Projectplan

Laravel Horizon Insights Applicatie

Scrumble

Tilburg

Datum	**:**	**14-02-2023**
Versie	**:**	**Versie 0.5**
Status	**:**	**Work in Progress**
Auteur	**:**	**Servi Huijbregts**

Versie

Versie	Datum	Auteur(s)	Wijzigingen	Status
0.1	07-02-2023	Servi Huijbregts	Projectopdracht, projectorganisatie en tijdplan toegevoegd	Feedback received
0.2	08-02-2023	Servi Huijbregts	Onderzoeksvragen met methodes en testaanpak/strategie toegevoegd	Feedback received
0.3	13-02-2023	Servi Huijbregts	Feedback Lennart verwerkt v1	Submitted for feedback
0.4	14-02-2023	Servi Huijbregts	Testaanpak aangevuld met test driven development hoofdstuk.	Submitted for feedback
0.5	14-02-2023	Servi Huijbregts	Feedback Lennart verwerkt v2	Submitted for feedback

Verspreiding

Versie	Datum	Aan	Reden
0.1 & 0.2	08-02-2023	Lennart de Graaf	Opgeleverd voor tussentijdse feedback
0.3, 0.4 & 0.5	14-02-2023	Lennart de Graaf	Opgeleverd voor tussentijdse feedback
0.5	20-02-2023	Herman Jurjus	In GRAD8-CMK course opgeleverd

Inhoudsopgave

1. Projectopdracht

1.1 Context

Het softwarebedrijf Scrumble heeft als missie om bedrijven een klantvriendelijke mogelijkheid te bieden om te bouwen aan organisaties die klaargestoomd zijn voor de toekomst. Ze willen de digitale wereld begrijpbaar en tastbaar maken en doen dit op een transparante, eerlijke en persoonlijke wijze. Naast transparantie, eerlijkheid en persoonlijkheid zijn innovatie, professionaliteit en kwaliteit ook belangrijke waarden voor het bedrijf.
Sinds het ontstaan van Scrumble zijn er een hele hoop applicaties ontwikkeld. Om precies te zijn hebben staan er op het moment 34 live. Deze applicaties verschillen enorm, maar één ding hebben ze met elkaar gemeen: dat ze complexe asynchrone logica uitvoeren. Dit varieert van zware processen om bijvoorbeeld vakantiepakketten te genereren, tot bedrijfsprocessen binnen het magazijn van een webshop automatiseren en optimaliseren. Het is van groot belang dat deze processen succesvol lopen, dag en nacht, 7 dagen per week, zonder support.

1.2 Doel van het project

De complexe logica die ik zojuist benoemt heb wordt momenteel gedraaid door middel van Laravel Horizon. Laravel Horizon is een first-party package voor Laravel die het mogelijk maakt om “jobs” in een “queue” te zetten, om op een later moment, a-synchroon af te handelen. Een voorbeeld hiervoor is het versturen van een welkomst mail naar een nieuwe gebruiker. Deze mail moet bijvoorbeeld verstuurd worden via SMTP (Simple Mail Transfer Protocol), dit kan even duren. Je wilt voorkomen dat het aanmaken van een gebruiker langer duurt, doordat je moet wachten op het versturen van deze mail. Laravel heeft hiervoor een “dispatch” functionaliteit, je verzend je job dan om later uitgevoerd te worden, je gebruiker aanmaken is echter wel direct afgerond.
Dit “dispatchen” van “jobs” wordt heel veel gedaan bij Scrumble, in sommige applicaties draaien zelfs duizenden tot tienduizenden jobs per dag. Er gaan echter soms nog wat dingen fout. Dit kan zijn doordat de gebruiker of klant foutieve data heeft ingevoerd, een externe API errors geeft, of natuurlijk een bug die door Scrumble veroorzaakt is. Laravel Horizon biedt hiervoor een interface om in te kunnen zien dat er iets fout gaat. Je kunt zien welke “jobs” er in de queue staan, welke afgerond zijn en welke fout zijn gegaan. Als je wilt inzien wat er precies fout is gegaan wordt dat lastig. Al helemaal als er bijvoorbeeld 2000 jobs op een dag fout zijn gegaan. Naast dat je dus totaal geen inzicht hebt, vereist het ook vaak nog een handmatige actie. De klant moet zelf vaak aangeven dat er een fout is binnen het systeem en Scrumble moet deze fout dan handmatig opzoeken.
Het zou natuurlijk ideaal zijn als Scrumble automatisch een melding krijgt als er iets fout gaat en ook meteen inzicht krijgt wat er fout gaat en waar dit precies gebeurt. Daarvoor wordt de Laravel Horizon tool ontwikkeld.

Ik heb een aantal requirements op een rijtje gezet om een duidelijker overzicht te bieden van hetgeen dat gerealiseerd moet worden:
Om onderstaande requirements tabel beter te begrijpen is het belangrijk om de volgende begrippen uit te leggen:
Applicatie: Met applicatie in de requirements tabel wordt een applicatie bedoeld die Scrumble heeft ontwikkeld, als voorbeeld geef ik de applicatie “Pulse”. Als ik in de requirements tabel spreek van een applicatie aanmaken betekent dit niets meer dan een rij in de database toevoegen met een id, naam en omschrijving van deze applicatie.
Alerts: Als ik het in onderstaande tabel heb over alerts bedoel ik hiermee een melding die in de frontend getoond wordt. Als ik het heb over het instellen van deze alerts bedoel ik het via de frontend invullen van een formulier waarin je de parameters geeft over wanneer de backend een alert moet terugsturen naar mijn frontend.
Jobs: Als ik het in onderstaande tabel heb over jobs bedoel ik hiermee een laravel job. Een job in Laravel Queue is een taak die asynchroon wordt uitgevoerd op de achtergrond en die kan worden geactiveerd door een gebeurtenis in de applicatie, zoals een gebruikersactie of een geplande taak. Hier zijn enkele voorbeelden van jobs die in een Laravel Queue kunnen voorkomen:

- Versturen van e-mails: een job kan worden toegevoegd aan de Laravel Queue om e-mails in de achtergrond te verzenden, zodat de gebruiker de applicatie kan blijven gebruiken zonder te hoeven wachten op de voltooiing van het e-mailproces.
- Verwerken van betalingen: een job kan worden gebruikt om betalingsverwerking uit te voeren, zoals het controleren van de betalingsstatus en het bijwerken van de database, zonder de gebruiker te storen.
- Genereren van rapporten: een job kan worden gebruikt om rapporten te genereren, zoals PDF's of Excel-bestanden, zodat de gebruiker de applicatie kan blijven gebruiken terwijl de job in de achtergrond wordt uitgevoerd.
- Importeren of exporteren van gegevens: een job kan worden gebruikt om gegevens te importeren of exporteren, zoals het uploaden van bestanden of het bijwerken van de database met externe gegevens.

De Laravel Queue is een handig hulpmiddel om taken op de achtergrond uit te voeren en de prestaties van de applicatie te verbeteren door het verminderen van de wachttijden voor de gebruiker.

Vereiste	User story	Prioriteit
De mogelijkheid om applicaties aan te maken en te beheren.	Als projectbeheerder wil ik applicaties kunnen aanmaken en beheren, zodat ik binnen de front-end van deze tool makkelijk kunnen zoeken naar applicaties.	HOOG
De mogelijkheid om alerts in te kunnen stellen wanneer er een bepaalde drempel bereikt wordt.	Als projectbeheerder wil ik alerts kunnen instellen, zodat ik zelf controle heb over wanneer er een notificatie komt.	GEMIDDELD
Per applicatie moet er een overzicht komen om inzichten te krijgen in bijvoorbeeld het aantal fouten, verwerkte jobs en verwerkingstijden van jobs.	Als projectbeheerder wil ik een overzicht hebben per applicatie, zodat ik inzicht kan krijgen in de status van de queues in de applicaties.	HOOG
Je moet de geschiedenis van jobs kunnen inzien en kunnen zoeken op bijvoorbeeld ID of datum.	Als projectbeheerder wil ik de geschiedenis van jobs in kunnen zien en kunnen filteren, zodat ik makkelijker kan zoeken naar een specifieke job.	GEMIDDELD
Een job moet open te klikken zijn waardoor je de precieze stappen van de job in kunt zien.	Als projectbeheerder wil ik een job kunnen openen, zodat ik de precieze stappen van een job kan zien en kan zien waar het fout gaat.	HOOG
Een totaaloverzicht van alle applicaties met daarin de belangrijkste data.	Als projectbeheerder wil ik een algemeen overzicht, zodat ik een dashboard altijd open kan hebben en de meest kritische problemen in één oogopslag kan zien.	GEMIDDELD
Het systeem moet gebruiksvriendelijk zijn.	Als projectbeheerder wil ik dat het systeem gebruiksvriendelijk is, zodat ik vanuit de front-end direct zaken kan regelen zoals het opnieuw uitvoeren van jobs.	GEMIDDELD

1.3 Begrenzing en Randvoorwaarden

Tot het project behoort:	Tot het project behoort niet:
Projectplan	Onderhoudswerkzaamheden aan het geleverde systeem
Uitgewerkte onderzoeksvragen (Analyserapport)	Vergroting van de scope bij verandering van requirements
Frontend ontwikkeling	
Backend ontwikkeling	
Rekening houden met de niet-functionele requirements zoals performance en security	
Rapporteren van voortgang	
Project management via het intranet van Scrumble genaamd Pulse	
Sprint reviews met de product owner	
Unit testen	
Integratie testen	
Acceptatie testen	
Usability testing (gelinkt aan requirement “het systeem moet gebruiksvriendelijk zijn”)	

1.4 Strategie

De werkwijze die ik hanteer tijdens de stageperiode is een Scrum werkwijze. Agile Scrum is een effectieve en flexibele manier van werken die het mogelijk maakt om projecten op een productievere manier op te leveren en problemen snel te onderscheppen. Scrum is een ideale werkwijze voor projecten waarbij vooraf nog geen gedetailleerde planning kan worden opgesteld. Bij scrum werk je in sprints, dit zijn periodes van een vooraf bepaalde lengte waarin je kunt werken. Deze sprints worden ingedeeld met issues die vervolgens weer taken bevatten.
De sprints voor mijn project zijn 2 weken lang. Aan het einde van iedere sprint vind een sprint review en een retrospective plaats. Tijdens de sprint review wil ik een samenvatting geven van de afgelopen sprint, een demo geven van het gemaakte werk, eventuele bespreekpunten bespreken, een planningsvoorstel voor de komende sprint geven en een retrospective uitvoeren.
De retrospective zal worden ingedeeld in de vragen:

- Wat ging er tijdens de afgelopen sprint goed?
- Wat ging er minder goed en kan in de volgende sprint beter?

Uiteindelijk is het dan weer een kwestie van de besproken planning verwerken in issues en in een sprint zetten.
Voor nu ziet de sprint planning er als volgt uit:

Sprint #	Startdatum	Einddatum
Sprint 1	6 Feb 2023	17 Feb 2023
Sprint 2	20 Feb 2023	3 Mar 2023

Sprint 3	6 Mar 2023	17 Mar 2023
Sprint 4	20 Mar 2023	31 Mar 2023
Sprint 5	3 Apr 2023	14 Apr 2023
Sprint 6	17 Apr 2023	28 Apr 2023
Sprint 7	1 May 2023	12 May 2023
Sprint 8	15 May 2023	26 May 2023
Sprint 9	29 May 2023	9 June 2023
Sprint 10	12 June 2023	23 June 2023

Deze sprint planning staat nog niet vast en kan altijd nog aangepast worden als er geconstateerd wordt dat de huidige sprint grootte niet handig of fijn werkt.

1.5 Onderzoeksvragen

De onderzoeksvragen zullen in de loop van de stage nog aangevuld worden met eventuele verdere onderzoeksvragen. Om de stageperiode en de doelen zo goed mogelijk af te kaderen is het belangrijk om voorafgaand aan de stage een goede set met onderzoeksvragen te hebben. Op dit moment zit ik in de vooronderzoeksfase. In deze fase is het vooral belangrijk om de scope volledig te bepalen en de opdracht volledig af te bakenen. Een aantal vragen die hiervoor relevant kunnen zijn, zijn de volgende vragen:

- Wat is Laravel Horizon en hoe werkt het?
    - Bij deze vraag ga ik kort in op wat Laravel Horizon is en hoe het werkt om de lezer van het vooronderzoek meer context te geven van het te realiseren product.

- Welke gegevens zijn er momenteel wel en niet beschikbaar in Laravel Horizon?
    - Bij deze vraag ga ik dieper in de materie duiken en een lijst van momenteel beschikbare endpoints opstellen en laten zien wat voor data deze endpoints terugsturen. Ook is het belangrijk om naar de requirements te kijken en te controleren wat er juist niet beschikbaar is vanuit de Laravel Horizon applicatie.

- Wat zijn de uitdagingen bij het samenvoegen van gegevens uit meerdere Laravel Horizon-instanties?
    - Bij deze vraag ga ik verder in op de mogelijke uitdaging die bij het samenvoegen naar voren komen zoals problemen met autorisatie en hoe projecten geconfigureerd moeten worden om tot het uitlezen van api data te komen.

- Zijn er bestaande tools voor het samenvoegen van gegevens van meerdere Laravel Horizon-instanties?
    - Hierbij ga ik onderzoeken of er momenteel al tools zijn die ervoor zorgen dat de gegevens van verschillende horizon instanties samengevoegd worden, mocht dit zo zijn is het namelijk een afweging of er niet beter gebruik gemaakt kan worden van deze tool in plaats van het zelf te ontwikkelen.

- Wat zijn de belangrijkste items die in het dashboard moeten worden bijgehouden?
    - Hierbij ga ik de requirements vanuit de projectomschrijving met betrekking tot de frontend op een rijtje zetten en verder definiëren. Als deze requirements uitgewerkt zijn is er ook duidelijk wat er vanuit de backend aan de frontend geleverd moet worden en kunnen er uiteindelijk diagrammen gemaakt worden vanuit deze analyse.

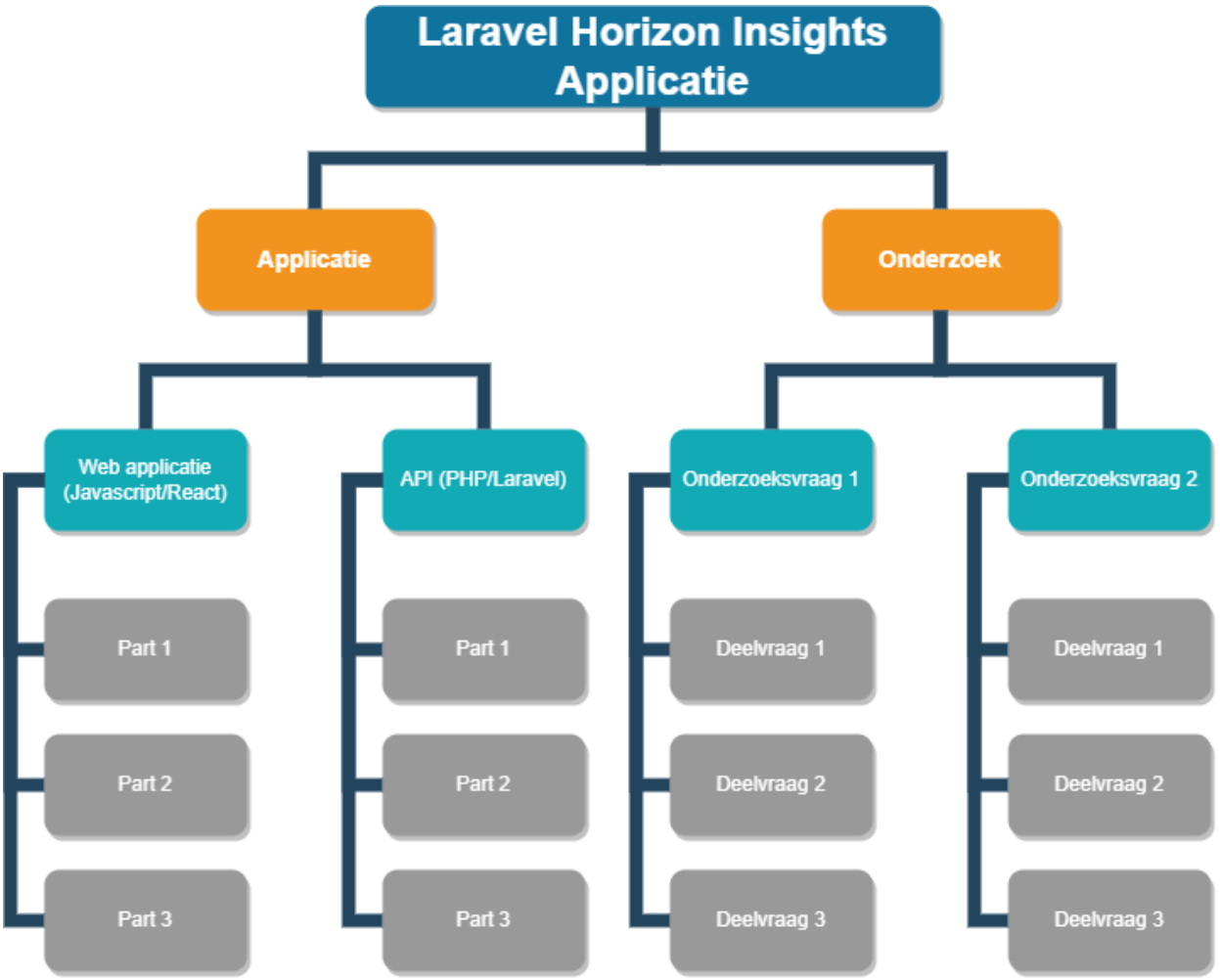
- Wat zijn de beveiligings implicaties van het samenvoegen van gegevens van meerdere Laravel Horizon-instanties?
    - Hierbij ga ik onderzoeken hoe de beveiliging van de Laravel Horizon API in elkaar zit en of er mogelijke beveiligingsrisico's zijn die ervoor zorgen dat het realiseren van dit systeem niet veilig genoeg is. Mochten er weinig beveiliging implicaties zijn ga ik nog kijken naar een mogelijkheid om het systeem alleen op het ip-adres van Scrumble te draaien om zo nog meer beveiliging aan de data toe te voegen.

- Hoe kunnen de API en het dashboard worden geoptimaliseerd voor prestaties en schaalbaarheid?
  (Performance best practices laravel, react & uitzoeken verticaal of horizontaal schaalbaar voor usecase)
    - Hierbij ga ik kijken naar de best practices voor zowel Laravel als React (de tools waarmee ik het systeem ga ontwikkelen) met betrekking tot performance. Ook ga ik uitzoeken of het systeem het beste verticaal, horizontaal of überhaupt schaalbaar moet zijn.

Nadat ik deze vragen heb beantwoord is de scope een stuk duidelijker en kan ik een afgebakende onderzoeksvraag vaststellen. Vanuit deze onderzoeksvraag zullen een aantal deelvragen naar voren komen die ik ga beantwoorden met behulp van de DOT-methodes.

1.6 **Eindproducten**

Voor nu heb ik een tijdelijke variant van mijn PBS ingevoegd. Deze wordt nog aangevuld als ik meer duidelijkheid heb geschept met mijn analysefase.


Laravel Horizon Insights Applicatie
Applicatie
Onderzoek
Web applicatie (Javascript/React)
API (PHP/Laravel)
Onderzoeksvraag 1
Onderzoeksvraag 2
Part 1
Part 2
Part 3
Part 1
Part 2
Part 3
Deelvraag 1
Deelvraag 2
Deelvraag 3
Deelvraag 1
Deelvraag 2
Deelvraag 3

2. Projectorganisatie

2.1 Teamleden

Naam + tel + e-mail	Afk.	Rol/taken	Beschikbaarheid
Luuk de Weijer Luuk@scrumble.nl 0135356210	Luuk	Bedrijfseigenaar, lead developer en product owner.	Deze persoon is 5 dagen per week op kantoor om ondersteuning te bieden waar nodig. Is noodzakelijk tijdens scrum meetings aanwezig (sprint review etc.)
Rico Clark Rico@scrumble.nl 0135356210	Rico	Bedrijfseigenaar en lead developer.	Deze persoon is 5 dagen per week op kantoor om ondersteuning te bieden waar nodig.
Lennart de Graaf L.degraaf@fontys.nl 0653804656	Lennart	Docentbegeleider vanuit FHICT - Open learning	Lennart werkt 5 dagen per week, de locatiebezoeken kunnen het beste op maandag en vrijdag uitgevoerd worden aangezien Lennart dan fysiek in Tilburg aanwezig is.

2.2 Communicatie

Iedere maandag: Week kick-off (Bedrijfsactiviteiten van afgelopen week en komende week worden tijdens deze kick-off besproken met alle medewerkers. Zo worden bijvoorbeeld de mogelijke en succesvolle leads genoemd)

Iedere dag rond 10:00: Daily stand up (Dagelijkse stand up waarin iedereen aangeeft wat hij/zij gisteren heeft gedaan en van plan is vandaag te gaan doen. Ook geef je aan of je eventueel ondersteuning nodig hebt van anderen)

Iedere 2 weken op woensdagmiddag: Sprint review en Retrospective (Hierin bespreek ik samen met de product owner wat er gebeurt is tijdens de sprint en wat er de volgende sprint gaat gebeuren. Bij de retrospective bespreken we wat en goed en minder goed ging.)

3. Activiteiten en tijdplan

3.1 Opdeling en aanpak van het project

Bij scrum werk je in sprints, dit zijn periodes van een vooraf bepaalde lengte waarin je kunt werken. Deze sprints worden ingedeeld met issues die vervolgens weer taken bevatten.
De sprints voor mijn project zijn 2 weken lang. Aan het einde van iedere sprint vind een sprint review en een retrospective plaats. Tijdens de sprint review wil ik een samenvatting geven van de afgelopen sprint, een demo geven van het gemaakte werk, eventuele bespreekpunten bespreken, een planningsvoorstel voor de komende sprint geven en een retrospective uitvoeren.
De retrospective zal worden ingedeeld in de vragen:

- Wat ging er tijdens de afgelopen sprint goed?
- Wat ging er minder goed en kan in de volgende sprint beter?

Uiteindelijk is het dan weer een kwestie van de besproken planning verwerken in issues en in een sprint zetten.

3.2 Tijdplan

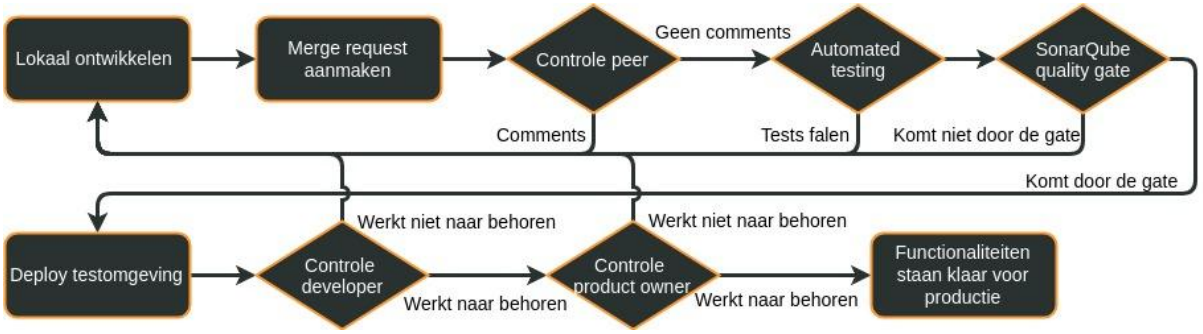
Fasering	Start	Gereed
1 Vooronderzoek & Analysefase	Sprint 1	Einde van sprint 2
2 Ontwerp & Realisatiefase	Sprint 3	Einde van sprint 8
3 Afronding & Adviesfase	Sprint 9	Einde van sprint 10

4. Testaanpak en Configuratiemanagement

4.1 Testaanpak/strategie

Voor de productie-implementatie moet alle code door een release-proces. Het begint met een merge request van een feature branch naar de development branch op GitLab, die beoordeeld wordt door een andere developer. Feedback wordt gegeven om de code efficiënter, beter leesbaar of conform codestyle te maken. Vervolgens worden unit tests uitgevoerd en als de code goedgekeurd is, wordt deze gemerged naar de development branch. Jenkins start vervolgens een SonarQube inspectie van de code. Als de code niet door de kwaliteitspoort komt, moeten de issues opgelost worden en het proces herstarten. Als het wel door de poort komt, wordt de uiteindelijke build voor de testomgeving opgeroepen en komt de code online. De developer controleert vervolgens of de functionaliteiten correct werken en de product owner controleert of alles naar wens is geïmplementeerd.

Als je dit proces in een flowchart zou moeten visualiseren ziet het er als volgt uit:



4.2 Test driven development

Bij Scrumble wordt er test driven gewerkt. Dit houdt in dat voorafgaand aan het maken van de software er eerst tests worden geschreven die het uiteindelijke doel van de software valideren. Door middel van deze tests kom je erachter welke onderdelen je allemaal nodig hebt om de software op een juiste manier te schrijven. Ik kan een voorbeeld geven van een zeer simpele functionaliteit:

Usecase: Ik wil een gebruiker kunnen aanmaken in de backend.
Test: Test schrijven met daarin de functie om een gebruiker aan te maken. Hierin gebruik ik bijvoorbeeld het model “User”, de route /api/user/store (deze gebruikt de UserController), de UserController wijst vervolgens naar de UserService en de UserService wijst vervolgens naar de UserRepository.
Wat betekent dit? Dit betekent dat de test dus aan het begin bij het gebruik van het User model al faalt omdat deze nog niet bestaat, de eerste stap is dus een User model maken. Vervolgens gaat de test de route aanroepen maar er is nog geen Controller, dus de test slaagt niet. Als je dit helemaal tot het einde doortrekt heb je alle facetten ontwikkeld om een user aan te maken en deze gevalideerd door middel van tests en kun je dus geen onderdeel missen.

4.3 Testomgeving en benodigdheden

Er wordt een onderscheid gemaakt tussen twee soorten testomgevingen. De eerste is lokaal, dit is de computer van de developer en hij test hier al zijn code. De tweede is op de acceptatieomgeving (subdomein op de server) zodat andere belanghebbenden het kunnen testen. Hiervoor zijn verder alleen de gewenste apparaten nodig om te testen (desktop of mobiel).

4.4 Configuratiemanagement

Scrumble gebruikt GitLab als VCS en heeft een repository opgezet in drie branches: Master, Development, en Feature Branches. De Master-branch is voor de productieomgeving, Development voor

de acceptatieomgeving en Feature Branches worden voornamelijk gebruikt onder de naam @Feature/{feature naam} voor gebruiksgemak en overzicht.

5. Risicoanalyse

5.1 Risicoanalyse

Hieronder zie je een tabel met daarin de risicoanalyse, de “Risico” kolom geeft een korte omschrijving van een mogelijk risico. De “Activiteiten …” kolom geeft een omschrijving van activiteiten die ik voorafgaand aan het project in plan om te voorkomen dat dit risico leidt tot falen van het project. De “Uitwijkactiviteiten” kolom geeft aan wat een uitwijkmogelijkheid is als het risico, ondanks de activiteiten ter voorkoming, toch plaatsvindt.

Risico	**Activiteiten ter voorkoming opgenomen in plan**	**Uitwijkactiviteiten**
De Laravel Horizon API biedt niet genoeg mogelijkheden.	- Onderzoeken wat de huidige mogelijkheden van de Laravel Horizon API zijn. - Vergelijken of deze mogelijkheden overeenkomen met de requirements.	Een andere queue handler stack vinden en onderzoeken.
De opdracht is te groot om binnen mijn stage te realiseren.	- Tijdens mijn analysefase ga ik vaststellen wat de omvang van de scope moet zijn om de opdracht wel binnen mijn stageperiode te realiseren. - Duidelijke backlog opstellen zodra de analyse klaar is om de sprints al te vullen en te controleren of de opdracht haalbaar is.	Meer uren maken, langer doorwerken, afspraken maken over eventuele interne doorontwikkeling.
Bedrijfsbegeleider valt weg/uit.	- De bedrijfsbegeleider is mede eigenaar van het bedrijf en zal dus niet uitvallen vanwege ontslag. - Een mogelijkheid is faillissement maar dat is niet een risico waar men bang voor is. - Een andere mogelijkheid is ziekteverzuim en in dit geval kan ik voor vragen bij andere developers terecht en officiële meetings kunnen opgevangen worden door een eventuele andere stagebegeleider of door middel van een online meeting.	Een andere bedrijfsbegeleider aangewezen krijgen vanuit het bedrijf en deze doorgeven bij Fontys.