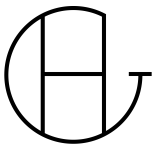

HoGent

Faculteit Bedrijf en Organisatie

JavaScript frameworks voor webapps: een vergelijkende studie

Jef Braem

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

Promotor:
Tom Antjon
Co-promotor:
Kristof Van Miegem

Instelling: —

Academiejaar: 2017-2018

Tweede examenperiode

Faculteit Bedrijf en Organisatie

JavaScript frameworks voor webapps: een vergelijkende studie

Jef Braem

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

Promotor:
Tom Antjon
Co-promotor:
Kristof Van Miegem

Instelling: —

Academiejaar: 2017-2018

Tweede examenperiode

Woord vooraf

Voor u ligt de bachelorproef 'JavaScript frameworks voor webapps: een vergelijkende studie'. In deze studie vergelijk ik drie verschillende JavaScript frameworks. Ik heb dit onderwerp gekozen in het kader van mijn afstudeerrichting toegepaste informatica aan de HoGent campus Aalst. Tot en met mei 2018 heb ik gewerkt en geschreven aan deze proef.

Het onderwerp voor deze bachelorproef heb ik zelf gekozen. Sinds dit jaar is mijn interesse in JavaScript frameworks zeer versterkt. We hebben in het vak webapps uitgebreid het framework Angular besproken. Na enkele weken bleek dit vak mij sterk te interesseren. Dit was ook de aanzet om dit onderwerp te kiezen.

Deze interesse was ook de reden waarom ik een stageplaats in verband met het web gekozen heb. Ik heb tijdens deze periode stage gelopen bij Codifly en heb daar gewerkt in React. Door met deze technologie te werken in de werkomgeving is mijn interesse enkel gestegen.

In de eerste plaats wil ik mijn promotor Tom Antjon bedanken voor al het geduld in deze periode. Daarnaast wil ik ook mijn co-promotor Kristof Van Miegem bedanken voor al zijn enthousiasme en inzet. Ten slotte wil ik al mijn klasgenoten bedanken die mij geholpen hebben tijdens deze proef. Het zei om vragen te beantwoorden of mijn teksten na te lezen.

Samenvatting

JavaScript is al vele jaren een belangrijk onderdeel van web development en is één van de meest gebruikte front-end programmeer talen voor het web. De laatste jaren zijn er veel JavaScript frameworks in opkomst gekomen. Hierdoor is er een grote verandering in hoe developers JavaScript gebruiken om web applicaties te bouwen. Deze proef heeft dus belang voor veel mensen.

Met de opkomst van al deze JavaScript frameworks wordt je als developer overspoeld met opties om je applicatie in te maken. Dit is een van de belangrijkste redenen waarom deze proef uitgevoerd werd.

Deze proef probeert de belangrijkste pijlers van een framework te vergelijken met elkaar zoals performance, compatibiliteit, stabiliteit, security en nog anderen. Deze worden onderverdeelt in een theoretisch deel en een praktisch deel.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

Zoals eerder in de samenvatting vermeld zijn JavaScript frameworks sterk gegroeid de voorbije jaren. Op dit moment zijn er tientallen frameworks in development. Niet elk framework is even efficiënt in taken uitvoeren die hiervan verwacht worden. Sommige frameworks hebben meer functionaliteit dan anderen, terwijl anderen gebruiksvriendelijker zijn voor de developer. In dit onderzoek zullen we deze verschillen proberen schetsen voor de drie populairste frameworks.

1.1 Probleemstelling

Met deze sterke opkomst van JavaScript frameworks is er een groot aanbod gekomen. Het voordeel van dit grote aanbod is dat er een geschikter framework gekozen kan worden voor een specifieke web applicatie. Het grote nadeel hiervan is dat de developer (bijna) nooit de moeite doet om het meest geschikte framework te kiezen. Meestal zal deze keuze gebaseerd worden op persoonlijke voorkeur of ervaring met een bepaald framework. In deze proef proberen we te achterhalen welke voordelen en nadelen bepaalde frameworks hebben. De personen die een meerwaarde aan deze proef zullen hebben zijn:

- web developers
- de JavaScript framework developers zelf.

1.2 Onderzoeksvraag

In dit onderzoek zullen we niet enkel performantie vergelijken maar ook pijlers zoals modulariteit, stabiliteit en nog anderen. Aan de hand van deze vergelijkingen proberen we een beter beeld te scheppen over de verschillen en gelijkenissen in deze frameworks. Welke JavaScript frameworks zijn het best resultaat gericht en development gericht?

1.3 Onderzoeksdoelstelling

Naar het einde toe van dit onderzoek proberen we een duidelijk beeld te kunnen scheppen van de verschillen en/of gelijkenissen tussen de verschillende frameworks die besproken worden. Het doel is niet om één framework als beste te beschouwen maar om de voor en nadelen van elk framework op te lijsten en te vergelijken.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

in Hoofdstuk **??** wordt het theoretisch en praktisch onderzoek uitgevoerd en worden de resultaten weergegeven om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

2.1 JavaScript

In dit onderdeel zal ten eerste de historiek van JavaScript beschreven worden. Hierna zullen de eigenschappen van JavaScript ten opzichte van andere programmeer talen besproken worden.

2.1.1 Historiek

JavaScript is uitgevonden in 1995 bij Netscape Communications. Zij hebben ook de Netscape browser gemaakt. In die tijd was Java de populaire taal voor het web. Hierdoor hebben ze besloten om de syntax van JavaScript op die van Java te baseren. De eerste versie van JavaScript was onder de naam Mocha in 1995. Hierna werd deze hernoemd naar LiveScript. Ten slotte werd de taal verandert naar JavaScript eind 1995. JavaScript is gestandaardiseerd bij ECMA International. Deze gestandaardiseerde versie van JavaScript noemen we ECMAScript. Deze gedraagt zich hetzelfde in alle applicaties die deze standaard accepteren.

2.1.2 Eigenschappen

Het is een object georiënteerde en dynamisch getypeerde scripttaal die gebruikt word om webpagina's interactief te maken. Deze bevat een standaard bibliotheek van objecten zoals Array, Date en Math. Deze kunnen dan gebruikt worden om de browser en zijn Document Object Model te besturen. Javascript kan HTML elementen plaatsen en/of verwijderen, reageren op events zoals muisclicks, een formulier indienen en navigeren naar een andere

pagina.

JavaScript en Java zijn te vergelijken op sommige vlakken maar ook heel verschillend. JavaScript lijkt op Java omdat het dezelfde uitdrukking syntaxis gebruikt. Dit is de wijze waarop een combinatie van waarden, variabelen, operatoren en functies kan worden uitgedrukt. De verschillen worden vervolgens besproken.

Javascript is een dynamisch getypeerde taal. Dit betekend dat het type van een variabele nog niet gekend is bij de compileer tijd. JavaScript ondersteunt een runtime system dat een klein aantal data types ondersteunt. Deze zijn number, boolean en string. Hierdoor kan tijd bespaard worden bij het maken van een project. Elke variabele word gedeclareerd door var, let of const. Een groot nadeel van zo'n dynamisch getypeerde taal is dat je door typfouten bugs kan maken die zeer moeilijk op te sporen zijn. Daartegenover is Java een statisch getypeerde taal. Dit betekend dat bij Java alle types gekend zijn bij compileer tijd. Het voordeel hiervan is dat er een heleboel checks kunnen gedaan worden door de compiler, hierdoor komen veel triviale bugs niet voor.

JavaScript bied meer vrijheid ten opzichte van Java. Je moet niet elke variabele, klasse of methode declareren. Methodes kunnen niet publiek, privaat of beschermd zijn. Er is ook geen nood om interfaces te implementeren. Parameters en functie retourneerwaarden zijn ook niet expliciet getypeerd. Dit kan veel tijd besparen maar ook voor moeilijk te vinden bugs zorgen.

Java is klasse gebaseerd, objecten zijn onderverdeeld in klassen en instanties. Er bestaat en vaste hiërarchie door de klassen heen. Klassen en instanties kunnen niet dynamisch attributen of methoden toegevoegd krijgen. Anderzijds is JavaScript object georiënteerd. Dit betekent dat er geen verschil gemaakt word tussen types van objecten. Overerving is door middel van het prototype systeem. Attributen en methoden kunnen dynamisch aan objecten worden toegevoegd (**Introduction_2018**).

2.2 Frameworks

In dit onderdeel zal ik het uitgebreid hebben over wat een framework is en de componenten waaruit een framework bestaat.

Een software framework is een set methodes en klassen die ontworpen zijn om het werk van een developer te vereenvoudigen. Het is een abstractie van veel kleine componenten die herbruikbaar zijn waardoor veel tijd gespaard kan worden. Een framework legt ook meestal een bepaalde structuur op bij de developer om de code te implementeren. Dit is goed voor consistente code en zorgt voor minder bugs. Er zijn meerdere onderdelen waaruit een framework kan bestaan en deze worden nader besproken (Clifton, 2003) (Eskelin, 2001).

Wrapper functie Een wrapper is een methode om één of meerdere functies te versimpelen, consistentie te geven en/of functionaliteit toevoegen. Een wrapper past het bestaande

gedrag aan en zal de functionaliteit niet compleet veranderen.

Architecturen Een architectuur is een stijl dat specifieke ontwerp patronen gebruikt. Een framework heeft een patroon nodig. Meestal ondersteund een framework het gebruik van meerdere ontwerp patronen. Dit patroon zorgt ervoor dat je een herbruikbare structuur maakt in je project. Eenmaal je een patroon gebruikt is het (bijna) onmogelijk om hier van af te stappen of je moet een grote refactor doen van je hele project.

Methodologie Een methodologie is de manier waarop iets gedaan kan worden. De methodologie is hoe de interactie tussen dingen gebeurt. Hoe objecten met elkaar kunnen communiceren, hoe met persistentie aanneemt of hoe er gereageerd kan worden op user events.

2.3 Benchmarking

In dit deel zal de term benchmarking verder uitgelegd worden, wat er onder deze term verstaan word en wat deze proef ermee wil bereiken. Benchmarking heeft verschillende betekenissen volgens het Engelse woordenboek „Benchmark" (g.d.), enkele zijn hier opgesomd.

"De kwaliteit van iets meten door het te vergelijken met de geaccepteerde standaard."

"Een standaard om iets te meten of over iets te oordelen van hetzelfde type."

"Een bepaalde grens van kwaliteit dat kan gebruikt worden als standaard om andere dingen mee te vergelijken."

Benchmarking is een belangrijk onderdeel van de informatica wereld. Het word overal gebruikt van hardware benchmarks tot database performance benchmarks. Benchmarking tools zijn meestal één of meerdere programma's die de performance van een applicatie meten onder bepaalde condities. Het doel van zo'n benchmark is om een eerlijke vergelijking te maken tussen verschillende dingen. In deze proef zullen er benchmarks gebruikt worden om de performance van JavaScript frameworks te vergelijken.

2.4 MVC

Om de tests in deze proef zo gelijk mogelijk te laten verlopen zal elke applicatie het MVC (Model View Controller) ontwerp patroon zo goed mogelijk proberen hanteren. In dit onderdeel zal ik de basis van het Model-View-Controller patroon beschrijven.

Het Model View Controller patroon is een software architectuur stijl of ontwerp patroon gebruikt voor seperation of concerns. Alle business logica zal zich bevinden in de Con-

troller. Dit is gescheiden van de View. De data die weergegeven wordt bevind zich in een Model. Het patroon beheert de fundamentele werking en data van de applicatie. Het kan reageren voor requests voor informatie, antwoorden met instructies om de state aan te passen en zelfs observers waarschuwen in event-driven systemen. Naast het MVC patroon zijn er meerdere ontwerp patronen ontwikkeld zoals MVVM (model-view-view-model), MVP (model-view-presenter), MVA (model-view-adapter) en nog veel meer. Deze zullen we niet bespreken in deze proef. Het MVC patroon was veel populairder ten opzichte van deze patronen. Verder zullen we de onderdelen van MVC nog kort bespreken (Atwood, 2008) („Model-View-Controller”, 2014).

Model Ten eerste zal het model besproken worden. Het model definieert de vorm van de data die de applicatie gebruikt. Een model kan een object zijn maar kan ook uit meerdere objecten bestaan. Het model en wat de gebruiker waarneemt hebben meestal een één-op-één relatie. Een model is dus blind, dit betekend dat de model enkel instaat voor de data bij te houden wat er verder met de data gebeurd weet de model niets van. De daadwerkelijke opslag van data wordt door een database gedaan.

View Informatie wordt weergegeven aan de gebruiker via de view. De view doet geen bewerkingen of berekeningen en dient enkel en alleen om data weer te geven. De user kan op de view bepaalde componenten aanklikken dat events kan triggeren. Deze kunnen doorgezonden worden naar de controller.

Controller De controller kan events opvangen en hierop reageren. Meestal worden er dan bewerkingen uitgevoerd op waarden uit de model. De model word dan aangepast en hierdoor zal de view dan weer geupdate worden.

3. Methodologie

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas

tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

4. Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit

lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

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

Er zijn reeds vele JavaScript frameworks in omloop. Als je aan een project begint weet je soms niet goed welke de beste keuze is en neem je waarschijnlijk het framework waar je het meest mee vertrouwt bent. Ik vroeg mij af, zou het niet efficiënter zijn om het meteen het meest geschikte framework te kiezen? Natuurlijk is het efficiënter om met iets te werken dat je kent. Maar als het een groot project is misschien niet.

Ik doe dit onderzoek omdat het vak webapps mij interesseert. Hierbij gebruiken we Angular 4. Ook heb ik al een kleine chat applicatie gemaakt als test voor mijn stageplaats gemaakt in react. Door deze twee gebeurtenissen is mijn interesse in web development gepiekt. Toen ik meer zoekwerk deed vond ik veel meer frameworks. Ik vroeg mij dus af welke beter is.

Met dit onderzoek wil ik een beter beeld vormen van 3 frameworks. Hiervoor ga ik mij baseren op de populariteit van JavaScript frameworks en heb besloten de 3 meest gebruikte te bestuderen. Het beste voor dit soort onderzoek is zo veel mogelijk verschillende frameworks, maar ik denk dat de tijd mij niet zal toestaan om meer dan 3 te bestuderen.

Bij dit onderzoek stel ik mij de volgende vragen:

- welk framework reageert het snelst?
- welk framework gebruikt het minste geheugen?
- waar implementeer je MVC het makkelijkst naar mijn gevoel?

A.2 State-of-the-art

A.2.1 Wat zijn JavaScript frameworks?

Een framework is een soort van skelet voor je project. Het heeft al een bepaalde lijst van functionaliteiten ter beschikking. Dankzij een framework zal een project opbouwen veel sneller verlopen.

Bij web development is er ook sprake van het multiplatformprobleem. JavaScript is zeer browseronafhankelijk maar het wordt veel gebruik tom de DOM te manipuleren. JavaScript frameworks lossen dit op door extra code voor verschillende browsers te genereren. Zo hoeft de ontwikkelaar zich hier geen zorgen meer over te maken. (Buckler, 2017)

A.2.2 Gelijkaardige onderzoeken

Er zijn zeker al gelijkaardige onderzoeken gedaan over dit onderwerp. Hetgeen me opviel is dat de meeste onderzoeen die al uitgevoerd zijn vooral over performantie van JavaScript frameworks gaan. Natuurlijk zal ik ook een deel van mijn tijd besteden aan performantie maar ik zou graag niet compleet in die richting gaan en ook een andere kijk op JavaScript frameworks hebben. Wat de moeilijkheidsgraad was om MVC te implementeren, hoe de leercurve was persoonlijk voor mij. Dit kan dan een beter beeld scheppen voor mensen die deze scriptie lezen.

Je mag gerust gebruik maken van subsecties in dit onderdeel.

A.3 Methodologie

In dit onderzoek zal ik starten met een uitgebreide literatuurstudie over JavaScript frameworks en over de gekozen frameworks. Daarna zal ik voor alle technologiën de performantie onderzoeken. Ten slotte zal ik voor elke framework één functioneel identieke MVC applicatie schrijven.

A.4 Verwachte resultaten

Ik verwacht dat de performantie van de frameworks niet ver van elkaar zal liggen. Computers zijn tegenwoordig zeer snel en er zullen dus maar kleine verschillen zijn.

A.5 Verwachte conclusies

Ik verwacht dat er wel een duidelijke conclusie zal zijn. Hiermee bedoel ik dat ik hoop op één framework waar de implementatie en/of leercurve veel makkeljiker of sneller zullen gaan dan bij de anderen. Aangezien ik al gewerkt heb met angular en met react zal ik hier mee rekening houden.

Bibliografie

Atwood, J. (2008, mei 5). Understanding the Model-View-Controller. Verkregen van https://blog.codinghorror.com/understanding-model-view-controller/

Benchmark. (g.d.), In *Cambridge Advanced Learner's Dictionary & Thesaurus*. Verkregen van https://dictionary.cambridge.org/dictionary/english/benchmark

Buckler, C. (2017). Top JavaScript frameworks, libraries and tools to use in 2017. *https://www.sitepoint.com/to javascript-frameworks-libraries-tools-use/*.

Clifton, M. (2003, november 3). What is a Framework [What is a Framework]. Verkregen van https://www.codeproject.com/Articles/5381/What-Is-A-Framework

Eskelin, P. (2001, november 3). Software Framework [Software Framework]. Verkregen van http://wiki.c2.com/?SoftwareFramework

Model-View-Controller. (2014, maart 17). Verkregen van https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ff649643(v=pandp.10)