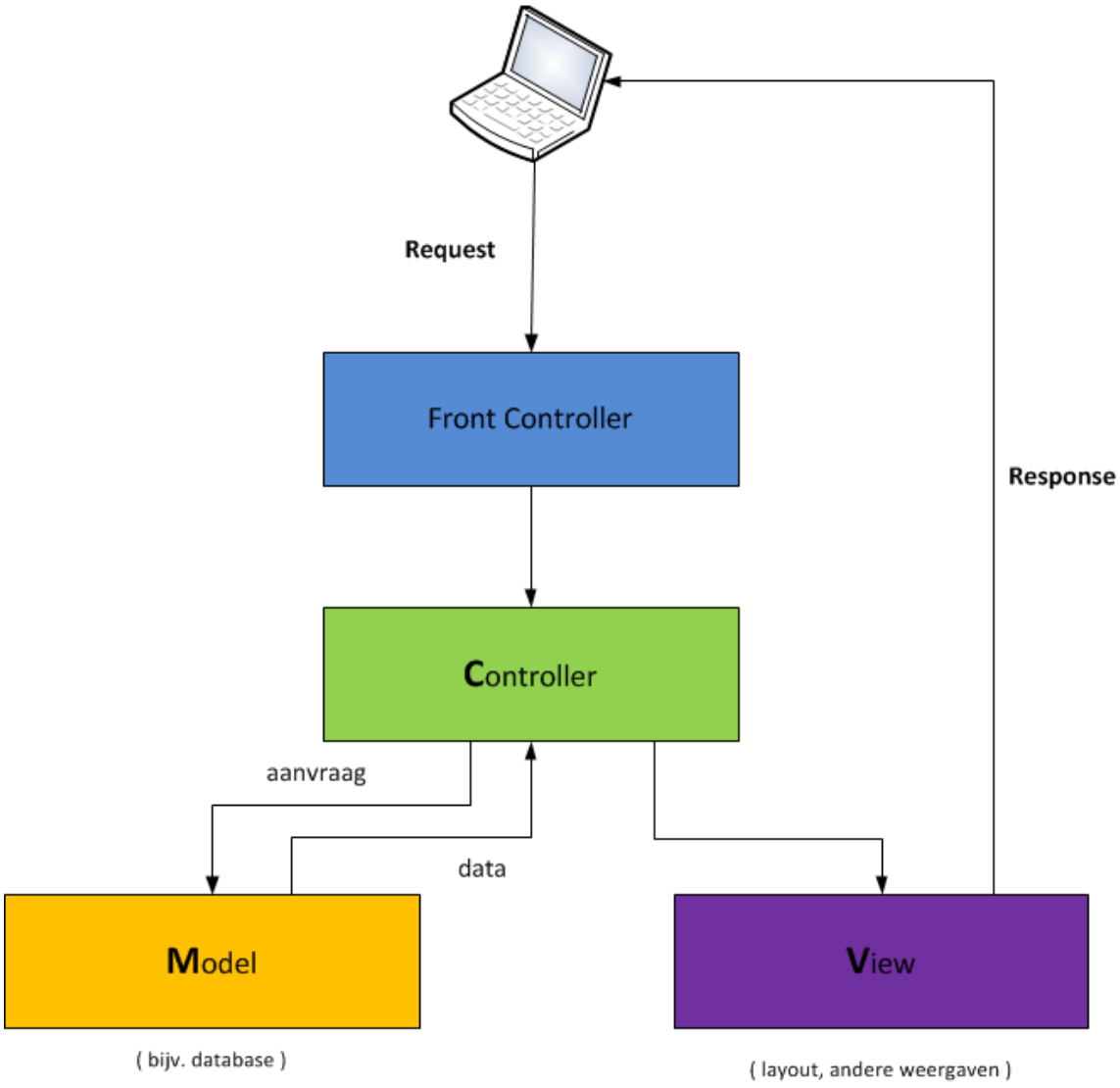
Figuur 9: Een ontwerp van één van de schermen van het systeem

Om van tevoren te weten hoe ik het systeem zou testen heb ik een Algemeen Testplan geschreven. Ik heb voor een algemeen plan gekozen, omdat de drie sprints qua opzet hetzelfde zijn en het niet nodig is om daar drie verschillende testplannen voor te schrijven. Deze zouden qua inhoud dan nauwelijks verschillen.

7.2 Design Patterns

Gedurende dit project zijn verschillende design patterns gebruikt. Van twee design patterns wist ik van tevoren dat ik ze wilde gebruiken. Dit zijn het Model-View-Controller-pattern (MVC) en het Front Controller-pattern. Deze twee design patterns wilde ik graag gebruiken omdat ik daar in het verleden prettig mee heb gewerkt. De combinatie van de twee design patterns zorgt ervoor dat de opzet van een nieuw project eenvoudiger is.
Het MVC design pattern wordt veel gebruikt bij webapplicaties en zorgt door het opsplitsen van de code in verschillende eenheden dat de applicatie beter te onderhouden is. Het MVC-framework dat gebruikt is tijdens dit project, is een bestaand framework dat ontwikkeld is door de hoofdprogrammeur van Wodan Brothers. Het framework is gebaseerd op ideeën achter het CakePHP framework. Het framework is een stuk eenvoudiger dan het CakePHP framework, maar wel

erg krachtig en doet wat het moet doen. Tijdens de ontwikkeling van het online checklistsysteem heb ik in het framework geen functionaliteit gemist. Het heeft mij erg geholpen bij het maken van het systeem. Iedere controller heeft zijn eigen functionaliteit en daardoor is het eenvoudiger om het overzicht te bewaren. Ook helpt het MVC-framework bij het onderhouden van het systeem. Als iets verplaatst moest worden binnen het systeem dan was dit eenvoudig te doen, omdat iedere controller zijn eigen verantwoordelijkheden heeft.



Figuur 10: Een voorbeeld van MVC met een Front Controller

Als aanvulling op het MVC framework is het Front Controller-pattern gebruikt (Gervasio, 2012). Ook dit is een veel gebruikt design pattern bij webapplicaties. De front-controller is binnen het MVC framework de eerste controller die

aangeroepen wordt. De controller handelt de pagina aanvragen af. Het zorgt voor duidelijke URLs, ook wel 'Friendly URLs' genaamd. Het maakt van URLs als:

> *www.website.nl/pagina.php?cat=13&id=1&prod=2131*

Een nette, duidelijke URL als:

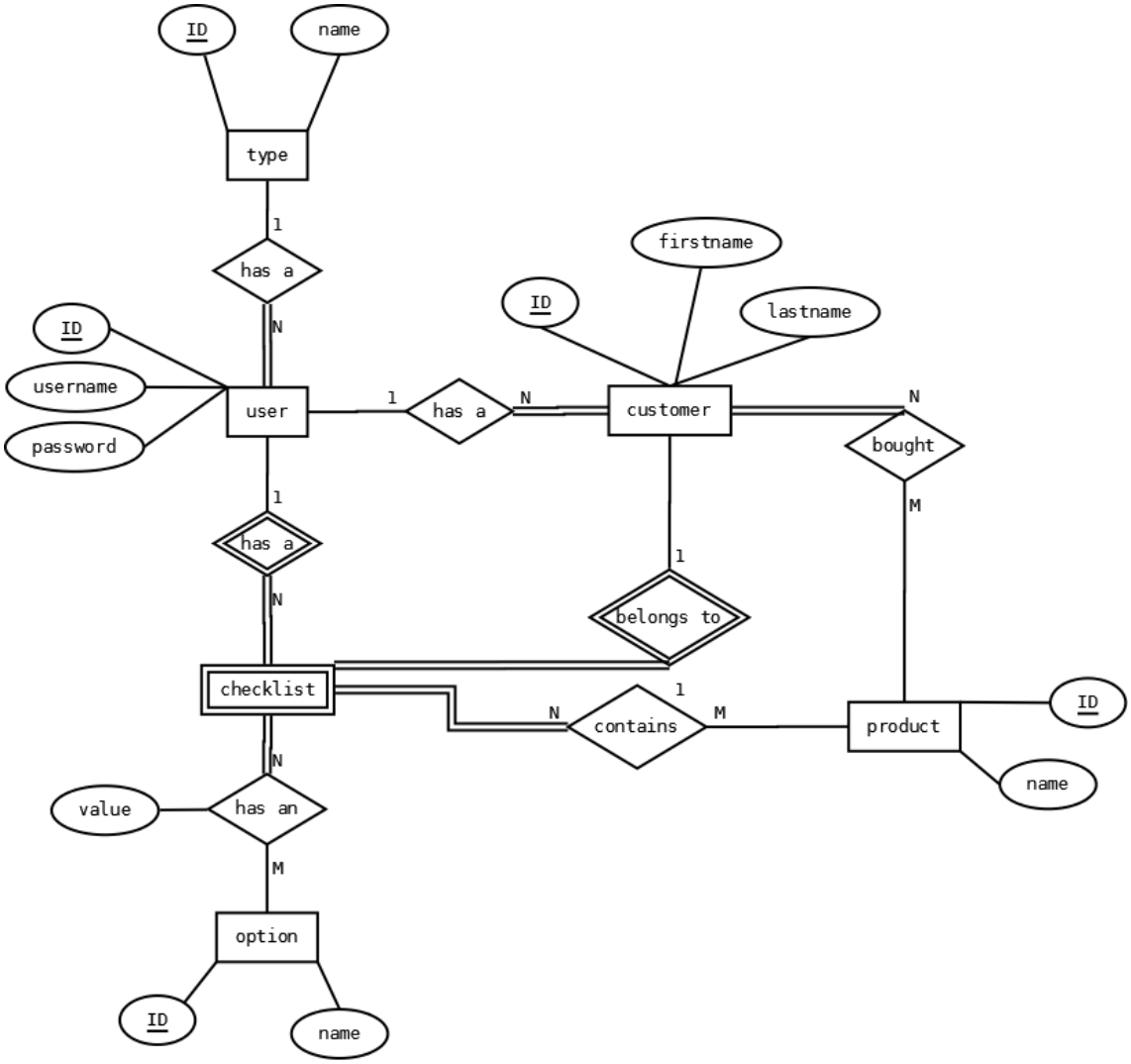
> *www.website.nl/13/1/2131*

Dit afhandelen van 'Friendly URLs' wordt routing genoemd. De front-controller zorgt dat de juiste pagina getoond wordt op basis van de URL die hij binnen krijgt.
Het gebruiken van 'Friendly URLs' zorgt ook voor een klein beetje extra beveiliging. Het is niet veel, maar gebruikers van de site kunnen niet direct zien dat voor de site PHP gebruikt wordt, wat bij een url als in het eerste voorbeeld wel het geval is.

Tijdens dit project zijn in zowel PHP als JavaScript verschillende Design Patterns gebruikt. De gebruikte design patterns worden besproken als ze van toepassing zijn. Omdat het gebruik van de design patterns erg verweven is met overige werkzaamheden zullen ze niet in een apart hoofdstuk besproken worden.

8. De database

Het opstellen van een gegevensmodel en het bijhouden daarvan is niet sprint-specifiek. Om die reden is er een apart hoofdstuk voor de database, zodat alle werkzaamheden aan de database bij elkaar besproken kunnen worden.

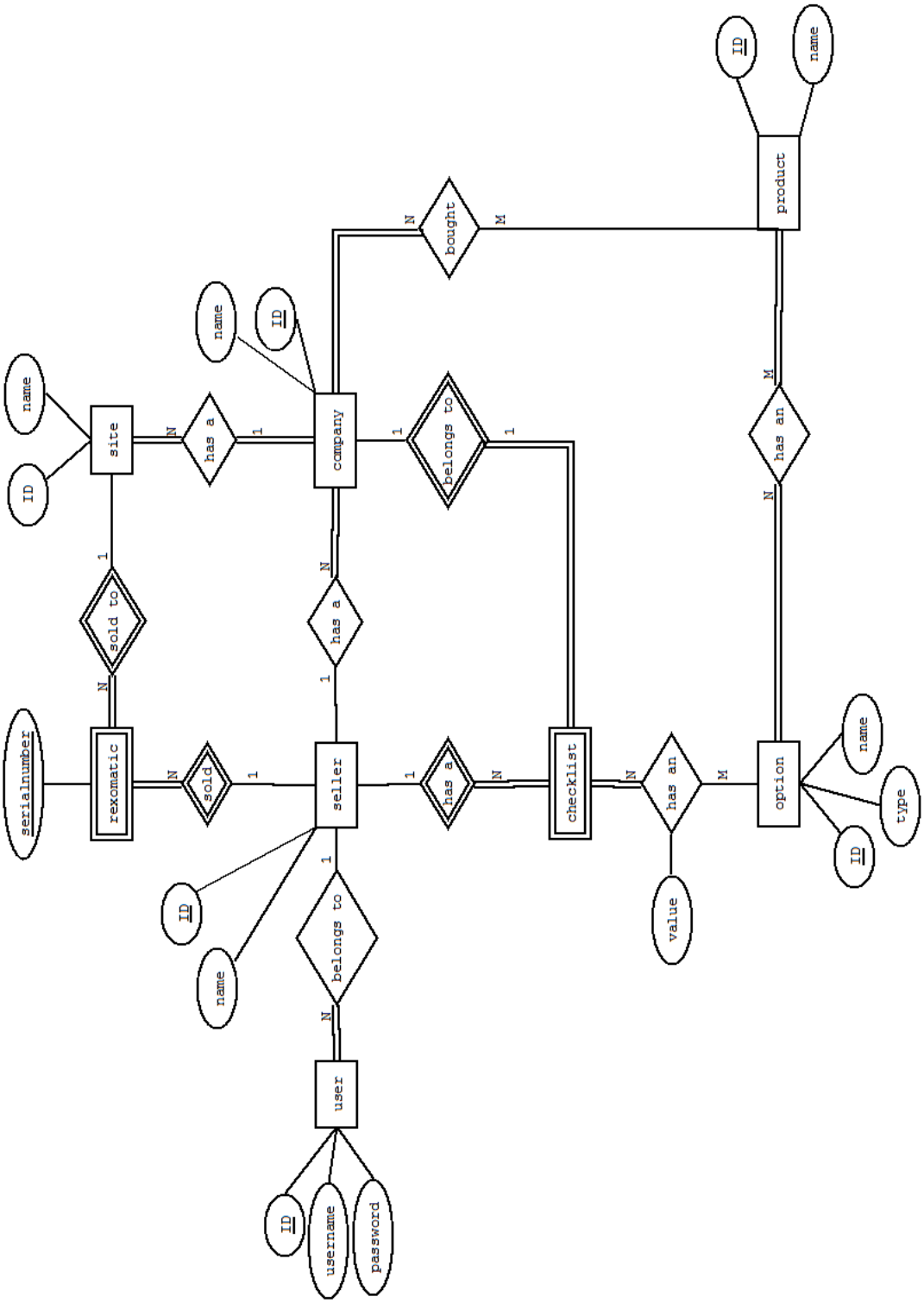


Figuur 11: De eerste versie van de database

De eerste versie van het gegevensmodel van de database is opgesteld voor de eerste sprint. Dit gegevensmodel is opgesteld na gesprekken met de product owner en de hoofdprogrammeur. Tijdens dit gesprek hebben we de oude situatie besproken, zodat ik er achter kon komen hoe er op dat moment bij Wodan Brothers gewerkt werd en hoe ze bestelde systemen klaarzetten. Het belangrijkste zijn de checklists. Deze worden door resellers ingevuld en naar

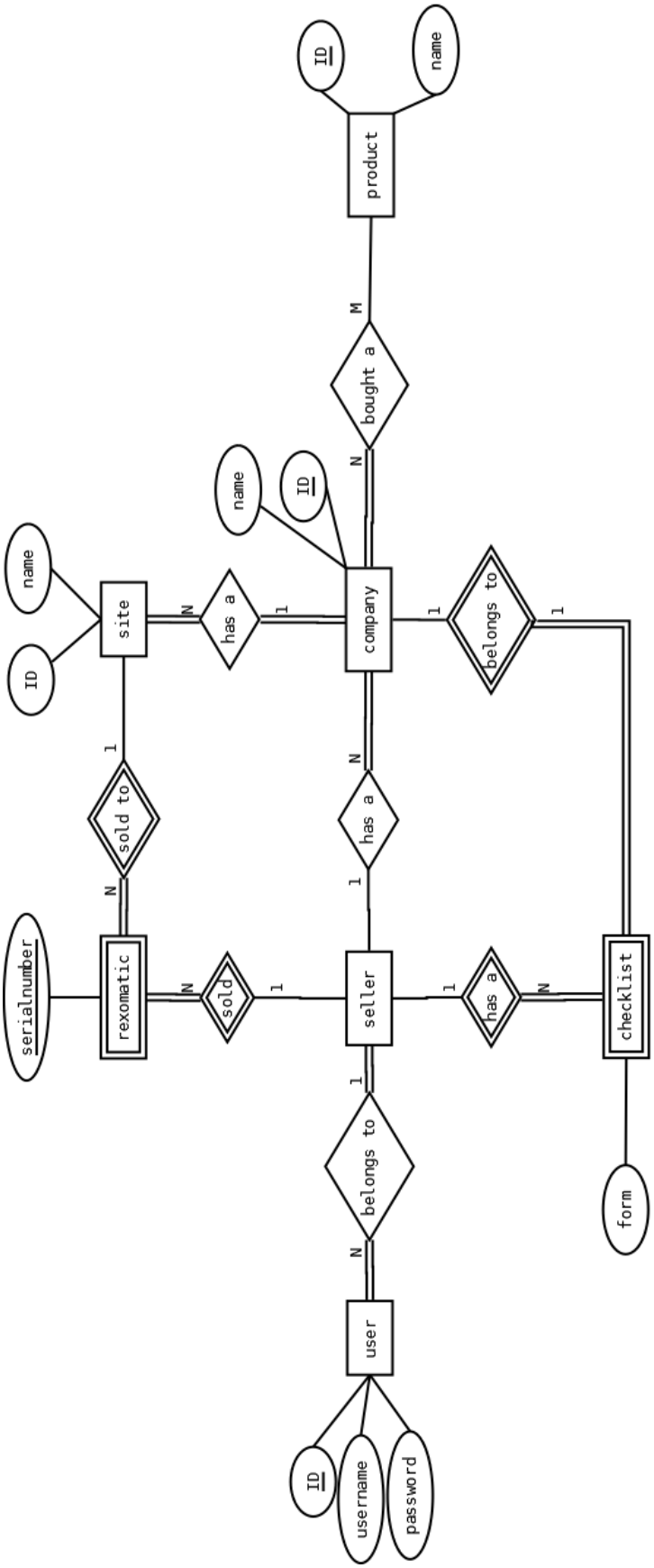
Wodan Brothers gestuurd. Medewerkers van Wodan Brothers vullen ook checklists in. Zij zijn immers ook verkopers van het systeem. In eerste instantie leek het mij verstandig om alle opties in een aparte tabel te zetten. De opties zouden dan overeenkomen met de opties uit Dyflexis en elke checklist zou dan een verwijzing bevatten naar alle opties met een waarde die door de gebruiker was ingevoerd.

Na een vervolggesprek met de product owner, om achter zijn wensen voor het systeem te komen, heb ik het gegevensmodel aangepast. Dit is de tweede versie geworden die u kunt zien in **Figuur 12**. Duidelijk was geworden dat klanten meerdere vestigingen konden hebben en dat deze ook in het systeem opgegeven zouden moeten worden. Omdat Wodan Brothers naast het Dyflexis planningsysteem ook prikklokken levert, zou het ook nodig zijn om in het systeem bij te houden welke klok aan welke vestiging geleverd was of zou worden. Een andere aanpassing is dat de verkopende partijen meerdere gebruikers kunnen hebben. Het gegevensmodel is aangepast om dit mogelijk te maken.

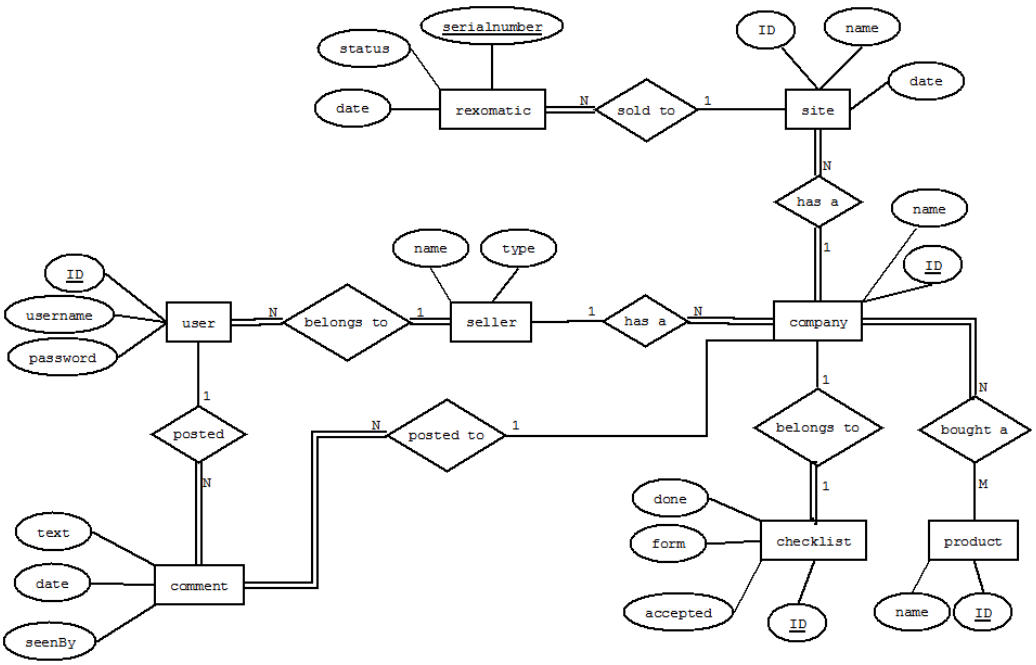


Figuur 12: De tweede versie van de database

Het gesprek met de product owner was het laatste gesprek dat ik heb gehad voor het begin van de eerste sprint – meer over deze sprint kunt u lezen in hoofdstuk 9. In deze sprint ben ik begonnen met het omzetten van de papieren checklists naar een digitale versie. Tijdens het omzetten kwam ik er achter dat de tabel met opties geen goed oplossing zou zijn. Het systeem moet flexibel zijn en eenvoudig aan te passen als dat nodig is. Een tabel met alle verschillende opties is niet flexibel genoeg. Het is mogelijk om een nieuwe optie in de tabel te zetten en deze ook in de digitale versie van de checklist te plaatsen. Voor één enkele optie is dat prima te doen. Ook als daarbij het Database Design Document aangepast wordt zal de hoeveelheid werk meevallen. Als er een compleet nieuwe checklist aan het systeem toegevoegd moet worden zal het aanzienlijk meer werk zijn. Om die reden heb ik de tabel met opties verwijderd uit het gegevensmodel. De tabel 'checklist' bevat in de derde versie van het gegevensmodel, te zien in **Figuur 13**, de hele, ingevulde checklist. Dit is gedaan door de hele checklist te serialiseren. Dit betekent dat alle invoervelden met hun waarden omgevormd worden naar één lange string. Op deze manier hoeft geen enkele checklist hetzelfde te zijn en zullen er ook geen problemen optreden als er opties verwijderd of toegevoegd worden. De checklist zal altijd terug te halen zijn.

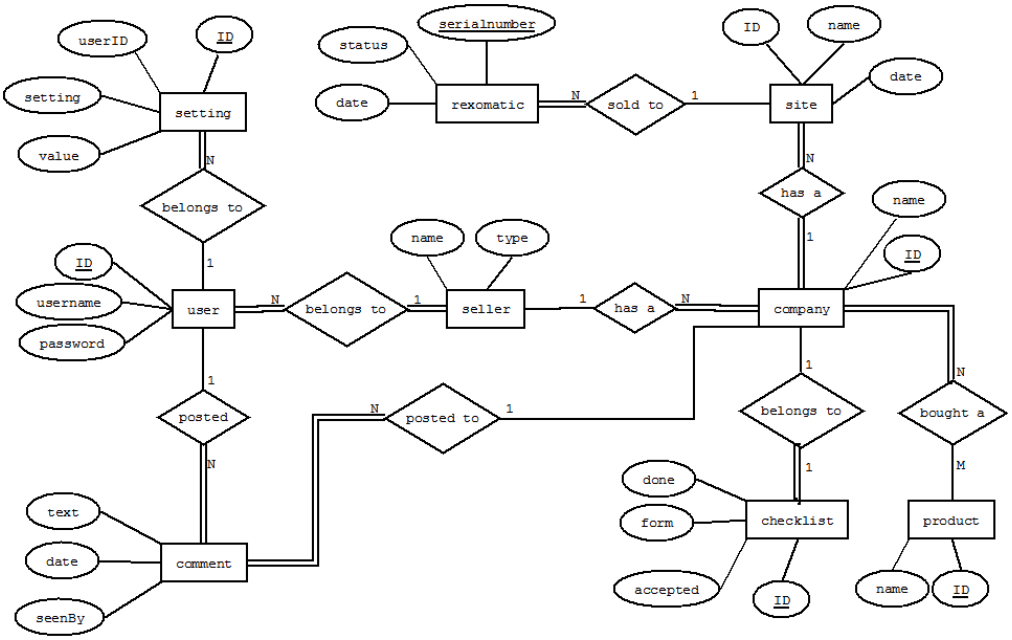


Figuur 13: De derde versie van de database



Figuur 14: De vierde versie van de database

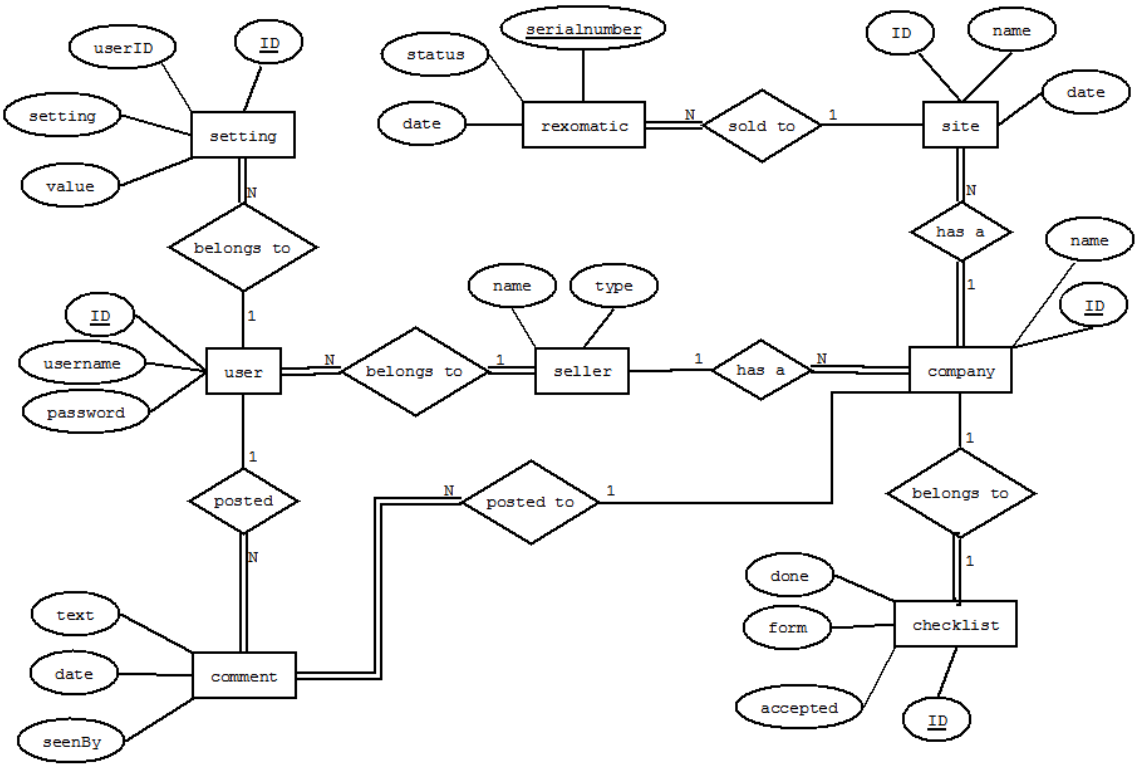
De vierde versie van het gegevensmodel is gemaakt na het toevoegen van het commentaar voor de resellers (zie hoofdstuk 10.3). Daarnaast is er een aantal kleine aanpassingen gedaan aan enkele andere tabellen. Sommige tabellen hadden extra velden nodig om opties binnen het systeem mogelijk te maken.



Figuur 15: De vijfde versie van de database

Omdat het systeem de mogelijkheid bevat om de gebruiker te waarschuwen wanneer er een oplevering aankomt, is het prettig voor de gebruiker om in te kunnen stellen op welk moment hij gewaarschuwd wil worden. Om die reden is er een tabel aan het gegevensmodel toegevoegd. Dit is de tabel die de instellingen van de verschillende gebruikers bevat.

De product owner heeft in een gesprek gezegd dat hij graag wilde dat de producten die een klant heeft besteld ook op de checklist kwamen. Uiteindelijk is besloten daar niets mee te doen en is deze tabel in de zesde versie van het gegevensmodel verwijderd.



Figuur 16: De zesde en laatste versie van de database

Het ontwerpen van het gegevensmodel voor de database is een doorlopend proces geweest gedurende het hele project. Meerdere stakeholders hebben invloed gehad op het ontwerp. De eerste en belangrijkste stakeholder is de product owner. Daarnaast heb ik nog input gekregen van de commercieel directeur van Wodan Brothers. Hij is degene die meestal de opdrachten in het systeem zal zetten. De derde stakeholder is ook een medewerker van Wodan Brothers. Hij controleert de checklists en zet de systemen klaar voor de klanten.

Een vierde stakeholder waar ik rekening mee heb moeten houden is een reseller. Dit is iemand die niet werkt voor Wodan Brothers, maar wel moet werken met het systeem. Halverwege het project is het systeem online gegaan, zodat het ook in de praktijk getest kon worden. Dit heeft veel invloed gehad op het gegevensmodel en de bouw van het systeem.

9. De eerste sprint: 'van papier naar digitaal'

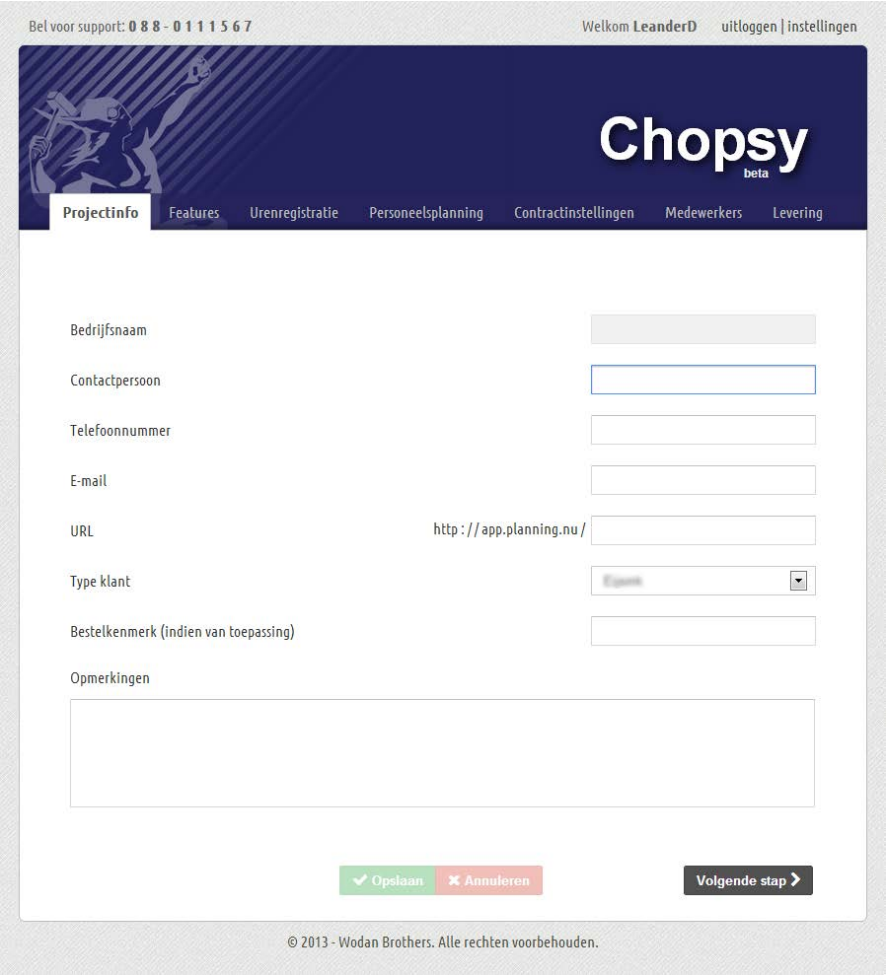
Voor iedere sprint heb ik een Sprint Backlog gemaakt. Een Sprint Backlog is de planning voor een sprint. Met de product owner heb ik besproken welke user stories hij graag geïmplementeerd wilde hebben in de eerste sprint. Het doel van de eerste sprint was het digitaliseren van de checklist. Voor ik begon met de eerste sprint heb ik alle user stories omgezet in voor mijzelf duidelijke taken. Deze uitgewerkte taken kon ik van het Kanban Board in de Sprint Backlog zetten en ze maakten voor mij duidelijk wat ik te doen had om een user story te realiseren. Vervolgens heb ik het aantal dagen dat beschikbaar was voor de sprint vermenigvuldigd met het aantal uren dat ik op een dag had om aan het systeem te werken. Met dit totaal aantal uren heb ik een planning gemaakt voor mijn taken. Dit heb ik gedaan op basis van de ervaring die ik heb.

Zoals u eerder hebt kunnen zien in **Figuur 3** bevat een Sprint Backlog ook een Burndown Chart. Deze grafiek geeft met de rode lijn het geplande aantal uren weer. Als alles volgens planning zou verlopen, zou het aantal uren per dag met acht af moeten nemen. De blauwe lijn geeft het werkelijke aantal uren weer. Het kan voorkomen dat er te veel of te weinig uren voor een taak gepland zijn. De blauwe lijn gaat dan respectievelijk omlaag of omhoog ten opzichte van de rode lijn. Het is met een Burndown Chart eenvoudiger te zien of er nog volgens planning gewerkt wordt.

9.1 Een digitale checklist

Na de bespreking met de product owner en het inplannen van de taken van de user stories was duidelijk dat het doel van de eerste sprint was om een digitale versie te hebben van de eerder besproken papieren checklists. Dit is een van de belangrijkste en grootste onderdelen van het systeem. Om alles overzichtelijk te houden heb ik ervoor gekozen om de verschillende onderdelen van de papieren checklist te verdelen over verschillende tabbladen. Dit is gebruiksvriendelijker omdat de gebruiker geen lange lijsten voorgeschoteld krijgt en daardoor een beter overzicht heeft. Het was ook mogelijk geweest om de verschillende onderdelen van de checklist op verschillende pagina's te zetten. Dit is wel gebruiksvriendelijker dan alles op één pagina, maar doordat de pagina's

tussendoor moeten laden is het voor de gebruiker minder prettig. Dit is zeker het geval als de gebruiker heen en weer wil tussen verschillende stappen. Om het voor de gebruiker zo eenvoudig en prettig mogelijk te maken is ervoor gekozen om de onderdelen op verschillende tabbladen te zetten.



Figuur 17: De digitale checklist

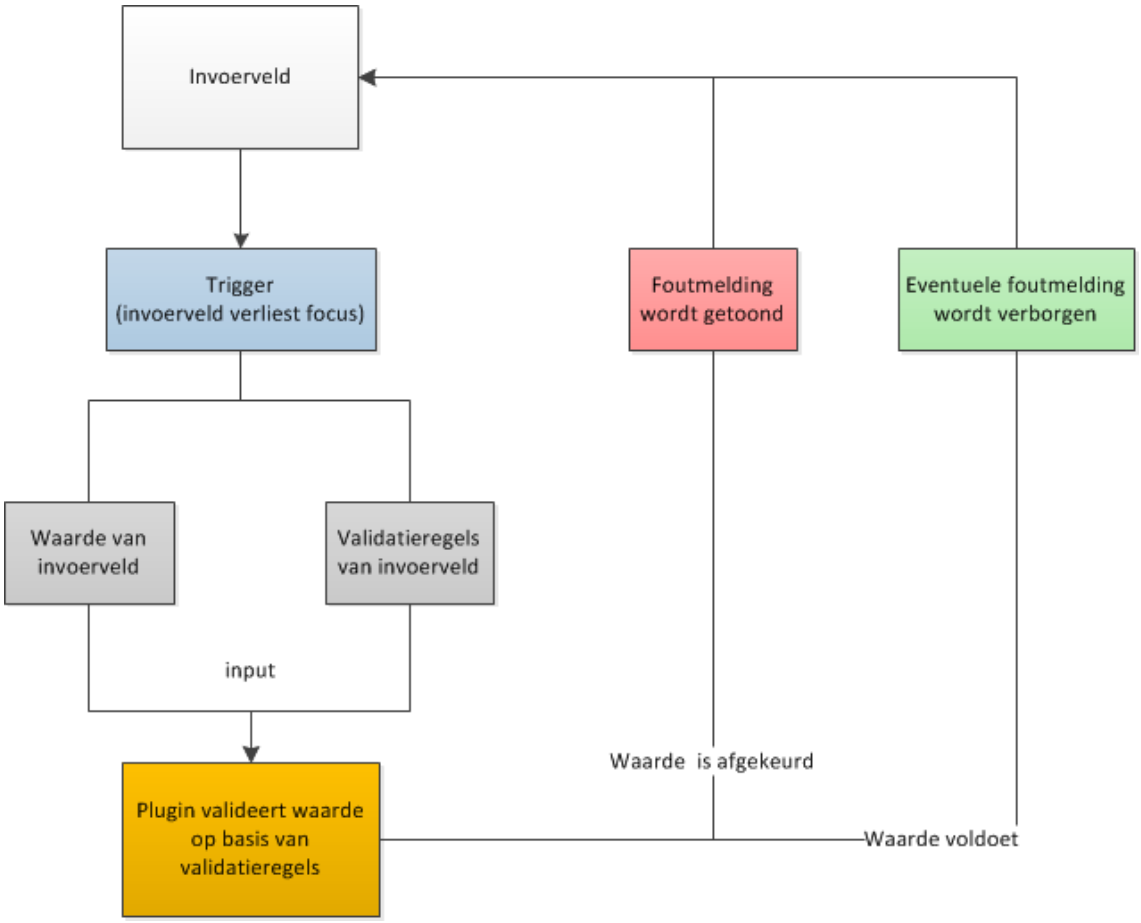
9.2 Front-end validatie

Een groot voordeel van het digitaliseren van de checklists is dat de input van de gebruikers gecontroleerd kan worden. Zo hoeft de user input niet achteraf gecontroleerd te worden. Dit kan erg veel tijd schelen voor de medewerkers van Wodan Brothers, maar voor de gebruiker is het ook prettiger werken. Hij krijgt immers direct feedback over fouten die hij heeft gemaakt en kan deze vervolgens oplossen, zonder dat achteraf nog te hoeven doen. Er zijn meerdere oplossingen

mogelijk voor het controleren van fouten. Het was ook mogelijk geweest om de fouten in de backend te controleren, maar dan was de gebruiker terug gestuurd naar de pagina waar hij vandaan kwam met een lijst met foutmeldingen. Deze oplossing is veel minder gebruiksvriendelijk en daarom niet toegepast.
De ingevoerde data van de gebruiker wordt in de backend uiteraard nog wel gecontroleerd en er komt ook een foutmelding als het nodig is, maar dat zijn vaak fouten zoals het ontbreken van een verbinding met de database.

Figuur 18: De front-end validatie van het systeem

Voor de front-end validatie is een eigen jQuery-plugin geschreven. Er zijn jQuery-plugins beschikbaar die front-end validatie doen, maar bieden helaas niet alle opties die nodig zijn voor de checklist. Soms moeten velden gecontroleerd worden aan de hand van een Regular Expression of aan de hand van een functie. Daarnaast kunnen de gewenste opties en regels voor validatie op meerdere manieren doorgegeven worden aan de plugin. De plugins die beschikbaar waren op het moment van ontwikkelen boden deze functionaliteiten niet.



Figuur 19: De werking van de validatie-plugin

De plugin maakt gebruik van validatieregels die via de code van het invoerveld zijn toegevoegd. Bij het online checklistsysteem wordt een veld gecontroleerd zodra het de focus verliest. De plugin gebruikt de waarde van het invoerveld en de validatieregels voor dat veld om de waarde te valideren. Als een waarde wordt afgekeurd krijgt de gebruiker een foutmelding te zien. Wordt de waarde van het invoerveld goedgekeurd, dan zal een eventuele eerdere foutmelding voor dat veld verborgen worden.

Om de jQuery-plugin te maken zijn verschillende design patterns gebruikt.
Binnen het validatie algoritme worden bepaalde stappen voor het valideren van de waarde van een invoerveld uitgevoerd door een subklasse. Dit is het Template Pattern.
De subklasse bevat alleen methodes die aangeven of een waarde voldoet aan een bepaalde eis. De validatie-methode roept de methode aan van de subklasse die hij op dat moment nodig heeft. De subklasse controleert de waarde die hij mee heeft gekregen en geeft aan of de waarde voldoet of niet. De validatie-methode

handelt vervolgens de rest van de validatie af. Het kan namelijk zo zijn dat een waarde juist wel of juist niet moet voldoen. Een voorbeeld hiervan is dat een invoerveld een bepaald woord wel of niet mag bevatten.
Naast het Template Pattern is het Prototype Pattern (Osmani, 2012) gebruikt. De Gang of Four (Carr, 2009) beschrijft het prototype pattern als een design pattern dat objecten creëert gebaseerd op een template van een bestaand object door middel van klonen. Bij de validatie-plugin heb ik dit design pattern gebruikt om een nieuw String-object te creëren. Het String-object van JavaScript miste voor mij een format functie. De validatie-plugin maakt gebruik van standaard foutmeldingen. Deze foutmeldingen moeten ook de invoer van de gebruiker bevatten om aan te geven wat er fout is. De invoer van de gebruiker zal niet altijd op dezelfde plaats in de foutmelding staan, bijvoorbeeld aan het einde van de zin. Om dit op te lossen heb ik het prototype pattern gebruikt om een nieuw String-object te maken met een extra functie: de format-functie. In werkelijkheid wordt er dus een nieuw object gemaakt, gebaseerd op een bestaand object. Ik vind het prettig om het te zien als het toevoegen van functionaliteit aan bestaande objecten.

Een derde design pattern dat ik gebruikt heb voor de validatie-plugin is het Mixin Pattern (Osmani, 2012). De validatie plugin bevat een object 'defaults'. Dit zijn de standaard instellingen van de plugin. Deze instellingen kunnen door de gebruiker van de plugin overschreven worden. Om dit mogelijk te maken is het Mixin Pattern gebruikt. De gebruiker kan aan de validatie-plugin een nieuw object meegeven. Dit object bevat de attributen en methoden die hij wil gebruiken als instellingen voor de validatie.
De reden voor het maken van een plugin is dat het dan bij meer projecten gebruikt kan worden. Dat houdt echter ook in dat de plugin eenvoudig in gebruik moet zijn voor programmeurs die niets van mijn plugin afweten. Om die reden heb ik gebruik gemaakt van het Façade Pattern. Het enige dat de programmeur hoeft te doen is de validatie-plugin te koppelen aan een invoerveld. Om een voorbeeld te geven:

```
$("id_van_invoerveld").validate( { instelling_object } );
```

De echte code van de validatie-plugin is verborgen en de programmeur maakt

gebruik van een façade. Het gebruik van de plugin op deze manier brengt mij ook tot het Composite Pattern (Osmani, 2012). In bovenstaand voorbeeld wordt de validatie-plugin aangeroepen op één enkel invoerveld. Maar de plugin is zo geschreven dat de façade ook gebruikt kan worden op een collectie van invoervelden. Dit is het Composite Pattern, waarbij een collectie van objecten behandeld kan worden op dezelfde manier als één enkel object.
Een laatste design pattern dat gebruikt is: 'A Lightweight Start'. Dit is geen traditioneel design pattern, maar een modern design pattern. Dat wil zeggen dat het nog niet erg lang bestaat omdat het een design pattern is speciaal voor jQuery-plugins. Het eerder beschreven 'defaults'-object is een onderdeel van dit design pattern. Daarnaast wordt er voor de 'function invocation' een puntkomma geplaatst, omdat een andere plugin misschien niet correct is afgesloten. Mocht dat het geval zijn, dan wordt die plugin alsnog afgesloten. Dit zijn twee voorbeelden van de best practices die gebruikt worden in dit design pattern.

9.3 Beveiliging

Chopsy is een online systeem en zal daarom goed beveiligd moeten zijn. Als eerste is het belangrijk dat niet iedereen toegang heeft tot het systeem. Om te zorgen dat het systeem afgeschermd is, is er een login gemaakt. Hiervoor heeft een gebruiker een gebruikersnaam en wachtwoord nodig. Om te kunnen controleren of de gegevens die de gebruiker invoert kloppen, worden de gebruikersnaam en het wachtwoord in de database opgeslagen. Een wachtwoord kan echter niet als 'plain-text' opgeslagen worden. De database zou gehackt kunnen worden en dan zouden de wachtwoorden van de gebruikers eenvoudig te lezen zijn. Om die reden zijn de wachtwoorden gehasht en daar zijn meerdere manieren voor.
Een aantal van deze manieren ken ik uit ervaring ermee. Ik heb in het verleden uitgezocht wat de verschillende hash-functies van PHP doen en hoe veilig ze zijn. Omdat het enige tijd geleden is dat ik dit heb uitgezocht, ben ik de voor mij bekende hash-functies opnieuw gaan uitzoeken. Ik ben begonnen met de minst veilige hash-functie die ik van PHP ken, namelijk: MD5.
De MD5 hash-functie werd tot voor kort nog veel gebruikt. Tegenwoordig zijn er zogenaamde 'rainbow tables' (tabellen met mogelijke wachtwoorden) en dat

betekent dat door MD5 gehashte wachtwoorden zonder salt (een reeks willekeurige karakters) niet veilig meer zijn. Een salt gebruiken kan een MD5-hash veiliger maken, maar computers worden sneller en sneller en daardoor zal het steeds makkelijker worden om ze via brute force te kraken. MD5 is geschikt om via brute force te kraken, omdat hoe meer rekenkracht een computer heeft hoe sneller hij verschillende wachtwoorden / tekenreeksen kan proberen. Met het oog op de toekomst is het dus beter een andere hash-functie te gebruiken. Er zijn nog SHA-hash-functies, maar ook deze kunnen sneller gekraakt worden met meer rekenkracht. Daardoor is ervoor gekozen om ook de SHA-hash-functies niet te gebruiken. BCrypt, gebaseerd op Blowfish, maakt gebruik van iteraties. Het aantal iteraties kan gekozen worden om het algoritme langzamer te maken. Doordat BCrypt gebruik maakt van iteraties zal het ook bestand zijn tegen brute force, zelfs als de rekenkracht van computers toeneemt. Deze iteraties houden in dat de hash-functie vertraagd wordt. Door de hash-functie te vertragen zal deze een maximaal aantal keer per uur uitgevoerd kunnen worden en daardoor wordt ook de brute-force aanval vertraagd.

Door de voor mij bekende hash-functies opnieuw uit te zoeken ben ik via-via op andere hash-functies gekomen die ik hier verder niet zal bespreken. Eén ervan wil ik echter nog wel even kort bespreken. Dit is de CRC32 hash-functie. Een probleem dat kan optreden bij deze hash-functie is 'hash collision'. Een probleem waar ik zelf nog niet aan had gedacht. Doordat deze hash-functie een 32-bit integer genereert als hash, zijn er maar 2^{32} (4.294.967.296) mogelijkheden (Guzel, 2013). Daardoor is het mogelijk dat verschillende input-waardes, dezelfde output hebben. Als een wachtwoord niet bekend is, maar iemand heeft wel de hash gevonden door bijvoorbeeld het hacken van de database, dan is het mogelijk een wachtwoord te vinden dat dezelfde hash oplevert. Dit wordt 'hash collision' genoemd. Meerdere input-waardes die dezelfde output opleveren. Omdat bcrypt een 448-bit hash oplevert bestaat hier het probleem van 'hash collision' niet. (Bcrypt, sd)

Om de wachtwoord-hashes nog veiliger te maken wordt er gebruik gemaakt van een salt. Deze salt staat alleen in de broncode en wordt dus niet in de database opgeslagen. Mocht er met brute force geprobeerd worden de wachtwoorden te kraken, dan zal dus ook de salt gevonden moeten worden. Wachtwoorden zijn

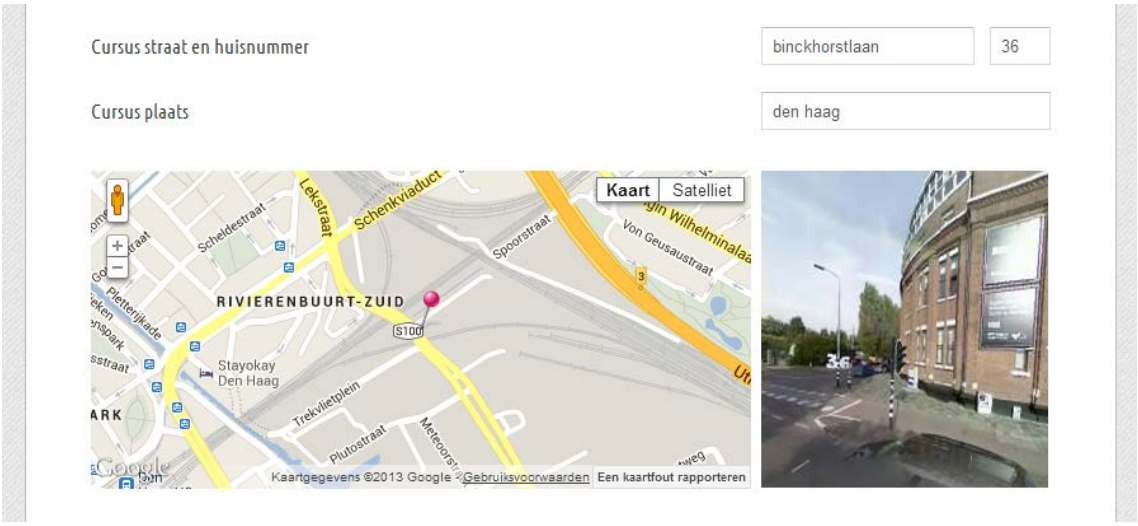
vaak bestaande woorden, de salt is echter een willekeurige reeks van tekens en zal daarom veel moeilijker zijn om te achterhalen.

Doordat BCrypt bestand is tegen brute force, en door de salt ook tegen rainbow tables, is er voor gekozen om deze hash-functie te gebruiken voor de wachtwoorden.

9.4 Feedback

Tijdens deze sprint heb ik een aantal keer aan mijn opdrachtgever laten zien hoe de ontwikkeling van het systeem vorderde. Bij deze demonstraties heb ik feedback gekregen die ik verwerkt heb in een Feedbacklog. De meeste feedback punten heb ik nog in deze sprint meegenomen, omdat het vaak om kleine punten ging. Punten die open zijn blijven staan heb ik verwerkt in het Sprintrapport. Daarin heb ik besproken waarom ik ze niet heb meegenomen en wat er mee zou gaan gebeuren.

De feedback die ik tijdens de eerste sprint heb gekregen gaat voornamelijk om kleine aanpassingen. Elementen moesten bijvoorbeeld verplaatst worden naar een ander gedeelte van het systeem. De meeste aanpassingen die ik moest doen tijdens deze sprint waren aanpassingen aan de vormgeving. De grootste toevoeging die ik heb moeten doen is een Google Map tonen.



Figuur 20: Google Maps met foto van het pand

Tijdens het invullen van het adres zal de Google Map zich focussen en aan de rechterkant zal een foto getoond worden van het adres. Mits mogelijk is dit een foto van de voordeur. De foto van het pand heb ik toegevoegd omdat het adres de locatie van de cursus is die gegeven moet worden. Voor de medewerkers die de cursus moeten geven is het handig om te kunnen zien waar zij moeten zijn. Ook deze toevoeging is gedaan om de gebruiksvriendelijkheid te verhogen.

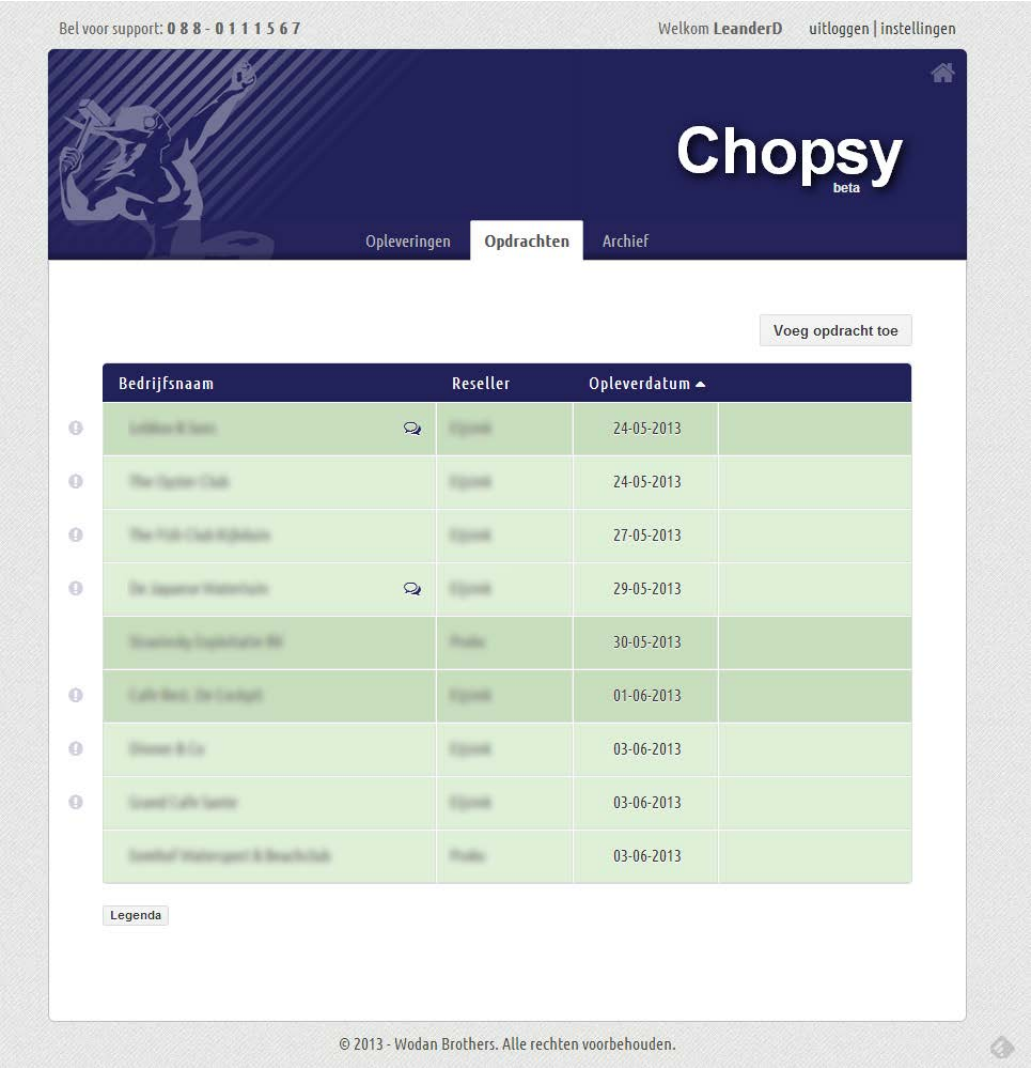
10. De tweede sprint: 'meer overzicht'

Ook voor deze sprint heb ik van tevoren een Sprint Backlog gemaakt. De product owner heeft een aantal user stories uitgekozen die hij graag in deze sprint geïmplementeerd wilde hebben. Daarnaast heeft hij bepaald welke punten van de feedback van de vorige sprint hij verwerkt wilde hebben. Bij de tweede sprint was het doel om meer overzicht te creëren in de opleveringen. Dit moest bereikt worden door het maken van een overzicht van de klanten en daarbij de status van de checklist en het medewerkersbestand.

10.1 Van whiteboard naar website

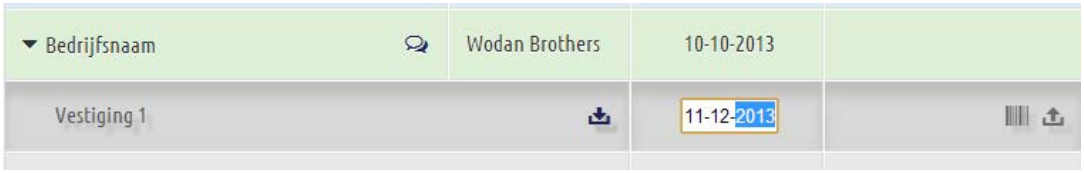
Op het kantoor van de Wodan Brothers hangt een groot whiteboard. Op dit bord worden de opleveringen van de door klanten bestelde systemen geschreven. De datum van de oplevering wordt opgeschreven, maar ook of de checklist compleet is en of het medewerkersbestand ontvangen is. De Wodan Brothers moeten in de gaten houden wanneer welke oplevering is. Sorteren op een whiteboard is met de hand erg veel werk, dus de opleveringen staan niet op volgorde van opleverdatum. In Chopsy kunnen de checklists worden ingevuld, maar de product owner wil ook graag de opleveringen in het systeem bijhouden.
Om dit te bereiken heb ik een tabel gemaakt met de klanten van Wodan Brothers die aan het systeem zijn toegevoegd. Door de rijen van de tabel verschillende kleuren te geven – de kleuren zijn afhankelijk van de status van de oplevering – is eenvoudig te zien hoe ver een oplevering gevorderd is. Daarnaast wordt met iconen aangegeven of er een medewerkersbestand is geüpload.
Een rij zal gaan knipperen als de deadline nadert en de oplevering aandacht nodig heeft. Het kan bijvoorbeeld zijn dat de checklist niet als voltooid is gemarkeerd of überhaupt nog niet is ingevuld.
Mocht de deadline erg dichtbij komen, dan zal de rij blijven knipperen, maar de tekst zal rood zijn in plaats van de standaard kleur. Het precieze moment dat een rij gaat knipperen wordt bepaald door de instellingen van de gebruiker.
Door het gebruik van kleuren en het vragen om aandacht van de gebruiker is het eenvoudiger om te zien wat er met welke opleveringen moet gebeuren.
Verder is het mogelijk om te sorteren in de tabel. Dit kan op naam, reseller of

datum. Daardoor zal de gebruiker ook een beter overzicht hebben van de opleveringen.



Figuur 21: De digitale versie van het whiteboard

Om het voor de gebruiker makkelijker te maken om bijvoorbeeld een bedrijfsnaam of datum aan te passen heb ik 'inline-editing' toegevoegd. De gebruiker kan nu klikken op het element dat hij wil bewerken. Er zal een tekstvak verschijnen waar direct de nieuwe waarde ingevoerd kan worden. Zodra de gebruiker op 'enter' drukt zal de nieuwe waarde opgeslagen worden. Dit is sneller en gebruiksvriendelijker omdat er geen nieuw formulier of scherm getoond hoeft te worden.

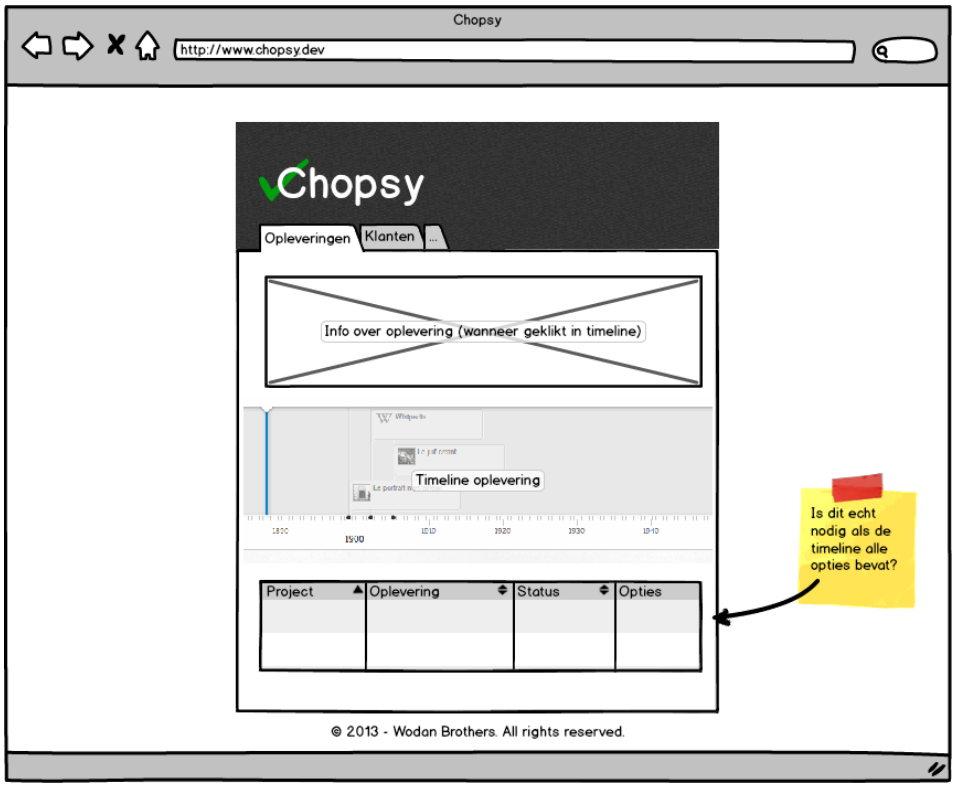


Figuur 22: Inline-editing van een opleverdatum

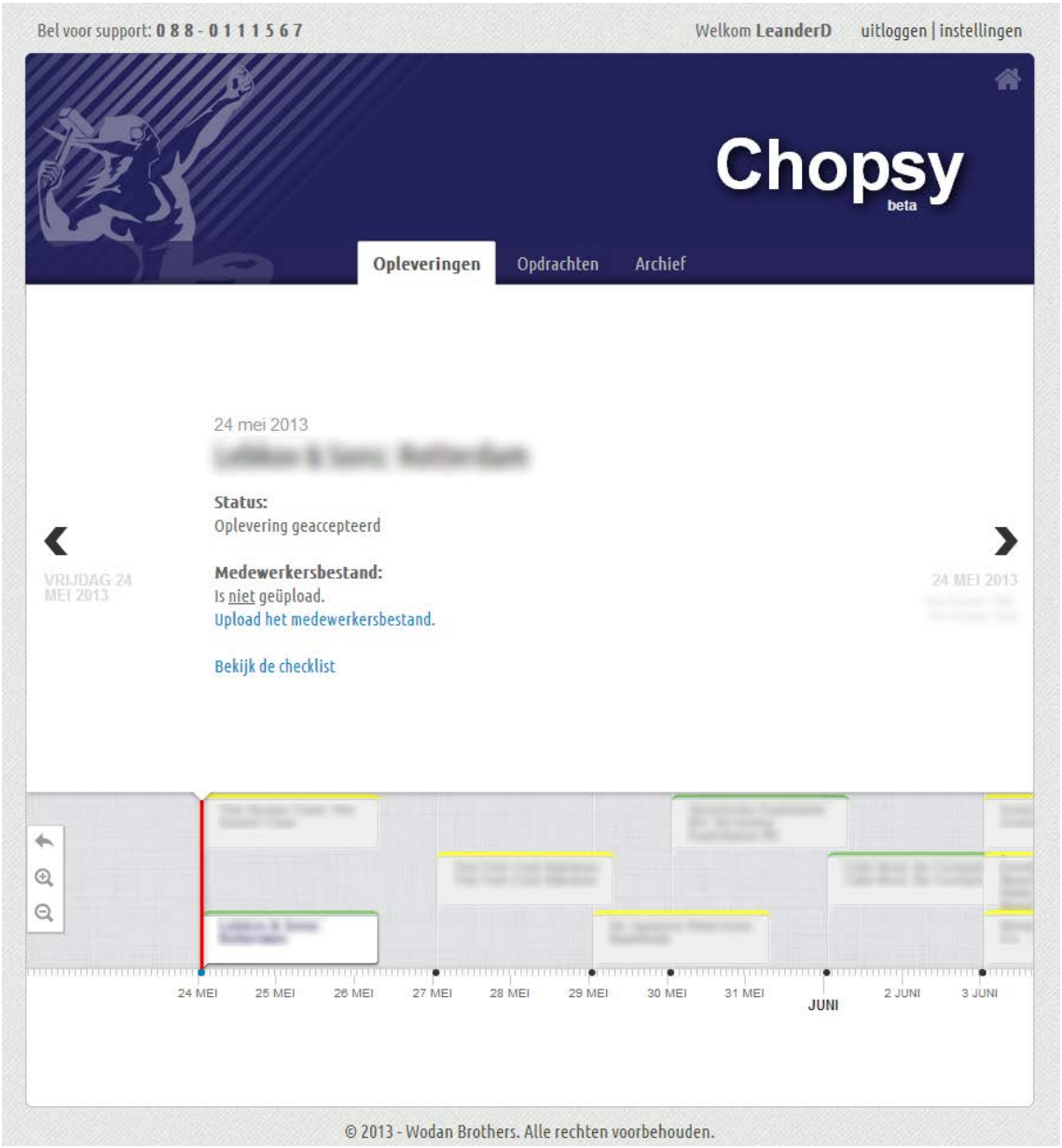
10.2 Een overzicht met tijdlijn

Een aanvulling op de tabel met opleveringen is een tijdlijn. De op te leveren systemen staan op chronologische volgorde. Aan de linkerkant wordt de huidige datum aangegeven met een rode lijn. Iedere dag zullen de op te leveren systemen dichterbij de rode lijn komen en is eenvoudig te zien hoeveel tijd er nog voor een oplevering is en wat de eerst volgende is.

Daarnaast is het mogelijk om op een op te leveren systeem te klikken. Boven de tijdlijn zal dan meer informatie verschijnen over het systeem. Dit gebeurt door middel van tekst in plaats van iconen zoals bij de tabel. De gebruiker krijgt met volledige zinnen de status van de oplevering te zien.



Figuur 23: Het ontwerp voor de tijdlijn

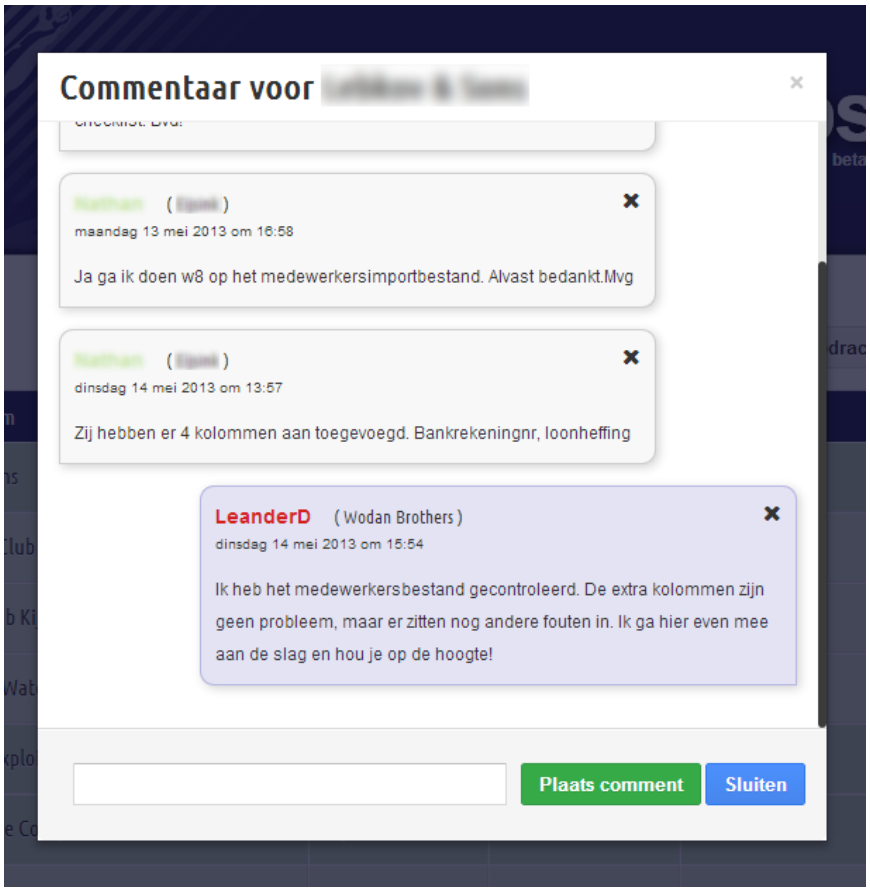


Figuur 24: Het uiteindelijke ontwerp van de tijdlijn

10.3 Commentaar

In de oude situatie ging alle communicatie met de resellers via de telefoon. Mochten er bepaalde taken nog niet uitgevoerd zijn, dan moest er gebeld worden om te vragen of dit alsnog gedaan kon worden. Hier ging veel tijd in zitten, zelfs als het om kleine dingen ging. In de nieuwe situatie zal er nog steeds gebeld moeten worden, maar dit zal een stuk minder vaak voorkomen dan dat eerst het geval was. Medewerkers van Wodan Brothers kunnen per opdracht commentaar achterlaten voor de resellers. De resellers kunnen zelf ook commentaar

achterlaten voor de Wodan Brothers. Het commentaar wordt live geüpdatet en ziet er uit als veel berichtenapplicaties die op smartphones te vinden zijn. Het zou dus ook kunnen werken als een soort van chat in het geval dat een reseller support nodig heeft.
Omdat de medewerkers van Wodan Brothers bijna al hun werk achter de pc doen, leek het mij handig om dit commentaar in te bouwen, het commentaar is namelijk niet voortgekomen uit een user story, maar gebouwd op eigen initiatief.
Ik heb ervoor gekozen om het plaatsen van commentaar mogelijk te maken omdat het typen van een kort berichtje beter past binnen de workflow van de medewerkers van Wodan Brothers. Dit is zeker het geval als het om iets eenvoudigs gaat, zoals een checklist markeren als voltooid of het toevoegen van een medewerkersbestand. Natuurlijk was het mogelijk om dit gewoon via mail te laten lopen, er kan dan ook een kort berichtje gestuurd worden. Maar nu blijft alles op de plaats waar het hoort: bij de opdracht. Het commentaar is te zien voor iedereen die de opdracht kan zien. Zo kan iedereen op de hoogte blijven van wat er is afgesproken.

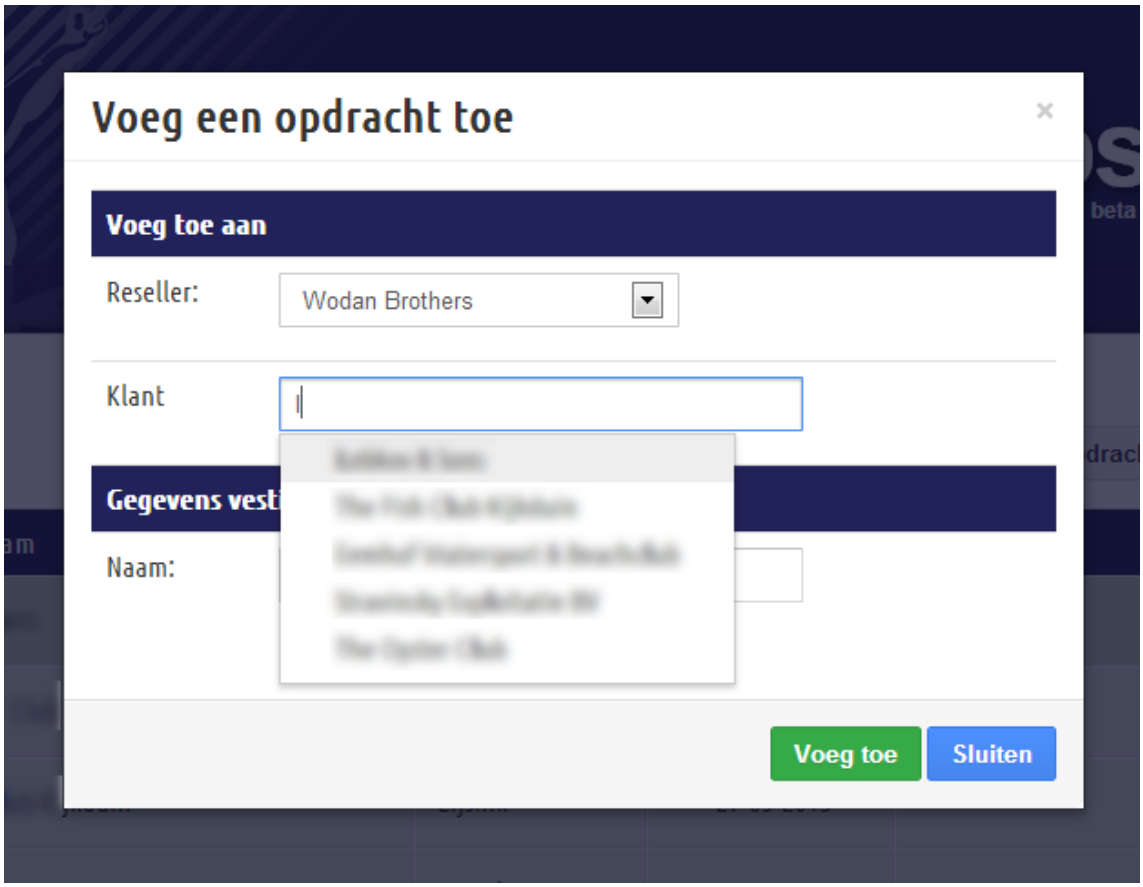


Figuur 25: Het commentaar bij een opdracht

10.4 Meerdere vestigingen per opdracht

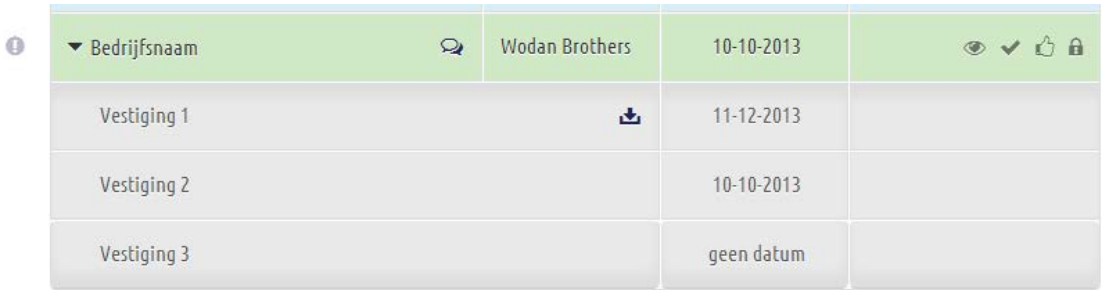
Het Dyflexis planningsysteem wordt het meest gebruikt in de horeca. Het komt vaak voor dat een klant meerdere horecagelegenheden heeft en hier moet in Chopsy rekening mee gehouden worden.

De checklist geldt nog steeds voor de klant, maar het invoeren van rex-o-matic serienummers en het uploaden van medewerkers bestanden gebeurt per vestiging.



Figuur 26: Het toevoegen van een opdracht

Bij het toevoegen van een opdracht is het dan ook mogelijk om de naam van een vestiging op te geven. Bij intypen van de naam van de klant geeft het systeem suggesties op basis van wat de gebruiker in heeft getypt. Bij klanten met meerdere vestigingen toont het systeem in het opdrachtenoverzicht de opties die voor een vestiging gelden bij de vestiging zelf in plaats van op de regel van de klant.



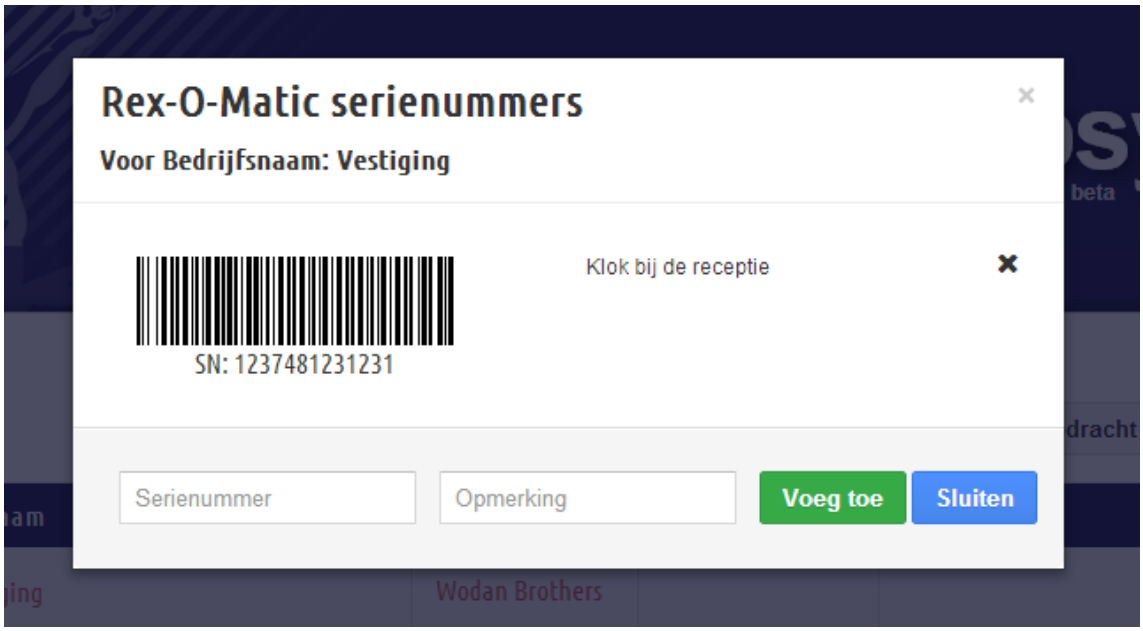
Figuur 27: Meerdere vestigingen per opdracht

10.5 Rex-O-Matic

Eerder hebt u kunnen lezen over de prikklokken die geleverd worden door Wodan Brothers. Deze prikklokken, met de naam Rex-O-Matic, zijn allemaal voorzien van een serienummer.

Figuur 28: Een Rex-O-Matic prikklok

Om bij te houden waar welke klok terecht is gekomen kunnen per vestiging serienummers toegevoegd worden. Daar kan ook een opmerking bij geplaatst worden als dat nodig is. Na het toevoegen wordt er ook direct een barcode gegenereerd. Dit is omdat er gebruik gemaakt wordt van een handscanner om de serienummers eenvoudig en snel over te nemen in verschillende documenten. Om het systeem zo gebruiksvriendelijk mogelijk te maken zijn de barcodes toegevoegd. De gebruikers kunnen nu blijven werken zoals zij gewend zijn.



Figuur 29: Het toevoegen van een serienummer aan een vestiging

Ook bij het het toevoegen van de serienummers kan de jQuery validatie plugin van pas komen. De validatie plugin heeft de mogelijkheid om de invoer van een gebruiker te controleren aan de hand van een functie. Als er een database bijgehouden zou worden met de serienummers van alle Rex-O-Matics die bij Wodan Brothers zijn binnen gekomen, dan zou die functie het ingevoerde serienummer kunnen controleren. De validatie plugin kan eenvoudig melden of het een geldig serienummer is.

10.6 Feedback

Ook tijdens de tweede sprint heb ik feedback gekregen. Net als bij de eerste sprint ging de feedback alleen over kleine punten. Kleine toevoegingen, aanpassingen e.d. Een aantal voorbeelden van feedback tijdens de tweede sprint:

- Het icoontje van het commentaar moet zichtbaar zijn als er comments zijn; verborgen als er geen comments zijn en knipperen als er nieuwe comments zijn
- Resellers mogen checklists wel afvinken als niet alle medewerkersbestanden zijn geüpload, maar er moet een waarschuwing komen
- Opdrachten moeten standaard gesorteerd worden op datum (oplopend)

- Indien er fouten zijn gemaakt op de checklist moet het toch mogelijk zijn deze op te slaan.

Het zijn dus geen grote aanpassingen die ik heb moeten doen. De feedback van de tweede sprint heb ik dan ook nog tijdens de tweede sprint kunnen afhandelen. Geen van de punten heb ik mee hoeven nemen naar de derde sprint.

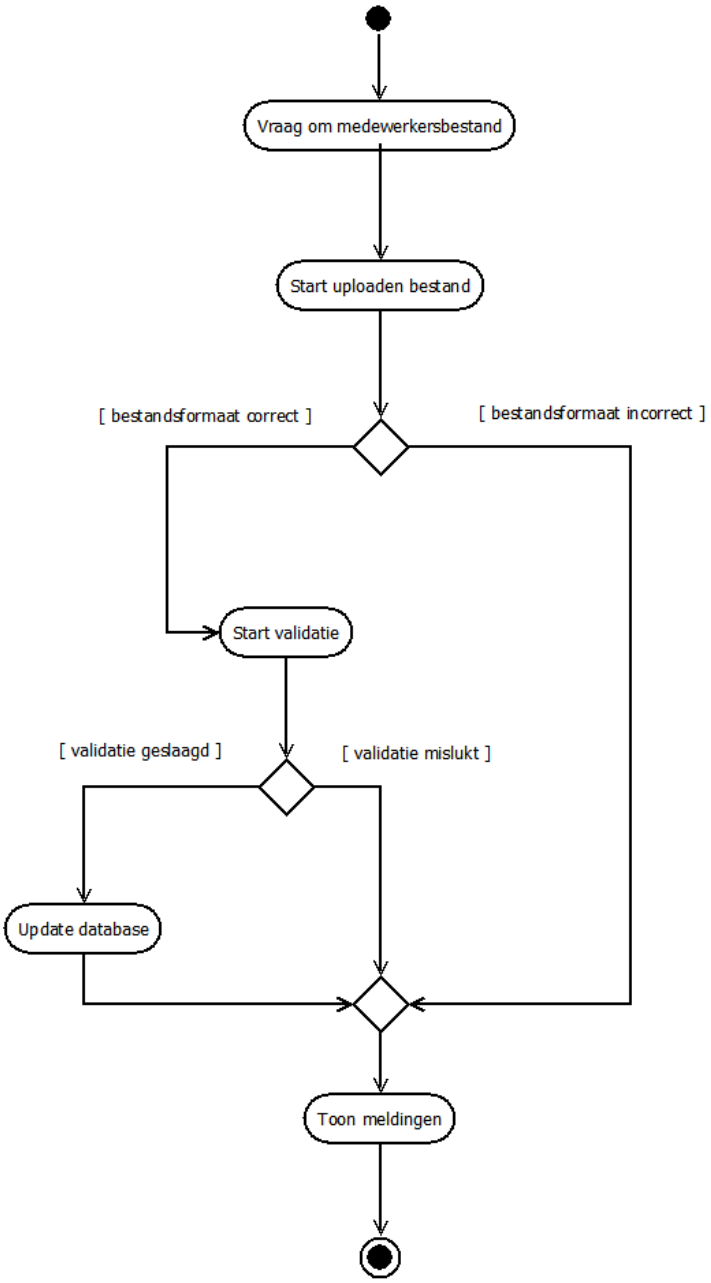
11. De derde sprint: 'de laatste fase'

Het doel van de derde en laatste sprint was om de eerste versie van het systeem af te ronden. Hiervoor was het nodig dat medewerkersbestanden gecontroleerd konden worden, moest er een koppeling gemaakt worden met het Dyflexis planningsysteem en moesten klanten een overzicht kunnen ontvangen van het door hen bestelde systeem. Daarnaast is nog een aantal toevoegingen gedaan om de gebruiksvriendelijkheid te verhogen.

11.1 Excel-file validator

Als in de oude situatie een papieren checklist volledig was ingevuld kon het systeem voor de klant klaar gezet worden. Bij dit klaarzetten is het ook mogelijk om de medewerkers te importeren die hun rooster in het Dyflexis systeem moeten kunnen zien. Het medewerkers bestand werd los, via e-mail aangeleverd bij Wodan Brothers. Een werknemer controleerde vervolgens het bestand en importeerde het in het systeem als het bestand correct was ingevuld. Aangezien nu alles via Chopsy gedaan wordt, scheelt het ook tijd als het medewerkersbestand via dit systeem aangeleverd kan worden. Net als de comments is het prettig als het medewerkersbestand altijd bij de opdracht blijft. Daarom kan er per vestiging een medewerkersbestand geüpload worden. Het was een optie geweest om het hierbij te laten. Het scheelt al tijd dat een medewerker van Wodan Brothers het medewerkersbestand niet meer in zijn mail hoeft te zoeken zodra hij het nodig heeft. Maar het kost veel tijd om het medewerkersbestand handmatig te controleren op fouten. Daarnaast moeten de fouten genoteerd worden, om vervolgens doorgegeven te worden aan de reseller. Die zal het bestand terug moeten sturen naar de klant om zo de fouten te laten herstellen. Zoals eerder besproken kan een reseller in de nieuwe situatie een bestand uploaden. Dit bestand wordt direct gecontroleerd door de Excel-file validator. Het bestand wordt gecontroleerd op verplichte velden en op foutieve invoer. Als het bestand gecontroleerd is krijgt de gebruiker een melding met de gemaakte fouten. Zo kan hij zien welke verplichte velden niet zijn ingevuld, maar hij kan ook zien als bepaalde velden onjuist zijn ingevuld. Het gaat dan onder andere om velden die een datum in een bepaald formaat verwachten, ongeldige e-mailadressen, ongeldige telefoonnummers of velden die maar enkele opties

hebben zoals man / vrouw, gehuwd / ongehuwd etc. Ook worden Burgerservicenummers gecontroleerd op geldigheid door middel van een elfproef. Omdat de gebruiker direct te zien krijgt welke fouten er gemaakt zijn hoeven medewerkers van Wodan Brothers hier niet bij betrokken te zijn. Zij hoeven de bestanden niet meer te controleren of de fouten door te geven. De resellers kunnen deze fouten nu zelf oplossen. Dit scheelt voor de Wodan Brothers en de resellers tijd, omdat het heen en weer sturen van e-mails niet meer nodig is.



Figuur 30: Activity diagram van de Excel-file validator

Het Activity-diagram dat u in **Figuur 30** kunt zien is gemaakt om de werking van de Excel-file validator voor mijzelf duidelijk te krijgen. Het heeft mij geholpen te bepalen welke tests nodig waren om de code voor de validator te kunnen schrijven.

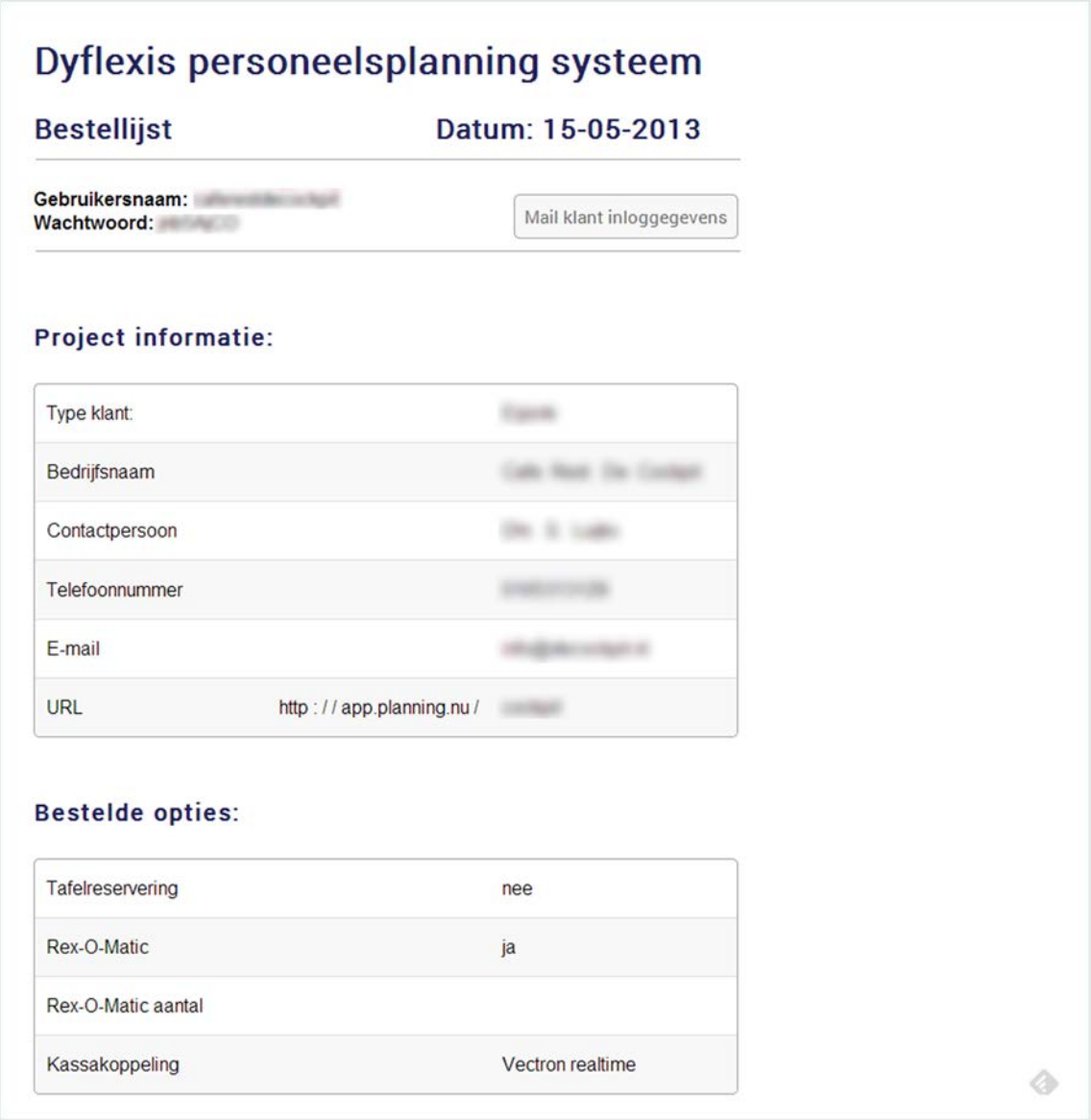
De Excel-file validator is, net als andere onderdelen van het systeem, volgens Test Driven Development geschreven.
In de validator kan het pad naar het te valideren bestand gezet worden.

Ik ben begonnen met het schrijven van een test die de bestandslocatie test als deze nog niet gezet is. Het pad naar het bestand zou NULL moeten zijn. Vervolgens heb ik de get-methode geschreven om de test te laten slagen. De test die ik daarna heb geschreven is een test die de bestandslocatie zet en vervolgens via de get-methode test of het pad correct is gezet. Om deze test te laten slagen heb ik de set-methode geschreven. Dit zijn uiteraard twee eenvoudige methoden, die eenvoudig volgens Test Driven Development te maken zijn.
Na het testen van deze twee methoden ben ik verder gegaan met de validatie-methode. Ik ben begonnen met een hele eenvoudige test, om vervolgens de code te schrijven die nodig was om deze test te laten slagen. Daarna heb ik bepaald wat het volgende was dat de validatie-methode zou moeten doen. Daar heb ik eerst een test voor geschreven en daarna pas de werkelijke code. Op deze manier heb ik mijn hele validatie-methode geschreven.
Ook de tests die ik heb geschreven voor de validator-class werden getest met het PHPUnit-framework. Doordat PHPUnit gebruik maakt van Code Coverage heb ik kunnen zien of alle te testen code ook daadwerkelijk getest werd, omdat Code Coverage bijhoudt welke code wel en niet is uitgevoerd tijdens het testen.

11.2 Overzicht van de configuratie

In deze sprint heb ik me ook bezig gehouden met de overzichten van de ingevulde checklists. Als een checklist als voltooid is gemarkeerd is het mogelijk om de ingevulde checklist te bekijken. In eerste instantie kreeg de gebruiker hetzelfde formulier te zien als wat was ingevuld, maar was het niet mogelijk de waarden nog aan te passen. Zoals eerder besproken is dit formulier geschikt om in te vullen, het is echter niet geschikt als overzichtspagina. Omdat de product owner graag wilde dat de klant een PDF zou krijgen met de bestelde configuratie heb ik

hiervoor een nieuwe pagina gemaakt. Deze pagina toont alle gekozen instellingen op één pagina. Het was de bedoeling om deze pagina om te zetten naar een PDF-bestand dat via e-mail naar de klant gestuurd zou kunnen worden. Omdat deze pagina een veel beter overzicht geeft van de ingevulde checklist dan het invulformulier, heb ik besloten deze pagina ook aan de gebruiker te tonen als hij een ingevulde checklist wil terugzien.



Dyflexis personeelsplanning systeem

Bestellijst Datum: 15-05-2013

Gebruikersnaam:
Wachtwoord:

Mail klant inloggegevens

Project informatie:

Type klant:	
Bedrijfsnaam	
Contactpersoon	
Telefoonnummer	
E-mail	
URL	http : / / app.planning.nu /

Bestelde opties:

Tafelreservering	nee
Rex-O-Matic	ja
Rex-O-Matic aantal	
Kassakoppeling	Vectron realtime

Figuur 31: Een deel van de overzichtspagina van de bestelde configuratie

Het genereren van een PDF-bestand op basis van het overzicht was geen enkel probleem. Wodan Brothers maakt gebruik van een betaalde dienst die HTML om kan zetten naar PDF. Het toevoegen van de code die de pagina converteerde naar PDF was erg eenvoudig. Echter was er geen controle over de plaatsen waar het

overzicht opgedeeld werd in verschillende pagina's binnen het PDF-bestand. Het kwam dus voor dat een tabel in twee delen werd gesplitst, wat er niet netjes uitzag. Om deze reden is ervoor gekozen om geen PDF naar de klanten te sturen, maar om de klant inloggegevens te geven voor alleen de overzichtspagina. Als een gebruiker in Chopsy is ingelogd kan hij elke overzichtspagina bekijken. Een klant kan met zijn inloggegevens alleen de pagina zien met de configuratie die hij besteld heeft.

11.3 Koppeling Dyflexis

Een groot gedeelte van het klaarzetten van systemen is al geautomatiseerd. Het kan echter nog een stuk beter. In de oude situatie was het nog nodig om gegevens van de checklist over te typen en hier ging behoorlijk wat tijd in zitten. Nu de checklist digitaal is, is het mogelijk om de gegevens die het planningsysteem Dyflexis nodig heeft automatisch over te zetten. In eerste instantie was het de bedoeling dat Chopsy hiervoor zou zorgen. Het checklistsysteem zou dan scripts van Dyflexis aanroepen, om vervolgens de benodigde gegevens in te voeren. In dat geval had Chopsy veel moeten weten van hoe Dyflexis werkt. Om veiligheidsredenen is ervoor gekozen om dit niet te doen. Nu is er een special script voor Dyflexis gemaakt. Het planningsysteem kan dit script aanroepen en krijgt direct de gegevens die hij nodig heeft. In PHP is het mogelijk om gebruik te maken van HTTP Basic Authentication en HTTP Digest Authentication. Het script voor Dyflexis is afschermt door middel van HTTP Digest Authentication. Deze variant maakt gebruik van MD5-hashes en is de veiligste van de twee methoden.
Er is voor HTTP Authentication gekozen omdat er dan via de URL van het script ingelogd kan worden. Dyflexis kan inloggen op het script door gebruikersnaam en wachtwoord in de URL mee te geven:

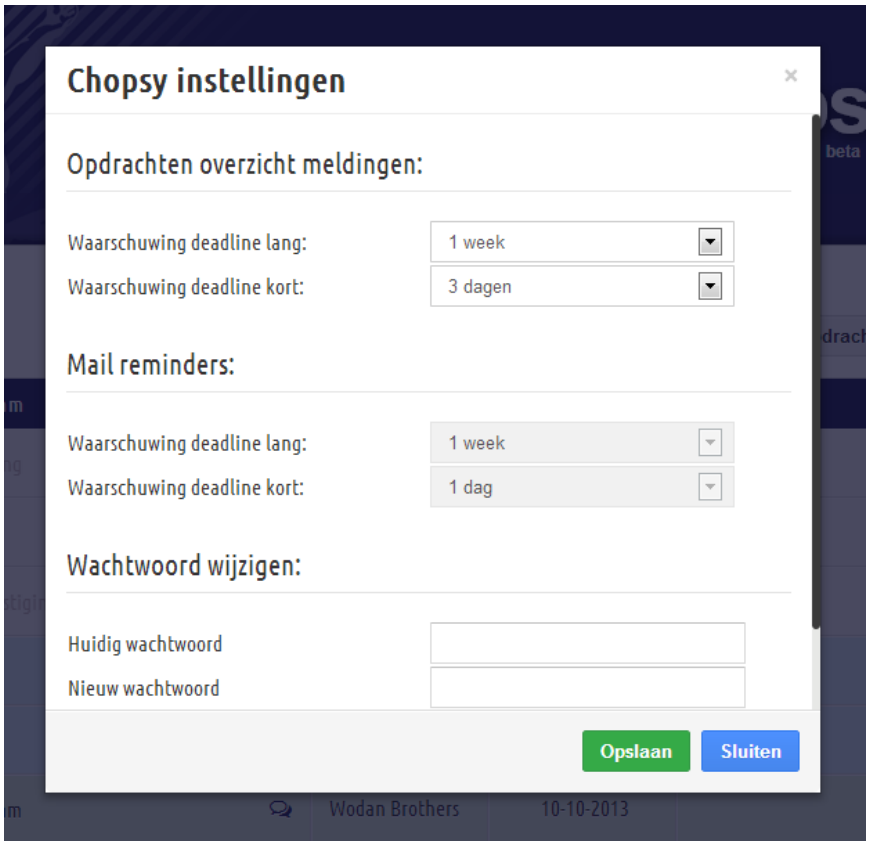
> *http://gebruikersnaam:wachtwoord@url.nl/*

Om systemen geautomatiseerd klaar te kunnen zetten is het erg handig dat Dyflexis ook geautomatiseerd kan inloggen. Op deze manier kan een medewerker van Wodan Brothers een systeem klaarzetten door te klikken op een knop en

hoeft hij niet zelf nog in te loggen. Toch is het script van Chopsy met de configuraties van bestelde systemen beveiligd.

11.4 Instellingen

Om het systeem nog iets gebruiksvriendelijker te maken heb ik een scherm met instellingen toegevoegd. De gebruiker kan hier opgeven op welk moment hij gewaarschuwd wil worden voor een oplevering. Er zijn aparte instellingen voor waarschuwingen binnen het systeem en voor waarschuwingen via e-mail. Mocht een gebruiker zelf niets instellen, dan worden er standaard instellingen gebruikt. Naast het instellen van de waarschuwingen is het mogelijk voor de gebruiker om zijn wachtwoord te wijzigen.



Figuur 32: Het instellingenscherm van Chopsy

11.5 Feedback

Bij de derde sprint kreeg ik meer feedback en ook grotere punten. In het begin heb ik één papieren checklist gekregen. Deze checklist heb ik in de eerste sprint

omgezet naar een digitale versie. Pas halverwege de derde sprint kreeg ik te horen dat er nog een andere checklist was. Verschillende resellers moesten een andere checklist te zien krijgen. Dit is iets waar ik van tevoren niet echt rekening mee had gehouden, omdat ik niet wist dat er meerdere checklists waren. Omdat ik het opslaan van de eerste checklist zo flexibel had gemaakt was het een stuk eenvoudiger om een tweede checklist toe te voegen.

Zoals u in hoofdstuk 11.2 hebt kunnen lezen was het de bedoeling dat de klant een PDF-bestand zou ontvangen met daarin de bestelde configuratie. De product owner heeft besloten dat het beter was om de gebruiker een gebruikersnaam en wachtwoord te geven voor de overzichtspagina. De commercieel directeur vond het niet nuttig, maar er is voor gekozen om deze functionaliteit beschikbaar te houden.

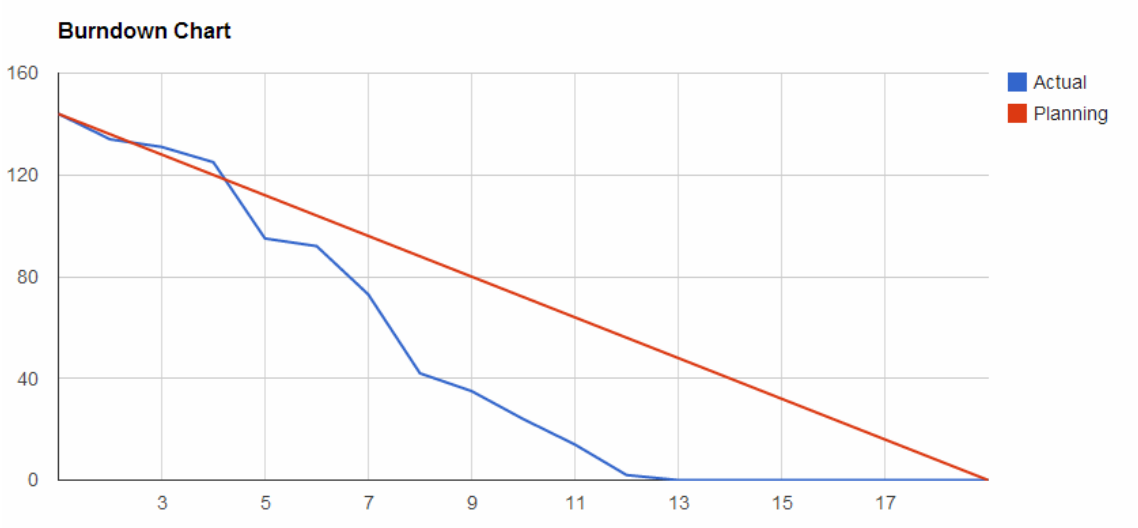
De andere punten die ik uit de feedback heb gekregen waren voornamelijk kleine punten. Er werd ook gevraagd om nieuwe functionaliteit, maar dit paste niet meer in de planning en is om die reden niet toegevoegd.

12. Evaluatie gebruikte aanpak

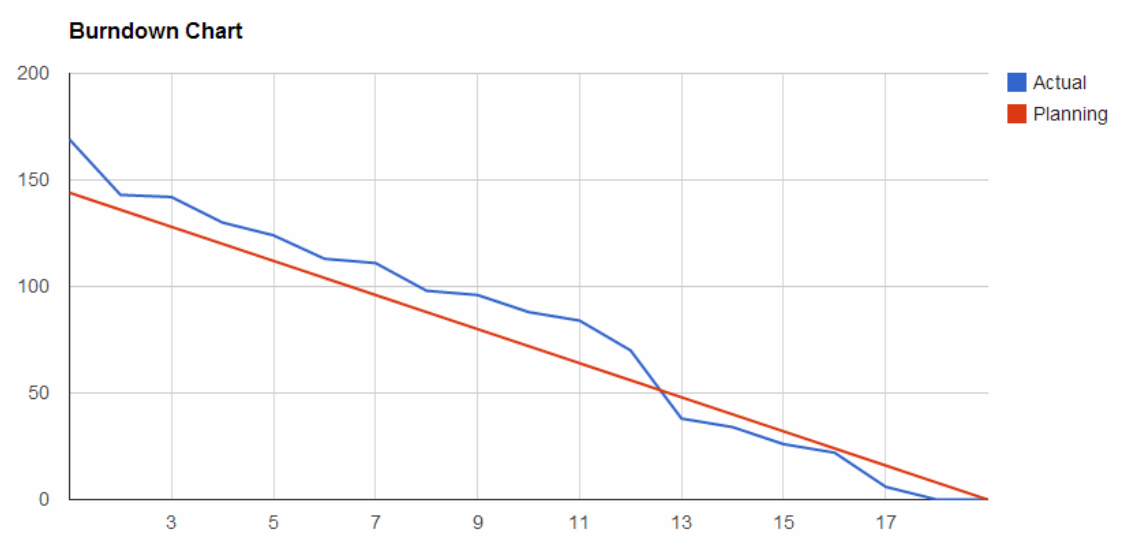
In dit hoofdstuk bespreek ik de aanpak die ik heb gebruikt tijdens mijn afstudeerproject en licht ik toe wat er wel en niet goed is gegaan.

12.1 SCRUM

Voor ik begon aan het project heb ik besloten gebruik te maken van SCRUM. Het leek mij een geschikt methode voor dit project, omdat ik een volledig werkend product wilde opleveren. Bij een andere softwareontwikkelingsmethode is dit ook mogelijk, maar had ik meer documentatie gehad en een kleiner product. Ik ben redelijk tevreden met mijn keuze. SCRUM is – met enkele aanpassingen – ook geschikt voor één persoon. Ik had op elk moment een duidelijk overzicht van mijn taken en of ik nog volgens planning aan het werk was. De planning kan echter wel beter. Vaak is het zo dat mensen te krap plannen en daardoor in tijdnood komen. Bij mij was het vaak zo dat ik te ruim had gepland en dus minder tijd nodig had dan gedacht. Vooral bij de eerste sprint was ik veel eerder klaar dan gedacht.

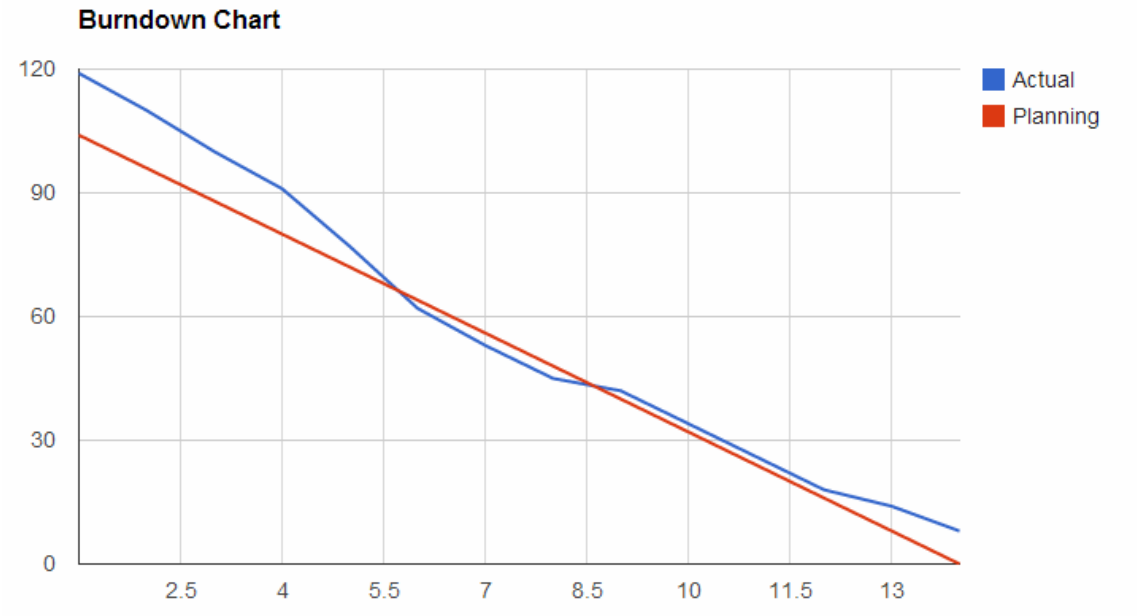


Figuur 33: Planning en gewerkte uren van de eerste sprint



Figuur 34: Planning en gewerkte uren van de tweede sprint

Bij de tweede sprint klopte mijn planning al veel beter – de lijn van de werkelijke uren volgde de lijn van de geplande uren beter dan bij de eerste sprint. Ook bij de derde sprint was het probleem van te ruim plannen veel minder aanwezig. Bij de derde sprint leek het eerst dat ik te krap had gepland, daarna kwam ik toch weer onder de planning. Maar doordat ik extra werk kreeg heb ik de planning uiteindelijk toch niet kunnen halen.



Figuur 35: Planning en gewerkte uren van de derde sprint

Zoals u hebt kunnen zien klopte mijn planning voor de eerste sprint niet. Ik had te weinig taken voor de gehele sprint ingepland. Bij de volgende sprints klopte mijn planning beter. Toch heb ik gemerkt dat mijn planning nog beter kan.

RS1	digitale versie maken van de checklist
	database koppeling programmeren
	gegevens validatie programmeren
	checklist validatie programmeren (frontend)

Figuur 36: Taken behorende bij een user story voor de checklist

De taken die ik heb opgesteld voor de user stories – zie voor een voorbeeld **Figuur 35** – zijn niet specifiek. Deze taken kunnen nog verder opgesplitst worden. Het opdelen van de taken in nog kleinere, specifiekere taken zou mij erg geholpen hebben bij het plannen van dit project.

Ik heb het werken met SCRUM als erg prettig ervaren. Wel vraag ik mij af of het echt geschikt is voor één persoon. Het Kanban Board, de Burndown Charts en de User Stories vond ik prettig. Toch merkte in gedurende het project dat ik een langere ontwerpfase had moeten plannen. Toen ik begon aan de eerste sprint had ik het idee dat ik genoeg informatie had om aan het project te beginnen. Ook tijdens de eerste twee sprints leek dat nog het geval te zijn. Ergens rond 70% van het project kwamen er nieuwe taken bij en dit paste eigenlijk niet in de planning. Dit had waarschijnlijk voorkomen kunnen worden door een langere ontwerpfase, waarbij de requirements beter uitgezocht worden. Bij een volgend project zou ik misschien een andere softwareontwikkelingsmethode gebruiken of meer tijd nemen voor het ontwerpen van het systeem.

De reden voor mijn keuze voor SCRUM is geweest dat ik bij een gesprek voor mijn afstuderen te horen kreeg dat ze bij Wodan Brothers met SCRUM werken. Omdat ik tijdens mijn studie SCRUM al interessant vond, leek mij dit een mooie kans om er meer van te weten te komen. Helaas heb ik bij Wodan Brothers niets van SCRUM gezien. Mijn manier van werken met SCRUM is daarom ook een manier die samengesteld is uit verschillende bronnen op Internet. Dat neemt niet weg dat ik SCRUM nog altijd interessant vind en er graag beter mee zou willen leren werken. (Krug, 2006)

12.2 Test Driven Development

Het was voor mij de eerste keer dat ik werkte volgens Test Driven Development. Maar het is een ontwikkelmethode die ik zeer waarschijnlijk zal blijven gebruiken. Ik vond het erg prettig om op deze manier te werken, omdat het ervoor zorgt dat je op een andere manier nadenkt over je code. Het zorgt er ook voor dat je alleen code schrijft die echt nodig is om iets werkend te krijgen en deze code slimmer schrijft dan zonder TDD. Daarnaast is het hebben van unit testen ook erg prettig, omdat je op elk moment kunt kijken of alle code nog naar behoren functioneert. Net als met SCRUM moet ik het werken met TDD nog beter onder de knie krijgen, maar dat is iets dat ik zeker van plan ben. Ik vond het lastig om te bepalen wat ik wel of niet volgens TDD moest ontwikkelen. Dit heeft ook te maken met het gebruik van verschillende programmeertalen. Vooral in JavaScript / jQuery had ik vaak het idee dat ik code niet volgens TDD kon schrijven. Het gaat dan om code dat een popup scherm toont of een ander element op de pagina verbergt / toont. Daarnaast vond ik de *refactoring* (het herschrijven van mijn code) vaak ook lastig. In veel gevallen had ik het idee dat mijn code niet verder opgeschoond kon worden, omdat ik alleen de code had geschreven waarvan ik vond dat het echt nodig was om de test te laten slagen. In veel gevallen hield het opschonen van mijn code in dat ik kleine aanpassingen deed. Bijvoorbeeld de return-value van een functie niet eerst opslaan in een variabele, maar direct in een return-statement zetten. Naarmate het project vorderde hoefde ik dit ook minder te doen, omdat ik hier tijdens het schrijven van de code al rekening mee hield.

Het werken volgens TDD heeft er bij mij voor gezorgd dat ik nog beter ben gaan inzien hoe belangrijk het testen van code is en dat in de meeste gevallen de code die geschreven wordt netter en duidelijker is.

12.3 Project of verslag

Tijdens mijn afstuderen heb ik gemerkt dat het toch lastig kan zijn om je aan een planning te houden. Ik heb al eerder gemerkt dat opdrachtgevers altijd meer willen dan van tevoren afgesproken. Dat was bij dit project ook het geval. Tijdens dit project voelde ik toch wel druk om voornamelijk met het project bezig te zijn en verloor daardoor af en toe mijn afstuderen uit het oog. Doordat ik fulltime

voor hen aan het werk was vergat ik soms dat ik daar ook zat voor mijn afstuderen. Wodan Brothers wil graag een product dat af is en waar zij mee kunnen werken. Af en toe lag mijn focus daardoor te veel op het product en had ik het idee dat ik niet echt de ruimte had om rustig aan mijn verslagen te werken. Dit komt misschien ook doordat ik de eerste afstudeerder ben bij Wodan Brothers. Maar dit had ik beter in gaten moeten houden en ik had mijn focus beter moeten verdelen. Ik vind dat ik te vaak 'ja' heb gezegd tegen nieuwe functionaliteit op momenten dat ik dat waarschijnlijk niet had moeten doen. In het Plan van Aanpak in dit document schreef ik dat het mogelijk was om de product owner tegemoet te komen tijdens de bouw, zonder daarbij de planning in gevaar te brengen. Daar sta ik nog steeds achter. Ik schreef ook dat er wel grenzen zijn aan de hoeveelheid extra werk of aanpassingen. Deze grens heb ik toch enigszins overschreden.

13. Evaluatie

13.1 De verwachtingen

Voor ik met het project begon had ik een aantal verwachtingen. Eén daarvan was dat ik bij Wodan Brothers veel zou kunnen leren. Niet alleen over het maken van software, maar ook over runnen van een jong bedrijf. Daarnaast verwachtte en hoopte ik dat ik genoeg uitdagingen tegen zou komen. De grootste uitdagingen bij het bouwen van het systeem waren voor mij het maken van de validatie-plugin, het maken van de Excel-file validator en de koppeling met Dyflexis. Ook vond ik het werken met verschillende stakeholders op sommige momenten een uitdaging, omdat ze allemaal hun eigen wensen hadden en die kwamen niet altijd overeen. Ik heb veel geleerd van deze uitdagingen. Het maken van een jQuery-plugin was nieuw voor mij en ook het verwerken van Excelbestanden was nieuw voor mij. Maar wat ik ook geleerd heb is dat ik de uitdagingen nodig heb. Op de momenten dat ik bezig was met taken die niet uitdagend genoeg waren, vond ik het moeilijk om me te blijven concentreren. Het kwam voor dat ik op zoek ging naar taken die wel uitdagend waren. Naast dat ik graag uitdagend werk heb, wil ik dat het afwisselend is. Dat is ook een reden waarom ik graag een eigen bedrijf zou willen hebben. Daarbij zal ik naast het maken van websites / software me ook bezighouden met het bedrijf zelf. Dat houdt het voor mij interessant, afwisselend en uitdagend. Dat Wodan Brothers een jong bedrijf is, was voor mij ook een reden om te kiezen voor deze afstudeeropdracht. Ik heb goed gelet op wat ik wel en niet zou doen als ik een eigen bedrijf zou hebben. Toch heb ik hier minder over geleerd dan ik van tevoren hoopte. Achteraf gezien waren mijn verwachtingen hiervan iets te hoog.

13.2 Het product

Met het online checklistsysteem ben ik zeker tevreden. Op bepaalde punten zou het product beter kunnen. In hoofdstuk 6 heb ik beschreven dat usability voor mij erg belangrijk is. Toch heb ik gemerkt dat het lastig is om het product voor iedereen gebruiksvriendelijk te maken. Sommige dingen waren voor mij erg logisch. Voor mij waren de icoontjes erg duidelijk of waren er logische kleuren gebruikt. Het bleek niet voor iedereen even duidelijk te zijn. De gedachte achter

veel van de dingen die ik heb gedaan om de usability te verbeteren is wat Steve Krug zegt in zijn boek (Krug, 2006) en dat is dat een gebruiker niet zou moeten nadenken over wat te doen. Dat is echter niet eenvoudig, omdat ik het product gemaakt heb. Ik hoef dus niet meer na te denken over waar ik moet klikken of hoe iets werkt. Daardoor zie je snel dingen over het hoofd, die toch voor de gebruiker duidelijker of eenvoudiger kunnen.

Daarnaast vind ik dat op bepaalde plekken mijn code netter en beter had gekund. Het werken met Test Driven Development heeft er wel voor gezorgd dat ik anders ben gaan programmeren. De code die volgens TDD is geschreven is duidelijk en netjes. Ook door te 'refactoren'. Niet alles code kon echter volgens TDD geschreven worden. Vooral de aanvullende jQuery code kan netter en meer gestructureerd.

Met de jQuery-validatie plugin en de Excel-file validator ben ik erg tevreden. Ze zijn volgens TDD geschreven, de code is duidelijk en beiden zijn flexibel. Dat betekent dat ze niet alleen geschikt zijn voor dit project, maar ook gebruik kunnen worden voor andere projecten.

Verder vind ik het prettig dat ik een compleet product heb kunnen afleveren, dat ook daadwerkelijk gebruikt wordt. Op het moment van schrijven is niet alle feedback verwerkt. Met de product owner is afgesproken dat dit nog gedaan zal worden en verwacht wordt dat dat lukt voor het einde van mijn afstuderen.

13.3 De doelstellingen

Het product is compleet en bevat de wensen van de verschillende stakeholders. Aan het begin van dit verslag heb ik de doelstellingen van de opdracht besproken. Het online checklistsysteem moest ervoor zorgen dat het invullen van de checklists eenvoudiger zou worden voor de gebruikers. Deze doelstelling is zeker behaald. Een gebruiker krijgt tijdens het invullen van de checklist feedback van de validatie-plugin, zodat altijd duidelijk is wat er moet gebeuren. Daarbij kan het systeem opties tonen of verbergen bij bepaalde keuzes. Sommige instellingen zijn namelijk niet altijd beschikbaar en deze hoeven in dat geval niet ingevuld te worden in de checklist. Voor de gebruiker is het invullen van de checklist daardoor duidelijker en eenvoudiger. Doordat het systeem de gebruiker

feedback geeft over gemaakte fout is het probleem dat verplichte velden niet ingevuld werden ook opgelost.

Een tweede doelstelling was dat het systeem de bestelde configuratie over zou kunnen zetten naar een ander systeem, het Dyflexis planningsysteem. Deze doelstelling is behaald, maar anders opgelost dan in eerste instantie was bedacht. Het idee was dat Chopsy, het online checklistsysteem, de configuratie zou aanbieden aan Dyflexis. Het initiatief zou dan bij Chopsy vandaan komen en Dyflexis zou hierop reageren. De hoofdprogrammeur van Wodan Brothers vond het uiteindelijk toch prettiger en veiliger als dat op een andere manier gedaan zou worden. Nu komt het initiatief bij Dyflexis vandaan. Deze vraagt aan Chopsy welke configuraties er overgezet moeten worden. Het online checklistsysteem zal deze vervolgens aan Dyflexis aanbieden. Daar zal het planningsysteem wel (automatisch) voor moeten inloggen. Met het behalen van deze doelstelling is het probleem met de foutgevoeligheid van het overtypen van checklists ook opgelost.

Het waarschuwen van een gebruiker als er een oplevering aankomt, was ook een doelstelling van de opdracht. De medewerkers van Wodan Brothers hoeven daardoor niet zelf de opleveringen in de gaten te houden. Het systeem waarschuwt een gebruiker als hij is ingelogd, door het laten knipperen van de opdrachten die aandacht nodig hebben. Maar het systeem waarschuwt de gebruikers ook via e-mail voor een oplevering.

Het controleren van de Excel-files met gegevens van medewerkers, het medewerkersbestand, was een tijdrovende klus. Een andere doelstelling van de opdracht was daarom dat het systeem de bestanden zou kunnen controleren en aan zou kunnen geven welke fouten er gemaakt zijn. Als een gebruiker een Excel-file upload, zal het direct gecontroleerd worden. Als het bestand geen fouten bevat, zal het bestand geaccepteerd worden. Is dit niet het geval dan wordt het bestand afgekeurd en de gebruiker krijgt een lijst met gemaakte fouten te zien. Omdat de gebruiker in veel gevallen een reseller is, krijgt hij direct de fouten te zien. Wodan Brothers hoeft het bestand niet te controleren en hoeft ook de fouten niet door te geven. Het is voor beide partijen sneller en eenvoudiger.

In de oude situatie had de klant geen overzicht van de bestelde configuratie. Het online checklistsysteem zou dit moeten oplossen. Als de checklist ingevuld en

geaccepteerd was, dan zou de klant een e-mail krijgen met een PDF-bestand met de bestelde configuratie. De klant ontvangt wel een overzicht van de bestelde configuratie, maar dit gebeurt niet via een PDF-bestand. Hij krijgt via e-mail inloggegevens voor de pagina met de configuratie voor zijn planningsysteem.

In hoofdstuk 2.2 staat een aantal problemen beschreven. De doelstellingen die hierboven besproken zijn lossen een aantal van deze problemen op. Een aantal problemen is echter nog niet besproken. In de oude situatie was het mogelijk dat checklists kwijtraakten. Door het verwijderen van checklists binnen het systeem te voorkomen is het niet meer mogelijk dat checklists kwijtraken. Ook is het niet mogelijk dat resellers verschillende versies van een checklist gebruiken, omdat iedereen altijd de laatste versie heeft via updates die online doorgevoerd worden. Een laatste probleem is dat er aanpassingen buiten de checklist om gedaan werden. Dit is een probleem dat helaas niet opgelost kan worden met het systeem. Resellers kunnen er nog steeds voor kiezen een aanpassing via de telefoon door te geven en dit niet op te geven in de checklist.

13.4 De praktijktest

In hoofdstuk 8 hebt u kunnen lezen dat het checklistsysteem halverwege het project online is gezet om in de praktijk getest te worden. Na een demonstratie van het systeem op het kantoor van Wodan Brothers heeft een reseller van het planningsysteem de overstap gemaakt van de papieren checklists. Na afloop van de demonstratie vertelde hij mij dat hij erg enthousiast was over het systeem en ook later heb ik dit meerdere malen gehoord.
Tijdens het werken met het systeem – dat nog in ontwikkeling was – heeft hij mij feedback gegeven. Soms wilde hij graag extra functionaliteit, soms wilde hij aanpassingen. Ook heeft hij geholpen bij het opsporen van bugs. Voor mij is het testen in de praktijk erg nuttig geweest. Op deze manier heb ik ook feedback kunnen krijgen van iemand die met het systeem moet werken. Zoals eerder gezegd heeft het testen in de praktijk de bouw van het systeem en het opstellen van het gegevensmodel van de database beïnvloed. Ik had in mijn planning geen rekening gehouden met de praktijktest. Wel had ik rekening gehouden met het verwerken van feedback en daar paste dit prima in. De feedback heb ik verwerkt in het systeem en als het nodig was ook in het gegevensmodel van de database.

13.5 De beroepstaken

1.4 Uitvoeren analyse door definitie van requirements

Voor deze beroepstaak heb ik requirements opgesteld van de product owner. Daarnaast heb ik gesprekken gehad met verschillende stakeholders die allemaal hun eigen wensen hadden. Deze wensen spraken elkaar soms tegen, maar door dit te bespreken met de andere stakeholders was er altijd een oplossing te vinden. De wensen van de stakeholders zijn verwerkt in user stories. Bij deze user stories heb ik use cases met use case beschrijvingen gemaakt die u terug kunt vinden in het 'User Stories Document'. Voor ik begon met het project had ik verwacht dat ik meer bezig zou zijn met het opstellen van de requirements. Uiteindelijk zijn alle wensen van de verschillende stakeholders verwerkt in het product, maar vaak is dit gedaan doordat tijdens de bouw van het systeem de stakeholders met nieuwe wensen kwamen.

2.1 Opstellen gegevensmodel voor database

Het gegevensmodel van de database en het ontstaan daarvan kunt u terugvinden in hoofdstuk 8. Het gegevensmodel voor de database van het online checklistsysteem is minder uitgebreid dan ik had verwacht voor ik aan de opdracht begon. Dit komt ook doordat ik gebruik heb gemaakt van geserialiseerde data, om het systeem zo flexibel mogelijk te houden. Het gegevensmodel bevat binaire relaties, één op veel verbanden en één op één verbanden. Unaire relaties en veel op veel verbanden zijn niet toegepast.

2.2 Ontwerpen, bouwen en bevragen van een database

Om aan deze beroepstaak te voldoen heb ik een database ontworpen en gebouwd. Het bevragen van de database wordt door één gebruiker gedaan. Deze gebruiker is het online checklistsysteem. Zoals u in hoofdstuk 8 hebt kunnen zien is het nodig om meerdere tabellen te raadplegen om een query te beantwoorden. De integriteit van de database wordt gewaarborgd door het gebruik van 'foreign keys'. Hoe de 'foreign keys' toegepast zijn in de database kunt u terugvinden in het 'Database Design Document'.

3.2 Ontwerpen systeemdeel

Bij het ontwerp van de validator-plugin en de Excel-file validator heb ik rekening gehouden met mogelijk hergebruik. Beiden zijn niet alleen geschikt voor dit project, maar kunnen ook gebruikt worden in andere projecten. Bij het bouwen van deze delen van het systeem is ook rekening gehouden met toekomstige wijzigingen. Bij de validatie-plugin is het mogelijk de error-functie, de succes-functie en de lexer-functie via de façade te overschrijven. Zo blijft de plugin ook bruikbaar als er veranderingen aangebracht moeten worden in hoe de plugin fouten aangeeft of hoe de regels voor de plugin worden opgesteld.
Ook bij de Excel-file validator heb ik rekening gehouden met wijzigingen in de template van het medewerkersbestand. Alle opties voor hoe een file gevalideerd moet worden staat bovenaan in de klasse. Daar kan opgegeven worden welke velden verplicht zijn en wat welk veld zou moeten bevatten (enumeratie, telefoonnummer, e-mailadres etc.). Daarbij kan de programmeur zelf Regular Expressions opgeven om de velden aan te toetsen. Ook is het mogelijk om de 'mime types' van geaccepteerde bestanden te wijzigen. Het is mogelijk om bestanden te accepteren op de bestandsextensie, maar deze kan eenvoudig aangepast worden. Het aanpassen van de 'mime types' is mogelijk gemaakt omdat Excel-files van verschillende Office versies ook verschillende 'mime types' hebben. In de toekomst zou dit weer kunnen wijzigen. Meer over de validatie-plugin kunt u lezen in hoofdstuk 9.2 en meer informatie over de Excel-file validator kunt vinden in hoofdstuk 11.1.

3.3 Bouwen applicatie

Om het online checklistsysteem 'Chopsy' te maken heb ik gebruik gemaakt van een framework dat gebouwd is door de hoofdprogrammeur van Wodan Brothers. Hierover hebt u kunnen lezen in hoofdstuk 5 en hoofdstuk 7. Omdat het klaarzetten van systemen veel tijd kostte, heb ik het online checklistsysteem gekoppeld aan het planningsysteem. Meer hierover hebt u kunnen lezen in hoofdstuk 11.3.
Het bouwen van het systeem is gedaan in Sublime Text 2 en voor het testen is gebruik gemaakt van PHPUnit met Code Coverage voor het testen van PHP-code en QUnit voor het testen van JavaScript code.

Geraadpleegde literatuur

Bcrypt. (sd). Opgehaald van SourceForge: http://bcrypt.sourceforge.net/

Carr, R. (2009, 08 22). *Gang of Four Design Patterns*. Opgehaald van Black Wasp: http://www.blackwasp.co.uk/GofPatterns.aspx

Cohn, M. (2008, 04 25). *Advantages of the "as a user, I want" user story template*. Opgehaald van Mountain Goat Software: http://scrummethodology.com/scrum-user-stories/

Derksen, E. (2010, 03 29). *SCRUM en Exploratory Testing*. Opgehaald van De Nieuwe Tester: http://nieuwetester.blogspot.nl/2010/03/scrum-en-exploratory-testing.html

Gervasio, A. (2012, 07 30). *An Introduction to the Front Controller pattern*. Opgehaald van PHP Master: http://phpmaster.com/front-controller-pattern-1/

Goldstein, I. (2012, 07 03). *Testing in SCRUM - We still love testers*. Opgehaald van Scrum Shortcuts: http://www.scrumshortcuts.com/blog/quality-testing/testing-in-scrum-still-love-testers/

Guzel, B. (2011, 01 27). *Advanced Regular Expression Tips and Techniques*. Opgehaald van NetTuts+: http://net.tutsplus.com/tutorials/php/advanced-regular-expression-tips-and-techniques/

Guzel, B. (2013, 6 26). *Understanding Hash Functions and Keeping Passwords Safe*. Opgehaald van NetTuts+: http://net.tutsplus.com/tutorials/php/understanding-hash-functions-and-keeping-passwords-safe/

Hemert, V. v. (sd). *Een pragmatische aanpak van testen binnen SCRUM*. Opgehaald van Softwaretesten.net: http://www.softwaretesten.net/agile-scrum/scrum-ervaringen-uit-de-praktijk-2/scrum-ervaringen-uit-de-praktijk/

Huang, G. (2010, 02 03). *How to Test your JavaScript Code with QUnit*. Opgehaald van NetTuts+: http://net.tutsplus.com/tutorials/javascript-ajax/how-to-test-your-javascript-code-with-qunit/

Krug, S. (2006). *Don't Make Me Think.* New Riders.

Methodology, S. (2009, 10 24). *Scrum user stories*. Opgehaald van Scrum Methodology: http://scrummethodology.com/scrum-user-stories/

Osmani, A. (2012). *Learning JavaScript Design Patterns*. Opgehaald van Addy Osmani: http://addyosmani.com/resources/essentialjsdesignpatterns/book/#prototypepatternjavascript

Santos, S. (2008, 04 20). *.htaccess - gzip and cache your site for faster loading and bandwith saving*. Opgehaald van Samaxes: http://www.samaxes.com/2008/04/htaccess-gzip-and-cache-your-site-for-faster-loading-and-bandwidth-saving/

Ward, D. (2013, 01 11). *Adding your own callbacks to existing Javascript functions*. Opgehaald van Encosia: http://encosia.com/adding-your-own-callbacks-to-existing-javascript-functions/

Way, J. (2012, 10). *Test-Driven PHP*. Opgehaald van NetTuts+: http://net.tutsplus.com/sessions/test-driven-php/

Zaefferer, J. (2012, 06 27). *Introduction To JavaScript Unit Testing*. Opgehaald van Smashing Magazine: http://coding.smashingmagazine.com/2012/06/27/introduction-to-javascript-unit-testing/

Afkortingenlijst

Zie document 'Verklarende woordenlijst en gebruikte afkortingen'.