Orbit

Onderzoek

Wat zijn de specifieke eisen van de developers?

08-03-2023

Naar	Luuk de Weijer
	Scrumble
Betreft	Orbit implementatie
Auteur(s)	Servi Huijbregts
Email	servi@scrumble.nl

Inleiding en doelstelling

Inleiding

Orbit is een applicatie die ervoor zorgt dat er meer inzicht komt in de verwerking van jobs en queues. Deze applicatie moet gemaakt worden omdat het huidige systeem dat dit faciliteert, Laravel Horizon, niet toereikend en gebruiksvriendelijk genoeg is.

Doelstelling

Het doel is om een eigen variant van Laravel Horizon te ontwikkelen die wel het inzicht en de gebruiksvriendelijkheid biedt wat de ontwikkelaars van Scrumble vereisen. Het is echter nog niet bekend wat de einddoelstellingen van het project nou precies inhouden, dit moet nog verder onderzocht/aangevuld worden als de mogelijkheden van het project verder onderzocht zijn.

Het doel van dit onderzoek staat echter wel vast. Dit doel is namelijk vaststellen wat de eisen zijn vanuit de eindgebruikers met betrekking tot filteren en overzicht. In opdracht omschrijving staan namelijk veel requirements die niet volledig afgebakend zijn.

Onderzoeksvragen

Hoofdvraag	Wat zijn de specifieke eisen van de developers?
Deelvragen	Welke requirements hebben volgens de developers welke prioriteit? Welke implementatiewijze lijkt de developers de effectiefste? Als we verder op de requirements ingaan, welke zaken kunnen dan specifieker?

Deelvraag 1

Vraag Welke requirements hebben welke prioriteit volgens de developers?

Resultaat Hiervoor stel ik een overzicht op van de requirements zoals deze voorkomen in de opdrachtomschrijving. De requirements tabel ziet er dan als volgt uit, de derde kolom kan de developer een keuze maken uit een van de volgende prioriteiten: Hoogst, hoog, normaal, laag, laagst.
Op basis van een vragenlijst die ik gestuurd heb naar de developers heb ik het gemiddelde genomen van alle antwoorden en hieraan een prioriteit gekoppeld. Antwoorden vragenlijst

Requirement	Description	Prioriteit
Applicatie aanmaken en beheren	Een formulier met de mogelijkheid om een naam en een unieke identifier in te stellen om gemakkelijk te zoeken naar applicaties binnen de front-end van de tool.	Normaal Niet van t...
Alerts instellen	Een functie om alerts in te stellen wanneer er bijvoorbeeld 20% meer errors zijn dan de dag ervoor. Het systeem moet filters instellen en iedere minuut checken als basis hebben.	Hoog
Applicatie-dashboard	Per applicatie een overzicht (dashboard) om inzichten te krijgen in het aantal fouten, het totale aantal verwerkte jobs, de job verwerkingstijden, enz. Het dashboard moet inzicht creëren, en het kan een aantal grafieken bevatten, zoals het aantal jobs per uur, fouten per uur en alert notificaties.	Hoog

Geschiedenis van jobs	Een functie om de geschiedenis van jobs te kunnen inzien en te kunnen zoeken op ID, gebruiker, datum, naam, enz. Waarschijnlijk een simpele tabel met filter functionaliteiten.	Hoogst
Job-informatie inzien	Een functie om een job te kunnen openen om de precieze stappen in te kunnen zien. Hier kunnen bijvoorbeeld constructor data, return data, data die naar de database zou gaan en exception stack traces getoond worden. Dit is de belangrijkste functie van de applicatie.	Hoogst
Algemeen dashboard	Een algemeen dashboard van alle applicaties met daarin de belangrijkste zaken op één scherm inzichtelijk. Het dashboard kan bijvoorbeeld globale (gegroepeerde) grafieken bevatten en de meest kritische problemen die spelen, zoals ingestelde alerts "+20% more exceptions in the last hour for SomeProject!" of als ergens Horizon niet draait.	Normaal
User Friendliness	Een functie om een job te kunnen retrying vanuit de front-end nadat de error is gefixt. Eventueel kunnen een hele batch jobs in 1x gerepareerd worden. Het moet ook mogelijk zijn om queues of hele Horizon's in 1x te clearen, vooral als er een backlog ontstaat en geen enkele job meer doorgaat.	Hoog
Security	Al onze applicaties sturen en verwerken gevoelige bedrijfsgerelateerde data, persoonsgegevens, errors, en ingevulde klantdata in jobs. Het is van het hoogste belang dat dit voor	Hoogst

	niemand buiten Scrumble, wellicht niemand buiten ons kantoor in te zien is / gehackt kan worden.	
Performance	Voor applicaties die veel verkeer(en users) hebben kun je je inbeelden dat het gaat over duizenden jobs die per dag naar de applicatie gestuurd gaan worden. De applicatie, alle inzichten, en natuurlijk alle verbonden applicaties (en Horizons) moeten allemaal snel blijven werken, zelfs met deze grote hoeveelheden data.	Hoog

Verder heb ik in deze vragenlijst ook nog wat extra verdieping gevraagd voor sommige requirements. Dit ging specifiek over de requirements: Alerts instellen, Applicatie-dashboard, algemeen dashboard en geschiedenis van jobs.

De vragen inclusief de antwoorden volgen hier:

Requirement: Alerts instellen *Een functie om alerts in te stellen wanneer er bijvoorbeeld 20% meer errors zijn dan de dag ervoor. Het systeem moet filters instellen en iedere minuut checken als basis .* Mijn invulling momenteel is een alert in kunnen stellen met een bepaalde waarde via de front-end en wanneer deze wordt overschreden een Discord notificatie sturen. Wat zou jouw invulling zijn voor deze requirement?
Ik zou het baseren op een tijdframe waar het op gebaseerd moet worden (1 dag, 1 week, 1 maand, etc van tevoren) en een threshold in procenten. Idealiter zou dit gaan met een discord notification. Echter moet er wel iets van een backoff inzitten zodat er niet gespamd gaat worden.
Discord notificatie is ok
Ik kan me voorstellen dat sommige jobs die falen niet meteen een probleem zijn, maar wel als ze 100 keer falen. En andere jobs zoals van betalingen die niet eens 1 keer mogen falen. dan wil je bij de ene job al na 1 error een alert en bij de

andere pas na 100, dus op basis van een job tag oid de threshold kunnen instellen.
Discord notificatie is goed, ik zou dit ook binnen het dashboard zelf tonen (bijv een grote rode balk). Ik denk ook dat er verschillende selecties moeten zijn, oa: - errored jobs - total jobs - pending jobs (queue size growing) - applicatie-specifieke selectie (specifieke Exception class afvangen)
Het in kunnen stellen dat er alerts op bepaalde jobs zitten, dus wanneer belangrijke job #1 faalt er een discord notificatie verstuurd wordt. Dit zou ook in de vorm van batches kunnen, zijn er 10 jobs die altijd moeten lopen, faalt er 1/2 van, laat dat weten via een alert.

Requirement: Applicatie-dashboard *Per applicatie een overzicht (dashboard) om inzichten te krijgen in het aantal fouten, het totale aantal verwerkte jobs, de job verwerkingstijden, enz. Het dashboard moet inzicht creëren, en het kan een aantal grafieken bevatten, zoals het aantal jobs per uur, fouten per uur en alert notificaties.* Het overzicht wat Laravel Horizon momenteel zelf biedt omvat momenteel aardig wat metrics in de vorm van cijfers. (Zie afbeelding hieronder) Hoe zie jij als eindgebruiker dit dashboard voor je? Wil je bijvoorbeeld grafieken zien, en wat voor data zou er dan naar jou mening het belangrijkst zijn om te weergeven op dit dashboard?
Jobs per minute en jobs per hour zou wel nice zijn als grafiek. Rest is prima als cijfer. Op het moment zou ik van het stuk overview niet echt weten wat ik nog meer zou willen.
Grafiek van hoeveel jobs en hoeveel hiervan gefaald zijn lijkt me een duidelijke weergave van de data. Eventueel dat je hier dan binnen filters meerdere projecten kan selecteren wanneer je kiest voor de microservice.
Ik denk dat de data in vorm van grafieken minder belangrijk is dan daadwerkelijke controle en inzicht hebben over individuele jobs. Maar als er dan toch een dashboard komt is failure rate per tag misschien wel nice om te hebben. Dan kun je bijvoorbeeld zien dat job syncs in Droppery vaak falen.
Het standaard Horizon dashboard is een goed begin. Het allerbelangrijkste is dat het dashboard *inzicht* biedt, voornamelijk in *problemen*. Als er geen problemen zijn, kijk ik hier waarschijnlijk niet naar. Dus snel kunnen zien of er veel errors zijn bijvoorbeeld (ten opzichte van anders). Verder zou ik graag een grafiek zien met het aantal jobs wat wordt afgehandeld over verloop van tijd.
Voor deze applicatie lijkt het me goed om in 1 oogopslag te zien waar dingen fout

zijn gegaan. Wanneer het een dashboard van 1 project is kan er ingezoomd worden op specifieke jobs. Bij een globaal dashboard wil je zien in welke projecten de grootste fouten zitten. Misschien een idee om jobs te kunnen indelen in hoe cruciaal ze zijn voor het functioneren van het project, daar kan dan ook mee gespeeld worden op het dashboard.

Requirement: Algemeen-dashboard *Een algemeen dashboard van alle applicaties met daarin de belangrijkste zaken op één scherm inzichtelijk. Het dashboard kan bijvoorbeeld globale (gegroepeerde) grafieken bevatten en de meest kritische problemen die spelen, zoals ingestelde alerts "+20% more exceptions in the last hour for SomeProject!" of als ergens Horizon niet draait.* Het overzicht wat Laravel Horizon momenteel zelf biedt omvat momenteel aardig wat metrics in de vorm van cijfers. (Zie afbeelding hierboven bij de voorgaande vraag) Hoe zie jij als eindgebruiker dit algemene dashboard voor je? Wil je bijvoorbeeld grafieken zien, en wat voor data zou er dan naar jouw mening het belangrijkst zijn om te weergeven op dit dashboard?
Ik denk niet dat deze feature nodig is. Het enige wat je er naar mijn mening zou moeten laten zien is een overzicht van projecten/queues waar problemen mee zijn.
Grafieken zijn altijd nice, qua kritische problemen is het handig om in 1 oogopslag een samenvatting van de unieke errors in te kunnen zien.
Ik denk inderdaad dat +20% more exceptions for Someproject wel fijn is inderdaad. dan valt het op zodra dingen ineens niet meer goed lopen. Verder lijkt het me niet nuttig om projecten onderlingen te vergelijken, want een project als Droppery gaat altijd meer errors gooien dan een project dat minder complex is.
Duplicate volgens mij
Check het vorige antwoord. Hou het simpel en met core informatie waar op door te klikken is, een idee om zoveel mogelijk op het dashboard te houden en met expandable/accordeon onderdelen te werken.

Requirement: Geschiedenis van jobs *Een functie om de geschiedenis van jobs te kunnen inzien en te kunnen zoeken op ID, gebruiker, datum, naam, enz. Waarschijnlijk een simpele tabel met filter functionaliteiten.*

Momenteel kun je in Laravel Horizon niet filteren, de jobs worden per 50 gepaginate ingeladen. Stel dat je de mogelijkheid had om via de Scrumble Quick Table te filteren door alle jobs, waarop zou jij als eindgebruiker dan willen filteren?
Ik zou vooral graag willen zoeken op: - daterange - job type - en text search in params
Queue, eventuele parameters v.d. job
Het grootste probleem waar ik nu tegenaan loop met horizon is dat je alleen maar op tags kan filteren en dat volstaat bijna nooit. Aantal filter/sort manier die ik graag zou zien zijn: - Filteren op een datum/tijd range. - Sorten op datum - Filteren op queue - Filteren op tags - Status: pending/completed/failed
Job name, job ID, job tag(s), queue name, status pending/completed/errored, datum/tijd
Hoe vaak een job voor komt, project, niveau van belang, datum.

Conclusie

Op basis van de opdrachtomschrijving en de feedback van de ontwikkelaars is er een overzicht gemaakt van de vereisten voor de ontwikkeling van de applicatie. Dit overzicht bevat een tabel met de vereisten en hun prioriteiten, waarbij prioriteit kan worden gekozen uit vijf niveaus, van hoogste tot laagst. Door middel van een vragenlijst en het gemiddelde van de antwoorden van de ontwikkelaars is er een prioriteit toegekend aan elke vereiste.

De belangrijkste functies van de applicatie zijn het inzien van jobinformatie en de beveiliging van gevoelige bedrijfsgegevens. Ook is er grote nadruk gelegd op de prestaties van de applicatie, met het oog op de verwerking van duizenden jobs per dag. Daarnaast is er ook aandacht

besteed aan het gebruiksgemak van de applicatie, zoals het kunnen retryen van een job vanuit de front-end en het clearen van queues en hele Horizons.

Er is extra verdieping gevraagd voor sommige vereisten, zoals het instellen van alerts en het maken van dashboards. Uit de antwoorden van de ontwikkelaars blijkt dat het instellen van alerts moet worden gebaseerd op een tijdsframe en een threshold in procenten, en dat er een backoff moet zijn om spamming te voorkomen. Discord-notificaties zijn geschikt voor dit doel. Bij het maken van dashboards moet rekening worden gehouden met de behoeften van de gebruikers en de informatie die ze nodig hebben om hun taken uit te voeren.

Al met al biedt het overzicht van vereisten een duidelijk beeld van wat er van de ontwikkelaars wordt verwacht en wat de prioriteiten zijn bij het ontwikkelen van de applicatie.

Deelvraag 2 & 3

Vraag Welke implementatiewijze lijkt de developers de effectiefste?

Resultaat Uit onderzoek blijkt dat ik twee mogelijkheden heb voor de implementatie van het Laravel Horizon Insights project, namelijk: Package-release of volledige applicatie.

Hierbij wil ik graag weten wat de eindgebruikers denken dat het meeste

waarde toevoegt aan hun workflow. Hiervoor heb ik een vragenlijst gemaakt die eerst naar de lead developer gestuurd wordt voor feedback en vervolgens naar al de andere developers.

De vragenlijst kan hier gevonden worden: Vragenlijst

De antwoorden hierop zijn als volgt:

Er zijn twee mogelijkheden waartussen ik momenteel nog twijfel, onderzoek heeft tot nu toe uitgewezen dat beiden zouden kunnen voldoen aan de requirements. Mogelijkheid 1: Microservice, hierbij maak ik 1 gehele applicatie die alle horizon queues/jobs vanuit de Scrumble applicaties kan monitoren. Mogelijkheid 2: Package, hierbij maak ik een package die per applicatie gebruikt kan worden net als Laravel Horizon maar meer functionaliteit biedt zoals die gevraagd wordt in de requirements. (Denk aan Linguist) **De vraag is nu: Wat lijkt jou als eindgebruiker de beste mogelijkheid en waarom?**
Mogelijkheid 2. Ik zie geen meerwaarde van het draaien als losse applicatie omdat je dit applicatie specifiek wil managen. En alles op 1 plek alleen maar de complexiteit onnodig ver4d.
Ligt aan het einddoel, als het om het monitoren van meerdere applicaties tegelijk gaat dan microservice. Bij het optimaliseren van bijvoorbeeld 1 project lijkt me package het beste. De microservice heeft ook als nadeel dat dit voor lokaal dan apart moet draaien waardoor je het idee van de microservice kwijt bent.
Mogelijkheid 1 lijkt mij voor gebruik makkelijker.
Optie 1 zodat ik makkelijk een verzamel dashboard kan zien en in 1 oogopslag zien of ALLE applicaties goed draaien. Optie 2 zodat we meer gerichte features kunnen ontwikkelen, waarschijnlijk betere performance garanderen, en makkelijker updates doorvoeren aan applicaties aangezien het een package is Ofwel, ik ben on the fence, beide hebben voor- en nadelen
Ik neig meer naar 1 dan naar 2, het lijkt me handig om alles op 1 plek te hebben zodat we het snel kunnen inzien en kunnen switchen tussen projecten. Dit kunnen

we dan weer koppelen aan andere monitor tools die we hebben/aan het ontwikkelen zijn. Met 1 wordt het meer ons product, een optie tot het brengen van een saas product.

Conclusie

Op basis van de feedback van de eindgebruikers lijkt er geen duidelijke voorkeur te zijn voor de implementatie van het Laravel Horizon Insights project. Beide opties hebben hun voor- en nadelen en het hangt af van het einddoel en de specifieke vereisten van de gebruiker.

Een deel van de eindgebruikers lijkt de voorkeur te geven aan de microservice-optie (optie 1) omdat dit hen in staat stelt om alle applicaties op één plek te monitoren en snel tussen projecten te schakelen. Dit kan handig zijn voor het overzicht en het gebruik van andere monitor tools.

Een ander deel van de eindgebruikers geeft de voorkeur aan de package-optie (optie 2) vanwege de gerichte functionaliteit en mogelijke betere performance, evenals de mogelijkheid om gemakkelijker updates door te voeren aan de applicaties.

Al met al lijkt het verstandig om te blijven onderzoeken en te overleggen met de eindgebruikers om te bepalen welke optie de meeste waarde toevoegt aan hun workflow.

Conclusie

Algehele conclusie

Op basis van de deelvragen kunnen we concluderen dat het belangrijk is om te achterhalen welke requirements voor de developers de hoogste prioriteit hebben en welke implementatiewijze zij als de effectiefste beschouwen. Daarnaast is het relevant om verder in te gaan op de specifieke details van de requirements om zo een duidelijk beeld te krijgen van de benodigdheden van de developers. Door deze informatie te verzamelen, kunnen we beter tegemoetkomen aan de eisen en behoeften van de developers en zo het ontwikkelproces optimaliseren.