Orbit

Ontwerp

Verschillende ontwerpen

Naar	Luuk de Weijer
	Scrumble
Betreft	Orbit ontwerpen
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.

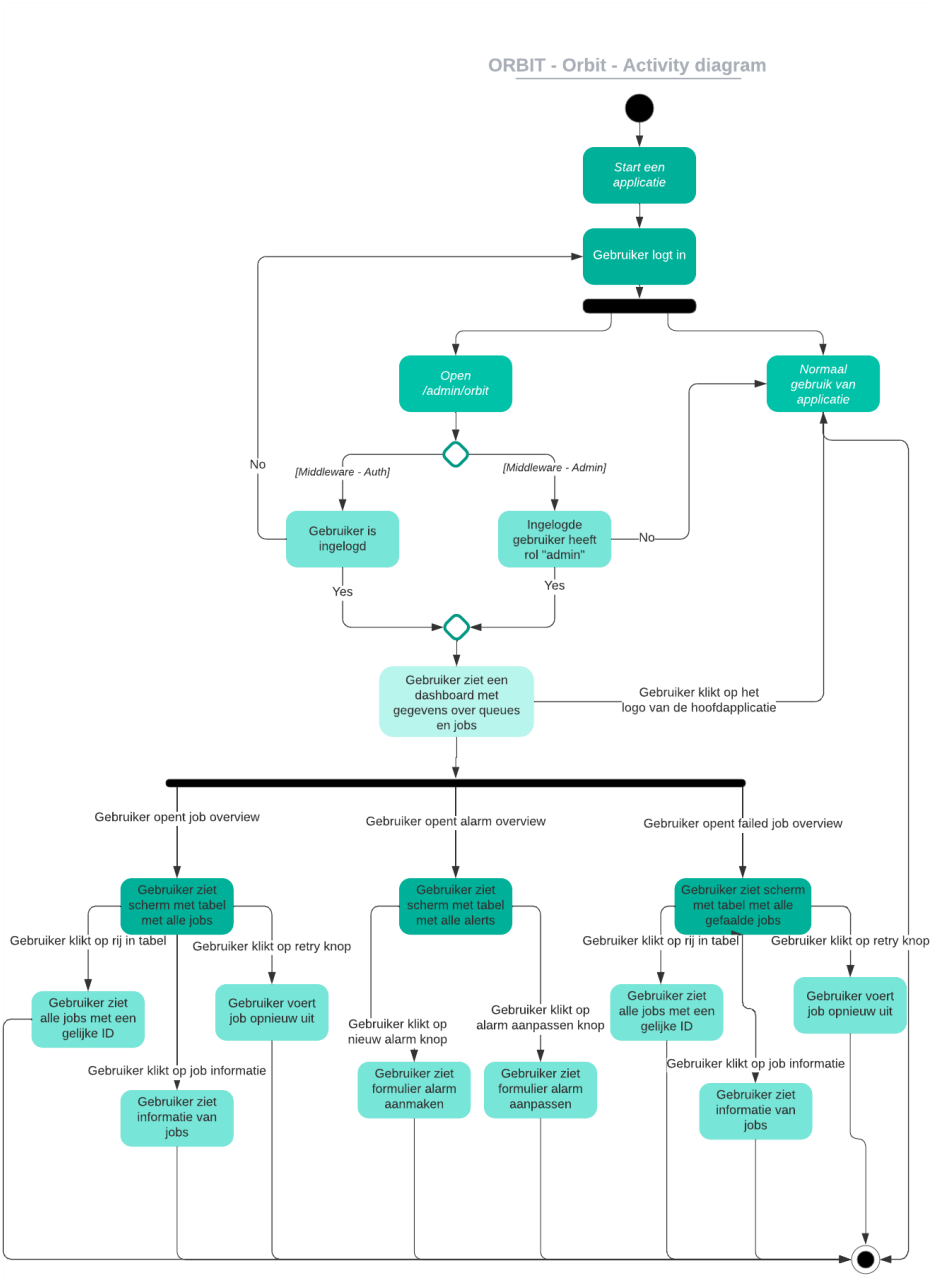
Doelstelling

Door middel van verschillende ontwerptechnieken wil ik op visuele wijze een duidelijk beeld geven van de volgende zaken: Waar staat Orbit gepositioneerd binnen de workflow van Scrumble? Wat is de workflow van Orbit zelf? Hoe is de databasestructuur van Orbit opgebouwd? Met gebruik van welke software architectuur is Orbit gebouwd?

Door middel van het beantwoorden van deze vragen met zowel ontwerptekeningen als tekstuele uitleg wil ik meer context bieden aan lezers.

Wat is de workflow van Orbit?

Activity diagram Orbit

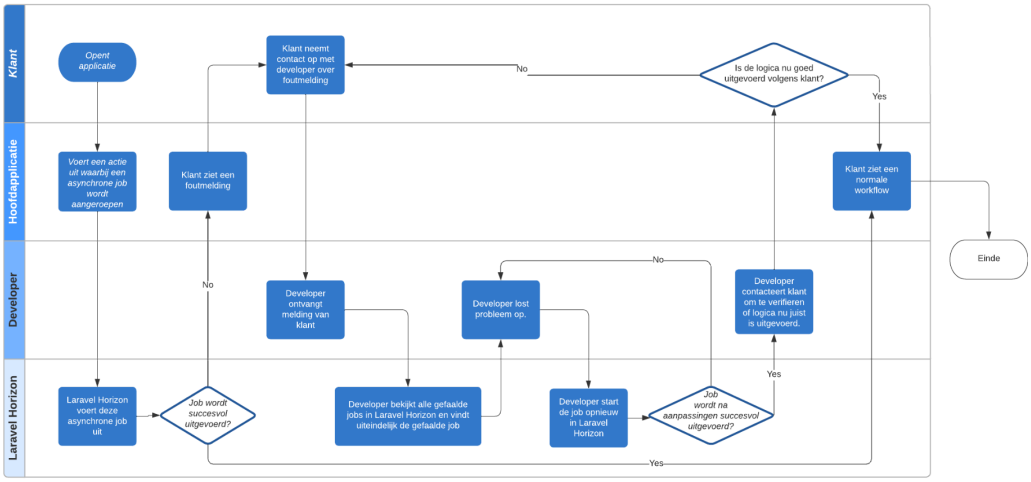


Toelichting Om een duidelijker beeld te krijgen van de functionaliteit in de Orbit package heb ik een activity diagram gemaakt. Hierin zie je de core functionaliteiten terugkomen en de stappen om daar te komen.

Waar is Orbit gepositioneerd binnen de workflow van Scrumble?

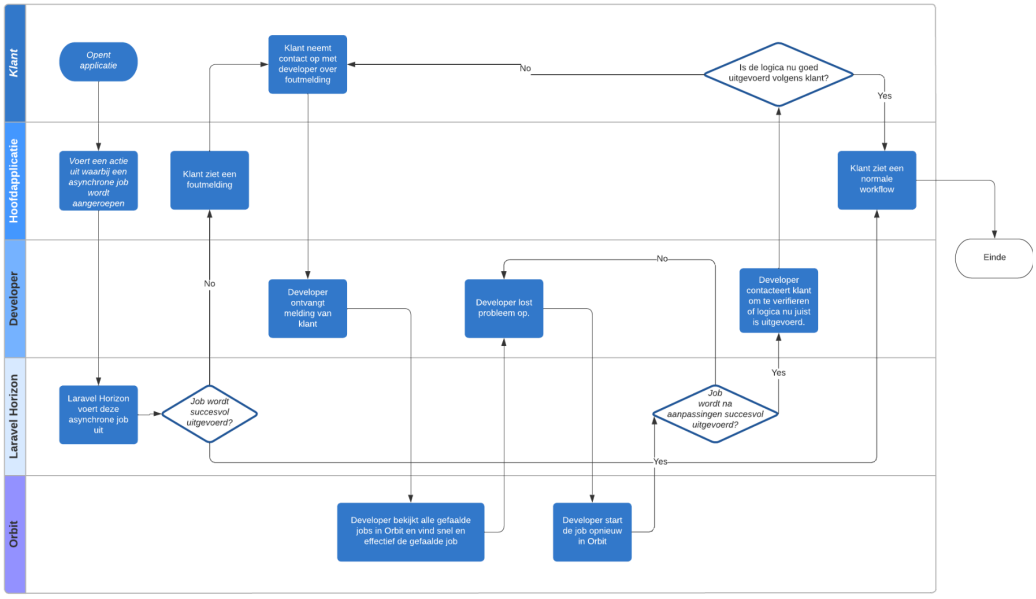
Flowchart werkwijze scrumble nu zonder Orbit

Hieronder zie je een flowchart van de situatie zoals hij op dit moment is.



Flowchart werkwijze scrumble met Orbit

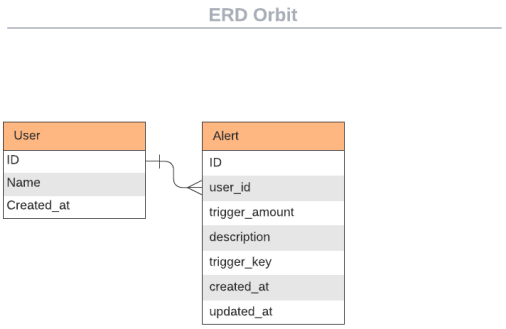
Hieronder zie je een flowchart van de situatie zoals hij zou moeten zijn als Orbit geïmplementeerd is. Zoals je ziet is er qua workflow niets veranderd. Alleen de tooling verandert, echter zit er in snelheid van het oplossen een verbetering door Orbit te implementeren omdat het doel hiervan is de filtering en gebruiksvriendelijkheid verbeteren.



Wat is de databasestructuur van Orbit?

ERD Orbit

Orbit zal zelf geen database hebben omdat Orbit gebruik gaat maken van de database van de applicatie waarin hij geïmplementeerd wordt. Het enige wat Orbit nodig heeft is een tabel voor het opslaan van de Alarmgegevens. Deze tabel zal er als volgt uitzien:



Zoals je ziet is er alleen een connectie met de ID van de gebruiker uit de hoofd applicatie.

Een kleine omschrijving van de belangrijke items in Alert:

- Trigger Key: een string waarmee gecontroleerd wordt, zit deze in de redis key dan wordt deze bijgehouden in een count.
- Trigger Amount: Als de count van de key boven dit getal komt wordt het alarm getriggerd.
- Description: Een beschrijving van het alarm.

Met gebruik van welke software architectuur is Orbit gebouwd?

CSR-Structuur

Binnen Scrumble maakt men gebruik van de CSR-structuur, dit staat voor Controller-Service-Repository structuur.
De "Controller-Service-Repository" software architectuur is een veelgebruikt ontwerppatroon in softwareontwikkeling. Het biedt een gestructureerde manier om de logica en de gegevens van een applicatie te organiseren. Het patroon bestaat uit drie belangrijke componenten: de controller, de service en de repository.

Controller:
De controller is verantwoordelijk voor het ontvangen van verzoeken van de gebruikersinterface of externe systemen. Het fungeert als een tussenpersoon tussen de gebruikersinterface en de onderliggende logica van de applicatie. De controller verwerkt de ontvangen verzoeken en roept de juiste services aan om de benodigde logica uit te voeren.

Service:

De service bevat de kernlogica van de applicatie. Het biedt functionaliteit en verwerkt de complexe bedrijfsregels. De service laadt de benodigde gegevens vanuit de repository en voert de vereiste bewerkingen uit. Het kan ook gegevens valideren, berekeningen uitvoeren of externe services aanroepen. Het doel van de service is om de controller te ondersteunen bij het uitvoeren van de gewenste acties.

Repository:
De repository is verantwoordelijk voor het opslaan en ophalen van gegevens. Het fungeert als een abstractielaag tussen de service en de gegevensopslag, zoals een database of externe API. De repository biedt methoden om gegevens te lezen, te schrijven en te wijzigen. Hierdoor kan de service zich concentreren op de logica, terwijl de repository de details van de gegevensopslag afhandelt.

Interfaceservice en interfacerepository zijn concepten die binnen deze architectuur worden gebruikt om de flexibiliteit en modulariteit te vergroten.

Interfaceservice:
Een interfaceservice definieert een contract of een set van methoden waarmee de service kan communiceren met andere componenten in het systeem, zoals controllers of andere services. Het biedt een gestandaardiseerde manier om functies aan te roepen en gegevens uit te wisselen. Door een interfaceservice te gebruiken, kunnen componenten gemakkelijk met elkaar communiceren zonder afhankelijk te zijn van de specifieke implementatie van de service.

Interfacerepository:

Een interfacerepository speelt een vergelijkbare rol als de interfaceservice, maar dan voor de repository. Het definieert methoden waarmee de service gegevens kan opvragen en manipuleren zonder zich bewust te zijn van de specifieke details van de gegevensopslag. Door een interfacerepository te gebruiken, kan de service eenvoudig schakelen tussen verschillende implementaties van de repository, zoals een SQL-database of een NoSQL-database, zonder dat de logica van de service hoeft te veranderen.

Kort samengevat biedt de "Controller-Service-Repository" softwarearchitectuur een gestructureerde manier om de logica en de gegevens van een applicatie te organiseren. De controller ontvangt verzoeken, de service voert de logica uit en de repository handelt de gegevensopslag af. Interfaceservice en interfacerepository worden gebruikt om de communicatie tussen componenten te standaardiseren en de flexibiliteit van het systeem te vergroten.