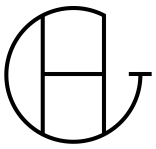

HoGent

Faculteit Bedrijf en Organisatie

Monitoring mogelijkheden in NodeJS toepassingen voor feilloos debuggen in 2019

Michiel Leunens

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

Promotor:
Tom Antjon
Co-promotor:
Michiel Cuvelier

Instelling: beSports bvba - Kayzr

Academiejaar: 2018-2019

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Monitoring mogelijkheden in NodeJS toepassingen voor feilloos debuggen in 2019

Michiel Leunens

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

Promotor:
Tom Antjon
Co-promotor:
Michiel Cuvelier

Instelling: beSports bvba - Kayzr

Academiejaar: 2018-2019

Tweede examenperiode

Woord vooraf

Backend debug toepassingen zijn vaak slecht onderhouden, incompleet of gewoonweg verschrikkelijk duur. Daarom is deze bachelorproef gemaakt met als doel een open-source debug tool te schenken aan iedereen die hun project in NodeJS en Express ontwikkelt. Mede mogelijk gemaakt dankzij Kayzr, is het de bedoeling om deze tool online te zetten zodat deze effectief in de praktijk gebruikt kan worden. Graag bedank ik ook Dhr. Tom Antjon, voor altijd paraat te staan, snel mijn vragen te kunnen beantwoorden en mij een goed inzicht te geven in het werk dat nog gedaan moest worden.

Samenvatting

[1-4]

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

1.1 Probleemstelling

Het zoeken naar fouten in je software kan een frustrerend, hectisch en vermoeiend proces zijn. Daarom zijn, als ontwikkelaar zijnde, toepassingen die dat proces voor jou kunnen automatiseren of beter analyseren altijd welkom. Kayzr, een Belgische start-up die zich bezighoudt met het mainstream maken van E-Sports in de Benelux en het organiseren van toernooien ervan, wenst zulke software zodat hun productieproces versneld kan worden. Het concreet probleem gaat als volgt: Kayzr's processen werden op verschillende plekken gemonitord. De tools zijn incompleet, en daardoor heeft men een gefragmenteerde collectie aan logs, foutmeldingen, informatie, enz... Per API call moet men foutmeldingen controleren in de terminal, IP-adressen in het Google Cloud platform, andere nuttige informatie staat dan weer eens ergens anders opgeslagen. Hierdoor duurt het lang om fouten te analyseren en te komen tot een mogelijke oplossing. Een dergelijke tool zoals PM2, een monitoringstoepassing met veel meer functionaliteiten, meer dan dat ze nodig hadden, is meteen een kost dat ze zich niet kunnen permitteren. Aan €50 per server per maand, kost hen dit met een vijftigtal servers al te veel. Deze kost zou schalen zeer moeizaam of zelfs onmogelijk maken.

1.2 Onderzoeksvraag

De onderzoeksvraag gaat dus als volgt: bestaat er een mogelijkheid om de monitoringstechnieken van Kayzr, en mogelijks andere bedrijven, op een goedkope manier te stroomlijnen en verbeteren binnen een NodeJS omgeving, zodat hun debugproces geoptimaliseerd kan worden?

1.3 Onderzoeksdoelstelling

Kayzr is tevreden met de toepassing en gaat in hun project hiervan gebruik maken. De gemiddelde tijd om een probleem op te lossen is gedaald en dit proces verloopt efficiënter. De data die per proces is verzameld kan op een goede manier gevisualiseerd worden zodat ze die ook voor andere doeleinden kunnen hanteren.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 volgt een literatuurstudie dat een groot deel van Javascript en NodeJS, Express en middelware gaat omvatten

In Hoofdstuk 3 bestuderen we het huidige proces. Hoe verloopt dit? Wat kan er beter? Wat kan er sneller? Welke functies moet de nieuwe toepassing bevatten? Na deze studie schrijven we software naargelang de requirements, en toetsen we die op verschillende factoren aan de oude processen.

In Hoofdstuk 4, wordt er gekeken of er een oplossing bestaat op deze onderzoeksvragen. We kijken of Kayzr deze gaat gebruiken naar de toekomst toe en wat er gaat gebeuren met de eigendom van dit softwareproject.

2. Stand van zaken

Dit hoofdstuk bestaat uit een zeer uitgebreide literatuurstudie, waarin de kern van het probleem wordt opgesplitst in verschillende delen. In het eerste deel wordt er uitleg gegeven over Javascript en diens evolutie. Vervolgens volgt er een groot deel over de frameworks NodeJS en Express, erna bekijken we het belang van testen en monitoren van software en breiden we uit naar een analyse van zulke monitoringssoftware. In het tweede deel wordt er onderzoek gedaan naar de vraag van Kayzr. Wat wordt er verwacht, welk inzicht moet er gegeven worden, welke features moeten worden uitgewerkt.

2.1 Javascript

Javascript is een programmeertaal gemaakt voor het web, waarmee u statische webpagina's kan omzetten naar dynamische en interactieve websites. Doordat het een enorm krachtige scripttaal is dat speciaal werd ontwikkeld om de functionaliteiten van een doorsnee HTML/CSS-pagina uit te breiden, wordt het bijna onmogelijk om nog iets in te beelden dat niet geïmplementeerd kan worden m.b.v. Javascript (**Javascript2019**). De taal is weakly-typed, functioneel, event-driven en dynamisch.

2.1.1 Tijdlijn

Javascript heeft echter een hele evolutie achter de rug. Ontstaan in mei 1995, wordt de taal na vele updates nog steeds dagdagelijks gebruikt en kan het wereldwijde web niet meer ingebeeld worden zonder. Brendan Eich, de auteur van de taal, werkte in 1995 samen met Netscape Communications, de makers van de eerste grote webbrowser genaamd

Netscape Navigator, om een taal te implementeren in hun browser waar webontwikkelaars gebruik van zouden kunnen maken. Java, een zeer zware programmeertaal met tal van functionaliteiten was de eerste keuze van Netscape, maar Eich schreef uiteindelijk zijn eigen idee uit van een scripttaal in nog geen 10 dagen en overtuigde Netscape om de lichte, schaalbare en Java-complementerende-taal te adopteren (**Rangpariya2019**). Javascript, toen onder de naam Mocha en vervolgens LiveScript, was geboren.

3. Methodologie

[21-25]

4. Conclusie

[76-80]

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

Node.js is een backend Javascript framework wiens populariteit in de afgelopen jaren hard is toegenomen. Ontwikkelaars genieten van verschillende voordelen. Het werkt asynchroon, het is makkelijk schaalbaar en het is zeer functioneel. Doordat het Javascript is, kan elk besturingssysteem gebruik maken van de krachtige functies die Node.js te bieden heeft. Dit maakt het een uitstekend framework om webapplicaties te ontwikkelen. Node.js is echter niet makkelijk om te debuggen doordat het asynchroon is opgebouwd. Het toepassen van de juiste monitortechnieken kan de slaagkansen van een project echter goed verhogen, net als de levensduur van de applicatie. Op welke manieren kunnen we het monitoren van zulke applicaties aanpakken? Welke software en tools worden hiervoor gebruikt? Welke technieken worden het best toegepast? En kan er ook intern in het proces van Node.js gekeken worden om daaruit nuttige informatie te halen? En zijn al deze technieken drastisch veranderd sinds het framework werd uitgebracht in maart 2009?

A.2 State-of-the-art

Node.js is reeds een matuur framework. Zoals gezegd wordt in (Ancona e.a., 2017): Node.js is sterk afhankelijk van asynchrone en continue programmeerstijl. I/O-bewerkingen

worden uitgevoerd door middel van oproepen naar asynchrone functies waarbij een call-back moet worden doorgegeven om aan te geven hoe de berekening wordt voortgezet zodra de genoemde I/O-bewerking asynchroon is voltooid. Het Node.js executiemodel bestaat uit een hoofdgebeurtenislus die wordt uitgevoerd op een single-threaded proces. Het is daarom niet makkelijk om deze soort frameworks te debuggen, en wordt uiteindelijk een uitdagende taak. Gelukkig zijn hiervoor dan ook weer verschillende hulpmiddelen en tools uitgebracht om dit te vereenvoudigen. Zoals hierboven vermeld in (Ancona e.a., 2017), kan asynchrone code subtiele bugs opleveren die niet meteen zichtbaar zijn. Hier is nog niet echt onderzoek over gedaan. Waarin er honderden vergelijkende studies bestaan over de frameworks zelf en of Node.js een goede optie is, zijn er geen of amper studies over de beste manier om dit te debuggen en hoe men dit het best aanpakt.. (Ancona e.a., 2017) vertelt ons meer over het identificeren van schaalbaarheidsproblemen en het aanrijken van mogelijke oplossingen. Ze maken gebruik van parametrische uitdrukkingen voor runtime monitoring van Node.js toepassingen, maar ze geven toe dat dit nog maar de eerste stap is en dat hier nog meer onderzoek naar kan gedaan worden. Doordat hier nog niet veel over staat neergepend, leek me dit een zeer interessant onderwerp dat hand in hand gaat met mijn stageopdracht. Kayzr, mijn stagebedrijf, zoekt namelijk zelf goede manieren om hun Node.js applicaties goed te kunnen debuggen, en deze studie zou zeker een goede bijdrage kunnen zijn aan deze soort doelgroep. Mijn onderzoek zal zich onderscheiden door een goed overzicht te behouden in die ongedocumenteerde zee van beschikbare frameworks, tools en software, waardoor deze studie kan gebruikt worden als een guideline voor best-practices.

A.3 Methodologie

We starten door literatuuronderzoek te doen naar de verschillende mogelijkheden van software en tools. Ook kunnen we literatuuronderzoek doen naar Javascript van ES1 naar ES6, NodeJS en zijn asynchrone processen, monitoring en het monitoren van async processen. Na de literatuurstudie maken we een steekproef van een aantal node-developers door hen te contacteren en via een enquete hen te vragen welke tools zij gebruiken om hun Node.js applicaties te monitoren, uitgebracht tussen 2009 en 2019. We kunnen erna kijken welke tools performanter zijn dan anderen dankzij het gebruik van de servers van Kayzr. We kunnen er uitgebreider onderzoeken door:

- Monitoren van api calls volgens het aantal keren opgeroepen
- Monitoren van api calls volgens duratie tot een response gestuurd wordt
- Het gemak om errors op te slaan en later te debuggen/analyseren
- Monitoren van deze node process en hun taxatie op de server waar ze op draaien (CPU, RAM, netwerk. . .)

De tools voor de bovenstaande metingen te testen gaan uiteraard de te onderzoeken tools, en hulpprogramma's als taakbeheer zijn. We maken daarna gebruik van R Studio om deze metingen en enquetes om te zetten tot mooie data waaruit we hopelijk een conclusie kunnen trekken.

A.4 Verwachte resultaten

Ik vermoed dat de resultaten uiteen zullen lopen, en dat verschillende developers zich liever vasthouden aan hun eigen methodes. Ook zal er een breuk zijn tussen ontwikkelaars die liever oudere maar matuurdere tools zullen blijven gebruiken, en developers die liever het nieuwste van het nieuwste wensen te gebruiken. Dit kunnen we dan visualiseren a.d.h.v. een staafdiagram met op de x-as deaww tool en op de y-as het aantal developers.

Qua effectieve data van de overwogen tools zal elk waarschijnlijk zijn voor en nadelen hebben. Toch vermoed ik dat we er een effectieve winnaar gaan uit kunnen halen die over het algemeen goed scoort. Dit zullen we dan toetsen aan de literatuurstudie om zo best practices te bekomen.

A.5 Verwachte conclusies

De wereld van Javascript is nog in volle groei, maar is toch al een pak matuurder dan vroeger. We zien dat de tools beter zijn geworden. De concurrentie is enorm groot aangezien Node.js een enorm populair ontwikkelaarsplatform is. We verwachten dat nieuwere tools meer functionaliteiten gaan bieden maar minder stabiel gaan zijn dan de reeds bestaande. De manier van aanpak zal variëren, maar uit de steekproef kunnen we dat toch veralgemenen.

Bibliografie

Ancona, D., Franceschini, L., Delzanno, G., Leotta, M., Ribaudo, M. & Ricca, F. (2017). Towards Runtime Monitoring of Node.js and Its Application to the Internet of Things. In *Proceedings First Workshop on Architectures, Languages and Paradigms for IoT, ALP4IoT@iFM 2017, Turin, Italy, September 18, 2017.* (pp. 27–42). doi:10.4204/EPTCS.264.4

Creeger, M. (2009). CTO Roundtable: Cloud Computing. *Communications of the ACM*, *52*(8), 50–56.

Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.

Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Gent: Academia Press.