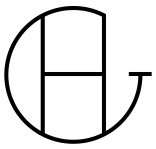


HoGent

Faculteit Bedrijf en Organisatie

# De selectie van JavaScript gebaseerde functionele test tools voor een KMO gespecialiseerd in Drupal

Maarten De Roover

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Steven Vermeulen
Co-promotor:
Nick Veenhof

Instelling: Dropsolid

Academiejaar: 2017-2018

Derde examenperiode

Faculteit Bedrijf en Organisatie

# De selectie van JavaScript gebaseerde functionele test tools voor een KMO gespecialiseerd in Drupal

Maarten De Roover

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Steven Vermeulen
Co-promotor:
Nick Veenhof

Instelling: Dropsolid

Academiejaar: 2017-2018

Derde examenperiode

# Woord vooraf

Deze bachelorproef is geschreven in opdracht van Dropsolid, het bedrijf waar ik stage en vakantiejob heb gedaan. Dropsolid is een KMO gelegen te Gent die in 2012 werd opgericht en ondertussen al meer dan 50 werknemers telt. De missie van Dropsolid is om digitaal ondernemen voor bedrijven makkelijker te maken. Dit doen ze door de klant tijdens elke fase van hun digitale transformatie bij te staan. Concreet wil dit zeggen dat ze een digitaal platform voor de klant opzetten met behulp van Drupal en hen trainen in hoe ze dit digitaal platform kunnen gebruiken. Ook ondersteunen ze klanten door hen advies te geven over digitale marketing, Search Engine Optimization (SEO) en Search Engine Advertising (SEA).

Het onderwerp van deze bachelorproef werd mij aangereikt tijdens mijn stage door Nick Veenhof, de CTO van Dropsolid. Tijdens mijn stage bespraken we enkele onderzoeken die interessant voor Dropsolid zouden zijn. Ik koos voor dit onderwerp omdat dit mij het interessantste leek en het dichtst tegen mijn sterkten aanleunde.

Graag zou ik Rembrand Le Compte, Technical Lead bij Dropsolid, en mijn co-promotor Nick Veenhof bedanken om mij in de juiste richting te sturen tijdens mijn onderzoek. Verder zou ik graag alle developers van Dropsolid bedanken omdat ik steeds bij hen terecht kon voor vragen.

Ook zou ik graag mijn promotor Steven Vermeulen bedanken voor het vertrouwen en de hulp tijdens het onderzoeksproces. Steven was altijd beschikbaar en antwoordde snel op al mijn vragen.

# Samenvatting

Tegenwoordig bestaan er een hele hoop verschillende tools voor het functioneel testen van een website. Sommige van deze tools bestaan al langer en hebben een groot gebruikersbestand waardoor er veel documentatie over te vinden is. Andere zijn recenter en daar is moeilijk documentatie over te vinden, maar omdat ze dingen vanaf de basis anders doen, kunnen ze beter zijn dan deze. Sommige tools werken op zichzelf en voor andere dient een extra tool gebruikt te worden. Dit alles zorgt er voor dat het voor Dropsolid niet makkelijk is om de beste keuze te maken.

Het eerste hoofdstuk van dit onderzoek is de inleiding en hierin wordt de probleemstelling, de onderzoeksvraag en de onderzoeksdoelstelling geformuleerd. ormt de literatuurstudie van dit onderzoek en hierin wordt de stand van zaken in verband met functioneel testen besproken en worden de verschillende tools toegelicht. Ook de tools die niet onderzocht werden maar wel significant kunnen zijn komen hier aan bod en wordt toegelicht waarom deze niet geselecteerd werden. In het daarop volgende hoofdstuk, Methodologie, wordt de installatie en implementatie van de tools toegelicht. De tools die in dit onderzoek onderzocht en getest worden zijn: TestCafé, Cypress, Nightwatch, WebdriverIO, Nightmare en Puppeteer. Voor elk van deze wordt een test geschreven die hetzelfde doet om een ruim idee te krijgen over de verschillende uitvoeringstijden, de syntax en de mogelijkheden met de tool. De vergelijking van deze tools en de conclusie wordt in het hoofdstuk Conclusie beschreven.

In samenspraak met Dropsolid werd geconcludeerd dat Nightwatch de beste tool is voor hun use case. Deze kwam als beste naar voor grotendeels omdat de Drupal community naar deze tool toe groeit. Zo zit Nightwatch al in de core van de pre-release van Drupal 8.6. Verder is cross-browser en cross-platform testen met Nightwatch mogelijk en is er een tool specifiek voor cloud testen met Nightwatch in ontwikkeling, Nightcloud.

Tenslotte wordt in het laatste hoofdstuk, Proof of concept, de uitwerking van de testsuite voor de Paragraph Types van Dropsolid hun Drupal 8 installatieprofiel, genaamd Rocketship, besproken.

In de toekomst kan dit onderzoek uitgebreid worden met een onderzoek dat uitzoekt welke service zoals SauceLabs of Browserstack de beste is voor cross-browser testen in de cloud. Ook een onderzoek dat de Continue Integratie (CI) mogelijkheden uitzoekt en hoe deze kunnen geïmplementeerd worden zou een goed vervolg zijn op dit onderzoek.

# Inhoudsopgave

# Woordenlijst

**API** Een Application Programming Interface is een set aan definities waarmee softwareprogramma's onderling kunnen communiceren, van Tuil (2011). 11

**Assertie** Een assertie is een bewering. Tijdens het schrijven van testen worden asserties gemaakt over de output en vervolgens wordt nagegaan of de werkelijke output hetzelfde is als de beweerde output. 22, 24–28, 38, 47, 49, 51, 54, 57

**BDD** Behavior Driven Development is een ontwikkelingswijze waarbij voor er geprogrammeerd wordt eerst het gedrag wordt beschreven. 19

**CI** Continue Integratie is het proces waarbij de code die weggeschreven is met een versie controle systeem automatisch wordt getest en gebouwd naar een omgeving, Guckenheimer (2017). 6

**Contentmanagementsysteem** Een contentmanagementsysteem is een webapplicatie waarmee een website makkelijk kan beheerd worden en inhoud kan toegevoegd worden. Voorbeelden van bekende contentmanagementsystemen zijn WordPress, Drupal en Joomla. 36

**Dummy** Een dummy is een object dat nooit echt gebruikt wordt. Dummies worden meestal gebruikt voor het opvullen van parameterlijsten, Fowler (2007). 20

**Fake** Een fake is een object dat werkende implementaties heeft maar dat gebruik maakt van een kortere weg waardoor ze niet ideaal zijn voor productie, Fowler (2007). 20

**Framework** Een framework is een verzameling van libraries die gebruikt kunnen worden aan de hand van een API , „Software Framework" (g.d.). 12, 13, 17, 22, 24, 27–30, 36, 47, 49, 51

**Grunt** Grunt is een tool om zaken te automatiseren tijdens het ontwikkelingsproces van een applicatie. Het automatisch compileren van SASS code naar CSS code bij het

opslaan van een SASS bestand is een voorbeeld van een taak die Grunt kan uitvoeren. 26–28

**Gulp** Gulp is een tool om zaken te automatiseren tijdens het ontwikkelingsproces van een applicatie. Het automatisch compileren van SASS code naar CSS code bij het opslaan van een SASS bestand is een voorbeeld van een taak die Gulp kan uitvoeren. 26–28

**Headless** Headless wil zeggen dat er geen UI nodig is voor het uitvoeren van het programma. 29–31, 73, 124

**Library** Een library is een verzameling van code die gebruikt wordt om programma's en applicaties te maken, „Software Library" (g.d.). 11, 17, 22, 24, 27–30, 36, 47, 49, 51, 56, 57

**Mock** Een mock is een object dat voorgeprogrammeerd wordt met oproepen die ze verwachten te krijgen in gedachten, Fowler (2007) plural. 20

**Paragraph Types** Paragraph Types zijn inhoudstypes die gemaakt worden met de Drupal module Paragraphs. Paragraph Types kunnen zeer uiteenlopend zijn, van een simpele blok met een afbeelding tot een ingewikkelde slideshow, Bobbeldijk (2013). 6, 18, 59, 60

**SEA** Search Engine Advertising of zoekmachinemarketing is het tegen betaling laten verschijnen van advertenties voor en naast de zoekmachineresultaten. 3

**Selenium-Grid** Een Selenium-Grid is een netwerk van verschillende browsers, versies van browsers, browsers op verschillende besturingssystemen en verschillende formaten van browsers die gebruik maken van het Selenium framework. 22

**SEO** Search Engine Optimization of zoekmachineoptimalisatie is het optimaliseren van de kans dat een website hoog scoort in de organische zoekresultaten op een zoekmachine. 3

**Snapshot** Een snapshot is een afbeelding van de huidige staat van het systeem. 22, 26, 28, 29, 47

**Spy** Een spy is een stub die informatie terug geeft over wanneer die spy werd opgeroepen, Fowler (2007) plural. 20, 26

**Stub** Een stub wordt tijdens het testen gebruikt om een standaard antwoord te geven op een oproep, Fowler (2007). 12, 20, 26

**Testsuite** Een testsuite is een collectie van samenhorende testen. 6, 18, 25, 59

**Tool** Een software tool is een programma dat gebruikt wordt voor de ontwikkeling, herstelling of verbetering van een ander programma, „A Dictionary of Computing, Software Tool" (2004). 5, 11, 12, 17–19, 22–30, 33, 35, 36, 38, 39, 49, 53, 54, 56, 57

**TypeScript** TypeScript is een superset van JavaScript dit wil zeggen dat JavaScript een deelverzameling is van TypeScript en beiden kunnen door elkaar gebruikt worden.. 25, 28

**User story** Een user story is een beschrijving van wat een bepaalde gebruiker wil en waarom. 23

**Web server** Een web server is een programma dat HyperText Transfer Protocol verzoeken ontvangt en verstuurt naar een web client. Voorbeelden van web clients zijn Google Chrome, Mozilla Firefox, Internet Explorer,... Voorbeelden van web servers zijn Apache, IIS (Internet Information Services), NGINX,... 35

**WebDriver** Een WebDriver is een framework voor de automatisatie van een browser. Een WebDriver maakt het mogelijk om gebruikersinteracties zoals het navigeren naar een pagina of het invullen van een formulier te automatiseren. Voorbeelden van WebDrivers zijn: Selenium WebDriver, ChromeDriver, GeckoDriver, Microsoft WebDriver,... 22, 27, 30

**XPath** XPath is een taal om elementen te selecteren in een XML document. 27, 28

# Acroniemen

**API** Application Programming Interface. 11, 20, 25–27, 29

**BDD** Behavior Driven Development. 19, 23, 24, 27, 28, 57

**CI** Continue Integratie. 6, 25–28, 30, 53, 55, 57
**CPU** Central Processing Unit. 27, 34
**CSS** Cascading Style Sheets. 11, 12, 24, 27, 28

**HTML** HyperText Markup Language. 24, 27
**HTTP** HyperText Transfer Protocol. 34

**IDE** Integrated Development Environment. 25

**KMO** Kleine of Middelgrote Onderneming. 3, 17, 57

**RAM** Random Access Memory. 34
**REPL** Read–Eval–Print Loop. 28

**SASS** Syntactically Awesome Style Sheets. 11, 12
**SEA** Search Engine Advertising. 3
**SEO** Search Engine Optimization. 3

**TCP** Transmission Control Protocol. 34
**TDD** Test Driven Development. 24, 28

**UI** User Interface. 12, 20, 26, 29, 42

# 1. Inleiding

## 1.1 Probleemstelling

Tegenwoordig bestaat er een waaier aan tools, frameworks en libraries voor het functioneel testen van een website. Deze hebben elk hun eigen implementatie en vaak ook hun eigen filosofie in verband met testen.

Enerzijds zijn er de tools die al langer op de markt zijn en die gaandeweg een grote community hebben opgebouwd. Deze soort tools zijn over het algemeen goed gedocumenteerd en er is veel informatie over te vinden. Hierdoor is de instap vaak eenvoudiger dan de tools die minder lang bestaan.

Anderzijds zijn er de tools die recenter zijn ontwikkeld en die nog geen grote community hebben opgebouwd. Deze soort zijn vaak ontwikkeld met één specifieke toepassing in gedachten of doen dingen vanaf de basis anders waardoor ze beter kunnen zijn dan de oudere tools.

Deze waaier aan verschillende tools maakt het voor Dropsolid niet makkelijk om de juiste keuze te maken.

## 1.2 Onderzoeksvraag

Dit onderzoek wordt in opdracht van Dropsolid uitgevoerd dus de onderzoeksvraag is gebaseerd op hun specifieke toepassing. De onderzoeksvraag luidt als volgt: Welke op Javascript gebaseerde tools voor het schrijven van functionele testen kan een KMO

gespecialiseerd in Drupal het best gebruiken?

## 1.3 Onderzoeksdoelstelling

Het hoofddoel van dit onderzoek is om de verschillende op javascript gebaseerde tools voor het schrijven van functionele testen te vergelijken en de selectie te maken voor de tool of combinatie van tools die het beste bij Dropsolid passen. Een bijkomend doel is om deze tool of combinatie van tools te gebruiken om op een nieuwe installatie van Dropsolid hun Drupal 8 installatieprofiel, genaamd Rocketship, een testsuite op te zetten die de voorgeprogrammeerde Paragraph Types aanmaakt en test.

## 1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt de huidige stand van zaken in verband met het funtioneel testen van websites besproken aan de hand van een literatuurstudie. Hier worden de verschillende tools besproken.

In Hoofdstuk 3 wordt toegelicht hoe de installatie, configuratie en implementatie van de verschillende tools verliep.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en wordt toegelicht welke tool verkozen werd en waarom deze voor Dropsolid de beste is.

# 2. Stand van zaken

In dit hoofstuk worden eerst de belangrijkste testtypes voor het testen van websites besproken. Dit zijn unit testen, integratie testen en functionele testen.

Dan worden de verschillende soorten tools van het functionele testtype toegelicht, namelijk de tools voor het schrijven van testen, de tools voor cross-browser testen en de tools voor Behavior Driven Development (BDD).

Vervolgens worden de tools voor het schrijven van functionele testen verder opgedeeld in categorieën en worden per categorie de belangrijkste tools en hun kenmerken voorgesteld.

## 2.1 Belangrijkste testtypes voor webdevelopment

Dit deel bespreekt de belangrijkste testtypes voor webdevelopment: unit testen, integratie testen en functionele testen. In het volgende deel wordt dieper ingegaan op het testtype dat het belangrijkst is voor dit onderzoek, het functionele testtype.

### 2.1.1 Unit testen

Unit testen zijn het soort testen die er voor zorgen dat elke individuele component van een applicatie werkt. Met een unit wordt een module, een functie of een specifiek stukje code van een applicatie bedoeld, Elliott (2016).

Een unit test zou simpel en snel moeten zijn en zou wanneer de test faalt duidelijk moeten zeggen waar de fout ligt en wat het probleem is. Om unit testen zo snel mogelijk te maken wordt er vaak gebruik gemaakt van dummies, fakes, stubs, spys en mocks.

Unit testen zouden een vangnet moeten zijn die het grootste deel van de code omvat. Ze zouden ontwikkelaars het vertrouwen moeten geven om aanpassingen te maken omdat ze weten dat als er iets misloopt ze een duidelijk overzicht gaan krijgen van wat waar misliep zodat ze snel kunnen debuggen, Zilberfeld (2013).

Unit testen houden zeker niet alle bugs tegen maar ze vergroten de kans dat een nieuw stuk code correct werkt en ze zorgen er voor dat het debuggen van code minder tijd in beslag neemt.

### 2.1.2 Integratie testen

Integratie testen zijn het soort testen die nagaan of de verschillende componenten (units) correct met elkaar samenwerken. Bij integratie testen komen alle verschillende componenten samen en worden die in hun geheel getest. Hier wordt dus geen gebruik gemaakt van mocks zoals bij unit testen, Reselman (2017).

Integratie testen maken vaak gebruik van API's van derden zoals een API die zaken wegschrijft naar een databank. Integratie testen hebben hierdoor vaak een netwerk connectie nodig waardoor het uitvoeren van deze testen veel trager verloopt dan het uitvoeren van unit testen. Unit testen en integratie testen worden het best gescheiden gehouden omdat het voor unit testen belangrijk is dat deze zo snel mogelijk zijn, Elliott (2016).

### 2.1.3 Functionele testen

Functionele testen, ook wel browser testen, UI testen, front-end testen of end-to-end testen genoemd, zijn het soort testen die de applicatie testen vanuit het oogpunt van de gebruiker. Deze worden ook wel end-to-end testen genoemd omdat de volledige applicatie van front-end tot back-end correct moet werken voor het slagen van deze testen.

Functionele testen doorlopen de processen die de gebruiker ook doorloopt, zoals bijvoorbeeld het inloggen op een website, het navigeren naar een bepaalde pagina via het menu, het invullen van een formulier op een website,... Functionele testen krijgen hiervoor een input mee en een verwachting van wat de output zal zijn en gaan dan na of dit ook daadwerkelijk zo is.

## 2.2 Soorten tools voor het functioneel testen van een website

In dit gedeelte wordt onderscheid gemaakt tussen drie soorten tools die gebruikt worden bij het functioneel testen van een website. De eerste soort die besproken wordt, is voor dit onderzoek het belangrijkst. In het volgende hoofdstuk wordt de onderverdeling van die eerste soort nader toegelicht en worden de tools die in dit onderzoek vergeleken worden, besproken.

### 2.2.1 Tools voor het maken van functionele testen

De eerste soort zijn de tools die nodig zijn voor het schrijven van functionele testen. Deze soort wordt nog eens opgesplitst in drie verschillende categorieën:

- De all-in-one tools, dit zijn de tools die volledig onafhankelijk werken en waarvoor er dus geen extra tools, frameworks of libraries nodig zijn.
- De tools waarvoor er nog een WebDriver vereist is voor het kunnen uitvoeren van testen.
- De libraries die nood hebben aan een test framework en assertie libraries om als volwaardige test tool beschouwd te kunnen worden.

Meer over deze verschillende soorten in het volgende deel.

### 2.2.2 Tools voor cross-browser testen

Deze soort zijn de tools die nodig zijn om functionele testen op verschillende besturingssystemen, browsers en formaten uit te voeren. Deze tools maken het volgende mogelijk:

- Het testen van een website op verschillende browsers zoals Google Chrome, Mozilla Firefox, Internet Explorer, Safari, ...
- Het testen van een website op verschillende versies van browsers.
- Het testen van een website op browsers op verschillende besturingssystemen zoals Windows, MacOS, Linux, ...
- Het testen van een website op verschillende formaten zoals het smartphone formaat, tablet formaat, desktop formaat, ...

Deze tools bieden vaak nog extra diensten aan waarvan het in de cloud uitvoeren van testen een van de belangrijkste is. Dit zorgt er voor dat testen niet meer lokaal uitgevoerd worden maar wel op afstand in de cloud. Hierdoor is het parallel uitvoeren van testen mogelijk. Deze tools maken meestal gebruik van een Selenium-Grid. In het volgende deel wordt dieper ingegaan op Selenium en libraries die Selenium uitbreiden.

Een andere dienst die deze tools kunnen aanbieden zijn het nemen van snapshots na het falen van een test. Aan de hand van deze snapshots is het mogelijk om specifiek te gaan debuggen omdat de exacte staat van het systeem waar de test faalde beschikbaar is.

Voorbeelden van deze soort tools zijn SauceLabs, BrowserStack, CrossBrowserTesting,

Testingbot en BrowseEmAll.

### 2.2.3 Tools voor Behavior Driven Development

De laatste soort zijn optionele tools die een abstractielaag toevoegen bovenop de andere tools om Behavior Driven Development, gedragsgerichte ontwikkeling, mogelijk te maken. BDD is een methodologie waarbij de focus ligt op het gedrag van de gebruiker. De verschillende gedragen van de gebruiker (scenario's) worden meestal vastgelegd aan de hand van user stories.

Deze soort tools zijn optioneel en dienen enkel om het lezen en schrijven van testen te vereenvoudigen. Het voordeel van deze tools is dat ze het mogelijk maken voor niet-programmeurs om testen te schrijven. Het nadeel is dat ze extra complexiteit toevoegen die vaak overbodig is.

Voorbeelden van dergelijke tools zijn Cucumber.js en CodeceptJS.

## 2.3 Tools voor het maken van functionele testen

In dit deel worden eerst de tools, frameworks en libraries besproken die een ondersteunende rol hebben voor de tools die in dit onderzoek onderzocht worden. Daarna worden per categorie de tools toegelicht die in het volgende hoofdstuk, Methodologie, getest worden. Dit deel wordt afgesloten met de vermelding van tools die in dit onderzoek niet onderzocht worden maar wel belangrijk zijn of ooit geweest zijn.

### 2.3.1 Ondersteunende tools, frameworks en libraries

#### ChromeDriver

ChromeDriver is een open source tool voor het geautomatiseerd testen van websites. Concreet wil dit zeggen dat ChromeDriver een standalone server is die gebruik maakt van een Chrome of Chromium browser voor het uitvoeren van geautomatiseerde testen. ChromeDriver is beschikbaar op Windows, MacOS en Linux. Dit onderzoek maakt gebruik van ChromeDriver 2.41, ChromeDriver (2018).

#### Mocha

Mocha (2018) is een open source Node.js JavaScript framework voor het maken van asynchrone testen. Mocha wordt in dit onderzoek gebruikt als standaard test framework voor de tools die nog geen test framework hebben.

#### Chai

Chai (2017) is een open source Node.js library voor het maken van asserties in testen. Met Chai is het mogelijk om zowel BDD als TDD asserties te maken. Chai wordt in dit onderzoek gebruikt als standaard assertie library voor de tools die dit nog niet bevatten.

### 2.3.2 Electron

Electron (2018) is een framework dat gebruik maakt van Chromium en Node.js en dat dient voor het maken van applicaties met webtechnologieën als JavaScript, HTML en CSS. Electron is open source en beschikbaar op zowel Windows, MacOS als Linux.

### 2.3.3 All-in-one tools

Dit zijn tools die geen frameworks of libraries van derden nodig hebben om te werken, deze tools zijn zelfstandig en bevatten het hele pakket. Het voordeel van deze tools is dat het mogelijk is om testen te schrijven en uit te voeren zonder het downloaden van extra software. Ook de extra complexiteit die wordt toegevoegd bij de integratie van verschillende tools met elkaar valt hier dus weg. De twee die in dit onderzoek besproken

en vergeleken worden van dit type zijn: TestCafé en Cypress.

#### TestCafé

TestCafé (2018), niet te verwarren met TestCafé Studio, is een gratis open source Node.js tool voor het automatiseren van end-to-end web testen en is oorspronkelijk ontwikkeld door DevExpress. TestCafé Studio daarentegen is een IDE die ook door DevExpress ontwikkeld werd en die het mogelijk maakt om testen te maken zonder code te moeten schrijven. TestCafé Studio is betalend en op deze tool wordt in dit onderzoek niet verder ingegaan.

TestCafé werkt op volgende besturingssystemen: Windows, MacOS en Linux, en kan gebruik maken van volgende browsers: Google Chrome, Internet Explorer, Microsoft Edge, Mozilla Firefox, Safari, Android browser. TestCafé kan ook geïntegreerd worden met programma's zoals SauceLabs, BrowserStack of CrossBrowserTesting voor testen in de cloud op meerdere browsers en formaten uit te voeren.

Het schrijven van testen met TestCafé gebeurt in JavaScript of TypeScript. Een kenmerk van TestCafé is dat het een manier implementeert om slim te testen. Er kunnen wachttijden ingesteld worden voor elementen om te verschijnen op de pagina. Van zodra het element verschijnt gaat de test verder en als de wachttijd overschreden wordt dan faalt de test. Dit zorgt er voor dat er geen verplichte slaaptijden moeten inbouwd worden om te wachten op elementen die niet direct tevoorschijn komen. De code zelf is goed leesbaar omdat de API die gebruikt wordt voor elementen te selecteren en asserties te maken eenvoudig, leesbaar en logisch is. Verder kan er ook gebruik gemaakt worden van het Page Object patroon om testen te vereenvoudigen en dubbele code te vermijden. Het Page Object patroon zorgt er voor dat de pagina die getest wordt weg geabstraheerd kan worden en in de code van de test kan gerefereerd worden naar elementen van de pagina die getest moet worden.

Een ander belangrijk kenmerk van TestCafé is de mogelijkheid om testen parallel uit te voeren. Het is mogelijk om meerdere instanties van dezelfde browser of van verschillende browsers op te zetten. Vervolgens wordt bij het uitvoeren van een testsuite de testen over deze verschillende instanties verdeeld.

TestCafé rapporteert automatisch alle JavaScript errors die hij tegenkomt op een webpagina bij het uitvoeren van een test.

TestCafé heeft ook een plugin genaamd TestCafé live waarmee testen kunnen geschreven en aangepast worden terwijl deze worden uitgevoerd. Van zodra er een aanpassing wordt gemaakt stopt de huidige test, indien die nog bezig was, en start het de test gewoon opnieuw. Dit vereenvoudigt het proces van testen schrijven en debuggen sterk.

Om TestCafé te integreren in een CI systeem is er op de site van TestCafé zelf voldoende documentatie te vinden over hoe dit precies verloopt en zijn er ook door de community plugins ontwikkeld om dit proces te vereenvoudigen. Op de site van TestCafé zelf is de documentatie voor de implementatie in volgende CI systemen te vinden: AppVeyor, CircleCI, Jenkins, TeamCity en Travis. TestCafé testen kunnen ook uitgevoerd worden

tijdens het ontwikkelingsproces door de testen via plugins toe te voegen aan Gulp of Grunt.

### Cypress

Cypress (2018) is de tweede all-in-one tool die toegelicht wordt voor het maken van functionele testen en is ook open source. Het is net zoals TestCafé open source en heeft een nieuwe architectuur die niet bovenop Selenium werd gebouwd. Cypress is opnieuw het volledige pakket en is dus niet afhankelijk van servers of drivers voor de installatie of configuratie. Cypress heeft een zeer gebruiksvriendelijke eigen UI voor het uitvoeren en meevolgen van testen.

Cypress testen worden volledig in JavaScript geschreven en Cypress heeft zijn eigen API voor het automatiseren van de browser of voor de interne configuratie van Cypress te veranderen. Cypress heeft ook een automatische functie ingebouwd om te wachten op elementen om te verschijnen of om asserties te maken. Hierdoor moeten er geen vaste wachttijden ingebouwd worden en wordt er dus tijd uitgespaard.

De sterkte van Cypress zit hem in de eenvoud om testen te schrijven en te debuggen. Er zijn enkele redenen waarom dit zo is. De eerste reden is, dat het uitvoeren van Cypress testen live kan meegevolgd worden en dat indien er een aanpassing gemaakt wordt, de test opnieuw start. De tweede reden is dat tijdens het live uitvoeren van testen, de logs van de command line in het linkerdeel van de UI te zien zijn waardoor er direct als er iets misloopt een foutmelding te zien is. De derde reden is dat Cypress automatisch snapshots maakt tijdens het uitvoeren van testen. Deze worden vervolgens in de command log geplaatst en achteraf kan op elke stap nagegaan worden wat er dan precies gebeurde. De vierde en laatste reden is dat het met Cypress mogelijk is om screenshots te nemen nadat een test faalde of om de volledige uitvoering van de test te filmen. Hierdoor kan nadat de test is uitgevoerd en gefaald precies terugegaan worden naar het punt waar de test faalde om zo specifiek te debuggen.

Een ander kenmerk van Cypress is de mogelijkheid om gebruik te maken van spys of stubs zoals in unit testen. Stubs kunnen gebruikt worden voor het netwerk verkeer zodat de eigen server niet moet gebruikt worden.

De mogelijkheden om Cypress in te bouwen in een CI systeem zijn zeer ruim. Zo is Cypress al succesvol geïmplementeerd geweest in volgende systemen: Jenkins, Travis, CircleCI, CodeShip, GitLab, BuildKite, AppVeyor, Semaphore, Concourse, Solano, Docker.

Momenteel is cross-browser testen nog niet mogelijk met Cypress. Enkel varianten van de Chrome familie worden momenteel ondersteund. Cypress heeft wel plannen om in de toekomst meer browsers te ondersteunen, Mann (2017). Anderzijds kan over het nut van cross-browser testen ook gediscussieerd worden. Cross-browser testen vergroot de kost en infrastructuur nodig voor het testen erg terwijl dit het vertrouwen slechts licht vergroot. Ook de kans dat een fout browserspecifiek is is zeer klein, Mann (2018).

Parallel testen is ook niet mogelijk met enkel Cypress. Wel kan gebruik gemaakt worden van Docker om verschillende Docker containers op te zetten die parallel draaien en met

Cypress testen uitvoeren.

### 2.3.4 Frameworks die nood hebben aan een WebDriver

Dit zijn frameworks die hun eigen syntax en tools voor asserties hebben maar die nog geen WebDriver hebben voor het uitvoeren van deze testen. Voor het uitvoeren van testen met deze soort tools moet er dus nog een WebDriver van een derde partij voorzien worden zoals bijvoorbeeld de Selenium WebDriver, ChromeDriver of GeckoDriver. Voorbeelden van deze soort zijn Nightwatch en WebdriverIO.

#### Nightwatch

Nightwatch (2018) is een open source Node.js framework voor het maken van functionele testen. Voor het uitvoeren van testen met Nightwatch is een WebDriver van een externe partij vereist. Nightwatch ondersteunt ChromeDriver, GeckoDriver en Selenium Standalone Server.

Nightwatch maakt gebruik van de W3C WebDriver API voor het uitvoeren van commando's en asserties op HTML elementen. Hiermee kan er gebruik gemaakt worden van CSS en XPath selectors om elementen op de pagina te selecteren.

Het parallel uitvoeren van testen is mogelijk met Nightwatch, zo kunnen er verschillende instanties van dezelfde browser of van verschillende browsers aangemaakt worden. Het aantal parallelle testen dat kan uitgevoerd worden is gelijk aan het aantal CPU's, zijn er 4 CPU's dan kunnen er maximaal 4 testen parallel uitgevoerd worden.

Een ander kenmerk van Nightwatch is dat het al een BDD library bevat en er dus geen gebruik moet gemaakt worden van externe libraries zoals Cucumber.js of Codecept.js om Behavior Driven testen te schrijven.

Op vlak van cross-browser testen is de integratie van Nightwatch mogelijk met alle bekende cloud services zoals SauceLabs, BrowserStack, TestingBot of CrossBrowserTesting. Momenteel is Nightcloud, een cloud service specifiek voor Nightwatch, in ontwikkeling, „NightCloud.io" (2017).

Continue integratie met Nightwatch is mogelijk met alle bekende CI systemen zoals Teamcity, Jenkins, Hudson, CircleCI,... Voor de integratie van Nightwatch in het ontwikkelingsproces zijn plugins beschikbaar voor de integratie met Gulp of Grunt.

Nightwatch wordt door de Drupal community ondersteund en is momenteel in de core van de pre-release van Drupal 8.6 opgenomen. Voordien werd door de Drupal community voornamelijk gebruik gemaakt van PhantomJS voor functionele testen maar nu PhantomJS niet meer onderhouden wordt is de community overgestapt op Nightwatch.

#### WebdriverIO

WebdriverIO (2018) is een open source Node.js test framework voor het maken van functionele testen. In tegenstelling tot Nightwatch laat WebdriverIO de keuze van BDD of TDD test framework aan de gebruiker zelf over. De frameworks die compatibel zijn met WebdriverIO en waaruit er kan gekoze worden zijn: Mocha, Jasmine en Cucumber. Deze test frameworks worden aan de hand van Node.js adapter modules verbonden met WebdriverIO.

Standaard is het enkel mogelijk om testen in JavaScript te schrijven maar aan de hand van een Node.js pakket kan de functionaliteit van WebdriverIO uitgebreid worden om ook TypeScript te kunnen gebruiken. Elementen zijn met WebdriverIO gemakkelijk te selecteren met de CSS of XPath selectors. WebdriverIO implementeert $ en $$ functies die geketend worden om makkelijker elementen te selecteren. Dit kan gebruikt worden ter vervanging van ingewikkelde XPath selectors.

Tijdens het uitvoeren van testen wordt er een verslag uitgeprint in de command line over hoe deze testen verliepen. Dit verslag is aanpasbaar en er is keuze uit 10 verschillende reporters die de testverslagen anders uitprinten. Het is ook mogelijk om een zelfgemaakte reporter te gebruikten. WebdriverIO voorziet ook standaard een service die snapshots maakt bij het falen van testen.

Het parallel uitvoeren van testen is mogelijk en in het configuratiebestand van WebdriverIO kan gedefinieerd worden welke verschillende browsers er gebruikt moeten worden en hoeveel instanties van deze browsers.

Voor de ontwikkeling en het debuggen van testen voorziet WebdriverIO een REPL interface, hiermee is mogelijk om commando's uit te voeren in de command line en rechtstreeks in de browser te zien wat er gebeurt.

Op vlak van Cross-Browser testen voorziet WebdriverIO documentatie voor de integratie met de cloud services SauceLabs, Browserstack en TestingBot. Om WebdriverIO aan het lokaal ontwikkelingsproces toe te voegen bestaan er plugins voor de integratie met Gulp en voor de integratie met Grunt.

Op vlak van continue integratie is het mogelijk om WebdriverIO te integreren in een CI systeem. WebdriverIO voorziet enkel documentatie voor de integratie in Jenkins maar er zijn online voorbeelden te vinden over hoe de integratie verloopt in andere systemen zoals Travis.

### 2.3.5 Libraries voor webautomatisering

Dit zijn libraries die in eerste plaats niet specifiek dienen voor testen maar wel voor webautomatisering. Deze libraries kunnen in combinatie met andere tools en frameworks gebruikt worden voor testen te maken. Het eerste dat deze soort tools nodig heeft is een test framework, de populairste voorbeelden hiervan zijn Mocha en Jasmine. Het tweede dat deze soort tools nodig heeft, is een library voor asserties, het populairste voorbeeld

hiervan is Chai. Aan de hand van deze twee soorten tools kunnen libraries zoals Puppeteer en Nightmare fungeren als een volwaardig test framework.

### Nightmare

Nightmare (2018) is een open source library voor browser automatisatie origineel ontwikkeld door Segment maar later open source geworden. Met de API van Nightmare is het mogelijk om gebruikersinteracties na te bootsen met simpele commando's zoals goto, type, click,... Nightmare werd oorspronkelijk ontwikkeld voor het automatiseren van taken op een website maar wordt vooral gebruikt voor het functioneel testen van websites en voor het crawlen van websites.

Nightmare maakt gebruik van Electron en Electron is een combinatie van Chromium en Node.js. Dit wil zeggen dat testen of programma's die met Nightmare geschreven worden, uitgevoerd worden in een Chromium omgeving.

Twee handige tools voor het functioneel testen van een website met Nightmare zijn Niffy en Daydream. Niffy is een tool die bovenop Nightmare gebouwd is en die helpt om veranderingen of bugs in de UI te ontdekken. Daydream is een Chrome extensie waarmee acties die uitgevoerd worden op een website omgezet worden in een Nightmare of Puppeteer script.

Op vlak van cross-browser testen zijn er geen mogelijkheden met Nightmare, Nightmare is enkel ontwikkeld voor de automatisatie van Electron.

De mogelijkheden in verband met het parallel uitvoeren van testen en continue integratie hangen niet direct van Nightmare zelf af, maar hangen wel af van het gebruikte test framework. Concreet wil dit zeggen dat als het test framework bijvoorbeeld geïntegreerd kan worden in Jenkins en de testen parallel kan uitvoeren dan kan dit ook met Nightmare.

### Puppeteer

Puppeteer (2018) is een open source Node.js library voor het automatiseren van een Chrome of Chromium browser. Standaard voert Puppeteer een headless versie van Chrome of Chromium uit, maar er kan ingesteld worden dat een volledige zichtbare versie van Chrome of Chromium moet uitgevoerd worden. Puppeteer kan gebruikt worden voor alles waarvoor browser automatisatie vereist is, maar wordt voornamelijk gebruikt voor het crawlen van websites en voor het functioneel testen van websites.

Een van de standaard ingebouwde functies van Puppeteer is het nemen van snapshots.

Op vlak van cross-browser testen zijn er geen mogelijkheden met Puppeteer. Puppeteer is enkel ontwikkeld voor de automatisatie van Chrome of Chromium. Wel kunnen cloud services gebruikt worden om testen in de cloud uit te voeren.

Het parallel uitvoeren van testen is mogelijk met Puppeteer maar wordt niet standaard voorzien. Deze functionaliteit kan wel gerealiseerd worden met tools die de community

heeft ontwikkeld zoals bijvoorbeeld de puppeteer-cluster library. Het integreren van Puppeteer in een CI systeem is mogelijk maar hier voorziet Puppeteer geen documentatie over.

### 2.3.6 Overige tools

In dit deel worden de tools toegelicht die in dit onderzoek niet getest worden, maar die wel belangrijk genoeg zijn om besproken te worden.

#### Selenium

Selenium is een alles omvattende tool voor het automatiseren van browsers. Selenium bevat zijn eigen WebDriver maar kan ook gelinkt worden aan andere WebDrivers zoals ChromeDriver en GeckoDriver. Voor het uitvoeren van programma's met de Selenium WebDriver is de Selenium Standalone Server vereist.

Selenium wordt voornamelijk gebruikt voor het maken van testen maar het automatiseren van repititieve taken op een website is ook perfect mogelijk met Selenium.

Selenium zelf wordt in dit onderzoek niet onderzocht omdat het verouderd is. Maar sommige van de tools, frameworks en libraries die eerder besproken werden en die wel onderzocht worden zijn gebouwd bovenop Selenium ter uitbreiding.

#### Appium

Appium is een open source test framework voor het testen van mobiele applicaties. Met Appium kunnen iOS, Android en Windows applicaties bestuurd worden om ze te testen.

Appium wordt niet onderzocht omdat Appium specifiek voor mobiele applicaties is en Dropsolid een algemenere tool nodig heeft.

#### Protractor

Protractor is een end-to-end test framework specifiek voor het testen van Angular en AngularJS applicaties.

Protractor wordt om dezelfde reden als Appium niet onderzocht, Dropsolid heeft nood aan iets algemener.

#### PhantomJS, SlimerJS

PhantomJS is een headless tool voor browserautomatisatie met Webkit. PhantomJS wordt momenteel niet meer onderhouden sinds Vitaly Slobodin hiermee gestopt is. De reden was de komst van headless Chrome die sneller is, minder geheugen inneemt en stabieler is, Slobodin (2017).

SlimerJS is gelijkaardig aan PhantomJS, het verschil is dat SlimerJS bovenop Gecko werkt. Gecko is de browser engine van Mozilla Firefox. SlimerJS kan ook niet headless uitgevoerd worden.

PhantomJS wordt niet onderzocht omdat het niet meer onderhouden wordt en dus als verouderd kan gezien worden. SlimerJS daarentegen wordt gezien als experimenteel en is dus niet belangrijk genoeg.

# 3. Methodologie

In dit hoofdstuk wordt eerst de werkomgeving besproken. Daarna wordt het testscenario toegelicht, hierin staat hoe de tools getest zijn. Vervolgens wordt de basis configuratie besproken die voor het testen van elke tool hetzelfde is. Hierna wordt elke verschillende tool getest.

## 3.1 Werkomgeving

In dit deel wordt de werkomgeving waarvan voor dit onderzoek gebruik is gemaakt toegelicht. De verschillende programma's en de reden waarom deze verkozen werden worden individueel toegelicht. In dit onderzoek wordt telkens, indien dit mogelijk was, gebruik gemaakt van de laatste stabiele versie die beschikbaar was om dit onderzoek zo actueel mogelijk te maken. Deze programma's werden voornamelijk geïnstalleerd met de package manager van de Linux versie die wordt gebruikt maar er wordt wel vermeld waar deze programma's te verkrijgen zijn.

### 3.1.1 Virtuele machine

Dit onderzoek maakt gebruik van de laatste versie van VirtualBox die momenteel beschikbaar is, VirtualBox 5.2.16 Oracle (2018). VirtualBox is een programma waarmee andere besturingssystemen kunnen gedraaid worden binnen het huidige besturingssysteem. VirtualBox wordt gebruikt om een omgeving op te zetten die enkel dient voor dit onderzoek en dus niet beïnvloed wordt door externe factoren, zoals het downloaden van programma's voor andere doeleinden. Deze omgeving bevat enkel het nodige voor dit onderzoek en niet meer.

In VirtualBox werd er een linux virtuele machine opgezet. Er wordt gebruik gemaakt van Linux Mint 19 64-bit Cinnamon editie omdat dit besturingssysteem door Dropsolid wordt aangeraden. Linux Mint is een gratis en open source besturingssysteem dat gebaseerd is op Debian en Ubuntu en al 30000 packages voorziet, Mint (2018).

De specificaties van de virtuele machine zijn de volgende:

- 8192MB RAM geheugen.
- Een virtuele harde schijf met een vaste grootte van 50GB. Er wordt gebruik gemaakt van een vaste grootte en niet van een virtuele harde schijf die dynamisch vergroot omdat die laatste de performantie van zowel het huidig besturingssysteem als dat van de virtuele machine kan aantasten, Sanders (2006).
- 128MB video geheugen.
- 4 CPU's
- In de netwerkinstellingen werden 2 poortdoorverwijzingen ingesteld. De eerste voor HTTP met als protocol TCP en van hostpoort 80 naar gastpoort 80. De tweede voor MariaDB met terug TCP als protocol en van hostpoort 3306 naar gastpoort 3306.

### 3.1.2 LAMP-stack

LAMP-stack is de benaming voor een combinatie van softwarepakketten die dienen voor het draaien van websites. De LAMP-stack die voor dit onderzoek werd opgezet bestaat uit Linux Mint, Apache2, MariaDB en PHP 7.1. De Linux Mint versie werd reeds besproken in het vorige deel, de overige worden in dit deel toegelicht.

#### Apache

Apache (2018) is een gratis Web server en de laatste stabiele versie die is uitgekomen is 2.4.34. Voor dit onderzoek wordt gebruik gemaakt van deze laatste stabiele versie van Apache en niet van NGINX of IIS omdat Apache gedeelde hosting mogelijk maakt met .htaccess bestanden. Apache heeft momenteel met 46% ook het grootste marktaandeel, (Leslie, 2018).

#### MariaDB

MariaDB (2018) is een relationeel databasemanagementsysteem. Een relationeel databasemanagementsysteem is een programma waarmee een relationele database gecreëerd, geüpdatet en beheerd kan worden. Een database is een set van data dat op een computer wordt opgeslagen en relationeel wijst er op dat er een structuur in zit die werkt aan de hand van relaties tussen de verschillende stukken data, („What is a Relational Database Management System?", g.d.). Voor dit onderzoek wordt de laatste stabiele versie van MariaDB gebruikt, MariaDB 10.3.8. De keuze ging uit naar MariaDB en niet naar MySQL om 2 redenen. De eerste reden is dat MariaDB performanter is dan MySQL dit blijkt zowel uit het onderzoek van Lindstrom in 2014, (Lindstrom, 2014), als uit het recentere onderzoek in 2017 van Taranto, (Taranto, 2017). De tweede reden is dat Drupal websites beter met MariaDB samenwerken dan met MySQL.

#### PHP

PHP (2018), Hypertext preprocessor, is een scripting taal die specifiek voor web development ontwikkeld is. Aangezien PHP de programmeertaal is van Drupal is Php dus zeker een vereiste. De laatste stabiele versie van PHP is PHP 7.2.8 en deze versie wordt vanaf Drupal 8.5.0 ondersteund. De laatste stabiele versie van Drupal die uitgekomen is en die in dit onderzoek gebruikt wordt, is 8.5.6. Meer hierover in het gedeelte over Drupal.

### 3.1.3 Extra tools

#### PhpMyAdmin

PhpMyAdmin (2018) is een handige tool om een database te beheren vanuit een webinterface. De versie die gebruikt wordt is opnieuw de laatste stabiele versie en voor phpmyadmin is dat momenteel 4.8.2.

#### Drush

Drush (2018) is een hulpprogramma om Drupal vanuit de command line te bedienen. Drush bevat handige commando's om bijvoorbeeld de cache van een site te legen vanuit de command line. Aangezien dit allemaal vanuit de command line gebeurt is dit dus sneller dan via de webinterface. De versie van Drush die in dit onderzoek gebruikt wordt is 8.1.17.

#### Git

Git (2018) is een gratis open source versiebeheersysteem. Aan de hand van Git kunnen bestanden en mappen gemakkelijk online in een Git repository bijgehouden en geüpdatet worden. De laatste stabiele versie van Git dat uitgekomen is, is 2.18.0 en dit is ook de versie die gebruikt wordt.

#### Node.js & Npm

Node.js (2018) is een open source runtime-omgeving die het mogelijk maakt om javascript code uit te voeren buiten de browser. De versie van Node.js die voor dit onderzoek gebruikt wordt is Node.js 10.8.0. Bij de installatie van Node.js wordt Npm meegeleverd. Npm (2018) staat voor Node Package Manager en is dus het programma waarmee Node.js modules kunnen worden gedownload. De versie van Npm die meegeleverd werd met Node.js 10.8.0 is 6.2.0. Node.js en Npm wordt in dit onderzoek meerdere keren gebruikt om verschillende tools, frameworks of libraries te downloaden en uit te voeren.

#### Visual Studio Code

Visual Studio Code is de code-editor die in dit onderzoek gebruikt wordt om de testen te schrijven of om de configuratie bestanden aan te passen. Er werd gekozen voor Visual Studio Code omdat dit een veelzijdige lichte code-editor is die kan uitgebreid worden met plugins voor specifieke doeleinden.

### 3.1.4 Drupal

Aangezien dit onderzoek uitgevoerd wordt in opdracht van Dropsolid, wordt dus gebruik gemaakt van het Contentmanagementsysteem waarin zij gespecialiseerd zijn, namelijk Drupal (2018).

Drupal is een gratis open source Contentmanagementsysteem geschreven in PHP dat zeer krachtig is en waarmee flexibele websites kunnen gebouwd worden die zeer schaalbaar zijn. Voor het bouwen van een website in Drupal is een redelijke portie technische kennis vereist, het onderhouden en toevoegen van inhoud vereist echter minder technische kennis en kan makkelijk door iemand gedaan worden die niet technisch is. Drupal is open source en heeft een grote community die actief modules en thema's toevoegen aan Drupal.org zodat deze door anderen kunnen gebruikt worden. Een Drupal module is een extensie die kan toegevoegd worden aan een Drupal website en die een bepaalde functionaliteit voorziet. Thema's zijn extensies die aan een site kunnen toegevoegd worden en die de look en feel van de site aanpassen.

Momenteel zijn er 2 versies van Drupal in omloop die beiden nog onderhouden en ondersteund worden, Drupal 7 en Drupal 8. Drupal 7 kwam uit op 5 januari 2011 en is sinds dan uitgegroeid tot een volwaardig systeem met veel modules, veel documentatie en veel problemen die al opgelost werden door de community. Drupal 8 is nieuwer en werd

gelanceerd op 19 november 2015. Hierdoor is er over Drupal 8 minder documentatie te vinden en zijn er momenteel nog minder modules ontwikkeld. Drupal 8 bevat nog steeds de kernideeën van Drupal maar is ontwikkeld met de gedachte om Drupal toegankelijker te maken voor zowel gebruikers als voor ontwikkelaars. Drupal 8 is beter in lijn met het web zoals Jonathan zei in zijn vergelijking tussen Drupal 7 en Drupal 8, Ramael, 2015.

Voor dit onderzoek wordt gebruik gemaakt van de laatste versie van Drupal 8, namelijk Drupal 8.5.6. Het doel van dit onderzoek is om het voor Dropsolid mogelijk te maken om front-end testen te schrijven voor hun Drupal 8 installatieprofiel, genaamd Rocketship, dus het is logisch dat enkel Drupal 8 in aanmerking komt. Rocketship is door Dropsolid ontwikkeld en is een Drupal 8 installatieprofiel dat al modules, features en een basis thema bevat om het webontwikkelingsproces te versnellen.

## 3.2 Testscenario

### 3.2.1 Installatie en configuratie

In dit deel zal het installatieproces beschreven worden. Hier zullen ook de configuratie bestanden terechtkomen die aangemaakt werden voor de installatie van de tools.

### 3.2.2 Implementatie

In dit onderdeel zal de test, die met de tool geschreven werd, komen en hier zal ook het commando voor het uitvoeren van de test te vinden zijn.

De test is steeds op dezelfde manier opgebouwd en is geschreven met als doel het proces dat een normale gebruiker zou doorlopen zo goed mogelijk na te bootsen. Dit gebeurt door gebruik te maken van de menu's om te navigeren en van de buttons om bijvoorbeeld in te loggen. Er zal geen gebruik gemaakt worden van kortere wegen om het testen te versnellen.

De test start met het inloggen als administrator op een Drupal 8.5.6 website die reeds vooraf geïnstalleerd werd. Vervolgens wordt er een nieuwe node aangemaakt van het type "basic page"met de titel "Een nieuwe node". Daarna wordt deze node verwijderd en ten slotte wordt er uigelogd. Tijdens deze test worden er vier asserties gemaakt om na te gaan of de test foutloos verlopen is. De eerste assertie is direct bij het optarten van de site om te controleren of de correcte site is geopend en of deze correct geopend is door de titel van de pagina te controleren. De tweede assertie controleert of er correct is ingelogd door na het inloggen te controleren of de huidige pagina de account overzichtspagina is. De derde assertie gaat na of de node correct is aangemaakt door te controleren of de huidige pagina de node pagina is. De laatste assertie controleert of er tijdens het verwijderen van de pagina een bevestigingspagina getoond wordt.

### 3.2.3 Uitvoeringstijd

In dit deel zal de uitvoeringstijd van het tien maal uitvoeren van de test en de gemiddelde tijd hiervoor worden weergegeven. De uitvoeringstijden worden gemeten met de ingebouwde reporters van de tools, hierdoor zijn de tijden niet altijd even precies. De uitvoeringstijden die in dit onderzoek gemeten worden, moeten niet als een benchmark gezien worden maar eerder als een proof of concept om ruim de verschillen in tijd te vergelijken.

## 3.3 Basisconfiguratie

Voor het surfen naar de website te vereenvoudigen werd een apache2 configuratie bestand aangemaakt voor elk project dat in de folders /etc/apache2/sites-available en /etc/apache2/sites-enabled terechtkomt.

#### PROJECT_NAAM.conf

```
<VirtualHost *:80>
   ServerName PROJECT_NAAM
   ServerAlias *.PROJECT_NAAM

   DocumentRoot /websites/PROJECT_NAAM

   <Directory /websites/PROJECT_NAAM>
      Options Indexes FollowSymLinks MultiViews
      AllowOverride All
      Require all granted
   </Directory>

    ErrorLog /var/log/apache2/PROJECT_NAAM_error.log
    LogLevel error
    CustomLog /var/log/apache2/PROJECT_NAAM_access.log
       combined
</VirtualHost>
```

In het hosts bestand in de /etc/ folder werd volgende regel toegevoegd:

127.0.0.1 PROJECT_NAAM .

Deze configuratie maakt het mogelijk om lokaal via de projectnaam naar de website te surfen.

Verder wordt elke tool geïnstalleerd via Npm en hiervoor moet eerst een package.json bestand aangemaakt worden in de hoofdmap. Dit gebeurt met het volgende commando: "npm init". Bij de Npm installatie worden ook de dependencies van de tool geïnstalleerd.

## 3.4 TestCafé

### 3.4.1 Installatie en configuratie

De installatie van TestCafé gebeurt met het commando: "npm install testcafe". Hiermee wordt testcafe lokaal geïnstalleerd in de hoofdmap van de Drupal installatie. Met het commando: npm install -g testcafe, wordt testcafe ook globaal geïnstalleerd. Zowel lokaal als globaal installeren van TestCafé maakt het mogelijk om gebruik te maken van het commando "testcafe"voor het uitvoeren van de test.

Vervolgens wordt in dezelfde map een tests folder aangemaakt die de test bevat. De test wordt "testcafe.js"genoemd en is in het volgende deel, implementatie, te vinden.

### 3.4.2 Implementatie

Het uitvoeren van de test gebeurt met het commando: "testcafe 'chrome' tests/testcafe.js". Dit voert de test op Chrome uit.

#### testcafe.js

```
1  import { Selector } from 'testcafe';
2
3  fixture`TestCafé test`
4    .page`http://testcafe.local/`
5
6    .beforeEach(async t => {
7      await t.expect(Selector('#block-bartik-page-title').find('h1').innerText).
           contains('Welcome to TestCafé')
8        .click('#block-bartik-account-menu .content ul li a')
9        .typeText('input[name=name]', 'admin')
10       .typeText('input[name=pass]', 'admin')
11       .click('#edit-submit')
12       .expect(Selector('#block-bartik-page-title').find('h1').innerText).contains('
             admin');
13   })
14
15   .afterEach(async t => {
16     await t.click(Selector('#block-bartik-account-menu ul li a').withText('Log out')
           );
17   });
18
19 test('Creating a basic page and deleting that page', async t => {
20   await t.click('#toolbar-item-administration')
21     .click('#toolbar-link-system-admin_content')
22     .click(Selector('ul.action-links li a').withText('Add content'))
23     .click(Selector('ul.admin-list li a').withText('Basic page'))
24     .typeText('#edit-title-0-value', 'Een nieuwe node')
25     .click('#edit-submit')
26     .expect(Selector('#block-bartik-page-title').find('h1').innerText).contains('Een
           nieuwe node')
27     .click(Selector('nav.tabs ul li a').withText('Delete'))
28     .expect(Selector('#block-seven-page-title').find('h1').innerText).contains('Are
         you sure you want to delete the content Een nieuwe node?')
29     .click('#edit-submit');
30 });
```

### 3.4.3 Uitvoeringstijd

| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| tijd (s) | 23 | 11 | 11 | 11 | 11 | 12 | 11 | 11 | 11 | 11 |
| Gemiddelde | 12,3 seconden | | | | | | | | | |

## 3.5 Cypress

### 3.5.1 Installatie en configuratie

Cypress wordt geïnstalleerd met het commando: "npm install cypress". Vervolgens wordt de Cypress client geopend met het commando: "npx cypress open". In de client zijn momenteel enkel voorbeeldtesten terug te vinden die bij de installatie werden meegeleverd. Hier wordt naast de example folder een tests folder aangemaakt met de test "cypress.spec.js".

In de hoofdmap is er bij de installatie een cypress.json bestand aangemaakt. In dit bestand kan de standaard configuratie overschreven worden. Hier wordt de volgende regel toegevoegd om vast te leggen op welke site de testen moeten uitgevoerd worden:

"baseUrl": "http://cypress.local/".

### 3.5.2 Implementatie

De test verloopt volgens de specificaties beschreven in het vorige deel en wordt met behulp van de UI gestart. In deze test wordt error handling afgezet zodat de test niet zou stoppen wanneer er ergens in de JavaScript een error wordt tegengekomen.

#### cypress.spec.js

```
1  context('Cypress test', () => {
2
3    beforeEach(() => {
4      cy.visit('')
5      cy.get('#block-bartik-page-title h1').should('contain', 'Welcome to Cypress')
6      cy.get('#block-bartik-account-menu .content ul li a').click()
7      cy.get('input[name=name]').type('admin')
8      cy.get('input[name=pass]').type('admin')
9      cy.get('#user-login-form').submit()
10     cy.get('#block-bartik-page-title h1').should('contain', 'admin')
11   })
12
13   it('Create a basic page and deleting that page', () => {
14     cy.on('uncaught:exception', (err, runnable) => {
15       expect(err.message).to.include('uncaught error handling')
16       done()
17       return false
18     })
19     cy.get('#toolbar-link-system-admin_content').click()
20     cy.get('ul.action-links li a').contains('Add content').click()
21     cy.get('ul.admin-list li a').contains('Basic page').click()
22     cy.get('#edit-title-0-value').type('Een nieuwe node')
23     cy.get('#node-page-form').submit()
24     cy.get('#block-bartik-page-title h1').should('contain', 'Een nieuwe node')
25     cy.get('nav.tabs ul li').contains('Delete').click()
26     cy.get('#block-seven-page-title').should('contain', 'Are you sure you want to
           delete the content Een nieuwe node?')
27     cy.get('#node-page-delete-form').submit()
28   })
29
30   afterEach(() => {
31     cy.get('#block-bartik-account-menu ul li a').contains('Log out').click({ force:
           true })
```

```
32      })
33  })
```

### 3.5.3 Uitvoeringstijd

| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| tijd (s) | 8,70 | 6,70 | 5,88 | 6,68 | 6,64 | 6,55 | 6,46 | 6,46 | 6,44 | 5,88 |
| Gemiddelde | 6,64 seconden | | | | | | | | | |

## 3.6 Nightwatch

### 3.6.1 Installatie en configuratie

Eerst wordt Nightwatch lokaal geïnstalleerd door het volgende commando uit te voeren in de hoofdmap van het project dat hiervoor is aangemaakt: "npm install nightwatch". Vervolgens wordt Nightwatch ook globaal geïnstalleerd met het commando: "npm install -g nightwatch". Hierdoor kan gebruik gemaakt worden van het Nightwatch commando voor testen uit te voeren.

De volgende stap is de installatie van de ChromeDriver met het commando: "npm install chromedriver". Hierna wordt in de hoofdmap het configuratie bestand nightwatch.json aangemaakt waarin gedefinieerd wordt dat de ChromeDriver standaard moet gebruikt worden.

#### nightwatch.json

```
{
  "src_folders": [
    "nightwatch/tests"
  ],
  "output_folder": "reports",
  "custom_commands_path": "",
  "custom_assertions_path": "",
  "page_objects_path": "",
  "globals_path": "globals.js",
  "selenium": {
    "start_process": false,
    "server_path": "",
    "log_path": "",
    "port": 4444,
    "cli_args": {
      "webdriver.chrome.driver": "",
      "webdriver.gecko.driver": "",
      "webdriver.edge.driver": ""
    }
  },
  "test_settings": {
    "default": {
      "selenium_port": 9515,
      "selenium_host": "localhost",
      "default_path_prefix": "",
      "desiredCapabilities": {
        "browserName": "chrome",
        "chromeOptions": {
          "args": [
            "--no-sandbox"
          ]
        },
        "acceptSslCerts": true
      }
    },
    "chrome": {
      "desiredCapabilities": {
        "browserName": "chrome"
      }
    },
    "edge": {
      "desiredCapabilities": {
        "browserName": "MicrosoftEdge"
      }
    }
```

```
46     }
47 }
```

Vervolgens wordt in het globals.js bestand gedefinieerd dat ChromeDriver automatisch moet opstarten bij het uitvoeren van testen met Nightwatch.

#### globals.js

```
1  var chromedriver = require('chromedriver');
2
3  module.exports = {
4    before: function (done) {
5      chromedriver.start();
6
7      done();
8    },
9
10   after: function (done) {
11     chromedriver.stop();
12
13     done();
14   }
15 };
```

### 3.6.2 Implementatie

De test verloopt volgens de specificaties beschreven in het vorige deel en wordt met het commando "nightwatch nightwatch/tests/nightwatch.js"gestart.

#### nightwatch.js

```
1  module.exports = {
2    before: function (browser) {
3      browser
4        .url('http://nightwatch.local')
5        .assert.containsText('#block-bartik-page-title h1', 'Welcome to Nightwatch')
6        .click('#block-bartik-account-menu .content ul li a')
7        .setValue('input[name=name]', 'admin')
8        .setValue('input[name=pass]', 'admin')
9        .submitForm('#edit-submit')
10       .assert.containsText('#block-bartik-page-title', 'admin');
11   },
12
13   after: function (browser) {
14     browser
15       .useXpath()
16       .click("//a[text()='Log out']")
17       .end();
18   },
19
20   'Creating a basic page and deleting that page': (browser) => {
21     browser
22       .click('#toolbar-item-administration')
23       .click('#toolbar-link-system-admin_content')
24       .click("#block-seven-local-actions ul li a")
25       .useXpath()
26       .click("//span[text()='Basic page']")
27       .useCss()
28       .setValue('input[id=edit-title-0-value]', 'Een nieuwe node')
29       .submitForm('#edit-submit')
30       .assert.containsText('#block-bartik-page-title', 'Een nieuwe node')
31       .useXpath()
32       .click("//a[text()='Delete']")
```

```
33          .useCss()
34          .assert.containsText('#block-seven-page-title', 'Are you sure you want to
                delete the content Een nieuwe node?')
35          .submitForm('#edit-submit');
36      },
37  };
```

### 3.6.3 Uitvoeringstijd

| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| tijd (s) | 5,44 | 5,51 | 5,41 | 5,41 | 5,58 | 5,71 | 5,41 | 5,39 | 5,66 | 5,51 |
| Gemiddelde | 5,50 seconden | | | | | | | | | |

## 3.7 WebdriverIO

### 3.7.1 Installatie en configuratie

De installatie begint met WebdriverIO lokaal en globaal te installeren met de commando's "npm install webdriverionpm install -g webdriverio". Hierdoor kunnen testen uitgevoerd worden met het wdio commando. Hierna wordt ChromeDriver geïnstalleerd met "npm install chromedriver". Normaal gezien zou de volgende stap zijn om gebruik te maken van de Selenium Standalone Service om de ChromeDriver te kunnen aanspreken, maar deze stap kan vervangen worden door wdio ChromeDriver Service te installeren met het commando "npm install wdio-chromedriver-service". In de hoofdmap wordt een map webdriverio aangemaakt met daarin een map tests die de test zal bevatten. Voor de logs van Chromedriver in op te slaan wordt ook een folder aangemaakt. Om asserties te maken in de test, wordt de assertie library Chai gebruikt. Dit wordt geïnstalleerd met het commando "npm install chai". Om die library samen met WebdriverIO te gebruiken moet ook de module chai-webdriverio geïnstalleerd worden met "npm install chai-webdriverio".

Nu wordt de configuratie van WebdriverIO van uit de command line gestart met "wdio config". De volgende opties worden geselecteerd:

- Waar moeten de testen uitgevoerd worden? Op mijn lokale machine
- Welk test framework moet er gebruikt worden? Mocha
- Moet het framework automatisch geïnstalleerd worden? Ja
- Waar kunnen de testen teruggevonden worden? ./webdriverio/tests/*.js
- Welke reporter mag gebruikt worden? Dot reporter (standaard)
- Welke service moet er gebruikt worden voor het uitvoeren van testen? ChromeDriver
- Hoeveel moet gelogd worden? Stil (standaard)
- Waar moeten de snapshots van de errors opgeslagen worden? ./webdriverio/errorS-hots/
- Wat is de base url? http://webdriverio.local

Dit zorgt ervoor dat een basis configuratie bestand wdio.conf.js wordt aangemaakt. Dit bestand wordt nog aangepast om de ChromeDriver op de standaard 9515 poort te vinden.

#### wdio.conf.js

```
1  exports.config = {
2    port: '9515',
3    path: '/',
4    specs: ['./webdriverio/tests/*.js'],
5    exclude: [],
6    maxInstances: 10,
7    capabilities: [{ browserName: 'chrome' }],
8    sync: true,
9    logLevel: 'silent',
10   coloredLogs: true,
11   deprecationWarnings: true,
12   bail: 0,
13   screenshotPath: './webdriverio/errorShots/',
14   baseUrl: 'http://webdriverio.local',
15   waitforTimeout: 10000,
16   connectionRetryTimeout: 90000,
17   connectionRetryCount: 3,
```

```
18    framework: 'mocha',
19    mochaOpts: {
20      ui: 'bdd'
21    },
22    services: ['chromedriver'],
23    chromeDriverArgs: ['--port=9999'],
24    chromeDriverLogs: './chromedriver',
25  }
```

### 3.7.2 Implementatie

De test verloopt volgens het scenario beschreven in het vorige deel. Voor het uitvoeren van de test moet eerst de ChromeDriver service opgestart worden met het commando "./node_modules/.bin/chromedriver". Zodra de ChromeDriver opgestart is, kan met het commando "wdio wdio.conf.js"de test uitgevoerd worden.

#### webdriverio.js

```
1  var assert = require('assert');
2  var chai = require('chai');
3  var chaiWebdriver = require('chai-webdriverio').default;
4  chai.use(chaiWebdriver(browser));
5
6  describe('WebdriverIO ', function () {
7    before('going to the site and logging in', () => {
8      browser.url('/');
9      chai.expect('#block-bartik-page-title h1').to.have.text('Welcome to WebdriverIO'
           );
10     browser.click('#block-bartik-account-menu .content ul li a')
11       .setValue('input[name=name]', 'admin')
12       .setValue('input[name=pass]', 'admin')
13       .click('#edit-submit');
14     chai.expect('#block-bartik-page-title h1').to.have.text('admin');
15   });
16   after('logging out', async () => {
17     browser.url('/user/logout');
18   });
19   it('should be able to create and delete a basic page', async () => {
20     browser.click('#toolbar-item-administration')
21       .click('#toolbar-link-system-admin_content')
22       .click("#block-seven-local-actions ul li a")
23       .click("//span[text()='Basic page']")
24       .setValue('input[id=edit-title-0-value]', 'Een nieuwe node')
25       .click('#edit-submit');
26     chai.expect('#block-bartik-page-title h1').to.have.text('Een nieuwe node');
27     browser.click("//a[text()='Delete']");
28     chai.expect('#block-seven-page-title').to.have.text('Are you sure you want to
           delete the content Een nieuwe node?');
29     browser.click('#edit-submit');
30   });
31 });
```

### 3.7.3 Uitvoeringstijd

| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| tijd (s) | 6,7 | 6,8 | 6,5 | 7,0 | 6,7 | 6,5 | 6,7 | 6,7 | 6,1 | 6,7 |
| Gemiddelde | 6,6 seconden | | | | | | | | | |

## 3.8 Nightmare

### 3.8.1 Installatie en configuratie

De installatie van Nightmare start net zoals de andere tools met een Npm commando "npm install nightmare". Bij de installatie van Nightmare wordt ook al Electron meegeleverd dus dit moet niet zelf nog eens geïnstalleerd worden. Hierna wordt het test framework Mocha geïnstalleerd met "npm install mocha", en de assertie library Chai met "npm install chai". Het package.json bestand wordt ook aangepast om het uitvoeren van de testen met npm mogelijk te maken.

#### package.json

```
{
  "name": "nightmare.local",
  "version": "1.0.0",
  "description": "CONTENTS OF THIS FILE ---------------------",
  "main": "index.js",
  "scripts": {
    "test": "mocha --recursive test --timeout 15000"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "chai": "^4.1.2",
    "mocha": "^5.2.0",
    "nightmare": "^3.0.1"
  }
}
```

De laatste stap van de installatie is om een map met de naam test aan te maken. Deze map zal dan de test bevatten die uitgevoerd wordt.

### 3.8.2 Implementatie

Het uitvoeren van de test gebeurt met het commando "npm test". De test is opgebouwd volgens het reeds beschreven testscenario.

#### nightmare.js

```
const Nightmare = require('nightmare')
var nightmare = Nightmare({ show: true })
const chai = require('chai')
const expect = chai.expect

describe('Nightmare ', async () => {
  var title;

  before('Going to the site and logging in', async () => {
    await nightmare
      .goto('http://nightmare.local')
      .wait('#block-bartik-page-title', 1000);
    title = await nightmare.evaluate(() => { return document.querySelector('#block-
        bartik-page-title').textContent; });
    expect(title).to.have.string('Welcome to Nightmare');
    await nightmare
      .wait('#block-bartik-account-menu .content ul li a', 1000)
      .click('#block-bartik-account-menu .content ul li a')
```

```
18          .wait('input[name=name]', 1000)
19          .type('input[name=name]', 'admin')
20          .type('input[name=pass]', 'admin')
21          .click('#edit-submit')
22          .wait('.contextual', 1000);
23        title = await nightmare.evaluate(() => { return document.querySelector('#block-
              bartik-page-title').textContent; });
24        expect(title).to.have.string('admin');
25      });
26
27      it('going to the site and asserting the title is correct', async () => {
28        await nightmare
29          .wait('#toolbar-link-system-admin_content', 1000)
30          .click('#toolbar-link-system-admin_content')
31          .wait('#block-seven-local-actions ul li a', 1000)
32          .click('#block-seven-local-actions ul li a')
33          .wait('#block-seven-content ul li:last-child a', 1000)
34          .click('#block-seven-content ul li:last-child a')
35          .wait('input[id=edit-title-0-value]', 1000)
36          .type('input[id=edit-title-0-value]', 'Een nieuwe node')
37          .click('#edit-submit')
38          .wait('#block-bartik-page-title', 1000);
39        title = await nightmare.evaluate(() => { return document.querySelector('#block-
              bartik-page-title').textContent; });
40        expect(title).to.have.string('Een nieuwe node');
41        await nightmare
42          .click('#block-bartik-local-tasks nav ul li:last-child a')
43          .wait('#block-seven-page-title', 1000);
44        title = await nightmare.evaluate(() => { return document.querySelector('#block-
              seven-page-title').textContent; });
45        expect(title).to.have.string('Are you sure you want to delete the content Een
              nieuwe node?');
46        await nightmare.click('#edit-submit');
47      });
48
49      after('logging out and closing the browser', async () => {
50        await nightmare
51          .wait('#block-bartik-account-menu .content ul li:last-child a', 1000)
52          .click('#block-bartik-account-menu .content ul li:last-child a')
53          .end();
54      });
55  })
```

### 3.8.3 Uitvoeringstijd

| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| tijd (s) | 7,94 | 8,77 | 8,99 | 8,67 | 8,04 | 8,81 | 8,60 | 8,71 | 8,92 | 9,45 |
| Gemiddelde | 8,69 seconden | | | | | | | | | |

## 3.9 Puppeteer

### 3.9.1 Installatie en configuratie

De installatie van Puppeteer start terug met een Npm commando, "npm install puppeteer". Bij de installatie van Puppeteer wordt Chromium meegeleverd omdat dit de browser is die Puppeteer automatiseert. De volgende stap is om het test framework Mocha te installeren met "npm install mocha". Vervolgens wordt de assertie library chai met "npm install chai"geïnstalleerd. Hierna wordt het package.json bestand aangepast zodat de testen kunnen uitgevoerd worden met een Npm commando.

#### package.json

```
{
  "name": "puppeteer.local",
  "version": "1.0.0",
  "description": "CONTENTS OF THIS FILE ---------------------",
  "main": "index.js",
  "scripts": {
    "test": "mocha --recursive test --timeout 15000"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "chai": "^4.1.2",
    "mocha": "^5.2.0",
    "puppeteer": "^1.7.0"
  }
}
```

De laatste stap is om een map aan te maken in de hoofdmap met de naam "test", waarin het test bestand zal komen.

### 3.9.2 Implementatie

Het uitvoeren van de test gebeurt met het commando "npm test". De test is opgebouwd volgens het reeds beschreven testscenario.

#### puppeteer.js

```
const puppeteer = require('puppeteer')
const chai = require('chai')
const expect = chai.expect

describe('Puppeteer ', async () => {
  let browser, page;
  var title;

  before('Going to the site and logging in', async () => {
    browser = await puppeteer.launch({ headless: false , args: ['--no-sandbox'] });
    page = await browser.newPage();
    await page.setViewport({ width: 1280, height: 800 });
    await page.goto('http://puppeteer.local', { timeout: 0, waitUntil: '
        domcontentloaded' });
    title = await page.evaluate(() => { return document.querySelector('#block-bartik
        -page-title').textContent; });
    expect(title).to.have.string('Welcome to Puppeteer');
    await page.waitFor('#block-bartik-account-menu .content ul li a');
```

```
17      await page.click('#block-bartik-account-menu .content ul li a');
18      await page.waitFor('input[name=name]');
19      await page.type('input[name=name]', 'admin');
20      await page.type('input[name=pass]', 'admin');
21      await page.click('#edit-submit');
22      await page.waitFor('#block-bartik-page-title');
23      title = await page.evaluate(() => { return document.querySelector('#block-bartik
            -page-title').textContent; });
24      expect(title).to.have.string('admin');
25    });
26
27    it('going to the site and asserting the title is correct', async () => {
28      await page.waitFor('#toolbar-link-system-admin_content')
29      await page.click('#toolbar-link-system-admin_content')
30      await page.waitFor('#block-seven-local-actions ul li a')
31      await page.click('#block-seven-local-actions ul li a')
32      await page.waitFor('#block-seven-content ul li:last-child a');
33      await page.click('#block-seven-content ul li:last-child a')
34      await page.waitFor('input[id=edit-title-0-value]');
35      await page.type('input[id=edit-title-0-value]', 'Een nieuwe node')
36      await page.click('#edit-submit')
37      await page.waitFor('#block-bartik-page-title');
38      title = await page.evaluate(() => { return document.querySelector('#block-bartik
            -page-title').textContent; });
39      expect(title).to.have.string('Een nieuwe node');
40      await page.click('#block-bartik-local-tasks nav ul li:last-child a');
41      await page.waitFor('#block-seven-page-title');
42      title = await page.evaluate(() => { return document.querySelector('#block-seven-
            page-title').textContent; });
43      expect(title).to.have.string('Are you sure you want to delete the content Een
            nieuwe node?');
44      await page.click('#edit-submit');
45    });
46
47    after('logging out and closing the browser', async () => {
48      await page.waitFor('#block-bartik-account-menu .content ul li:last-child a');
49      await page.click('#block-bartik-account-menu .content ul li:last-child a');
50      await browser.close();
51    });
52  })
```

### 3.9.3 Uitvoeringstijd

| # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| tijd (s) | 5,09 | 4,93 | 5,67 | 5,14 | 4,96 | 5,34 | 4,71 | 5,00 | 5,25 | 5,18 |
| Gemiddelde | 5,13 seconden | | | | | | | | | |

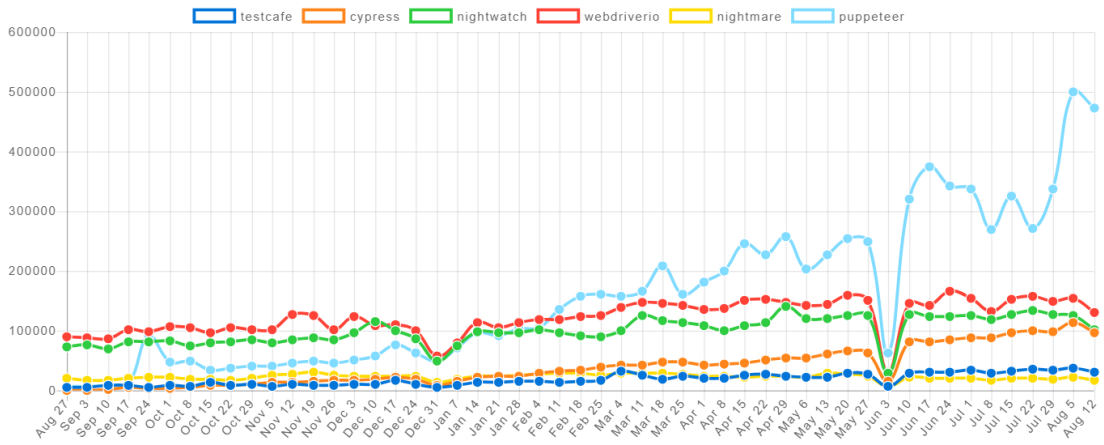
# 4. Conclusie

In dit hoofdstuk worden de verschillende tools die in het vorige hoodstuk getest werden, vergeleken op vlak van:

- populariteit
- syntax
- mogelijkheden voor cross-browser testen en integratie met cloud services
- mogelijkheden voor integratie in een CI systeem
- mogelijkheden voor het parallel uitvoeren van testen
- performantie

Hierna wordt de conclusie geformuleerd en wordt toegelicht welke tool verkozen werd en in het volgende hoofdstuk als toepassing voor Dropsolid gebruikt wordt.

Figuur 4.1: Aantal wekelijkse downloads met Npm op een tijdspanne van 1 jaar, grafiek van npm trends (2018)




## 4.1 Vergelijking

### 4.1.1 Populariteit

Puppeteer heeft de sterkste groei meegemaakt sinds de tool in januari van 2018 publiek werd gemaakt, met een piek van bijna 500000 wekelijkse downloads op 5 augustus. De drie populairste tools na Puppeteer met tussen de 100000 en 150000 wekelijkse downloads zijn WebdriverIO, Nightwatch en Cypress waarvan die laatste een langzame maar stabiele groei kende gedurende het afgelopen jaar. Nightmare en TestCafé zijn het minst populair met onder de 50000 wekelijkse downloads.

### 4.1.2 Syntax

TestCafé en Nightwatch zijn op vlak van syntax zeer gelijkaardig en zijn duidelijk te verstaan. Dit komt door de mogelijkheid om commando's te ketenen en om asserties zonder de ketting te breken uit te voeren. Kort daarop volgt WebdriverIO die ook zeer gelijkaardig is maar waarbij de ketting moet gebroken worden om asserties te maken.

Cypress code is ook goed verstaanbaar maar is net iets minder dan de voorgaande tools omdat opdrachten niet kunnen geketend worden.

Nightmare en Puppeteer zijn het minst goed leesbaar. Nightmare omdat de asserties niet met een simpel commando kunnen gemaakt worden en Puppeteer omdat de opdrachten niet kunnen geketend worden en er dus overal gebruik moet gemaakt worden van await zodat de commando's niet parallel uitgevoerd worden.

### 4.1.3 Cross-browser en cloud service mogelijkheden

| | |
|---|---|
| TestCafé | TestCafé ondersteund het testen op Google Chrome, Internet Explorer, Microsoft Edge, Mozilla Firefox, Safari en Android. Ook kan TestCafé gebruik maken van alle bekende cloud services voor cross-browser testen zoals SauceLabs, Browserstack en TestingBot. |
| Cypress | Cross-browser testen is momenteel niet mogelijk met Cypress, enkel Chrome varianten worden ondersteund. Er zijn wel plannen om in de toekomst meerdere browsers te ondersteunen, Mann (2017). Momenteel is er ook nog geen integratie met cloud services mogelijk. |
| Nightwatch | Integratie met alle bekende cloud services (BrowserStack, SauceLabs, TestingBot en CrossBrowserTesting) en alle bekende browsers (Google Chrome, Mozilla Firefox, Microsoft Edge en Internet explorer) is mogelijk. |
| WebdriverIO | Integratie met alle bekende cloud services en alle bekende browsers is mogelijk. |
| Nightmare | Nightmare kan enkel gebruik maken van de Electron applicatie en er zijn geen integratie mogelijkheden met cloud services. |
| Puppeteer | Puppeteer ondersteund enkel Chrome en Chromium. Wel zijn er integratie mogelijkheden met de bekende cloud services. |

### 4.1.4 Integratie mogelijkheden met CI systemen

| | |
|---|---|
| TestCafé | Voorziet documentatie voor integratie in AppVeyor, CircleCI, Jenkins, TeamCity en Travis. |
| Cypress | Voorziet documentatie voor integratie in Jenkins, Travis, CircleCI, CodeShip. |
| Nightwatch | Voorziet geen documentatie voor integratie in CI systemen, wel zijn online voorbeelden te vinden over de integratie in de bekende CI systemen zoals Jenkins en Travis. |
| WebdriverIO | Voorziet documentatie voor integratie in Jenkins en Docker. |
| Nightmare | Voorziet geen documentatie voor integratie in CI systemen, wel zijn online voorbeelden te vinden over de integratie in de bekende CI systemen. |
| Puppeteer | Voorziet geen documentatie voor integratie in CI systemen, wel zijn online voorbeelden te vinden over de integratie in de bekende CI systemen. |

### 4.1.5 Parallel uitvoeren van testen

| | |
|---|---|
| TestCafé | Het parallel uitvoeren van testen met TestCafé is standaard mogelijk. |
| Cypress | Het parallel uitvoeren van testen met Cypress is momenteel nog niet mogelijk, hier wordt wel aan gewerkt. |
| Nightwatch | Het parallel uitvoeren van testen met Nightwatch is standaard mogelijk. |
| WebdriverIO | Het parallel uitvoeren van testen met Nightwatch is standaard mogelijk. |
| Nightmare | Het parallel uitvoeren van testen met Nightmare is niet mogelijk. |
| Puppeteer | Het parallel uitvoeren van testen met Nightmare is standaard niet mogelijk. Er zijn wel libraries door de community gemaakt om deze functionaliteit te voorzien zoals bijvoorbeeld Puppeteer Cluster. |

### 4.1.6 Performantie

Deze tijden moeten niet gezien worden als een benchmark om de tools te vergelijken maar eerder als een proof of concept.

| | |
|---|---|
| TestCafé | Gemiddeld 12,3 seconden voor het uitvoeren van de test. |
| Cypress | Gemiddeld 6,64 seconden voor het uitvoeren van de test. |
| Nightwatch | Gemiddeld 5,5 seconden voor het uitvoeren van de test. |
| WebdriverIO | Gemiddeld 6,6 seconden voor het uitvoeren van de test. |
| Nightmare | Gemiddeld 8,69 seconden voor het uitvoeren van de test. |
| Puppeteer | Gemiddeld 5,13 seconden voor het uitvoeren van de test. |

## 4.2 Conclusie

De tool die uit dit onderzoek naar voor komt als beste voor een KMO gespecialiseerd in Drupal is Nightwatch. Nightwatch haalde het tweede beste resultaat op de performantietest en kan testen parallel uitvoeren. Er zijn voldoende mogelijkheden voor integratie met cloud services en ook cross-browser testen is geen probleem. Momenteel is er zelfs een cloud service specifiek voor Nightwatch in ontwikkeling. Ook de integratie in CI systemen is geen probleem.

Nightwatch voorziet standaard de mogelijkheid om BDD en unit testen te schrijven. Nightwatch heeft al een assertie library ingebouwd en er is ook de mogelijkheid om screenshots te nemen met een eenvoudig commando.

De doorslaggevende factor was echter het feit dat de Drupal community naar deze tool toegroeit voor het maken van hun functionele testen. Zo is Nightwatch opgenomen in de core van de voorlopige versie van Drupal 8.6. Er zijn al enkele commando's met Nightwatch ontwikkeld voor bijvoorbeeld het inloggen of uitloggen op een Drupal site te automatiseren.

# 5. Proof of concept

Dit hoofdstuk bevat de proof of concept die voor Dropsolid werd uitgewerkt. Dit is een testsuite die met Nightwatch gemaakt is om de Paragraph Types op Rocketship, de Drupal 8 installatie van Dropsolid, te testen. In dit hoofdstuk wordt eerst uitgelegd wat Rocketship is en wat de verschillende Paragraph Types zijn. Hierna wordt de installatie en configuratie van Nightwatch besproken. Vervolgens wordt de code van de testen opgelijst en wordt uigelegd wat er specifiek in de test gebeurt. Ten slotte wordt de code van de verschillende commando's die voor de testen werd geschreven opgelijst en toegelicht.

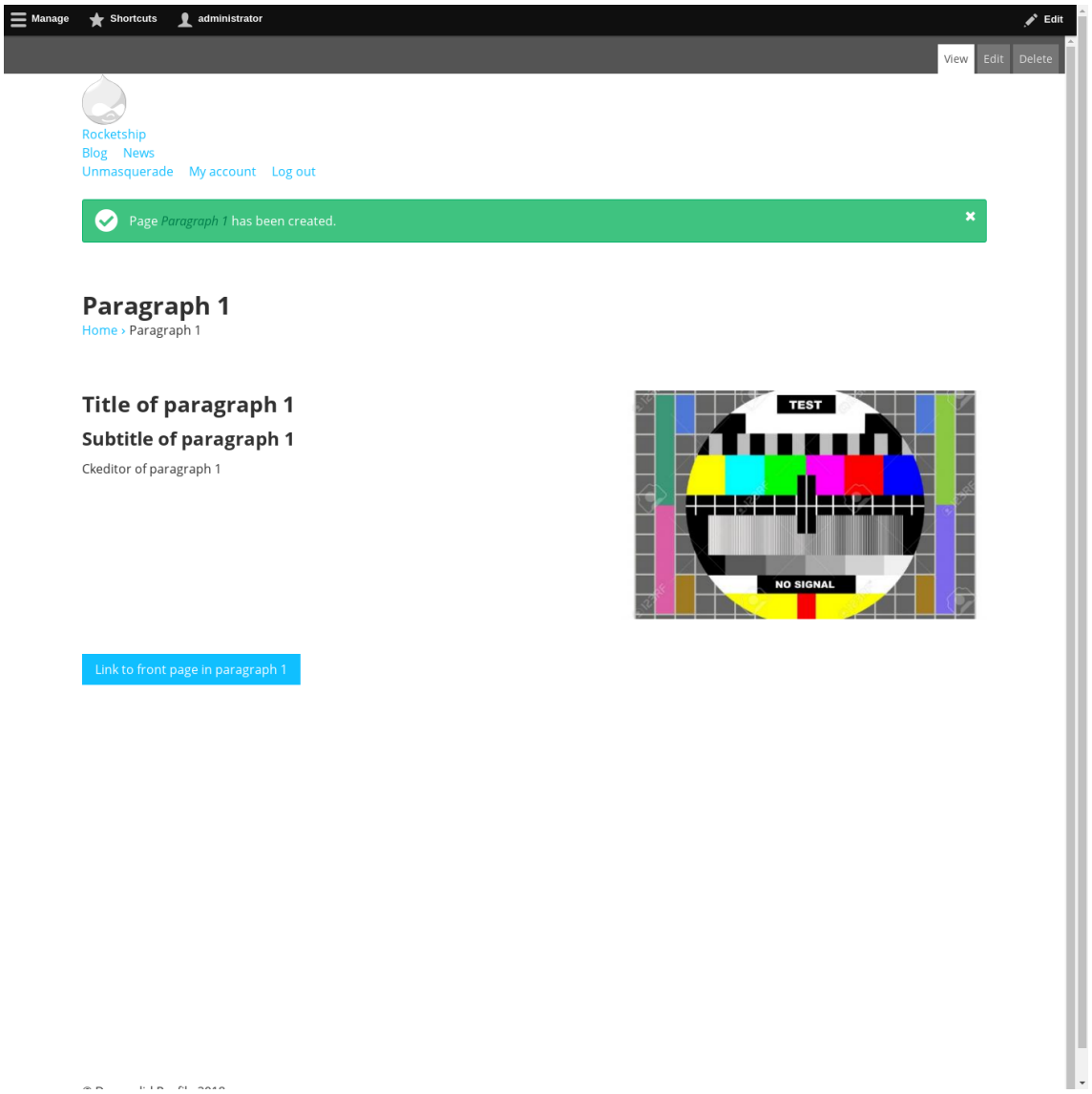
## 5.1 Rocketship & Paragraph Types

Rocketship is het installatieprofiel van Drupal dat Dropsolid heeft ontwikkeld. Dit wel zeggen dat er na de installatie van Rocketship, een werkende versie van een Drupal 8 site beschikbaar is die al voorgeprogrammeerde zaken bevat om het ontwikkelingsproces van een website te versnellen en vereenvoudigen.

Paragraph Types zijn voorgeprogrammeerde blokken die aan een website kunnen toegevoegd worden. Dit zijn bijvoorbeeld blokken waarin een afbeelding kan geplaatst worden of blokken die een titel en wat tekst bevat.

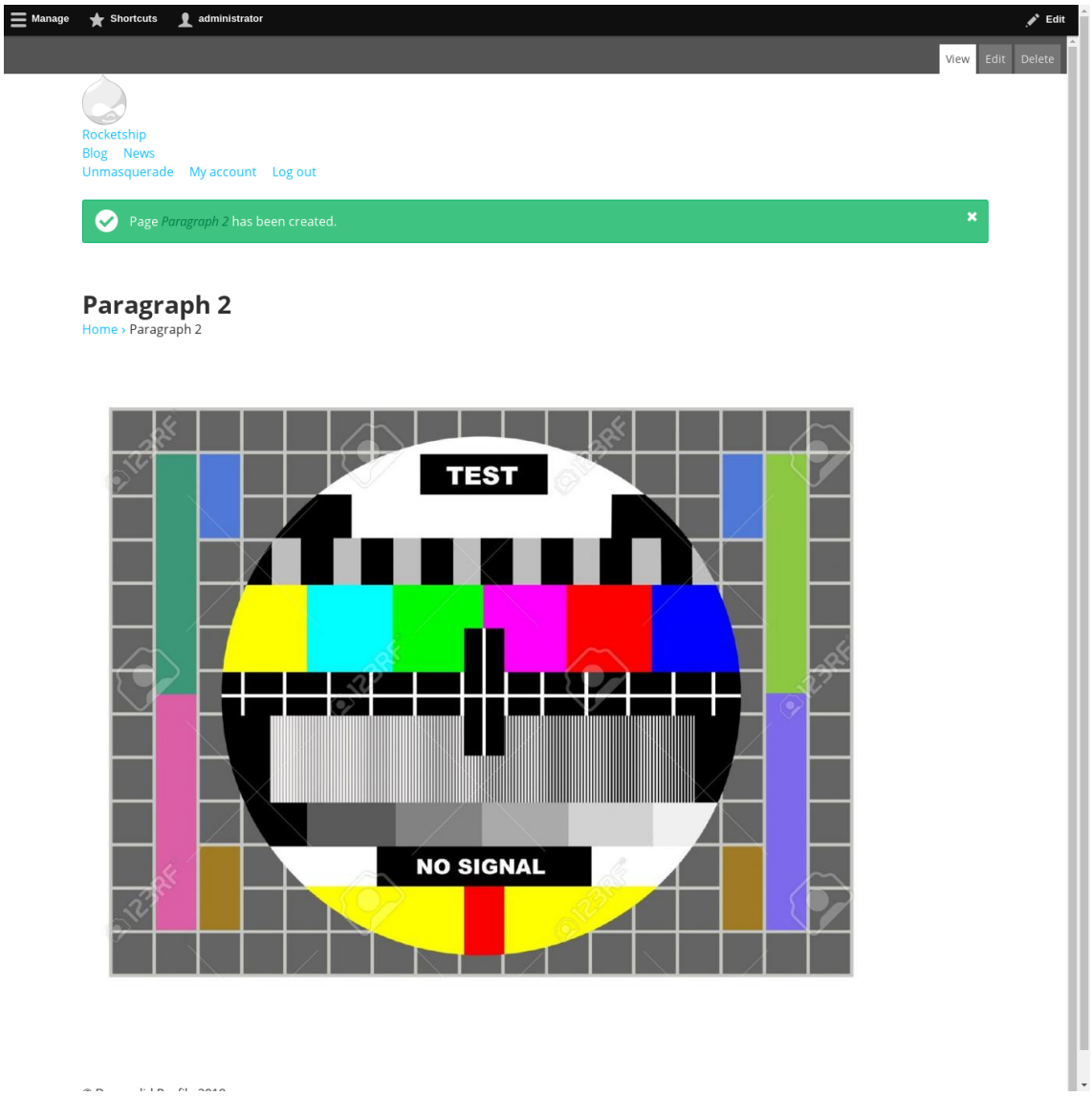
### Paragraph 1: Story

De eerste Paragraph is een blok met een titel, een subtitel, een tekstveld, een afbeelding en een button die linkt naar een andere pagina.



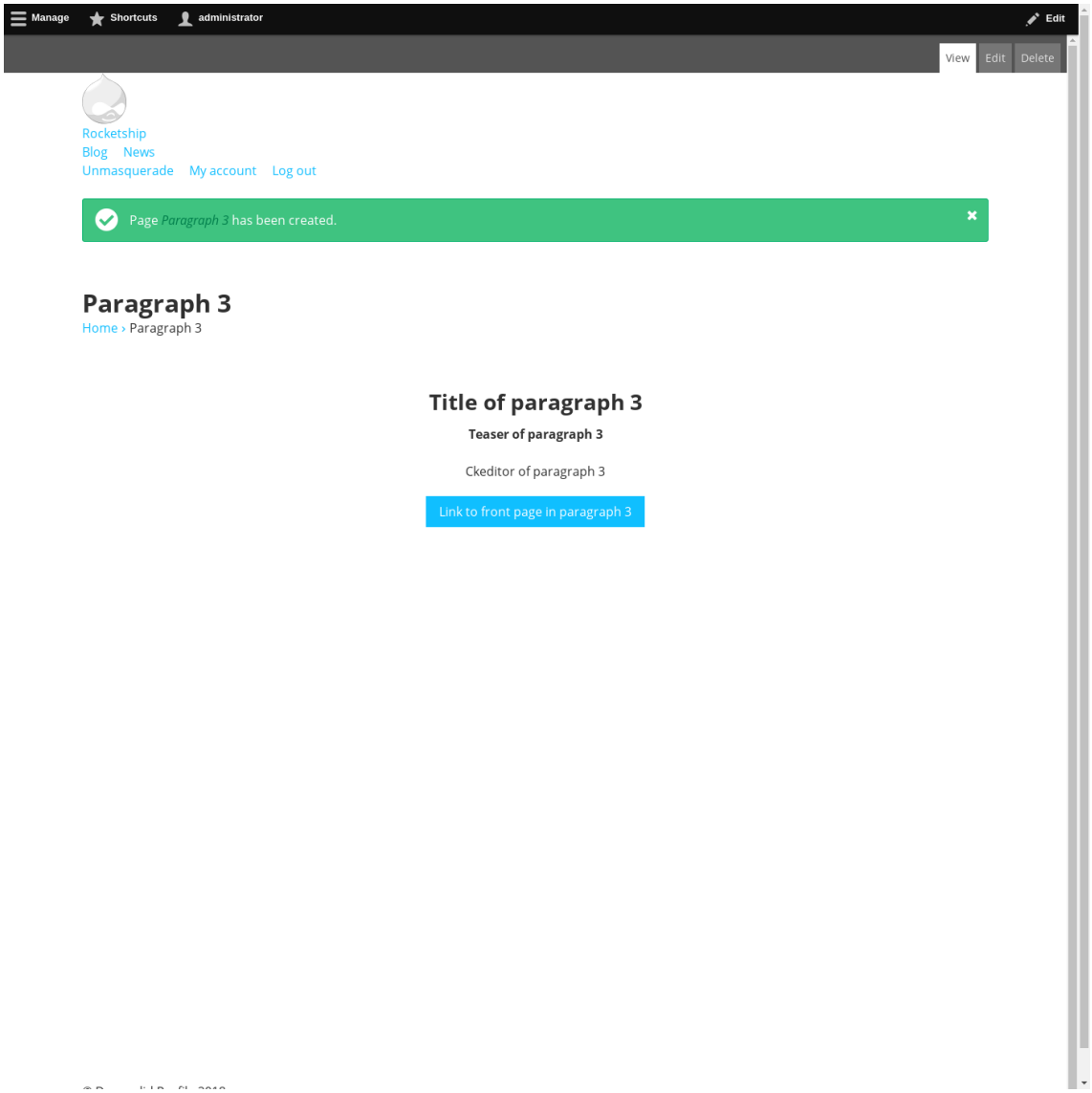
### Paragraph 2: Image

De tweede Paragraph is een blok die een afbeelding bevat en linkt naar een andere pagina.



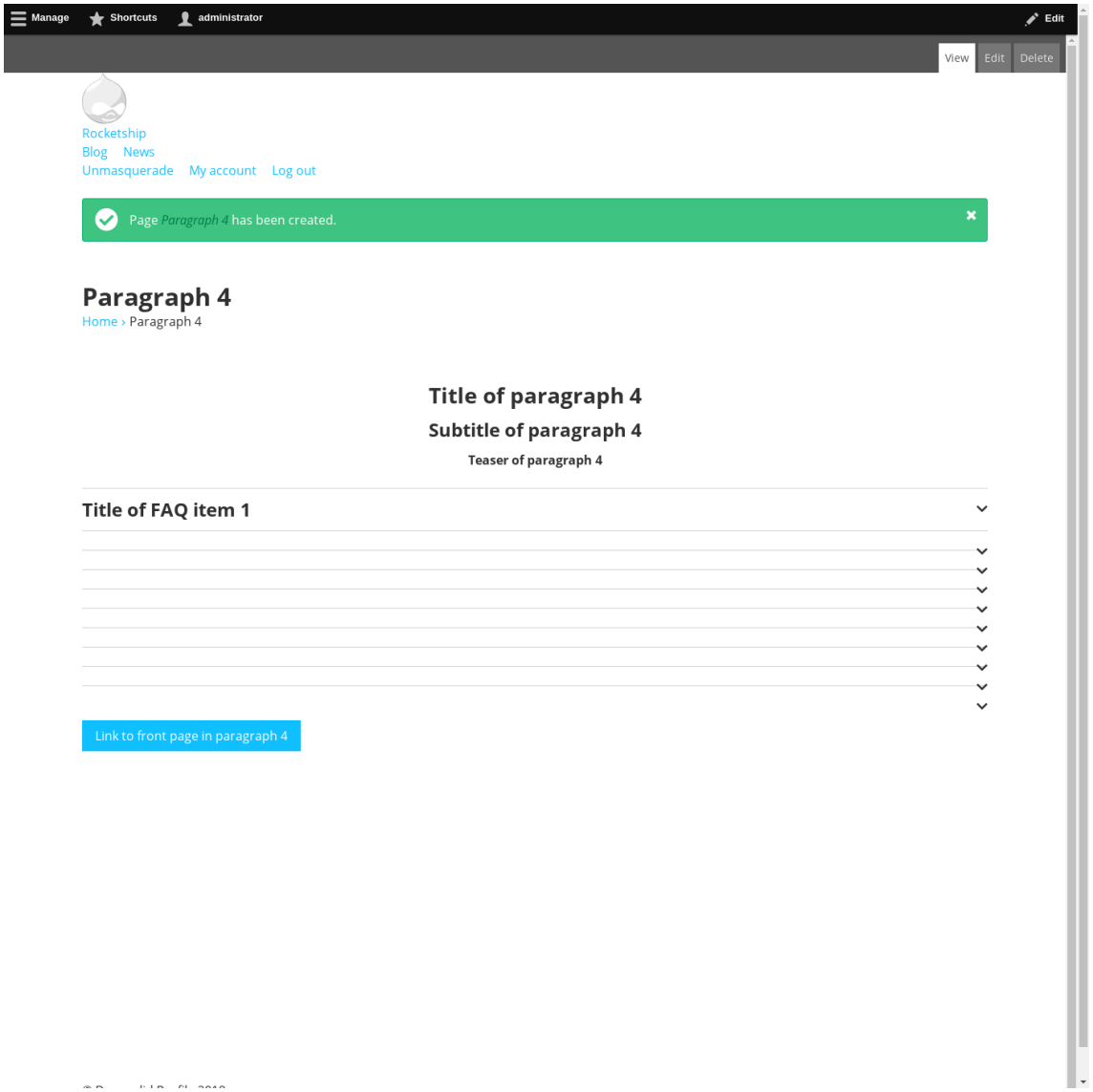
### Paragraph 3: Text main

De derde Paragraph is een blok met een titel, een teaser, een tekstveld en een button die linkt naar een andere pagina.



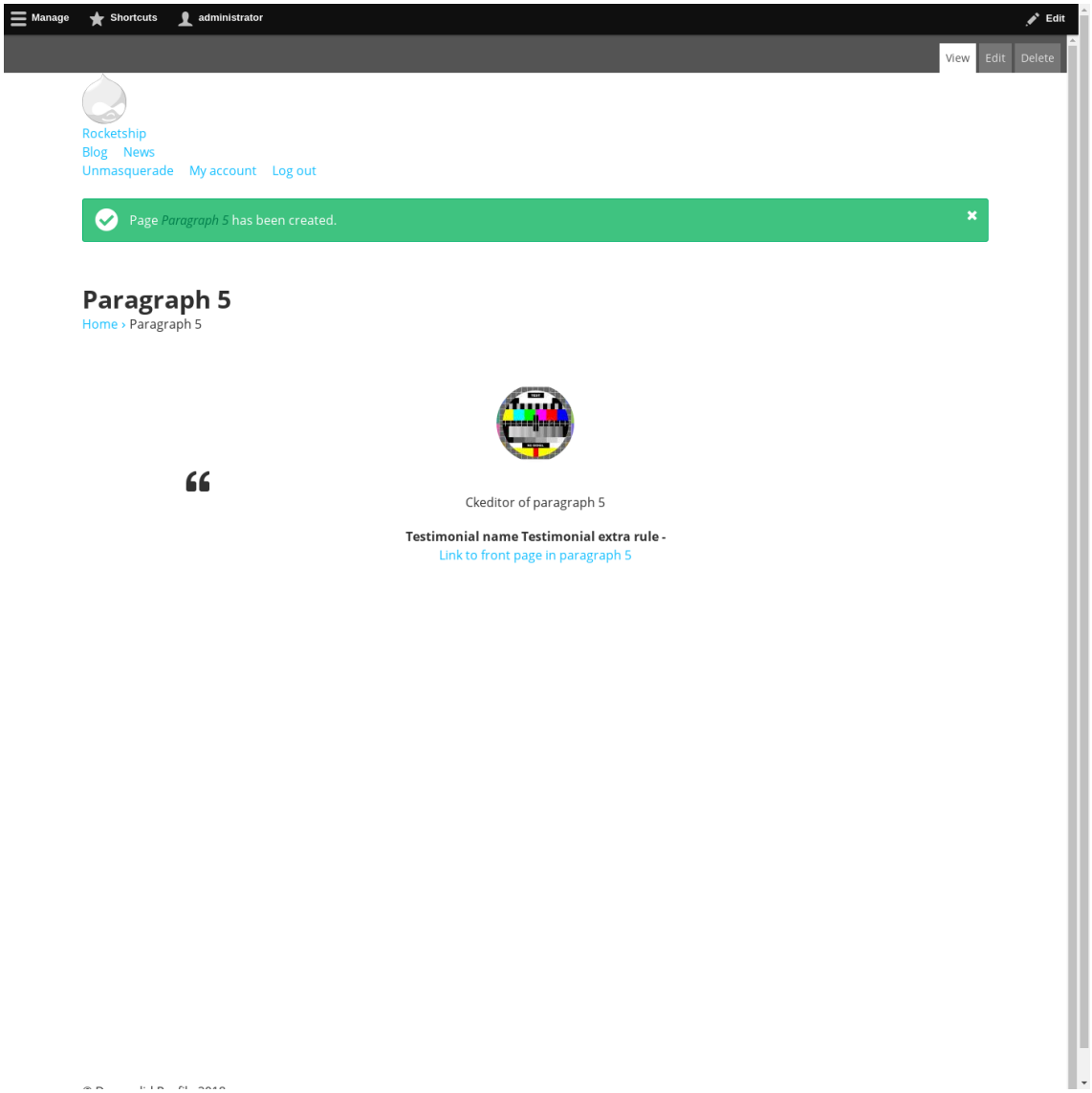
### Paragraph 4: FAQ

De vierde Paragraph is een blok met een titel, een subtitel en een teaser met daaronder veelgestgestelde vragen die kunnen open geklikt worden om het antwoord er op te zien.



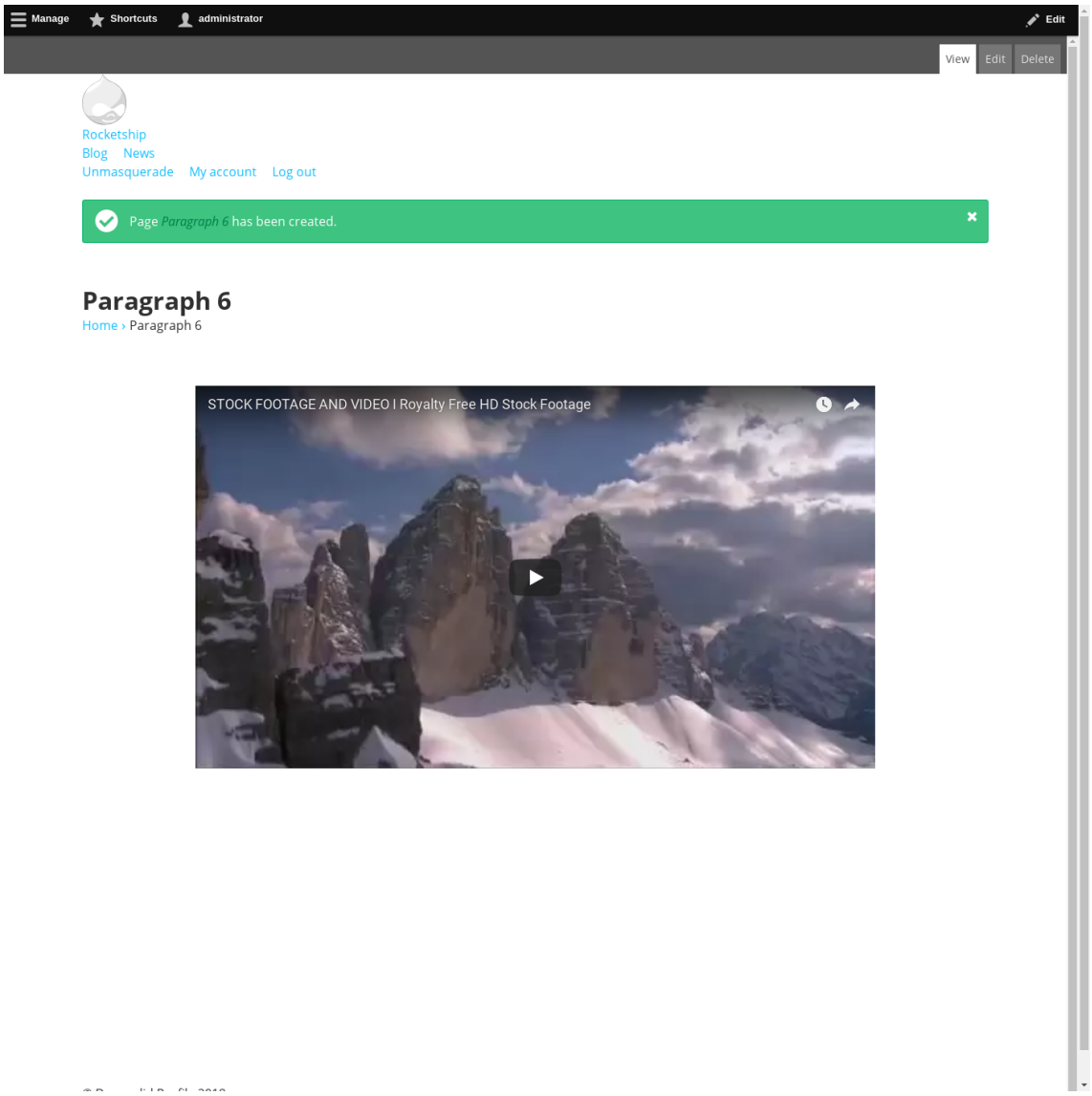
### Paragraph 5: Testimonial

De vijfde Paragraph is een blok die een afbeelding, een quote, een naam, een extra tekstveld en een link naar een andere pagina bevat.



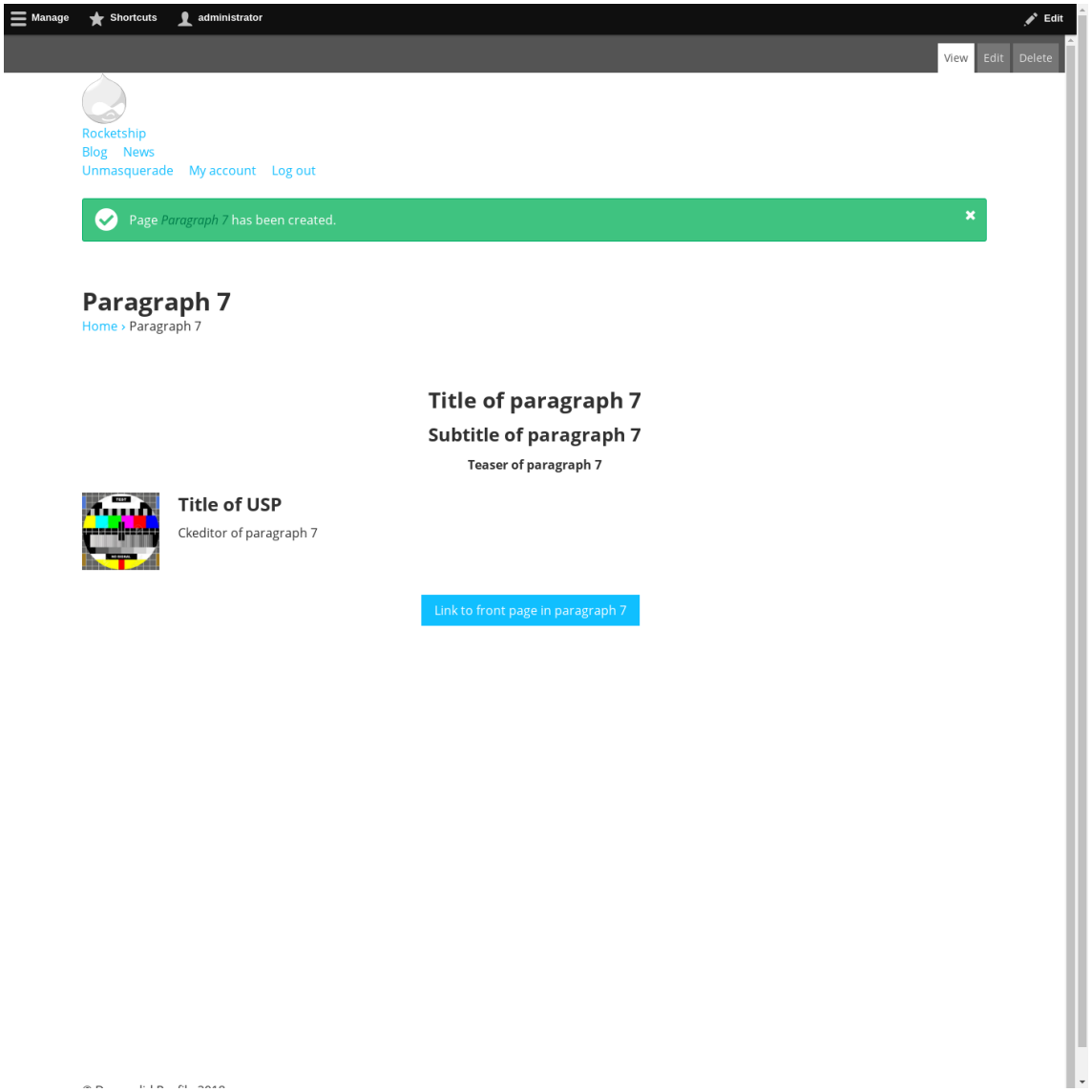
### Paragraph 6: Video

De zesde Paragraph is een blok die een youtube of vimeo video bevat.



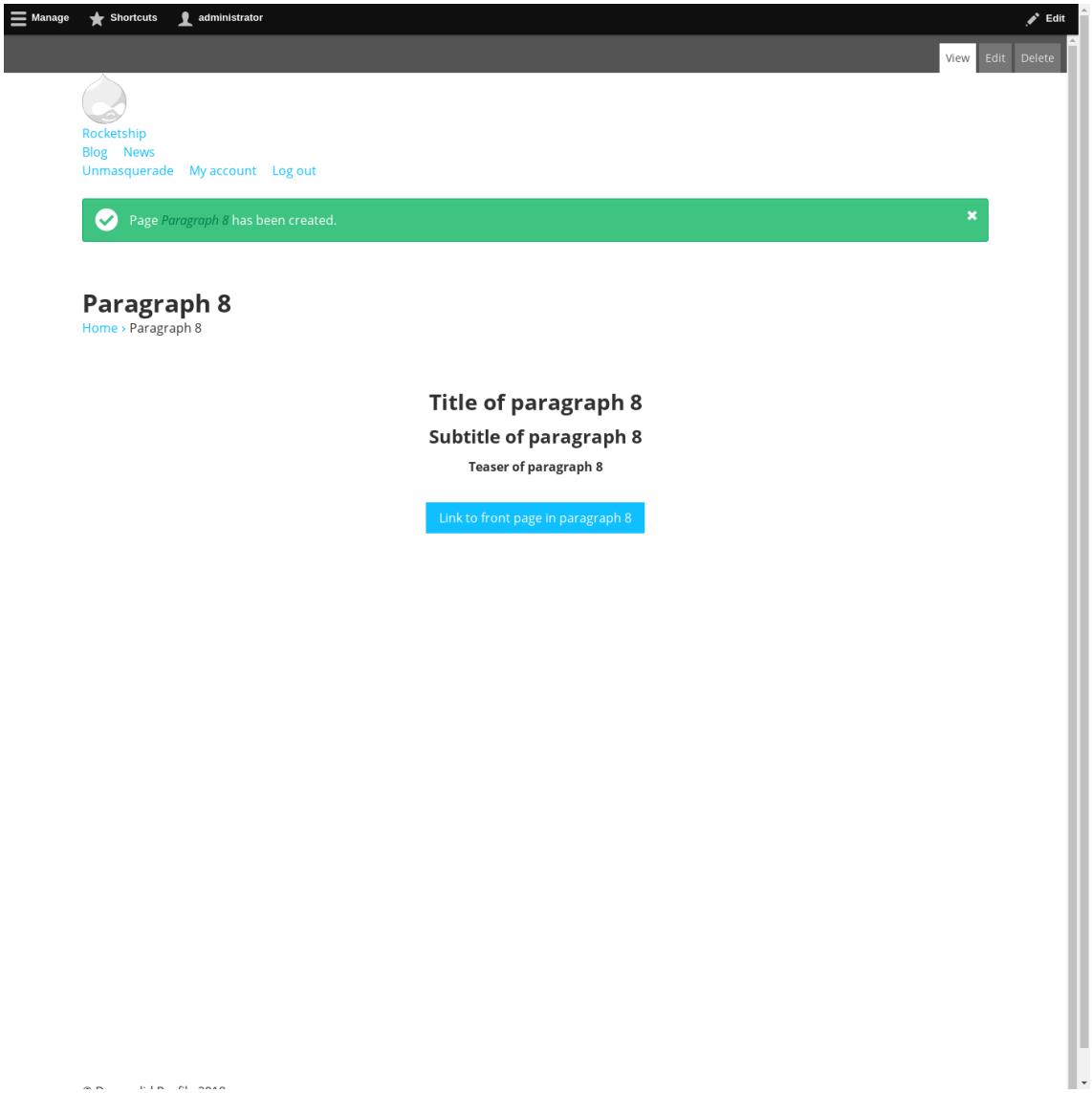
### Paragraph 7: USP

De zevende Paragraph is een blok met een titel, subtitel en teaser. Hieronder staat een USP item dat een afbeelding, titel en tekstveld bevat. Onder de USP items staat er ook nog eens een button die linkt naar een andere pagina.



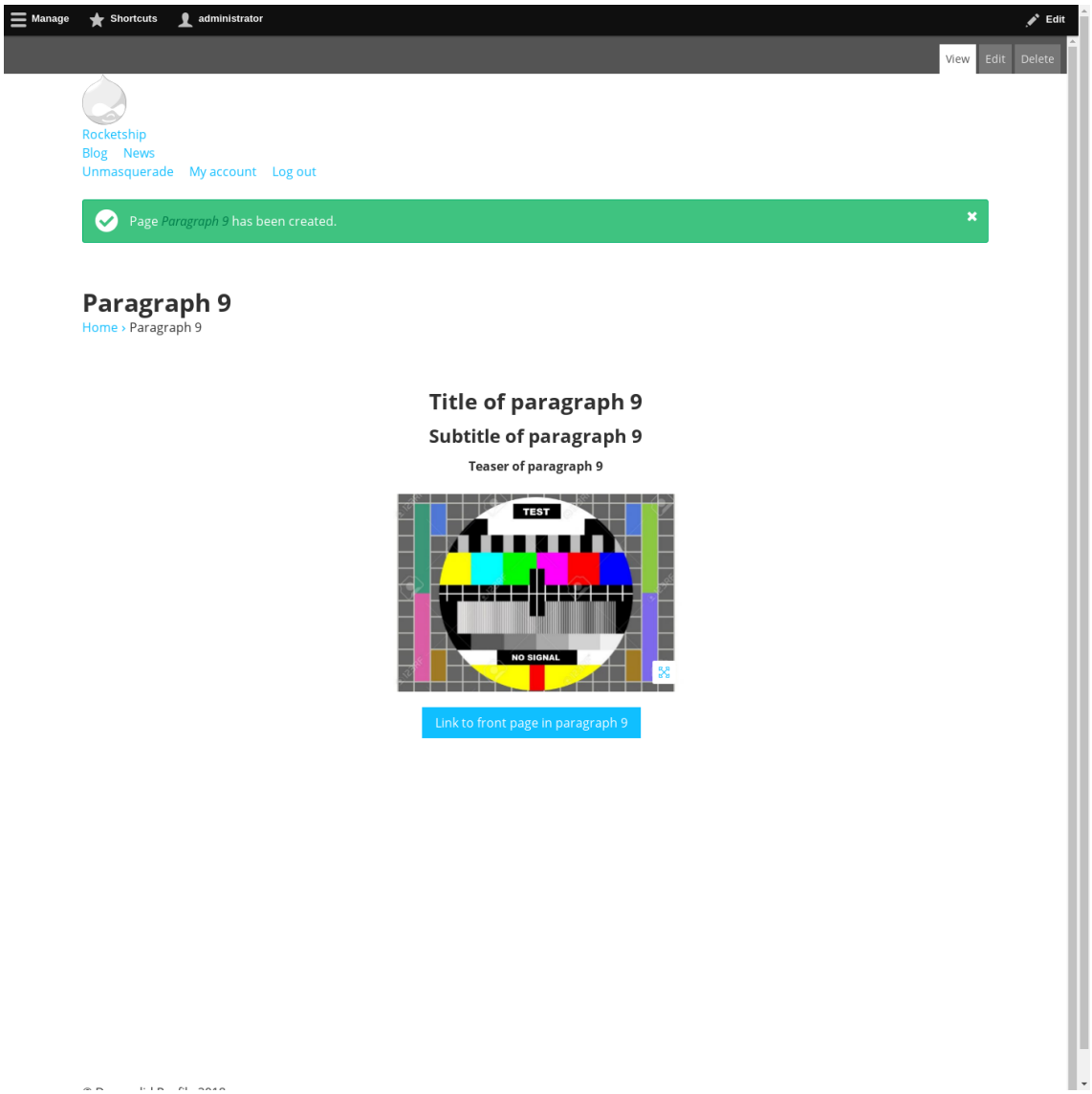
### Paragraph 8: Focus

De achtste Paragraph is gelijkaardig aan de derde Paragraph en is een blok met een titel, subtitel, teaser en button die linkt naar een andere pagina.



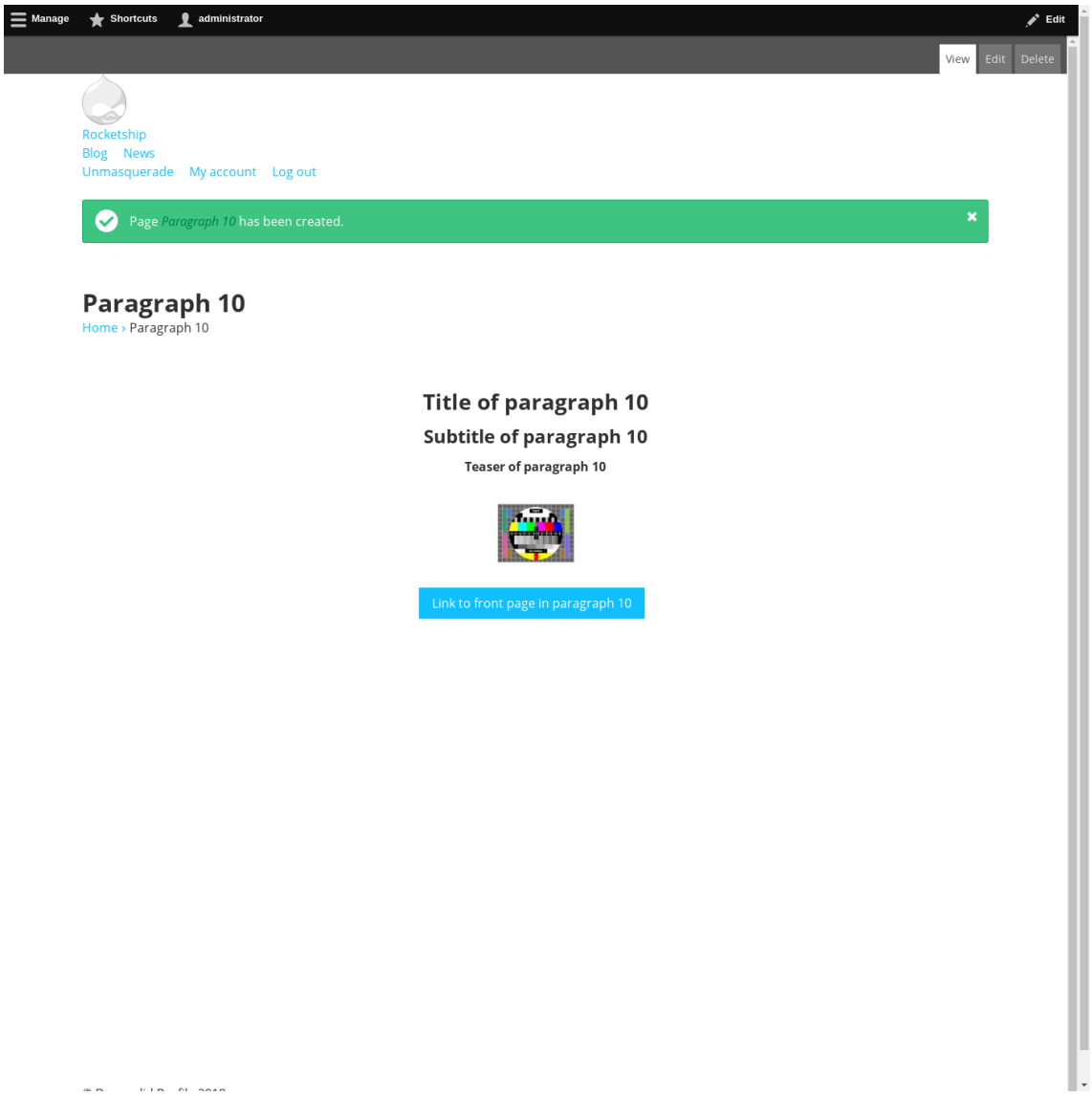
### Paragraph 9: Photo gallery

De negende Paragraph is een blok met een titel, subtitel en teaser. Hieronder staan afbeeldingen die linken naar een andere pagina en de blok wordt afgesloten met een button die ook een link bevat.



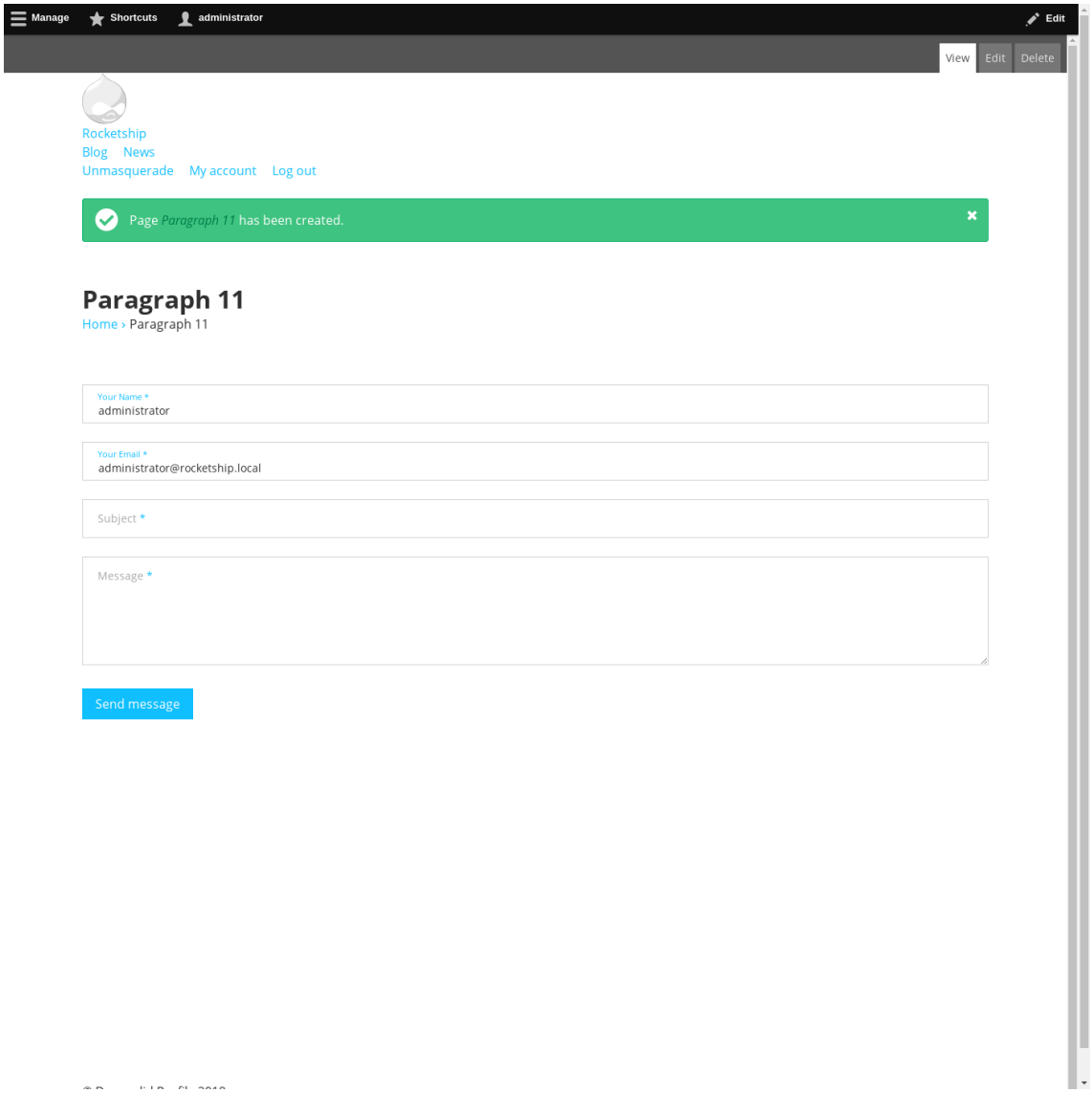
### Paragraph 10: Logo bar

De tiende Paragraph is een blok met een titel, subtitel, teaser, logo en button met een link.



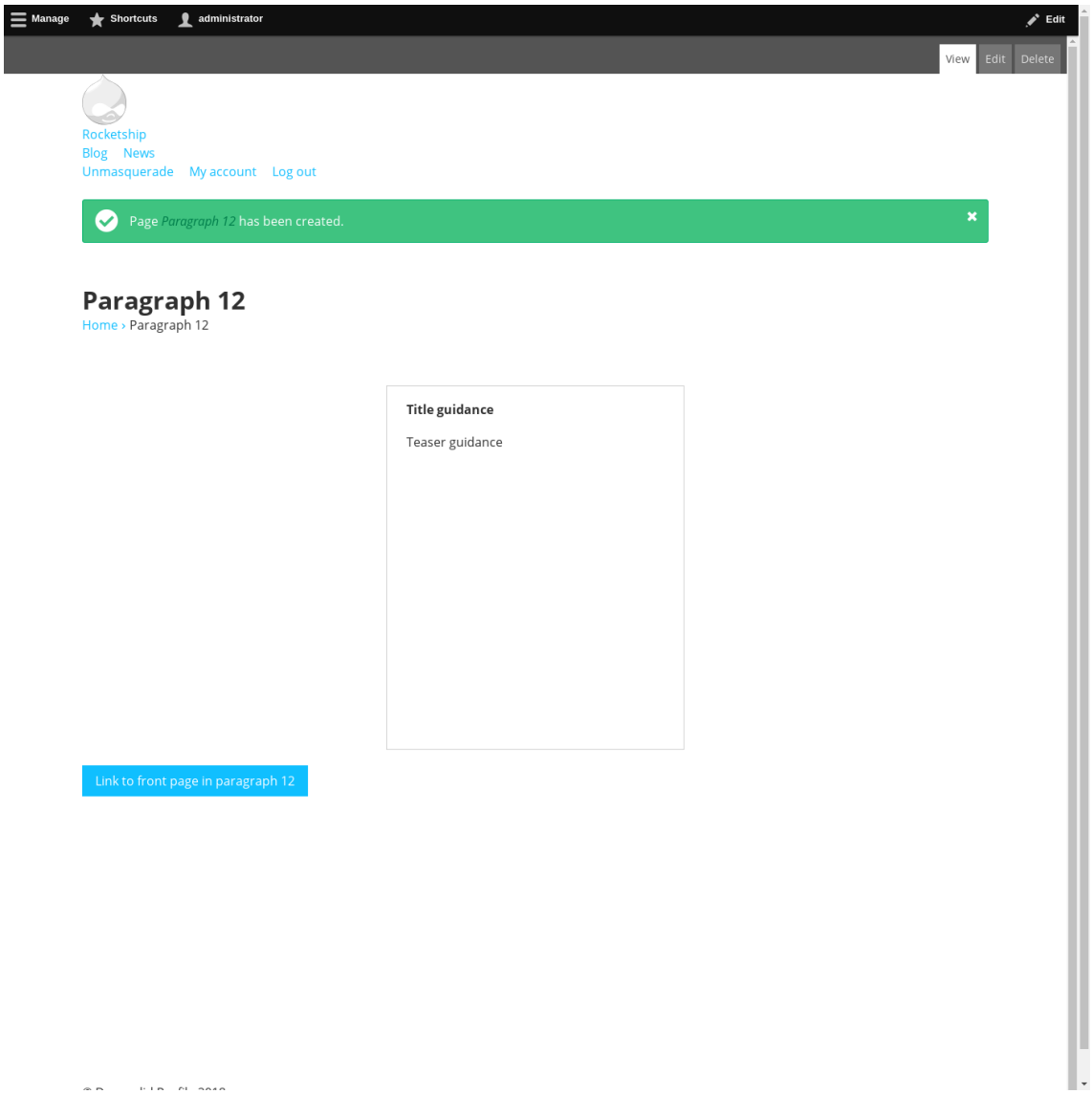
### Paragraph 11: Form

De elfde Paragraph is een blok met een formulier. Momenteel is het contact formulier het enige formulier dat met deze Paragraph kan aangemaakt worden.



### Paragraph 12: Guidance

De twaalfde Paragraph is een blok die Guidance items bevat en deze items kunnen op verschillende manieren op de pagina geplaatsts worden, bijvoorbeeld per 2 of per 4 naast elkaar. Zo een Guidance item bevat een titel, een teaser en linkt naar een andere pagina. De blok wordt afgesloten met een button die een link bevat.



## 5.2 Installatie en configuratie

De installatie begon net zoals in het hoofdstuk methodologie met de lokale en globale installatie van Nightwatch met de respectievelijke commando's "npm install nightwatch npm install -g nightwatch".

Voor deze proof of concept wordt gebruik gemaakt van de Server (2018) om de mogelijkheden in verband met het parallel uitvoeren van testen en cross-browser testen aan te tonen. ChromeDriver (2018) en GeckoDriver (2018) worden gedownload met "npm install chromedriver geckodriver". Hiermee kunnen de testen uitgevoerd worden op Google Chrome en op Mozilla Firefox.

Er werden twee configuratiebestanden aangemaakt. Een globals.js bestand dat de globale variabelen bevat en een nightwatch.json bestand dat de hoofdconfiguratie bevat. Hierin staat de configuratie voor het parallel uitvoeren van testen, de configuratie om Selenium Standalone Server, ChromeDriver en GeckoDriver te gebruiken. Hierin staan ook vijf omgevingen gedefinïeerd. De eerste omgeving is de standaard Chrome omgeving. De tweede omgeving is de Chrome desktop omgeving die de testen uitvoert op de Chrome browser met een resolutie van 1366 pixels breed en 768 pixels hoog. De derde omgeving is de Chrome headless omgeving. De vierde omgeving is de Firefox desktop omgeving die de testen uitvoert op de Firefox browser met een resolutie van 1366 pixels breed en 768 pixels hoog. De vijfde en laatste omgeving is de Firefox headless omgeving.

### globals.js

```
module.exports = {
  adminUsername: 'administrator',
  adminPassword: 'admin',
  timeoutTime: 10000,
  launch_url: 'http://rocketship.local',
  env: 'default',
  chromeDesktop: {
    adminUsername: 'administrator',
    adminPassword: 'admin',
    env: 'chromeDesktop'
  },
  chromeHeadless: {
    adminUsername: 'administrator',
    adminPassword: 'admin',
    env: 'chromeHeadless',
    headless: true
  },
  firefoxDesktop: {
    adminUsername: 'administrator',
    adminPassword: 'admin',
    env: 'firefoxDesktop'
  },
  firefoxHeadless: {
    adminUsername: 'administrator',
    adminPassword: 'admin',
    env: 'firefoxHeadless',
    headless: true
  }
}
```

### nightwatch.json

```
{
  "src_folders": [
    "nightwatch/tests"
  ],
  "output_folder": "reports",
  "custom_commands_path": "nightwatch/commands",
  "custom_assertions_path": "",
  "page_objects_path": "",
  "globals_path": "globals.js",
  "selenium": {
    "start_process": true,
    "server_path": "./selenium-server-standalone.jar",
    "log_path": "",
    "port": 4444,
    "cli_args": {
      "webdriver.chrome.driver": "./node_modules/.bin/chromedriver",
      "webdriver.gecko.driver": "./node_modules/.bin/geckodriver"
    }
  },
  "test_workers": {
    "enabled": true,
    "workers": "auto"
  },
  "test_settings": {
    "default": {
      "desiredCapabilities": {
        "browserName": "chrome"
      }
    },
    "chromeDesktop": {
      "desiredCapabilities": {
        "browserName": "chrome",
        "chromeOptions": {
          "args": [
            "window-size=1366,768"
          ]
        }
      }
    },
    "chromeHeadless": {
      "desiredCapabilities": {
        "browserName": "chrome",
        "chromeOptions": {
          "args": [
            "headless"
          ]
        }
      }
    },
    "firefoxDesktop": {
      "desiredCapabilities": {
        "browserName": "firefox",
        "moz:firefoxOptions": {
          "args": [
            "window-size=1366,768"
          ]
        }
      }
    },
    "firefoxHeadless": {
      "desiredCapabilities": {
        "browserName": "firefox",
        "moz:firefoxOptions": {
          "args": [
            "-headless"
          ]
        }
      }
    }
  }
}
```

```
71 }
```

## 5.3 Testsuite

Elke test start met het openen van een browser, te surfen naar de website en in te loggen als administrator. Dan wordt een pagina aangemaakt met de Paragraph die bij de test hoort en worden de velden van de Paragraph ingevuld. Ten slotte wordt elke test afgesloten met het uitloggen en sluiten van de browser.

### Lijst van testen

1. Test Paragraph 1
2. Test Paragraph 2
3. Test Paragraph 3
4. Test Paragraph 4
5. Test Paragraph 5
6. Test Paragraph 6
7. Test Paragraph 7
8. Test Paragraph 8
9. Test Paragraph 9
10. Test Paragraph 10
11. Test Paragraph 11
12. Test Paragraph 12

### Uitvoeringstijd testsuite

| | |
|---|---|
| Google Chrome standaard | 1 minuut 44 seconden |
| Google Chrome desktop | 1 minuut 59 seconden |
| Google Chrome headless | 1 minuut 19 seconden |
| Mozilla Firefox desktop | 2 minuten 10 seconden |
| Mozilla Firefox headless | 1 minuut 36 seconden |

### Test Paragraph 1

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- titel en subtitel invullen
- tekstveld invullen
- afbeelding met alt tekst toevoegen
- de viewmode selecteren, in deze test image_right
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel en subtitel correct zijn
- controleren dat het tekstveld correct is
- controleren dat de alt tekst van de afbeelding correct is
- controleren dat de correcte view mode is gebruikt
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph001.js

```
module.exports = {
  tags: ['paragraph1'],

  before: function (browser) {
    browser
      .url(browser.globals.launch_url)
      .drupalLoginAsAdmin();
  },

  after: function (browser) {
    browser
      .drupalDeletePage()
      .drupalLogout()
      .end();
  },

  'creating a page with paragraph 1 and asserting the paragraph is created correctly': (browser) => {
    browser
      .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 1', pageDescription: 'Page with paragraph 1', paragraphValue: 'p_001' })
      .drupalInputParagraphTitle({ title: 'Title of paragraph 1', subtitle: 'Subtitle of paragraph 1' })
      .drupalInputParagraphCkeditor({ text: 'Ckeditor of paragraph 1' })
      .drupalInputParagraphImage({ image: 'img.jpg', alt: 'Alt of image in paragraph 1', viewmode: 'image_right' })
      .drupalInputParagraphViewmode({ viewmodep001: 'image_right' })
      .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in paragraph 1', target: '_self' })
      .drupalSubmit()
      .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env + '/p001.png', headless: browser.globals.headless })
      .drupalAssertParagraphTitle({ title: 'Title of paragraph 1', subtitle: 'Subtitle of paragraph 1' })
      .drupalAssertParagraphCkeditor({ text: 'Ckeditor of paragraph 1' })
      .drupalAssertParagraphImage({ alt: 'Alt of image in paragraph 1' })
      .drupalAssertParagraphViewmode({ viewmodep001: 'image_right' })
      .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: 'Link to front page in paragraph 1' });
```

```
32      },
33  };
```

### Test Paragraph 2

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- afbeelding met alt tekst toevoegen
- link toevoegen aan de afbeelding
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de alt tekst van de afbeelding correct is
- controleren dat de link op de afbeelding werkt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph002.js

```
1  module.exports = {
2    tags: ['paragraph2'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 2 and asserting the paragraph is created correctly
         ': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 2', pageDescription: '
             Page with paragraph 2', paragraphValue: 'p_002' })
20       .drupalInputParagraphImage({ image: 'img.jpg', alt: 'Alt of image in paragraph
              2' })
21       .drupalInputParagraphLink({ url: '<front>', title: 'Link to front page in
             paragraph 2', target: '_self' })
22       .drupalSubmit()
23       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p002.png', headless: browser.globals.headless })
24       .drupalAssertParagraphImage({ alt: 'Alt of image in paragraph 2' })
25       .drupalAssertParagraphLink({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 2' });
26   },
27 };
```

### Test Paragraph 3

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- titel en teaser invullen
- tekstveld invullen
- de viewmode selecteren, in deze test centered
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel en teaser correct zijn
- controleren dat het tekstveld correct is
- controleren dat de correcte view mode is gebruikt
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel besproken

### testParagraph003.js

```
1  module.exports = {
2    tags: ['paragraph3'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 3 and asserting the paragraph is created correctly
         ': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 3', pageDescription: '
             Page with paragraph 3', paragraphValue: 'p_003' })
20       .drupalInputParagraphTitle({ title: 'Title of paragraph 3', teaser: 'Teaser of
              paragraph 3' })
21       .drupalInputParagraphViewmode({ viewmodep003: 'centered' })
22       .drupalInputParagraphCkeditor({ text: 'Ckeditor of paragraph 3' })
23       .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in
             paragraph 3', target: '_self' })
24       .drupalSubmit()
25       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p003.png', headless: browser.globals.headless })
26       .drupalAssertParagraphTitle({ title: 'Title of paragraph 3', teaser: 'Teaser
             of paragraph 3' })
27       .drupalAssertParagraphCkeditor({ text: 'Ckeditor of paragraph 3' })
28       .drupalAssertParagraphViewmode({ viewmodep003: 'centered' })
29       .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 3' });
30   },
31 };
```

### Test Paragraph 4

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- titel, subtitel en teaser invullen
- titel en tekstveld van een FAQ item invullen
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel, subtitel en teaser correct zijn
- controleren dat het FAQ item correct is aangemaakt
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel besproken

### testParagraph004.js

```
1  module.exports = {
2    tags: ['paragraph4'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 4 and asserting the paragraph is created correctly
         ': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 4', pageDescription: '
             Page with paragraph 4', paragraphValue: 'p_004' })
20       .drupalInputParagraphTitle({ title: 'Title of paragraph 4', subtitle: '
             Subtitle of paragraph 4', teaser: 'Teaser of paragraph 4' })
21       .drupalInputParagraphFaq({ text: 'Title of FAQ item 1' })
22       .drupalInputParagraphCkeditor({ text: 'Ckeditor of paragraph 4' })
23       .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in
             paragraph 4', target: '_self' })
24       .drupalSubmit()
25       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p004.png', headless: browser.globals.headless })
26       .drupalAssertParagraphTitle({ title: 'Title of paragraph 4', subtitle: '
             Subtitle of paragraph 4', teaser: 'Teaser of paragraph 4' })
27       .drupalAssertParagraphFaq({ title: 'Title of FAQ item 1', ckeditor: 'Ckeditor
             of paragraph 4' })
28       .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 4' });
29   },
30 };
```

### Test Paragraph 5

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- afbeelding met alt tekst toevoegen
- tekstveld invullen
- Testimonial naam en extra regel invullen
- Link aan testimonial toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat het tekstveld correct is
- controleren dat de naam en extra regel van het Testimonial item correct zijn
- controleren dat de alt tekst van de afbeelding correct is
- controleren dat de correcte view mode is gebruikt
- controleren dat de Testimonial link correct werkt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph005.js

```
module.exports = {
  tags: ['paragraph5'],

  before: function (browser) {
    browser
      .url(browser.globals.launch_url)
      .drupalLoginAsAdmin();
  },

  after: function (browser) {
    browser
      .drupalDeletePage()
      .drupalLogout()
      .end();
  },

  'creating a page with paragraph 5 and asserting the paragraph is created correctly': (browser) => {
    browser
      .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 5', pageDescription: 'Page with paragraph 5', paragraphValue: 'p_005' })
      .drupalInputParagraphImage({ image: 'img.jpg', alt: 'Alt of image in paragraph 5' })
      .drupalInputParagraphCkeditor({ text: 'Ckeditor of paragraph 5' })
      .drupalInputParagraphTestimonial({ name: 'Testimonial name', extrarule: 'Testimonial extra rule' })
      .drupalInputParagraphLink({ url: '<front>', title: 'Link to front page in paragraph 5', target: '_self' })
      .drupalSubmit()
      .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env + '/p005.png', headless: browser.globals.headless })
      .drupalAssertParagraphCkeditor({ text: 'Ckeditor of paragraph 5' })
      .drupalAssertParagraphTestimonial({ name: 'Testimonial name', extrarule: 'Testimonial extra rule' })
      .drupalAssertParagraphImage({ alt: 'Alt of image in paragraph 5' })
      .drupalAssertParagraphLink({ url: browser.globals.launch_url + '/', title: 'Link to front page in paragraph 5' });
  },
};
```

### Test Paragraph 6

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- video link toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren de video is correct is toegevoegd
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph006.js

```
1  module.exports = {
2    tags: ['paragraph6'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 6 and asserting the paragraph is created correctly
         ': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 6', pageDescription: '
             Page with paragraph 6', paragraphValue: 'p_006' })
20       .drupalInputParagraphVideo({ videourl: 'https://youtu.be/qvBX9VgPGSg' })
21       .drupalSubmit()
22       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p006.png', headless: browser.globals.headless })
23       .drupalAssertParagraphVideo();
24   },
25 };
```

### Test Paragraph 7

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- titel, subtitel en teaser invullen
- titel van het USP item invullen
- afbeelding met alt tekst toevoegen aan het USP item
- tekstveld van het USP item invullen
- link aan het USP item toevoegen
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel, subtitel en teaser correct zijn
- controleren dat de titel en tekstveld van het USP item correct zijn
- controleren dat de alt tekst van de afbeelding van het USP item correct is
- controleren dat de link van het USP item correct werkt
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph007.js

```
1  module.exports = {
2    tags: ['paragraph7'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 7 and asserting the paragraph is created correctly
         ': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 7', pageDescription: '
             Page with paragraph 7', paragraphValue: 'p_007' })
20       .drupalInputParagraphTitle({ title: 'Title of paragraph 7', subtitle: '
             Subtitle of paragraph 7', teaser: 'Teaser of paragraph 7' })
21       .drupalInputParagraphUsp({ title: 'Title of USP' })
22       .drupalInputParagraphImage({ imagep007: 'img.jpg', altp007: 'Alt of image in
             paragraph 7' })
23       .drupalInputParagraphCkeditor({ text: 'Ckeditor of paragraph 7' })
24       .drupalInputParagraphLink({ urlp007: '<front>', titlep007: 'Link to front page
              in paragraph 7', targetp007: '_self' })
25       .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in
             paragraph 7', target: '_self' })
26       .drupalSubmit()
27       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p007.png', headless: browser.globals.headless })
28       .drupalAssertParagraphTitle({ title: 'Title of paragraph 7', subtitle: '
             Subtitle of paragraph 7', teaser: 'Teaser of paragraph 7' })
29       .drupalAssertParagraphUsp({ title: 'Title of USP', text: 'Ckeditor of
             paragraph 7' })
```

```
30          .drupalAssertParagraphImage({ altp007: 'Alt of image in paragraph 7' })
31          .drupalAssertParagraphLink({ urlp007: browser.globals.launch_url + '/' })
32          .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: '
                Link to front page in paragraph 7' });
33    },
34  };
```

### Test Paragraph 8

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- titel, subtitel en teaser invullen
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel, subtitel en teaser correct zijn
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph008.js

```
1  module.exports = {
2    tags: ['paragraph8'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 8 and asserting the paragraph is created correctly
         ': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 8', pageDescription: '
             Page with paragraph 8', paragraphValue: 'p_008' })
20       .drupalInputParagraphTitle({ title: 'Title of paragraph 8', subtitle: '
             Subtitle of paragraph 8', teaser: 'Teaser of paragraph 8' })
21       .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in
             paragraph 8', target: '_self' })
22       .drupalSubmit()
23       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p008.png', headless: browser.globals.headless })
24       .drupalAssertParagraphTitle({ title: 'Title of paragraph 8', subtitle: '
             Subtitle of paragraph 8', teaser: 'Teaser of paragraph 8' })
25       .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 8' });
26   },
27 };
```

### Test Paragraph 9

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- titel, subtitel en teaser invullen
- afbeelding met alt tekst toevoegen
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel, subtitel en teaser correct zijn
- controleren dat de alt tekst van de afbeelding correct is
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph009.js

```
1  module.exports = {
2    tags: ['paragraph9'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 9 and asserting the paragraph is created correctly
         ': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 9', pageDescription: '
             Page with paragraph 9', paragraphValue: 'p_009' })
20       .drupalInputParagraphTitle({ title: 'Title of paragraph 9', subtitle: '
             Subtitle of paragraph 9', teaser: 'Teaser of paragraph 9' })
21       .drupalInputParagraphImage({ imagep009: 'img.jpg', altp009: 'Alt of image in
             paragraph 9' })
22       .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in
             paragraph 9', target: '_self' })
23       .drupalSubmit()
24       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p009.png', headless: browser.globals.headless })
25       .drupalAssertParagraphTitle({ title: 'Title of paragraph 9', subtitle: '
             Subtitle of paragraph 9', teaser: 'Teaser of paragraph 9' })
26       .drupalAssertParagraphImage({ altp009: 'Alt of image in paragraph 9' })
27       .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 9' });
28   },
29 };
```

### Test Paragraph 10

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- titel, subtitel en teaser invullen
- logo met alt tekst toevoegen
- link aan het logo toevoegen
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel, subtitel en teaser correct zijn
- controleren dat de alt tekst van het logo correct is
- controleren dat de link van het logo correct werkt
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph010.js

```
1  module.exports = {
2    tags: ['paragraph10'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 10 and asserting the paragraph is created
         correctly': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 10', pageDescription: '
             Page with paragraph 10', paragraphValue: 'p_010' })
20       .drupalInputParagraphTitle({ title: 'Title of paragraph 10', subtitle: '
             Subtitle of paragraph 10', teaser: 'Teaser of paragraph 10' })
21       .drupalInputParagraphImage({ imagep010: 'img.jpg', altp010: 'Alt of image in
             paragraph 10' })
22       .drupalInputParagraphLink({ urlp010: '<front>', titlep010: 'Link to front page
              in paragraph 10', targetp010: '_self' })
23       .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in
             paragraph 10', target: '_self' })
24       .drupalSubmit()
25       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p010.png', headless: browser.globals.headless })
26       .drupalAssertParagraphTitle({ title: 'Title of paragraph 10', subtitle: '
             Subtitle of paragraph 10', teaser: 'Teaser of paragraph 10' })
27       .drupalAssertParagraphImage({ alt: 'Alt of image in paragraph 10' })
28       .drupalAssertParagraphLink({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 10' })
29       .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 10' });
30   },
31 };
```

### Test Paragraph 11

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- contact formulier selecteren
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat het contact formulier correct is aangemaakt en werkt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph011.js

```
1  module.exports = {
2    tags: ['paragraph11'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 11 and asserting the paragraph is created
         correctly': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 11', pageDescription: '
             Page with paragraph 11', paragraphValue: 'p_011' })
20       .drupalInputParagraphForm({ form: 'contact' })
21       .drupalSubmit()
22       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p011.png', headless: browser.globals.headless })
23       .drupalAssertParagraphForm({ subject: 'Subject of contact form', message: '
             Message of contact form' });
24   },
25 };
```

### Test Paragraph 12

Opbouw test:

- test opstarten zoals in het begin van dit deel toegelicht
- de viewmode selecteren, hier default_large
- titel en teaser van Guidance item invullen
- afbeelding met alt tekst toevoegen aan het Guidance item
- link aan het Guidance item toevoegen
- button met link en link tekst toevoegen
- pagina opslaan en aanmaken
- screenshot nemen van de aangemaakte pagina
- controleren dat de titel en teaser van het Guidance item correct zijn
- controleren dat de link van het Guidance item correct werkt
- controleren dat de button correct is aangemaakt
- test afsluiten zoals in het begin van dit deel toegelicht

### testParagraph012.js

```
1  module.exports = {
2    tags: ['paragraph12'],
3
4    before: function (browser) {
5      browser
6        .url(browser.globals.launch_url)
7        .drupalLoginAsAdmin();
8    },
9
10   after: function (browser) {
11     browser
12       .drupalDeletePage()
13       .drupalLogout()
14       .end();
15   },
16
17   'creating a page with paragraph 12 and asserting the paragraph is created
         correctly': (browser) => {
18     browser
19       .drupalCreatePageWithParagraph({ pageTitle: 'Paragraph 12', pageDescription: '
             Page with paragraph 12', paragraphValue: 'p_012' })
20       .drupalInputParagraphViewmode({ viewmodep012: 'default_large' })
21       .drupalInputParagraphGuidance({ title: 'Title guidance', teaser: 'Teaser
             guidance' })
22       .drupalInputParagraphImage({ imagep012: 'img.jpg', altp012: 'Alt of image in
             paragraph 12' })
23       .drupalInputParagraphLink({ urlp012: '<front>', titlep012: 'Link to front page
              in paragraph 12', targetp012: '_self' })
24       .drupalInputParagraphButton({ url: '<front>', title: 'Link to front page in
             paragraph 12', target: '_self' })
25       .drupalSubmit()
26       .drupalScreenshot({ path: './nightwatch/screenshots/' + browser.globals.env +
             '/p012.png', headless: browser.globals.headless })
27       .drupalAssertParagraphGuidance({ title: 'Title guidance', teaser: 'Teaser
             guidance' })
28       .drupalAssertParagraphLink({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 12' })
29       .drupalAssertParagraphButton({ url: browser.globals.launch_url + '/', title: '
             Link to front page in paragraph 12' });
30   },
31 };
```

## 5.4 Commando's

Voor de testen zo leesbaar en gebruiksvriendelijk mogelijk te maken werden er commando's geschreven die de achterliggende code weg abstraheren.

### Lijst van commando's

1. drupalAssertParagraphButton
2. drupalAssertParagraphCkeditor
3. drupalAssertParagraphFaq
4. drupalAssertParagraphForm
5. drupalAssertParagraphGuidance
6. drupalAssertParagraphImage
7. drupalAssertParagraphLink
8. drupalAssertParagraphTestimonial
9. drupalAssertParagraphTitle
10. drupalAssertParagraphUsp
11. drupalAssertParagraphVideo
12. drupalAssertParagraphViewmode
13. drupalCreatePageWithParagraph
14. drupalDeletePage
15. drupalInputParagraphButton
16. drupalInputParagraphCkeditor
17. drupalInputParagraphFaq
18. drupalInputParagraphForm
19. drupalInputParagraphGuidance
20. drupalInputParagraphImage
21. drupalInputParagraphLink
22. drupalInputParagraphTestimonial
23. drupalInputParagraphTitle
24. drupalInputParagraphUsp
25. drupalInputParagraphVideo
26. drupalInputParagraphViewmode
27. drupalLogin
28. drupalLoginAsAdmin
29. drupalLogout
30. drupalRelativeURL
31. drupalScreenshot
32. drupalSubmit
33. drupalUserIsLoggedIn

### drupalAssertParagraphButton

Dit commando gaat na of een button op een Paragraph correct is aangemaakt door de titel te vergelijken met de waarde meegegeven en door op de button te klikken en de url van de bezochte pagina te vergelijken met de meegegeven waarde.

### drupalAssertParagraphButton.js

```
1  exports.command = function drupalAssertParagraphButton({ title, url }) {
2    if (typeof title !== 'undefined') {
3      this.assert.containsText('.paragraph .field--name-field-p-button a', title);
4    }
5
6    if (typeof url !== 'undefined') {
7      this.waitForElementVisible('.paragraph .field--name-field-p-button a',
           this.globals.timeoutTime)
8        .click('.paragraph .field--name-field-p-button a')
9        .assert.urlEquals(url)
10       .back();
11   }
12
13   return this;
14 };
```

### drupalAssertParagraphCkeditor

Dit commando gaat na of het tekstveld op een Paragraph correct is aangemaakt door het te vergelijken met de meegegeven waarde.

### drupalAssertParagraphCkeditor.js

```
1  exports.command = function drupalAssertParagraphCkeditor({ text }) {
2    if (typeof text !== 'undefined') {
3      this.assert.containsText('.paragraph .field--name-field-p-text p', text);
4    }
5
6    return this;
7  };
```

### drupalAssertParagraphFaq

Dit commando gaat na of de FAQ Paragraph correct is aangemaakt door de titel en het tekstveld te vergelijken met de meegegeven waarden.

### drupalAssertParagraphFaq.js

```
1  exports.command = function drupalAssertParagraphFaq({ title, ckeditor }) {
2    if (typeof title !== 'undefined') {
3      this.assert.containsText('.paragraph--type-p-004 .field__item--name-field-p-004-
           item:first-of-type h3', title);
4    }
5
6    if (typeof ckeditor !== 'undefined') {
7      this.waitForElementVisible('.paragraph--type-p-004 .field__item--name-field-p
           -004-item:first-of-type h3', this.globals.timeoutTime)
8        .click('.paragraph--type-p-004 .field__item--name-field-p-004-item:first-of-
             type h3')
9        .assert.containsText('.paragraph--type-p-004 .field__item--name-field-p-004-
             item:first-of-type .tab-item__content', ckeditor);
10   }
11
12   return this;
13 };
```

### drupalAssertParagraphForm

Dit commando gaat na of de Paragraph met een contact formulier correct is aangemaakt.

### drupalAssertParagraphForm.js

```
1  exports.command = function drupalAssertParagraphForm({ subject, message }) {
2    this.waitForElementVisible('input[name="name"]', this.globals.timeoutTime)
3      .assert.attributeEquals('input[name="name"]', 'value',
           this.globals.adminUsername)
4      .assert.attributeEquals('input[name="email"]', 'value',
           this.globals.adminUsername + '@rocketship.local');
5
6    if (typeof subject !== 'undefined') {
7      this.setValue('input[name="subject"]', subject);
8    }
9
10   if (typeof message !== 'undefined') {
11     this.setValue('textarea[name="message"]', message);
12   }
13
14   if (typeof subject !== 'undefined' && typeof message !== 'undefined') {
15     this.click('#edit-actions-submit')
16       .assert.urlEquals(this.globals.launch_url + '/')
17       .back();
18   }
19   return this;
20 };
```

### drupalAssertParagraphGuidance

Dit commando gaat na of de Paragraph Guidance titel en teaser correct zijn aangemaakt door deze te vergelijken met de meegegeven waarde.

### drupalAssertParagraphGuidance.js

```
1  exports.command = function drupalAssertParagraphGuidance({ title , teaser }) {
2    if (typeof title !== 'undefined') {
3      this.assert.containsText('.paragraph--type-p-012 .paragraph--type-p-012-child .
           field--name-field-p-title h4', title);
4    }
5
6    if (typeof teaser !== 'undefined') {
7      this.assert.containsText('.paragraph--type-p-012 .paragraph--type-p-012-child .
           field--name-field-p-teaser', teaser);
8    }
9
10   return this;
11 };
```

### drupalAssertParagraphImage

Dit commando gaat na of een Paragraph afbeelding correct is aangemaakt door de alt tag te vergelijken met de meegegeven waarde. Voor de afbeeldingen van Paragraph 7 en 9 moet een ander argument meegegeven worden.

### drupalAssertParagraphImage.js

```
1  exports.command = function drupalAssertParagraphImage({ alt, altp007, altp009 }) {
2    if (typeof alt !== 'undefined') {
3      this.assert.attributeEquals('.paragraph .field--name-field-p-image img', 'alt',
           alt);
4    }
5
6    if (typeof altp007 !== 'undefined') {
7      this.assert.attributeEquals('.paragraph--type-p-007 .field__item--name-field-p
           -007-children .field--name-image-url-field img', 'alt', altp007);
8    }
9
10   if (typeof altp009 !== 'undefined') {
11     this.assert.attributeEquals('.paragraph--type-p-009 .field--name-field-p-images-
           unlimited img', 'alt', altp009);
12   }
13
14   return this;
15 };
```

### drupalAssertParagraphLink

Dit commando gaat na of de Paragraph link correct is aangemaakt door de titel er van te vergelijken met de meegegeven waarde en door op de link te klikken en de url te vergelijken met de meegegeven waarde. Voor Paragraph 7 moet een andere url argument meegegeven worden.

### drupalAssertParagraphLink.js

```
1  exports.command = function drupalAssertParagraphLink({ title, url, urlp007 }) {
2    if (typeof title !== 'undefined') {
3      this.assert.containsText('.paragraph .field--name-field-p-link  a', title);
4    }
5
6    if (typeof url !== 'undefined') {
7      this.waitForElementVisible('.paragraph .field--name-field-p-link  a',
           this.globals.timeoutTime)
8        .click('.paragraph .field--name-field-p-link  a')
9        .assert.urlEquals(url)
10       .back();
11   }
12
13   if (typeof urlp007 !== 'undefined') {
14     this.waitForElementVisible('.paragraph--type-p-007 .field__item--name-field-p
           -007-children .field--name-image-url-field a', this.globals.timeoutTime)
15       .click('.paragraph--type-p-007 .field__item--name-field-p-007-children .field
             --name-image-url-field a')
16       .assert.urlEquals(urlp007)
17       .back();
18   }
19
20   return this;
21 };
```

### drupalAssertParagraphTestimonial

Dit commando gaat na of de Paragraph Testimonial correct is aangemaakt door de naam en extra regel te vergelijken met de meegegeven waarden.

### drupalAssertParagraphTestimonial.js

```
1  exports.command = function drupalAssertParagraphTestimonial({name, extrarule}) {
2    if(typeof name !== 'undefined') {
3      this.assert.containsText('.paragraph--type-p-005 .field--name-field-p-name',
           name);
4    }
5
6    if(typeof extrarule !== 'undefined') {
7      this.assert.containsText('.paragraph--type-p-005 .field--name-field-p-extra-rule
           ', extrarule);
8    }
9
10   return this;
11 };
```

### drupalAssertParagraphTitle

Dit commando gaat na of de Paragraph titel, subtitel of teaser correct zijn aangemaakt door deze te vergelijken met de meegegeven waarden.

### drupalAssertParagraphTitle.js

```
1  exports.command = function drupalAssertParagraphTitle({ title, subtitle, teaser }) {
2    if (typeof title !== 'undefined') {
3      this.assert.containsText('.paragraph .field--name-field-p-title h2', title);
4    }
5
6    if (typeof subtitle !== 'undefined') {
7      this.assert.containsText('.paragraph .field--name-field-p-subtitle h3', subtitle
           );
8    }
9
10   if (typeof teaser !== 'undefined') {
11     this.assert.containsText('.paragraph .field--name-field-p-teaser', teaser);
12   }
13
14   return this;
15 };
```

### drupalAssertParagraphUsp

Dit commando gaat na of de Paragraph USP correct is aangemaakt door de titel en tekst er van te vergelijken met de meegegeven waarden.

### drupalAssertParagraphUsp.js

```
1  exports.command = function drupalAssertParagraphUsp({ title, text }) {
2    if (typeof title !== 'undefined') {
3      this.assert.containsText('.paragraph--type-p-007 .field__item--name-field-p-007-
           children .field--name-field-p-title h3', title);
4    }
5
6    if (typeof text !== 'undefined') {
7      this.assert.containsText('.paragraph--type-p-007 .field__item--name-field-p-007-
           children .field--name-field-p-text p', text);
8    }
9
10   return this;
11 };
```

### drupalAssertParagraphVideo

Dit commando gaat na of de Paragraph video is aangemaakt en een video bevat.

### drupalAssertParagraphVideo.js

```
1  exports.command = function drupalAssertParagraphVideo() {
2    this.waitForElementVisible('.paragraph--type-p-006 .field--type-video-embed-field
         iframe', this.globals.timeoutTime);
3
4    return this;
5  };
```

### drupalAssertParagraphViewmode

Dit commando gaat na of de juiste Paragraph viewmode is gebruikt. De argumenten die kunnen meegegeven worden zijn specifiek voor Paragraph 1 of Paragraph 3.

### drupalAssertParagraphViewmode.js

```
1  exports.command = function drupalAssertParagraphViewmode({ viewmodep001,
       viewmodep003 }) {
2    if (typeof viewmodep001 !== 'undefined') {
3      switch (viewmodep001) {
4        case 'image_right':
5          this.assert.cssProperty('.paragraph .field--image', 'float', 'right');
6          break;
7        case 'image_left':
8          this.assert.cssProperty('.paragraph .field--image', 'float', 'left');
9          break;
10     }
11   }
12
13   if (typeof viewmodep003 !== 'undefined') {
14     switch (viewmodep003) {
15       case 'centered':
16         this.assert.cssProperty('.p-003--view-mode--centered', 'text-align', 'center
               ');
17         break;
18       case 'left':
19         this.assert.cssProperty('.p-003--view-mode--left', 'text-align', 'left');
20         break;
21     }
22   }
23
24   return this;
25 };
```

### drupalCreatePageWithParagraph

Dit commando maakt een pagina aan met de titel en beschrijving die meegegeven worden en selecteert de Paragraph die hoort bij de waarde die meegegeven is.

### drupalCreatePageWithParagraph.js

```
1  exports.command = function drupalCreatePageWithParagraph({ pageTitle,
       pageDescription, paragraphValue }) {
2    if (typeof pageTitle !== 'undefined') {
3      this.drupalRelativeURL('/node/add/page')
4        .waitForElementVisible('input[id=edit-title-0-value]',
             this.globals.timeoutTime)
5        .setValue('input[id=edit-title-0-value]', pageTitle);
6    }
7
8    if (typeof pageDescription !== 'undefined') {
9      this.waitForElementVisible('#edit-field-description-0-value',
           this.globals.timeoutTime)
10       .setValue('#edit-field-description-0-value', pageDescription);
11   }
12
13   if (typeof paragraphValue !== 'undefined') {
14     this.waitForElementVisible('#edit-field-paragraphs-add-more-add-more-select
           option[value=' + paragraphValue + ']', this.globals.timeoutTime)
15       .click('#edit-field-paragraphs-add-more-add-more-select option[value=' +
             paragraphValue + ']')
16       .click('#edit-field-paragraphs-add-more-add-more-button')
17       .waitForElementVisible('input[value="Add another Paragraph"]',
             this.globals.timeoutTime);
18   }
19
20   return this;
21 };
```

### drupalDeletePage

Dit commando verwijdert de pagina waarop de gebruiker aanwezig is.

### drupalDeletePage.js

```
1  exports.command = function drupalDeletePage() {
2    this.waitForElementPresent('#block-dropsolid-starter-local-tasks',
         this.globals.timeoutTime)
3      .execute(function () {
4        document.querySelector('#block-dropsolid-starter-local-tasks .tabs__nav--local
             -tasks li:last-child a').click();
5      })
6      .drupalSubmit();
7
8    return this;
9  };
```

### drupalInputParagraphButton

Dit commando voert de waarden voor het aanmaken van een Paragraph button in met de waarden die zijn meegegeven. Dit zijn de titel van de button, de link van de button en de target (open de pagina in hetzelfde of in een ander tabblad).

### drupalInputParagraphButton.js

```
1  exports.command = function drupalInputParagraphButton({ url, title, target }) {
2    if (typeof url !== 'undefined') {
3      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_button][0][uri]"]', this.globals.timeoutTime)
4        .setValue('input[name="field_paragraphs[0][subform][field_p_button][0][uri]"]'
             , url);
5    }
6
7    if (typeof title !== 'undefined') {
8      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_button][0][title]"]', this.globals.timeoutTime)
9        .setValue('input[name="field_paragraphs[0][subform][field_p_button][0][title
             ]"]', title);
10   }
11
12   if (typeof target !== 'undefined') {
13     this.waitForElementVisible('select[name="field_paragraphs[0][subform][
           field_p_button][0][options][attributes][target]"] option[value= ' + target +
            ']', this.globals.timeoutTime)
14       .click('select[name="field_paragraphs[0][subform][field_p_button][0][options][
           attributes][target]"] option[value= ' + target + ']');
15   }
16
17   return this;
18 };
```

### drupalInputParagraphCkeditor

Dit commando voert de waarde van een Paragraph tekstveld in met de waarde die is meegegeven.

### drupalInputParagraphCkeditor.js

```
exports.command = function drupalInputParagraphCkeditor({ text }) {
  if (typeof text !== 'undefined') {
    var ckeditor;
    this.execute(
      function (instance, content) {
        for (var i in CKEDITOR.instances)
          CKEDITOR.instances[i].setData(content);
      },
      [
        ckeditor,
        text
      ]
    );
  }

  return this;
};
```

### drupalInputParagraphFaq

Dit commando voert de tekst van de Paragraph FAQ in met de waarde die is meegegeven.

### drupalInputParagraphFaq.js

```
1  exports.command = function drupalInputParagraphFaq({ text }) {
2    if (typeof text !== 'undefined') {
3      this.waitForElementVisible('.field--name-field-p-004-item tbody tr:first-child
           summary[role="button"]', this.globals.timeoutTime)
4        .click('.field--name-field-p-004-item tbody tr:first-child summary[role="
           button"]')
5        .setValue('input[name="field_paragraphs[0][subform][field_p_004_item][0][
           container_1][title]"]', text);
6    }
7
8    return this;
9  };
```

### drupalInputParagraphForm

Dit commando selecteert de waarde van de form die in Paragraph form kan aangemaakt worden met de waarde die is meegegeven.

### drupalInputParagraphForm.js

```
1  exports.command = function drupalInputParagraphForm({ form }) {
2    if (typeof form !== 'undefined') {
3      this.waitForElementVisible('select[name="field_paragraphs[0][subform][
           field_webform][0][target_id]"]', this.globals.timeoutTime)
4        .click('select[name="field_paragraphs[0][subform][field_webform][0][target_id
             ]"] option[value=' + form + ']');
5    }
6
7    return this;
8  };
```

### drupalInputParagraphGuidance

Dit commando vult de titel en teaser van Paragraph Guidance in met de meegegeven waarden.

### drupalInputParagraphGuidance.js

```
1  exports.command = function drupalInputParagraphGuidance({ title, teaser }) {
2    this.waitForElementVisible('summary[role="button"]', this.globals.timeoutTime)
3      .click('summary[role="button"]');
4
5    if (typeof title !== 'undefined') {
6      this.setValue('input[name="field_paragraphs[0][subform][field_p_012_children
           ][0][subform][field_p_title][0][value]"]', title);
7    }
8
9    if (typeof teaser !== 'undefined') {
10     this.setValue('textarea[name="field_paragraphs[0][subform][field_p_012_children
           ][0][subform][field_p_teaser][0][value]"]', teaser);
11   }
12
13   return this;
14 };
```

### drupalInputParagraphImage

Dit commando voegt een afbeelding toe aan een Paragraph door de afbeelding en alt tekst in te vullen met de meegegeven waarden. Voor Paragraph 7, 9, 10 en 12 moeten andere argumenten gebruikt worden.

### drupalInputParagraphImage.js

```
1  exports.command = function drupalInputParagraphImage({ image, alt, imagep007,
       altp007, imagep009, altp009, imagep010, altp010, imagep012, altp012 }) {
2    if (typeof image !== 'undefined') {
3      this.waitForElementVisible('input[name="files[
           field_paragraphs_0_subform_field_p_image_0]"]', this.globals.timeoutTime)
4        .setValue('input[name="files[field_paragraphs_0_subform_field_p_image_0]"]',
             require('path').resolve(__dirname + '/images/' + image));
5    }
6
7    if (typeof alt !== 'undefined') {
8      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_image][0][alt]"]', this.globals.timeoutTime)
9        .setValue('input[name="field_paragraphs[0][subform][field_p_image][0][alt]"]',
              alt);
10   }
11
12   if (typeof imagep007 !== 'undefined') {
13     this.waitForElementVisible('input[name="files[
           field_paragraphs_0_subform_field_p_007_children_0_subform_field_p_image_0]"]
           ', this.globals.timeoutTime)
14       .setValue('input[name="files[
             field_paragraphs_0_subform_field_p_007_children_0_subform_field_p_image_0
             ]"]', require('path').resolve(__dirname + '/images/' + imagep007));
15   }
16
17   if (typeof altp007 !== 'undefined') {
18     this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_007_children][0][subform][field_p_image][0][alt]"]',
           this.globals.timeoutTime)
19       .setValue('input[name="field_paragraphs[0][subform][field_p_007_children][0][
             subform][field_p_image][0][alt]"]', altp007);
20   }
21
22   if (typeof imagep009 !== 'undefined') {
23     this.waitForElementVisible('input[name="files[
           field_paragraphs_0_subform_field_p_images_unlimited_0][]"]',
           this.globals.timeoutTime)
24       .setValue('input[name="files[
             field_paragraphs_0_subform_field_p_images_unlimited_0][]"]', require('path
             ').resolve(__dirname + '/images/' + imagep009));
25   }
26
27   if (typeof altp009 !== 'undefined') {
28     this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_images_unlimited][0][alt]"]', this.globals.timeoutTime)
29       .setValue('input[name="field_paragraphs[0][subform][field_p_images_unlimited
             ][0][alt]"]', altp009);
30   }
31
32   if (typeof imagep010 !== 'undefined') {
33     this.waitForElementVisible('input[name="files[
           field_paragraphs_0_subform_field_p_010_children_0_subform_field_p_image_0]"]
           ', this.globals.timeoutTime)
34       .setValue('input[name="files[
             field_paragraphs_0_subform_field_p_010_children_0_subform_field_p_image_0
             ]"]', require('path').resolve(__dirname + '/images/' + imagep010));
35   }
36
37   if (typeof altp010 !== 'undefined') {
```

```
38      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
            field_p_010_children][0][subform][field_p_image][0][alt]"]',
            this.globals.timeoutTime)
39        .setValue('input[name="field_paragraphs[0][subform][field_p_010_children][0][
              subform][field_p_image][0][alt]"]', altp010);
40    }
41
42    if (typeof imagep012 !== 'undefined') {
43      this.waitForElementVisible('input[name="files[
            field_paragraphs_0_subform_field_p_012_children_0_subform_field_p_image_0]"]
            ', this.globals.timeoutTime)
44        .setValue('input[name="files[
              field_paragraphs_0_subform_field_p_012_children_0_subform_field_p_image_0
              ]"]', require('path').resolve(__dirname + '/images/' + imagep012));
45    }
46
47    if (typeof altp012 !== 'undefined') {
48      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
            field_p_012_children][0][subform][field_p_image][0][alt]"]',
            this.globals.timeoutTime)
49        .setValue('input[name="field_paragraphs[0][subform][field_p_012_children][0][
              subform][field_p_image][0][alt]"]', altp012);
50    }
51
52    return this;
53  };
```

### drupalInputParagraphLink

Dit commando vult de url, titel en target (open link in nieuw tabblad of niet) van een link in met de meegegeven waarden.

### drupalInputParagraphLink.js

```
1  exports.command = function drupalInputParagraphLink({ url, title, target, urlp007,
       titlep007, targetp007, urlp010, titlep010, targetp010, urlp012, titlep012,
       targetp012 }) {
2    if (typeof url !== 'undefined') {
3      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_link][0][uri]"]', this.globals.timeoutTime)
4        .setValue('input[name="field_paragraphs[0][subform][field_p_link][0][uri]"]',
             url);
5    }
6
7    if (typeof title !== 'undefined') {
8      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_link][0][title]"]', this.globals.timeoutTime)
9        .setValue('input[name="field_paragraphs[0][subform][field_p_link][0][title]"]'
             , title);
10   }
11
12   if (typeof target !== 'undefined') {
13     this.waitForElementVisible('select[name="field_paragraphs[0][subform][
           field_p_link][0][options][attributes][target]"] option[value= ' + target + '
           ]', this.globals.timeoutTime)
14       .click('select[name="field_paragraphs[0][subform][field_p_link][0][options][
             attributes][target]"] option[value= ' + target + ']');
15   }
16
17   if (typeof urlp007 !== 'undefined') {
18     this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_007_children][0][subform][field_p_link][0][uri]"]',
           this.globals.timeoutTime)
19       .setValue('input[name="field_paragraphs[0][subform][field_p_007_children][0][
             subform][field_p_link][0][uri]"]', urlp007);
20   }
21
22   if (typeof titlep007 !== 'undefined') {
23     this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_007_children][0][subform][field_p_link][0][title]"]',
           this.globals.timeoutTime)
24       .setValue('input[name="field_paragraphs[0][subform][field_p_007_children][0][
             subform][field_p_link][0][title]"]', titlep007);
25   }
26
27   if (typeof targetp007 !== 'undefined') {
28     this.waitForElementVisible('select[name="field_paragraphs[0][subform][
           field_p_007_children][0][subform][field_p_link][0][options][attributes][
           target]"] option[value= ' + targetp007 + ']', this.globals.timeoutTime)
29       .click('select[name="field_paragraphs[0][subform][field_p_007_children][0][
             subform][field_p_link][0][options][attributes][target]"] option[value= ' +
              targetp007 + ']');
30   }
31
32   if (typeof urlp010 !== 'undefined') {
33     this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_010_children][0][subform][field_p_link][0][uri]"]',
           this.globals.timeoutTime)
34       .setValue('input[name="field_paragraphs[0][subform][field_p_010_children][0][
             subform][field_p_link][0][uri]"]', urlp010);
35   }
36
37   if (typeof titlep010 !== 'undefined') {
```

```
38      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
            field_p_010_children][0][subform][field_p_link][0][title]"]',
            this.globals.timeoutTime)
39        .setValue('input[name="field_paragraphs[0][subform][field_p_010_children][0][
              subform][field_p_link][0][title]"]', titlep010);
40    }
41
42    if (typeof targetp010 !== 'undefined') {
43      this.waitForElementVisible('select[name="field_paragraphs[0][subform][
            field_p_010_children][0][subform][field_p_link][0][options][attributes][
            target]"] option[value= ' + targetp010 + ']', this.globals.timeoutTime)
44        .click('select[name="field_paragraphs[0][subform][field_p_010_children][0][
              subform][field_p_link][0][options][attributes][target]"] option[value= ' +
               targetp010 + ']');
45    }
46
47    if (typeof urlp012 !== 'undefined') {
48      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
            field_p_012_children][0][subform][field_p_link][0][uri]"]',
            this.globals.timeoutTime)
49        .setValue('input[name="field_paragraphs[0][subform][field_p_012_children][0][
              subform][field_p_link][0][uri]"]', urlp012);
50    }
51
52    if (typeof titlep012 !== 'undefined') {
53      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
            field_p_012_children][0][subform][field_p_link][0][title]"]',
            this.globals.timeoutTime)
54        .setValue('input[name="field_paragraphs[0][subform][field_p_012_children][0][
              subform][field_p_link][0][title]"]', titlep012);
55    }
56
57    if (typeof targetp012 !== 'undefined') {
58      this.waitForElementVisible('select[name="field_paragraphs[0][subform][
            field_p_012_children][0][subform][field_p_link][0][options][attributes][
            target]"] option[value= ' + targetp012 + ']', this.globals.timeoutTime)
59        .click('select[name="field_paragraphs[0][subform][field_p_012_children][0][
              subform][field_p_link][0][options][attributes][target]"] option[value= ' +
               targetp012 + ']');
60    }
61
62    return this;
63  };
```

### drupalInputParagraphTestimonial

Dit commando voert de naam en extra regel van Paragraph Testimonial in met de meegegeven waarden.

### drupalInputParagraphTestimonial.js

```
1  exports.command = function drupalInputParagraphTestimonial({ name, extrarule }) {
2    if (typeof name !== 'undefined') {
3      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_name][0][value]"]', this.globals.timeoutTime)
4        .setValue('input[name="field_paragraphs[0][subform][field_p_name][0][value]"]'
             , name);
5    }
6
7    if (typeof extrarule !== 'undefined') {
8      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_extra_rule][0][value]"]', this.globals.timeoutTime)
9        .setValue('input[name="field_paragraphs[0][subform][field_p_extra_rule][0][
             value]"]', extrarule);
10   }
11
12   return this;
13 };
```

### drupalInputParagraphTitle

Dit commando vult de titel, subtitel of teaser in met de meegegeven waarden.

### drupalInputParagraphTitle.js

```
exports.command = function drupalInputParagraphTitle({ title, subtitle, teaser }) {
  if (typeof title !== 'undefined') {
    this.waitForElementVisible('input[name="field_paragraphs[0][subform][
        field_p_title][0][value]"]', this.globals.timeoutTime)
      .setValue('input[name="field_paragraphs[0][subform][field_p_title][0][value]"]
        ', title);
  }

  if (typeof subtitle !== 'undefined') {
    this.waitForElementVisible('input[name="field_paragraphs[0][subform][
        field_p_subtitle][0][value]"]', this.globals.timeoutTime)
      .setValue('input[name="field_paragraphs[0][subform][field_p_subtitle][0][value
        ]"]', subtitle);
  }

  if (typeof teaser !== 'undefined') {
    this.waitForElementVisible('textarea[name="field_paragraphs[0][subform][
        field_p_teaser][0][value]"]', this.globals.timeoutTime)
      .setValue('textarea[name="field_paragraphs[0][subform][field_p_teaser][0][
        value]"]', teaser);
  }

  return this;
};
```

### drupalInputParagraphUsp

Dit commando vult de titel van de Paragraph USP in met de meegegeven waarde.

### drupalInputParagraphUsp.js

```
exports.command = function drupalInputParagraphUsp({ title }) {
  if (typeof title !== 'undefined') {
    this.waitForElementVisible('select[name="field_paragraphs[0][subform][
        field_p_007_view_mode]"]', this.globals.timeoutTime)
      .click('select[name="field_paragraphs[0][subform][field_p_007_view_mode]"]
          option[value="2_column"]')
      .setValue('input[name="field_paragraphs[0][subform][field_p_007_children][0][
          subform][field_p_title][0][value]"]', title);
  }

  return this;
};
```

### drupalInputParagraphVideo

Dit commando voegt een video toe aan Paragraph video door de youtube of vimeo url in te vullen met de meegegeven waarde.

### drupalInputParagraphVideo.js

```
1  exports.command = function drupalInputParagraphVideo({ videourl }) {
2    if (typeof videourl !== 'undefined') {
3      this.waitForElementVisible('input[name="field_paragraphs[0][subform][
           field_p_video][0][value]"]', this.globals.timeoutTime)
4        .setValue('input[name="field_paragraphs[0][subform][field_p_video][0][value]"]
             ', videourl);
5    }
6
7    return this;
8  };
```

### drupalInputParagraphViewmode

Dit commando selecteert de viewmode voor een Paragraph met de meegegeven waarde. Dit commando gebruikt specifieke argumenten voor Paragraph 1, 3 en 12.

### drupalInputParagraphViewmode.js

```
1  exports.command = function drupalInputParagraphViewmode({ viewmodep001, viewmodep003
       , viewmodep012 }) {
2    if (typeof viewmodep001 !== 'undefined') {
3      this.waitForElementVisible('input[value="' + viewmodep001 + '"]',
           this.globals.timeoutTime)
4        .click('input[value="' + viewmodep001 + '"]');
5    }
6
7    if (typeof viewmodep003 !== 'undefined') {
8      this.waitForElementVisible('select[name="field_paragraphs[0][subform][
           field_p_003_view_mode]"] option[value=' + viewmodep003 + ']',
           this.globals.timeoutTime)
9        .click('select[name="field_paragraphs[0][subform][field_p_003_view_mode]"]
             option[value=' + viewmodep003 + ']');
10   }
11
12   if (typeof viewmodep012 !== 'undefined') {
13     this.waitForElementVisible('select[name="field_paragraphs[0][subform][
           field_p_012_view_mode]"] option[value=' + viewmodep012 + ']',
           this.globals.timeoutTime)
14       .click('select[name="field_paragraphs[0][subform][field_p_012_view_mode]"]
             option[value=' + viewmodep012 + ']');
15   }
16
17   return this;
18 };
```

### drupalLogin

Dit commando logt in op een Drupal site met de meegegeven naam en het meegegeven wachtwoord.

### drupalLogin.js

```
exports.command = function drupalLogin({ name, password }, callback) {
  if (typeof name !== 'undefined' && typeof password !== 'undefined') {
    const self = this;

    this.drupalUserIsLoggedIn(sessionExists => {
      if (sessionExists) {
        this.drupalLogout();
      }

      this.drupalRelativeURL('/user/login')
        .setValue('input[name="name"]', name)
        .setValue('input[name="pass"]', password)
        .submitForm('#user-login-form')
        .waitForElementVisible('#toolbar-bar', this.globals.timeoutTime);
    });

    if (typeof callback === 'function') {
      callback.call(self);
    }
  }

  return this;
};
```

### drupalLoginAsAdmin

Dit commando logt in op een Drupal site als administrator door de globale variabelen te gebruiken die gedefinïeerd worden in het globals.js bestand.

### drupalLoginAsAdmin.js

```
1  exports.command = function drupalLoginAsAdmin() {
2    if (typeof this.globals.adminUsername !== 'undefined' && typeof
         this.globals.adminPassword !== 'undefined') {
3      this.drupalLogin({ name: this.globals.adminUsername, password:
           this.globals.adminPassword });
4    }
5
6    return this;
7  };
```

### drupalLogout

Dit commando logt uit op een Drupal site.

### drupalLogout.js

```
1  exports.command = function drupalLogout(callback) {
2    const self = this;
3
4    this.drupalRelativeURL('/user/logout');
5
6    if (typeof callback === 'function') {
7      callback.call(self);
8    }
9
10   return this;
11 };
```

### drupalRelativeURL

Dit commando surft naar een relatieve url op een Drupal website door de meegegeven waarde te plaatsen na de launch_url variabele die in het globals.js bestand gedefinïeerd staat.

### drupalRelativeURL.js

```
1  exports.command = function drupalRelativeURL(pathname, callback) {
2    const self = this;
3
4    this.url(this.globals.launch_url + pathname);
5
6    if (typeof callback === 'function') {
7      callback.call(self);
8    }
9
10   return this;
11 };
```

### drupalScreenshot

Dit commando neemt een screenshot en plaatst deze in de map die als waarde moet meegegeven worden. Dit commando heeft een optioneel argument dat moet meegegeven worden wanneer de browser headless wordt uitgevoerd.

### drupalScreenshot.js

```
1  exports.command = function drupalScreenshot({ path, headless }) {
2    if (typeof path !== 'undefined') {
3      if (typeof headless !== 'undefined') {
4        if (headless) {
5          this.resizeWindow(1400, 1400);
6        }
7      }
8
9      this.saveScreenshot(path);
10   }
11
12   return this;
13 };
```

### drupalSubmit

Dit commando verzendt een formulier op een Drupal website.

### drupalSubmit.js

```
1  exports.command = function drupalSubmit(callback) {
2    const self = this;
3
4    this.waitForElementVisible('#edit-submit', this.globals.timeoutTime)
5      .click('#edit-submit');
6
7    if (typeof callback === 'function') {
8      callback.call(self);
9    }
10
11   return this;
12 };
```

### drupalUserIsLoggedIn

Dit commando controleert of de gebruiker ingelogd is.

### drupalUserIsLoggedIn.js

```
exports.command = function drupalUserIsLoggedIn(callback) {
  if (typeof callback === 'function') {
    this.getCookies(cookies => {
      const sessionExists = cookies.value.some(cookie =>
        cookie.name.match(/^SESS/),
      );

      callback.call(this, sessionExists);
    });
  }

  return this;
};
```

# A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

## A.1 Introductie

Heden ten dage bestaan er veel verschillende tools, frameworks en libraries voor UI testen die elk hun eigen implementatie en vaak ook hun eigen filosofie in verband met testen hebben. Bijkomend kunnen sommige van deze tools gecombineerd worden voor verschillende use cases.

Dropsolid, een KMO gespecialiseerd in Drupal, gebruikt momenteel nog geen front-end testtools en zou dit in de toekomst graag doen maar weet niet welke tools het beste bij hen zouden passen. De ontwikkelaars binnen Dropsolid die later deze testen zouden schrijven zijn het meest vertrouwd met Javascript en dus verkiest Dropsolid tools die op Javascript gebaseerd zijn. Ook zou dit het beste passen bij de systemen die ze momenteel al gebruiken en de mogelijke integratie versoepelen.

De onderzoeksvraag luidt als volgt: Welke op Javascript gebaseerde front-end testtools kan een KMO gespecialiseerd in Drupal het best gebruiken? Een bijkomende doelstelling is om de verkozen testtools te gebruiken om een test te schrijven die op een nieuwe installatie van Dropsolids Drupal 8 skelet, genaamd Rocketship, de voorgeprogrammeerde Paragraph Types aanmaakt en test.

## A.2 State-of-the-art

De drie belangrijkste testtypes voor websites zijn: unit testen, integratie testen en UI testen. Unit testen zorgen dat de individuele componenten (functies, klassen, ...) van een applicatie correct werken, integratie testen zorgen dat de verschillende componenten met elkaar werken zoals verwacht en UI testen, ook wel functionele of front-end testen genoemd, zorgen dat de applicatie vanuit het perspectief van de gebruiker correct werkt. (Elliott, 2016)

Dit onderzoek zal gaan over het laatste type, de UI testen. Voor UI testen heb je een combinatie nodig van verschillende tools, frameworks en libraries, de voornaamste bespreek ik hieronder.

De eerste soort tools zijn diegene die een dienst verlenen om je te helpen met het uitvoeren van testen op verschillende browsers en apparaten. Enkele voorbeelden zijn BrowserStack, Sauce Labs, BrowseEmAl en Browsershots. (Jackson, 2017)

De tweede soort tools zijn diegene waarmee je je testen schrijft. Selenium, Testcafe, Cypress, Puppeteer, PhantomJS, Nightmare en Casper zijn hier voorbeelden van. Selenium is een tool die de browser automatiseert en heeft een eigen WebDriver maar die kan uitgebreid worden met libraries zoals Appium, Protractor, WebdriverIO of Nightwatch. Testcafe en Cypress werken niet zoals Selenium, deze automatiseren de browser niet, maar injecteren Javascript scripts in de browser zelf. Puppeteer en PhantomJS zijn tools die gebruikt worden om headless Chrome of Chromium te besturen. Nightmare is een library die bovenop Electron werkt, Electron is gelijkaardig aan PhantomJS. Casper is een hulpprogramma voor PhantomJS voor navigatie scripting en testen.

De derde soort tools zijn diegene die een abstractie laag bovenop een andere testtool toevoegen met Behavior Driven Development als doel. Voorbeelden hiervan zijn Cucumber en CodeceptJS. (Zaidman, 2018)

## A.3 Methodologie

Dit onderzoek zal uit twee delen bestaan. In het eerste deel ga ik de verschillende tools en verschillende combinaties van deze tools vergelijken op verschillende vlakken zoals snelheid waarop de testen uitgevoerd worden, aantal extensies of plug-ins, support van de community, gebruiksvriendelijkheid, hoe goed deze tool past bij Dropsolid, ... In het tweede deel zal ik de verkozen tools gebruiken om een test te schrijven die op Rocketship voorgeprogrammeerde Paragraph Types aanmaakt en deze test. Hiervoor zou ik gebruik maken van een virtuele machine waarin ik de site lokaal host. Telkens na het uitvoeren of proberen uitvoeren van deze test zou ik de databank terugstellen naar zijn oorspronkelijke staat zodat ik steeds start van volledig dezelfde website.

## A.4 Verwachte resultaten

Voor het eerste deel van het onderzoek verwacht ik dat ik resultaten ga hebben over de snelheid waarop de testen uitgevoerd worden. Voor de gebruiksvriendelijkheid ga ik stukken code hebben om de syntax, leesbaarheid en complexiteit aan te tonen. Ook voor het aantal extensies en plug-ins en voor de support van een bepaalde tool verwacht ik dat ik resultaten ga hebben. Voor het tweede deel verwacht ik dat het resultaat in staat is om op een nieuwe installatie van Rocketship voorgeprogrammeerde Paragraph Types aan te maken en te testen.

## A.5 Verwachte conclusies

Ik verwacht dat ik aan de hand van het eerste deel van het onderzoek een gegronde keuze ga kunnen maken over welke combinatie van op Javascript gebaseerde front-end testtools het beste passen bij Dropsolid. Dit zou samen met de test die ik in het tweede deel schrijf de basis kunnen worden van front-end testen binnen Dropsolid.

# Bibliografie


Apache. (2018, juli). Apache HTTP Server (2.4.34). Verkregen van https://httpd.apache.org/

Bobbeldijk, J. (2013, november). Paragraphs. Verkregen van https://www.drupal.org/project/paragraphs

Chai. (2017, augustus). Chai Assertion Library (4.1.2). Verkregen van http://www.chaijs.com/

ChromeDriver. (2018, juli). ChromeDriver (2.41). Verkregen van http://chromedriver.chromium.org/

Cypress. (2018, juli). Cypress (3.0.3). Verkregen van https://www.cypress.io/

Drupal. (2018, augustus). Drupal (8.5.6). Verkregen van https://www.drupal.org/

Drush. (2018, mei). Drush (8.1.17). Verkregen van https://www.drush.org/

Electron. (2018, maart). Electron (1.8.4). Verkregen van https://electronjs.org/

Elliott, E. (2016, april). JavaScript Testing: Unit Vs Functional Vs Integration Tests. Verkregen van https://www.sitepoint.com/javascript-testing-unit-functional-integration/

Fowler, M. (2007, januari). Mocks Aren't Stubs. Verkregen van https://martinfowler.com/articles/mocksArentStubs.html

GeckoDriver. (2018, augustus). GeckoDriver (1.12.2). Verkregen van https://github.com/mozilla/geckodriver

Git. (2018, juni). Git (2.18.0). Verkregen van https://git-scm.com/

Guckenheimer, S. (2017, april). What is Continuous Integration? Verkregen van https://docs.microsoft.com/en-us/azure/devops/what-is-continuous-integration

Jackson, B. (2017, juni). Top 12 Browser Compatibility Testing Tools. Verkregen van https://www.keycdn.com/blog/browser-compatibility-testing-tools/

Leslie, A. (2018, mei). NGINX vs. Apache (Pro Con Review, Uses, Hosting for Each. Verkregen van https://www.hostingadvice.com/how-to/nginx-vs-apache/

Lindstrom, J. (2014, augustus). Performance evaluation of MariaDB 10.1 and MySQL 5.7.4. Verkregen van https://mariadb.org/performance-evaluation-of-mariadb-10-1-and-mysql-5-7-4-labs-tplc

Mann, B. (2017, februari). Proposal: Support for Cross Browser Testing. Verkregen van https://github.com/cypress-io/cypress/issues/310

Mann, B. (2018, januari). Proposal: Support for Cross Browser Testing. Verkregen van https://github.com/cypress-io/cypress/issues/310#issuecomment-354925454

MariaDB. (2018, juli). MariaDB (10.3.8). Verkregen van https://mariadb.org/

Mint, L. (2018). Linux Mint 19 (64-bit Cinnamon editite). Verkregen van https://linuxmint.com/

Mocha. (2018, mei). Mocha (5.2.0). Verkregen van https://mochajs.org/

Nightmare. (2018, maart). Nightmare (3.0.1). Verkregen van http://www.nightmarejs.org/

Nightwatch. (2018, juli). Nightwatch (1.0.8). Verkregen van http://nightwatchjs.org/

Node.js. (2018, augustus). Node.js (10.8.0). Verkregen van https://nodejs.org/

Npm. (2018, juli). Npm (6.2.0). Verkregen van https://www.npmjs.com/

npm trends. (2018, augustus). testcafe vs cypress vs nightwatch vs webdriverio vs nightmare vs puppeteer. Verkregen van http://www.npmtrends.com/testcafe-vs-cypress-vs-nightwatch-vs-webdriverio-vs-nightmare-vs-puppeteer

Oracle. (2018). Oracle VM VirtualBox (5.2.16). Verkregen van https://www.virtualbox.org/

Software Framework. (g.d.). Verkregen van https://www.techopedia.com/definition/14384/software-framework

Software Library. (g.d.). Verkregen van https://www.techopedia.com/definition/3828/software-library

What is a Relational Database Management System? (g.d.). Verkregen van https://www.codecademy.com/articles/what-is-rdbms-sql

A Dictionary of Computing, Software Tool. (2004). Verkregen van https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/software-tool

NightCloud.io. (2017). Verkregen van https://nightcloud.io/

PHP. (2018, juli). PHP (7.2.8). Verkregen van http://php.net/

PhpMyAdmin. (2018, juni). phpMyAdmin (4.8.2). Verkregen van https://www.phpmyadmin.net/

Puppeteer. (2018, augustus). Puppeteer (1.7.0). Verkregen van https://github.com/GoogleChrome/puppeteer

Ramael, J. (2015, november). HET VERSCHIL: DRUPAL 7 VS DRUPAL 8. Verkregen van https://dropsolid.com/nl/blog/het-verschil-drupal-7-vs-drupal-8

Reselman, B. (2017, december). The Fundamentals of Integration Testing. Verkregen van https://www.mabl.com/blog/the-fundamentals-of-integration-testing

Sanders, C. (2006, augustus). Fixed Size vs. Dynamically Expanding Virtual Hard Disks. Verkregen van http://techgenix.com/fixedsizevs-dynamicallyexpandingvirtualharddisks/

Server, S. S. (2018, augustus). Selenium Standalone Server (3.14.0). Verkregen van https://www.seleniumhq.org/

Slobodin, V. (2017, maart). [Announcement] Stepping down as maintainer. Verkregen van https://groups.google.com/forum/#!topic/phantomjs/9aI5d-LDuNE

Taranto, T. (2017, februari). AWS RDS benchmark—MariaDB vs MySQL. Verkregen van https://medium.com/@ttaranto/aws-rds-benchmark-mariadb-vs-mysql-47af70602bb8

TestCafé. (2018, augustus). TestCafé (v0.21.0). Verkregen van http://devexpress.github.io/testcafe/

van Tuil, K. (2011, juni). Wat is een API? Verkregen van https://computerworld.nl/development/74796-wat-is-een-api

WebdriverIO. (2018, juni). WebdriverIO (4.13.1). Verkregen van http://webdriver.io/

Zaidman, V. (2018, februari). An Overview of JavaScript Testing in 2018. Verkregen van https://medium.com/welldone-software/an-overview-of-javascript-testing-in-2018-f68950900bc3

Zilberfeld, G. (2013, oktober). Why you need unit testing in web development. Verkregen van https://www.creativebloq.com/web-design/testing-times-10134991