

AFSTUDEER DOSSIER

Afstudeeropdracht ANT

Hogeschool	Saxion hogeschool Deventer
Opleiding	HBO-ICT
Versie	1.1
afstudeerbegeleider	Danny Plass – Oude Bos
Bedrijfsbegeleider	Jan-Willem Kattouw
Datum	12-6-2021



VERSIEBEHEER

VERSIE	AUTEUR	DATUM	STATUS	OPMERKING
0.1	Lars Meulenbroek	9-3-2021	Concept	Opzetten Sjabloon en hoofdstuk indeling
0.2	Lars Meulenbroek	24-4-2021	Concept	Functioneel en technisch ontwerp
0.3	Lars Meulenbroek	7-5-2021	Concept	Verbeteringen
0.4	Lars Meulenbroek	20-5-2021	Concept	Functioneel en Technisch ontwerp conceptuele afronding
0.5	Lars Meulenbroek	25-5-2021	Concept	Feedback bedrijfsbegeleider & afstudeerbegeleider
0.6	Lars Meulenbroek	27-5-2021	Concept	Testen, Conclusie, Aanbevelingen
0.7	Lars Meulenbroek	28-5-2021	Concept	Verbeteringen, reflectie, samenvatting
1.0	Lars Meulenbroek	29-5-2021	Concept	Spelling controle
1.1	Lars Meulenbroek	12-6-2021	Definitief	Feedback concept verwerkt

SAMENVATTING

“Het is tijd om te automatiseren”. Elk bedrijf heeft deze zin wel eens gehoord. Het automatiseren is op dit moment een van de uitdagingen waar ieder bedrijf tegenaan loopt. Zo ook in de civiele techniek. Het ontwerpen van een brug of gebouw is nou eenmaal geen simpel proces. Er moeten onder andere eisen worden vastgelegd, er moeten 2D en 3D modellen worden gemaakt en berekeningen voor bijvoorbeeld betonwapening. Dit allemaal krijgt goedkeuring zodat een aannemer de bouw van de brug kan gaan verwerkelijken.

ANT is een platform wat is ontstaan in een samenwerking tussen CollaborAll en Witteveen en Bos. ANT wordt gebruikt voor het parametrisch ontwerpen. Het idee achter ANT is om 2D en 3D modellen op te delen in objecten en deze objecten te koppelen aan parameters waar berekeningen mee kunnen worden gestart. Het proces van parametrisch ontwerpen wil ANT automatiseren. Dit noemen zij workflows.

Wat zijn workflows? Om antwoord te geven op deze vraag zijn er met de verschillende stakeholders zowel individueel als gezamenlijk sessies gehouden om de workflow functionaliteit uit te werken in requirements, modellen en diagrammen.

De workflow functionaliteit is op te delen in 3 onderdelen:

1. Het bouwen van een workflow
2. Het configureren van een workflow
3. Het uitvoeren van een workflow

ANT wil gebruikers de mogelijkheid geven om hun eigen workflows te maken. Dit kunnen zij doen door stappen te configureren en aan elkaar te knopen, zodat ze sequentieel en/of parallel uitgevoerd kunnen worden. Met de sbrcode wordt de data die ontstaat vanuit de workflows gekoppeld aan objecten binnen het model.

De workflows geven gebruikers van ANT de vrijheid om innovatief en creatief ontwerpprocessen te automatiseren. Dit kan gaan om kleine berekeningen of een hele kostencalculatie. Met de workflows kunnen gebruikers van ANT alles zelf configureren naar hun eigen wens.

Om de workflows goed te kunnen uitwerken zijn er meerdere sessies gehouden met de verschillende stakeholders in de eerste weken van de afstudeerperiode. Uit deze sessies zijn onder andere de requirements, modellen en diagrammen ontstaan om de workflow functionaliteit in kaart te brengen.

Tijdens de ontwikkeling zijn er vele contact momenten geweest om de functionaliteiten te presenteren. Dit leidde tot het toevoegen en/of aanpassen van functies naar de wens van de stakeholders. Een van de grotere uitdagingen was de agnostische en herbruikbare functionaliteit van de workflow. Dit houdt in dat een workflow op zichzelf moet kunnen doorlopen worden, maar ook als onderdeel gebruikt moet kunnen worden van een andere workflow.

Met de twee wekelijkse releases van de nieuwe functionaliteit konden gebruikers snel feedback geven. De functionaliteiten werden eerst lokaal afgetest, vervolgens op de alpha en beta omgeving van het ANT platform. Vanaf de beta omgeving werden de functionaliteiten op de productie omgeving gezet.

Het koppelen van data binnen de sessie aan objecten binnen het model via de workflows zorgt ervoor dat ANT gebruikers hun ontwerp processen kunnen verbeteren en automatiseren.

1 INHOUDSOPGAVE

2	Inleiding	6
3	Afstudeeropdracht.....	7
4	Aanpak	9
5	Requirements Analyse	10
5.1	Huidige situatie	10
5.2	Stakeholders	10
5.2.1	Stakeholder analyse	11
5.3	Requirements.....	12
6	Functioneel Ontwerp.....	15
6.1	Workflow concept.....	15
6.2	Use case diagram	17
6.3	Activity diagram	18
6.4	Design.....	21
6.4.1	Mockups.....	22
6.4.2	Gebruiker feedback.....	26
6.5	Autodesk Forge	27
7	Technisch ontwerp	29
7.1	stack	29
7.2	Architectuur model	30
7.2.1	Workflow builder	30
7.2.2	Workflow configureren.....	30
7.2.3	Workflow uitvoeren	32
7.3	Database diagrammen	33
7.3.1	Het bouwen van een workflow	34
7.3.2	Project specifieke tabellen	35
7.3.3	SBS.....	36
7.4	Service repository pattern	37
7.5	Api documentatie.....	39
7.6	Workflow import.....	41
7.7	Flow van een workflow	41

7.8	doorlopen van een workflow	42
8	Testen	45
9	Conclusie.....	48
10	Discussie	49
11	Aanbevelingen	51
12	Bibliografie	52
13	Bijlages	54
13.1	Bijlage 1, Api calls.....	54
13.2	Bijlage 2, workflow import JSON.....	61
13.3	Bijlage 3, Workflow import response	62
13.4	Bijlage 4, Maak jobs voor elke node met blok type	64
13.5	Bijlage 5, Afsluiten en openen van een job.....	65
13.6	Bijlage 6, unit tests voor het valideren van een body.....	68
13.7	Bijlage 7, Service Repository Pattern	69

2 INLEIDING

Een gebouw, een brug of een viaduct. Al deze vormen van bouw hebben een enorm proces aan de achterkant. Van ontwerpen, berekeningen, benodigdheden en nog veel meer. Om dit gehele proces goed te laten verlopen worden er verschillende programma's gebruikt voor onder andere projectmanagement, data-analyse, ontwerptekeningen en constructieve berekeningen.

Een van de grote problemen is dat bij een project binnen de civiele techniek¹ verschillende bedrijven samenwerken en hiermee vaak hun intellectueel eigendom moeten delen met andere partijen. Daarnaast zijn deze projecten zo ongestructureerd dat bij de start van een nieuw project alles opnieuw wordt uitgewerkt. Denk hierbij aan protocollen voor 3D ontwerpen, structuur in het proces en vele andere methodieken.

In opdracht van CollaborAll Service B.V. werd in de periode van 8 februari 2021 tot 4 juli 2021 gezocht naar een oplossing om bedrijven gemakkelijker met elkaar te laten samenwerken zonder hun intellectueel eigendom te delen met de concurrent en een manier om delen van voorgaande projecten op een juiste wijze te kunnen hergebruiken.

In hoofdstuk 3 wordt het probleem geschetst wat resulteerde in de afstudeeropdracht. Vervolgens wordt er in hoofdstuk 4 een analyse gegeven op de stakeholders en vanuit de probleemstelling een lijst met requirements opgesteld. Hoofdstuk 5 gaat in op deze requirements en zal de gevraagde functionaliteit verder toelichten door middel van diagrammen en modellen. Verder worden hier de mockups weergegeven en wordt er ingegaan op het ontwikkelproces. Op basis hiervan wordt in hoofdstuk 6 de functionaliteiten van het ontwikkelde component toegelicht. Vervolgens wordt er in hoofdstuk 7 ingegaan op hoe de ontwikkelde functionaliteit getest is. Tot slot wordt in hoofdstuk 8 de probleemstelling beantwoordt en volgen in hoofdstuk 9 de aanbevelingen aan de opdrachtgever.

¹ "Civiele techniek houdt zich bezig het ontwerpen, bouwen, realiseren en onderhouden van alle infrastructuur. Een andere benaming is weg- en waterbouwkunde." (Voort, 2017)

3 AFSTUDEEROPDRACHT

Binnen de civiele techniek werken verschillende bedrijven bijna altijd samen. Het ontwerpen, berekeningen, data-analyse, project management van de bouw wordt bijna nooit door één partij gedaan. Een van de grote ingenieurs en adviesbureaus van Nederland is Witteveen en Bos. Zij hebben samen met CollaborAll het ANT platform opgericht.

CollaborAll is een netwerkorganisatie. Een netwerkorganisatie wordt beschreven als een niet-hiërarchische organisatie waarin teams in wisselende samenstelling werken aan complexe producten (Rutten, 2016). CollaborAll is opgericht in 2015 door Jan Willem Kattouw en Peter Kleinjan. CollaborAll bevindt zich in Almere. CollaborAll focust zich op het stroomlijnen en optimaliseren van processen door BIM, ERP, EAM data en systemen met elkaar te integreren. (CollaborAll, 2021)

ANT is een van de producten van CollaborAll en Witteveen en Bos. ANT is een platform voor parametrisch ontwerpen. Parametrisch ontwerpen is een proces waarbij op basis van data of relaties tussen onderdelen een ontwerp kan worden gegenereerd (parametrisch-ontwerpen-voor-dummies, 2018). Met ANT kan data ingevoerd en beheert worden. Daarnaast kunnen er binnen ANT ook relaties tussen deze data worden gelegd. Het parametrisch ontwerpen is de laatste jaren erg toegenomen in de bouw.

Binnen ANT kunnen verschillende bedrijven of divisies worden toegevoegd aan een project en samenwerken aan de data. ANT is dynamisch configureerbaar naar de eigen wensen van de eindgebruiker. Zo kan de gebruiker bijvoorbeeld data tabellen aanmaken en deze invullen met eigen invulformulieren. Deze data kan vervolgens gebruikt worden om bijvoorbeeld een berekening uit te voeren.

Deze stappen moeten nu allemaal één voor één binnen ANT geconfigureerd en uitgevoerd worden. Dit is niet alleen erg foutgevoelig, maar ook heel tijdrovend. Door de vele tabellen binnen een project raakt men snel het overzicht kwijt. Voor de gebruiker is het dan onduidelijk welke data bij welk model of object van een model hoort. Daarnaast is al deze data nu voor iedereen beschikbaar die aan het project is toegevoegd. Dit leidt tot het publiek worden van intellectueel eigendom tussen verschillende partijen die aan hetzelfde project werken.

Deze problemen wil ANT in één keer tackelen door een nieuwe hoofdfunctionaliteit aan de ANT applicatie toe te voegen. Met deze functionaliteit wil ANT gebruikers de mogelijkheid geven om een eigen netwerk van stappen op te zetten en te configureren. Dit netwerk kan vervolgens door verschillende partijen met verschillende rechten worden doorlopen. Het uitwisselen van data gaat op dit moment vaak via Excel bestanden. ANT wil de centrale plek zijn waar deze data wordt vastgelegd en gekoppeld aan het model. Met het rechten systeem van ANT zorgt dit ervoor dat het intellectueel eigendom bij de partij blijft die gebruik maakt van deze data, berekeningen of ontwerpen. Daarnaast moeten deze netwerken modulair zijn en hergebruikt kunnen worden in andere netwerken, zodat een stap voor bijvoorbeeld het berekenen van de betonwapening gemakkelijk hergebruikt kan worden over verschillende projecten zonder dat dit helemaal opnieuw geconfigureerd moet worden.

Dit netwerk van stappen wordt door ANT workflows genoemd. Met de afstudeeropdracht wil ANT een basis-opzet realiseren waar eindgebruikers een workflow kunnen opzetten, configureren en kunnen inzetten op hun projecten waar de gebruikers stapsgewijs hun berekeningen, ontwerpen en controles kunnen doorlopen om zo sneller tot het juiste ontwerp te komen.

De kern functionaliteiten van het component is op te delen in 3 onderdelen:

1. Het bouwen van een workflow
2. Het configureren van een workflow
3. Het uitvoeren van een workflow

De ANT API wordt naast de UI veel benaderd door andere platformen of berekeningsservers. Hierdoor ligt de prioriteit om de 3 kernonderdelen eerst volledig uit te werken als REST API. Het bouwen, configureren, en uitvoeren moet benaderbaar zijn via verschillende API calls. Vanuit de stakeholders ligt de prioriteit voor het realiseren van de UI op de laatste twee onderdelen. Voor hen is het waardevoller om het configureren en uitvoeren in de UI te realiseren, omdat hier minder technische mensen mee gaan werken. Waarbij het bouwen, voor nu, gebeurt door techneuten.

Het ontwikkelen van de workflows heeft als doel om de drie kern functionaliteiten volledig uitgewerkt en getest te hebben, zodat deze benaderbaar is via de REST API. Daarnaast zal de focus in de UI gelegd worden op de laatste 2 kernfunctionaliteiten, zodat project leiders de workflows kunnen configureren en het team de workflows kan uitvoeren vanuit een gebruikersvriendelijke omgeving. Het bouwen van de workflow kan in een later stadia binnen de UI worden gerealiseerd wanneer hier tijd voor over is, maar dit heeft niet de prioriteit.

In het volgende hoofdstuk worden deze kernfunctionaliteiten onderzocht door middel van het op stellen van de requirements aan de hand van de stakeholder analyse.

4 AANPAK

Om de workflow functionaliteit goed te kunnen realiseren moesten eerst alle eisen van de verschillende stakeholders in kaart worden gebracht. Dit werd gedaan door met de verschillende stakeholders in gesprek te gaan in de eerste weken van de afstudeerperiode. Door de gevraagde complexiteit kostte dit veel tijd. Door middel van diagrammen, modellen en de requirements terug te leggen aan de stakeholders werden alle functionaliteiten in kaart gebracht.

De nieuwe functionaliteit maakte gebruik van een aantal aanknooppunten binnen het huidige ANT platform. Hiervoor werd een analyse gemaakt van de al bestaande software en welke delen gebruikt konden worden voor de nieuwe functionaliteiten.

De requirements werden geprioriteerd en waren opgesplitst in de drie workflow kernfunctionaliteiten:

1. Het bouwen van een workflow
2. Het configureren van een workflow
3. Het uitvoeren van een workflow

De requirements werden omgeschreven naar stories in de vorm van Gitlab tickets. Het ANT team gebruikt milestones om de twee wekelijkse releases bij te houden. Elke twee weken werd er een grove planning gemaakt welke grotere functionaliteit onderdelen werden opgepakt. Het ANT development team heeft wekelijkse contact momenten met de eindgebruikers en stakeholders, waardoor er niet te ver vooruit kon worden gepland, omdat er een directe feedback loop ontstond waardoor nieuw gevraagde functionaliteiten konden worden gerealiseerd.

De milestones voor de workflow functionaliteit werden grof gepland en aan de start van elke week werden de stories op het bord vastgezet welke functionaliteiten er die week werden afgerond. Dit werd vastgesteld tijdens de demo aan de stakeholders en eindgebruikers van de gerealiseerde functionaliteiten. De demo resulteerde altijd in de planning voor de aankomende week.

Deze manier van flexibel werken zorgden ervoor dat de gebruikers (klanten) ook sneller tevreden werden gehouden door nieuwe gevraagde functionaliteit binnen een week te kunnen realiseren. Het ANT development team werkte niet met een vaste SCRUM methodiek om met name de overhead over het project zo laag mogelijk te houden. Weliswaar kijkt het af van SCRUM door kort vooruit te plannen met twee wekelijkse releases en demo's en daarnaast te reflecteren, maar dan op een wat luchtere manier zonder hier specifiek vaste momenten voor in te plannen.

5 REQUIREMENTS ANALYSE

In dit hoofdstuk worden de stakeholders en hun belang bij het component in kaart gebracht. Daarnaast worden de requirements van het component opgesteld en geprioriteerd aan de hand van de stakeholder analyse.

5.1 HUIDIGE SITUATIE

ANT is een platform voor het vastleggen van relaties tussen data. Het platform beheert parameters en legt daarbij onderlinge relaties. Het leggen van deze relaties wordt nu allemaal handmatig gedaan in tabellen die gebruikers kunnen invullen binnen hun project. Deze relaties en data vormen de basis voor het parametrisch ontwerpen. De volgende stap voor ANT is het leggen van deze relaties in een configureerbaar stappenplan genaamd workflows.

5.2 STAKEHOLDERS

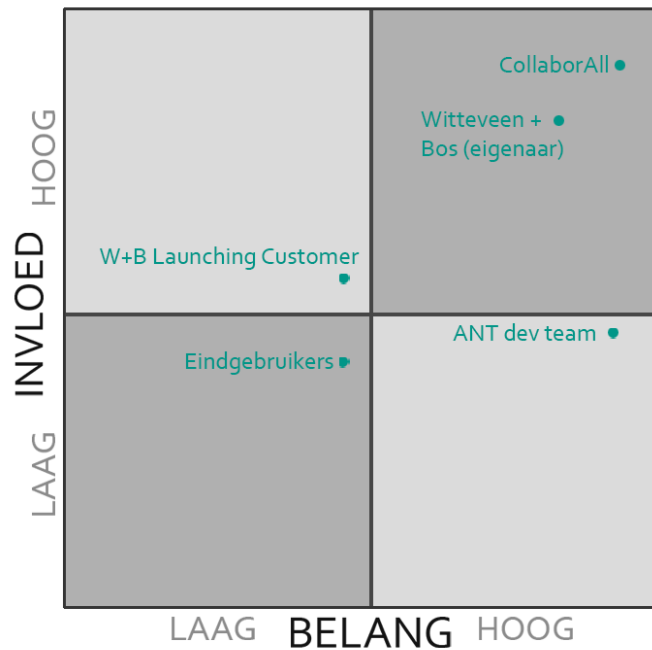
De stakeholders zijn opgedeeld in 3 categorieën. CollaborAll is het bedrijf waar het project wordt uitgevoerd. Witteveen en Bos is de samenwerking partner van CollaborAll voor het ANT platform, daarnaast gebruiken zij zelf ook het platform. De klanten zijn de eindgebruikers van het ANT platform.

Group	Stakeholders
CollaborAll	Eigenaar ANT development team
Witteveen +Bos	Eigenaar Launching customer
Klanten	Eindgebruikers

Witteveen en Bos bestaat uit veel verschillende disciplines: architecten, (geotechnische)-constructeurs, project leiders, ontwerpers. Al deze disciplines gebruiken ANT. Tevens zijn deze disciplines ook klanten. Onder klanten kan verder worden verstaan aan samenwerkende bedrijven met bijvoorbeeld Witteveen en Bos en in de toekomst andere bedrijven die ANT gebruiken bij hun projecten binnen de civiele techniek.

5.2.1 STAKEHOLDER ANALYSE

De volgende grafiek laat zien hoeveel macht en interesse de stakeholders hebben binnen het workflow project. Hoe hoger ze in de grafiek staan hoe meer macht ze hebben en hoe meer naar rechts hoe meer interesse er is in het project.



Figuur 1 Stakeholder analyse

De twee belangrijkste stakeholders zijn CollaborAll en Witteveen en Bos als eigenaar van het ANT platform. Zij hebben beiden een hoog belang en grote invloed op het product. CollaborAll staat hier hoger in de grafiek omdat zij direct in contact staan met het development team. Het development team heeft een stuk minder invloed op het product. Omdat Witteveen en Bos ook een launching customer is voor het product, hebben zij meer invloed dan de overige eindgebruikers van het ANT platform. Daarom staan zij gescheiden in de grafiek.

5.3 REQUIREMENTS

Tijdens de eerste weken van het project zijn de requirements uitgewerkt door vele meetings met onder andere Jan Willem Kattouw (CollaborAll), Gert Karrenbelt (W+B) en Maarten Visser (W+B). Tijdens dit proces zijn de requirements opgesteld en teruggelegd aan de stakeholders (zie figuur 1) en in overleg tot een consensus gekomen over de prioriteit van de requirements. Een van de grote uitdagingen tijdens het opstellen van de requirements is dat in de eerste weken het nog niet duidelijk was wat de verschillende partijen wilden van de workflow functionaliteit. Deze onduidelijkheid is verhelderd in de eerste weken door vele individuele en gezamenlijke gesprekken aan te gaan met zowel CollaborAll als Witteveen en Bos.

Door onder andere use case-, activity diagrammen (zie functioneel ontwerp, pagina 13) voor te leggen en de lijst met requirements terug te koppelen naar zowel CollaborAll als Witteveen en Bos zijn de requirements vastgesteld voor het eerste prototype van de workflows. CollaborAll had meer invloed op de functionaliteiten met een toekomstgerichte visie waarbij Witteveen en Bos (als klant) meer focus legde op het zo snel mogelijk afmaken van een MVP², zodat zij dit konden gebruiken als test voor een aantal projecten. Dit zorgde ervoor dat de requirements beide klanten tevreden moesten houden zodat het niet alleen haalbaar was op korte termijn, maar ook schaalbaar in de toekomst.

De requirements zijn opgesteld in dezelfde 3 delen van de workflow: bouwen, configureren en uitvoeren. Het uitvoeren en configureren van een workflow kunnen alleen worden gerealiseerd wanneer het bouwen van een workflow is voltooid. Het configureren van een workflow is dan ook een voorwaarde voor de uitvoer functionaliteit. Deze drie delen zijn dus opeenvolgend na elkaar en zullen ook op deze manier in hun prioritering worden gerealiseerd. Waarbij de must requirements opeenvolgend van bijvoorbeeld het bouwen de voorwaarde zijn voor de must requirements van het configureren en uitvoeren.

De requirements zijn als volgt:

Bouwen van een workflow	
MUST	Als een gebruiker wil ik mijn eigen workflows kunnen bekijken en beheren
MUST	Als een gebruiker wil ik een workflow blok kunnen bekijken, beheren en configureren
MUST	Als een gebruiker wil ik workflows aan elkaar kunnen knopen
MUST	Als een gebruiker wil ik workflow blokken aan elkaar kunnen knopen
MUST	Als een systeem wil ik alle project specifieke workflow tabellen aanmaken tijdens het aanmaken van een project
MUST	Als een gebruiker wil ik configuratie bestanden, in JSON, kunnen uploaden en toevoegen aan een workflow blok
SHOULD	Als een gebruiker wil ik een workflow kunnen importeren

² Minimum Viable Product

SHOULD	Als een gebruiker wil ik een media bestand kunnen uploaden en toevoegen aan een workflow blok configuratie
SHOULD	Als een gebruiker wil ik een Autodesk Forge configuratie toevoegen aan een workflow blok configuratie
SHOULD	Als een gebruiker wil ik ANT tabellen kunnen toevoegen aan een workflow blok configuratie

Configureren van een workflow	
MUST	Als een licentie houder wil ik een workflow binnen mijn licentie kunnen publiceren en beheren
MUST	Als een project admin wil ik een workflow binnen mijn licentie kunnen toevoegen aan mijn project
MUST	Als een project admin wil ik mijn project specifieke workflows kunnen bekijken en beheren
MUST	Als een project admin wil ik voor een workflow rollen kunnen bekijken en beheren
MUST	Als een project admin wil ik de relatie tussen workflow rollen en project teams kunnen bekijken en beheren
MUST	Als een project admin wil ik de relatie tussen workflow rollen en workflow nodes kunnen bekijken en beheren
MUST	Als een project admin wil ik sbs objecten kunnen bekijken en beheren
MUST	Als een project admin wil ik een sbscode ³ toekennen aan een workflow sessie
SHOULD	Als een licentie houder wil ik mijn workflow collecties kunnen bekijken en beheren
SHOULD	Als een licentie houder wil ik een workflow kunnen toevoegen aan een licentie binnen een collectie
COULD	Als een project admin wil ik een overzicht hebben van alle rollen, teams en blokken
WOULD	Als een project admin wil ik een versie bijhouden van mijn sbs objecten

³ Een sbscode is een interne code die aan een object wordt gekoppeld binnen het ontwerp. Ook wel te zien als een id van een object, maar dan binnen een system breakdown structure (SBS). Zie hoofdstuk SBS op pagina 29

Uitvoeren van een workflow	
MUST	Als een project admin wil ik een sessie kunnen starten
MUST	Als een project admin wil ik alle workflow sessies kunnen bekijken en beheren
MUST	Als een project admin wil ik gebruikers kunnen toekennen aan een bepaalde stap binnen een workflow
MUST	Als een gebruiker wil ik andere gebruikers binnen mijn team kunnen toekennen aan een bepaalde stap binnen een workflow
MUST	Als een gebruiker wil ik al mijn openstaande stappen zien binnen een workflow
MUST	Als een gebruiker wil ik parameters kunnen invullen binnen een workflow stap
SHOULD	Als een systeem wil ik workflow stappen koppelen aan ANT taken
SHOULD	Als een gebruiker wil ik mijn stappen kunnen filteren op sessie en status
WOULD	Als een gebruiker wil ik zelf kunnen indelen welke delen ik zie van een workflow stap

Een van de hoofdcomponenten, van het ANT platform, is de Autodesk Forge viewer. Deze viewer wordt in ANT gebruikt om 3D modellen te tonen aan de gebruiker. Uit de gesprekken met Witteveen en Bos kwam na de workflows functionaliteit naar voren dat zij de Forge viewer niet alleen een design update wilden geven, maar ook willen integreren met de workflows. De implementatie van Autodesk Forge binnen de workflows geldt als prototype voor het integreren, van software van derden, in de workflows. De volgende requirements zijn uit gesprekken met onder andere Maarten Visser (W + B), Gert Karrenbelt (W + B) en Marc Taken (W + B) naar voren gekomen:

Autodesk Forge	
MUST	Als een gebruiker wil ik alle objecten van een model kunnen weergeven
MUST	Als een gebruiker wil ik de eigenschappen van een object kunnen weergeven
MUST	Als een gebruiker wil ik data van een object kunnen updaten
MUST	Als gebruiker wil ik een overzicht kunnen genereren van alle eigenschappen van het model
SHOULD	Als een gebruiker wil ik een geselecteerd sbs object kunnen highlighten in de viewer
SHOULD	Als een gebruiker wil ik een geselecteerd model object kunnen highlighten in de viewer
SHOULD	Als gebruiker wil ik de Viewer componenten kunnen aan en uit schakelen

Non Functional	
MUST	De workflow functionaliteiten worden in dezelfde stack als het huidige ANT systeem gebouwd.

6 FUNCTIONEEL ONTWERP

In het functioneel ontwerp worden door middel van diagrammen en mockups de requirements nader toegelicht. Daarnaast wordt er dieper inzicht gegeven in de functionaliteiten die de workflows zullen bevatten. Door middel van mock-ups en wireframes wordt het design vastgesteld en geeft het een beeld hoe de functionaliteit naar de gebruiker zal worden vertaald.

6.1 WORKFLOW CONCEPT

ANT wil gebruikers de vrijheid geven om zelf workflows te maken. Het maken van een workflow is vergelijkbaar met een sandbox game⁴. ANT wil de gebruikers met de workflow functionaliteit de vrijheid geven om hele processen voor berekeningen en ontwerpen in te regelen voor hun projecten. Dit zorgt ervoor dat de innovatie en effectiviteit vanuit de organisatie aangewakkerd wordt, omdat gebruikers met de tooling van ANT zelf hun processen kunnen verbeteren en automatiseren.

Een workflow resulteert zich in een netwerk van stappen die synchroon en parallel naast elkaar kunnen lopen. Een workflow lijkt op een flowchart waarbij gebruikers stappen kunnen doorlopen om tot een goed ontwerp te komen. Stappen van een workflow kunnen ook geautomatiseerd zijn. Zo kan bijvoorbeeld een berekeningsproces worden opgestart door een server. Gebruikers moeten zelf instaat zijn om deze workflows op te kunnen bouwen. Als een workflow is opgebouwd moet deze door een project admin kunnen worden geconfigureerd met een rechten-systeem wie wat, binnen het project en binnen de workflow, mag zien en doen. Vervolgens moet na de configuratie de workflow kunnen worden doorlopen.

De workflow functionaliteit is in drie stappen op te delen:

1. Opbouwen van een workflow
2. Configureren van een workflow
3. Uitvoeren van een workflow

Deze 3 stappen vormen samen de basis voor het geautomatiseerd parametrisch ontwerpen binnen het ANT platform. In een periode van 20 weken zijn deze 3 delen in een eerste fase uitgewerkt, getest en in productie gezet.

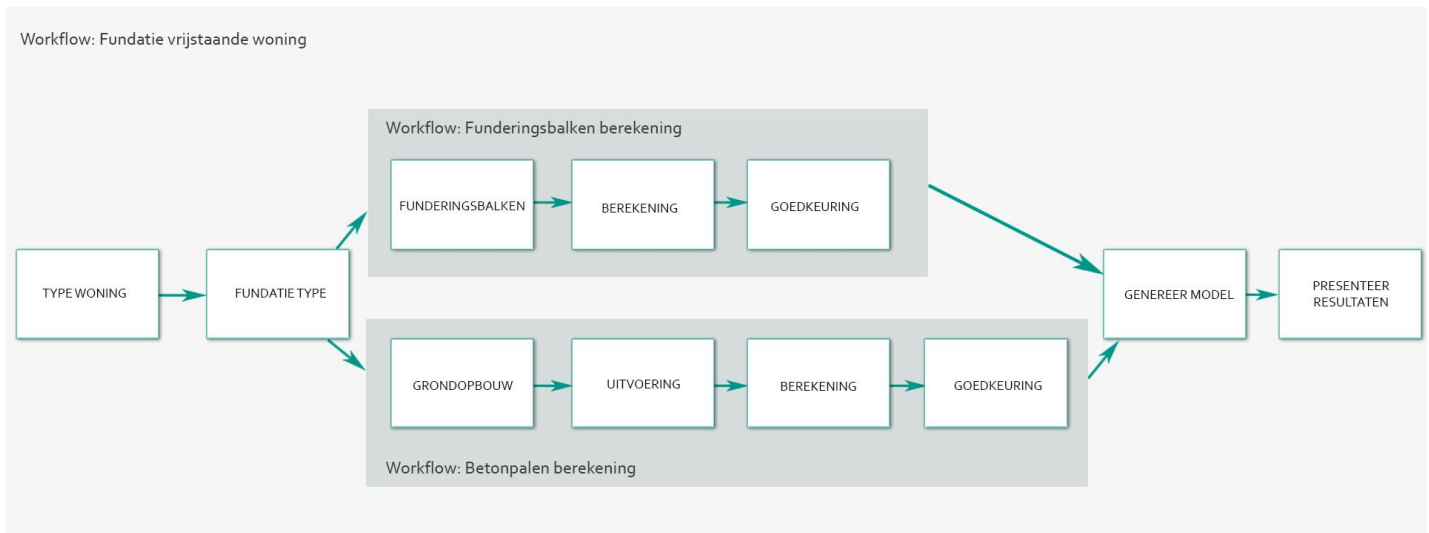
ANT wordt gebruikt door bedrijven die projecten uitvoeren binnen het ANT platform. Deze projecten hebben teamleden en projectleiders. De workflows kunnen straks worden ingezet op projecten. Een workflow is van zichzelf modulair en agnostisch. Een workflow die binnen een andere workflow valt weet niets af van deze andere workflow. Zo kan de zelfde workflow op verschillende ontwerpen of delen van hetzelfde ontwerp worden ingezet.

⁴ Een sandbox game geeft de gebruiker de vrijheid om zelf te bedenken wat hij/zij wilt doen met de middelen die de game aan hem of haar geeft.

Dit is vergelijkbaar met een franchise (het bedrijf). De franchise heeft verschillende pizzeria's (projecten). Elke pizzeria (project) heeft een aantal pizza's (workflows). Deze recepten zijn specifiek voor de pizzeria (project), maar kunnen gedeeld worden met andere pizzeria's (projecten) binnen de franchise (bedrijf). Het maken van een pizza (sessie ofwel doorlopen van de workflow) kan herhaaldelijk en tegelijk gedaan worden. Ook kunnen er meerdere pizza's tegelijk worden gemaakt. De pizza bestaat uit workflows op zichzelf. Het deeg maken, bereiden en de tijd in de oven kunnen worden gezien als workflows. Het recept is voor elke pizza (workflow) hetzelfde en kan dus binnen het project worden hergebruikt.

Een hoofd-workflow is een netwerk van kleinere workflows of individuele stappen. Deze kleinere workflows kunnen worden hergebruikt en ingezet op andere projecten. Een gebouw of brug kan bijvoorbeeld dezelfde betonwapening berekening (workflow) hergebruikt worden op meerdere plekken binnen hetzelfde project of ontwerp.

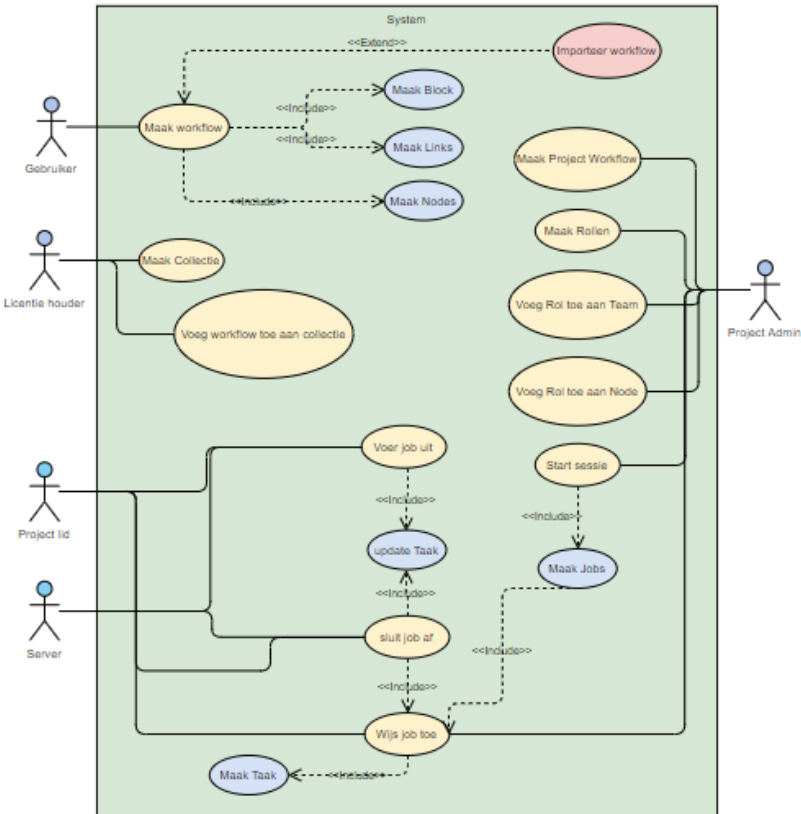
Hieronder volgt een voorbeeld van een workflow:



Hierboven is de workflow 'Fundatie vrijstaande woning' deze workflow heeft 2 kinder workflows die parallel uitgevoerd kunnen worden. Een van de voorwaarde van de workflow is dat een blok pas uitgevoerd als alle vorige blokken in de lijn succesvol zijn afgerond. De twee kinder workflows moeten op zichzelf kunnen werken en weten niet dat zij, in dit voorbeeld, in een andere workflow zitten (agnostisch). Deze workflows kunnen daardoor ingezet worden binnen andere workflows of op zichzelf kunnen worden doorlopen. Dit maakt hen herbruikbaar.

6.2 USE CASE DIAGRAM

Het volgende model is gemaakt tijdens de eerste weken als terugkoppeling tussen de gesprekken met onder andere Gert Karrenbelt (W+B) en Jan Willem Kattouw (CollaborAll). Het model is ontstaan uit de gesprekken over het bouwen, configureren en uitvoeren van een workflow. Nadat deze drie delen waren vastgesteld is de relatie tussen de acties besproken en in het model weergegeven.



Figuur 2 workflow use case diagram

Wanneer een licentiehouders een workflow heeft toegevoegd aan zijn licentie kan een project admin deze workflow in zijn project toevoegen. Deze kan hij configureren door rollen aan te maken en deze toekennen aan teams en nodes. Wanneer de gehele workflow is geconfigureerd kan hij een sessie starten. De sessie is het doorlopen van de stappen binnen de workflow door middel van jobs. Elke uitvoerbare node in een workflow vormt zich tot een job. Een job is toe te wijzen aan een gebruiker in het team met dezelfde rol als de node. Wanneer een job wordt toegewezen wordt er een taak voor de gebruiker of server aangemaakt om de job uit te voeren. De taak houdt de status van de job bij. Wanneer een taak is afgerond wordt de job gesloten en de volgende in de reeks geopend.

ANT wil elke gebruiker de mogelijkheid geven om zijn eigen workflow te maken. Een blok bevat alle configuratie van een stap binnen de workflow.

Een node zijn de werkelijke stappen van een workflow. Alle nodes vormen in combinatie met de links de flow van de workflow. Omdat een node zowel een blok als een andere workflow kan zijn kunnen workflows worden hergebruikt op meerdere plekken.

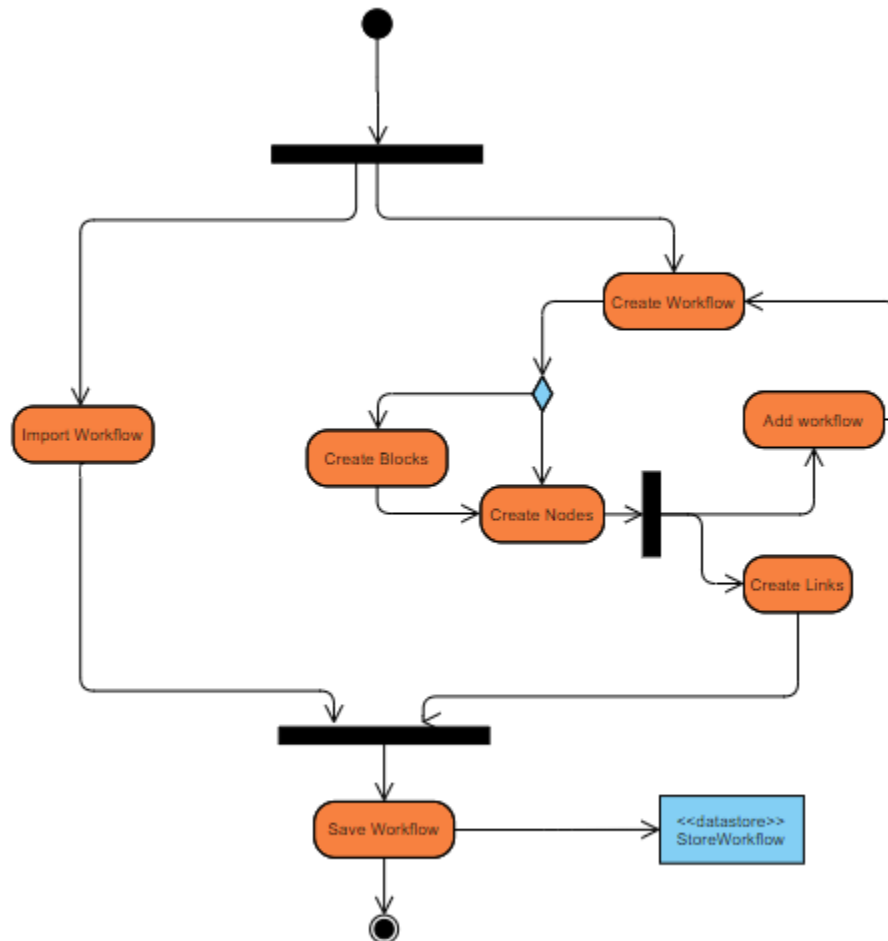
Een gebruiker kan dit geheel ook importeren in één keer.

De licentiehouders is de enige die een workflow kan publiceren binnen zijn licentie. Dit doet hij door middel van een workflow collectie die gekoppeld is aan zijn licentie.

6.3 ACTIVITY DIAGRAM

Nadat het bovengenoemde use case diagram (zie figuur 2) werd vastgesteld over de acties die de gebruiker moet maken voor de drie onderdelen van de workflow. Vanuit het development team was de vraag om de acties van de gebruiker gedetailleerder in kaart te brengen om zo de stappen van de gebruiker beter te begrijpen en relaties tot verschillende objecten in kaart te brengen.

Het volgende diagram focust zich op het bouwen van een workflow:

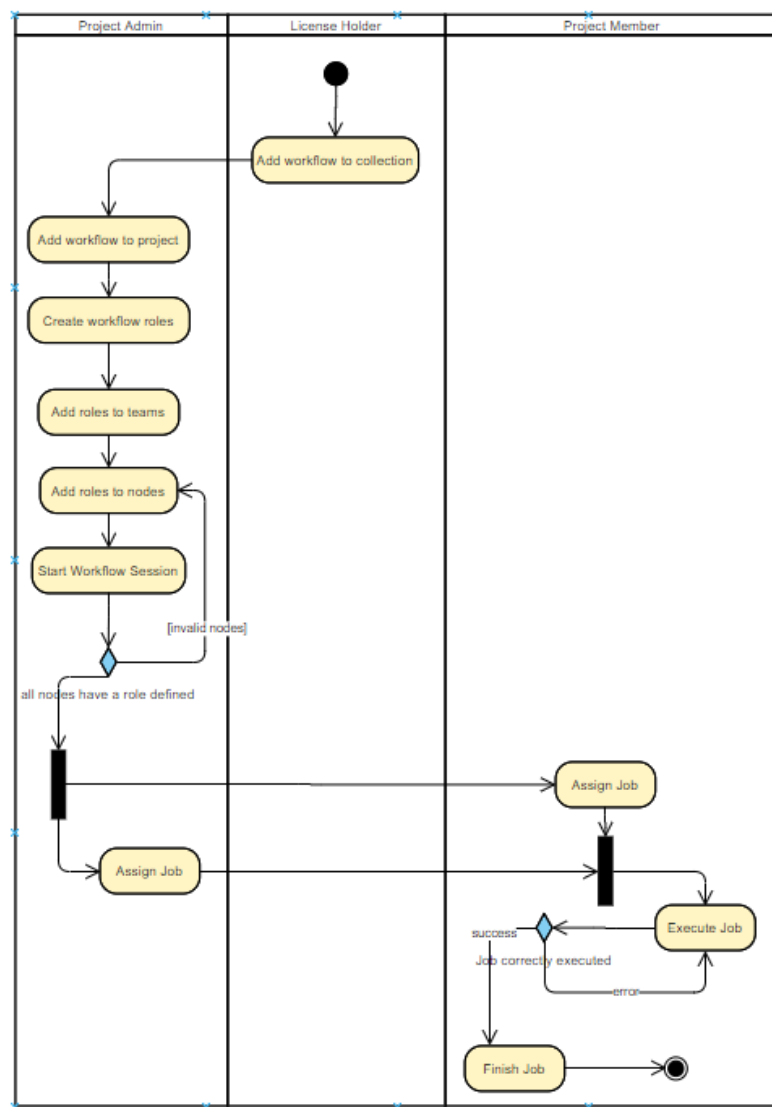


Elke gebruiker binnen de ANT omgeving mag een workflow maken. De gebruiker kan dit op verschillende manieren doen: (1) De gebruiker kan een gehele workflow importeren met een juist JSON format om zo een hele workflow met workflow kinderen aan te maken. (2) een gebruiker kan ook een workflow maken door zelf eerst een workflow aan te maken, vervolgens blokken binnen de workflow te creëren of nodes te maken en al bestaande workflows importeren. Wat een recursief proces is, zodat het modulaire en herbruikbare gedeelte van de workflow kan worden gerealiseerd. Wanneer alle nodes zijn gedefinieerd kan de gebruiker de nodes linken om de flow van de workflow te definiëren en vervolgens de workflow opslaan.

Tijdens de gesprekken met Gert Karrenbelt (W + B) kwam naar voren dat voor het configureren en uitvoeren van een workflow de verschillende gebruikers rollen en rechten binnen het ANT systeem van groot belang zijn. Om hier meer inzicht in te krijgen is met onder andere Jan Willem Kattouw (CollaborAll) het rechten systeem binnen ANT toegelicht en is vastgesteld welke rollen bepaalde acties mogen doen binnen het configureren en uitvoeren van een workflow.

Om deze rollen goed naar voren te laten komen is een activity diagram opgesteld met swimming lanes, waarbij elke rol binnen het ANT systeem een lane is binnen het diagram. Dit zorgde voor een duidelijk overzicht welke gebruiker welke acties mocht doen.

Hieronder volgt het diagram voor het configureren en uitvoeren van een workflow:



Wanneer een workflow door een gebruiker is aangemaakt kan deze gepubliceerd worden binnen een collectie. Een collectie wordt beheerd door de licentie houder en alleen hij kan een workflow toevoegen aan een collectie.

Een project admin kan vervolgens, wanneer hij binnen dezelfde licentie valt als de collectie, een workflow toevoegen aan zijn project. Wanneer een workflow is toegevoegd aan een project kan een project admin de workflow configureren door rollen te definiëren en deze toe te kennen aan de nodes van een workflow en de teams binnen een project. De rol van een node definieert welk team deze node mag uitvoeren binnen de gehele workflow.

Wanneer alle nodes een rol hebben kan de project admin een workflow sessie starten. Bij het starten van een sessie worden jobs aangemaakt voor alle uitvoerbare nodes binnen de workflow. Vervolgens kan de project admin de eerste job (node) toewijzen aan een persoon binnen het team met dezelfde toegekende rol van de node.

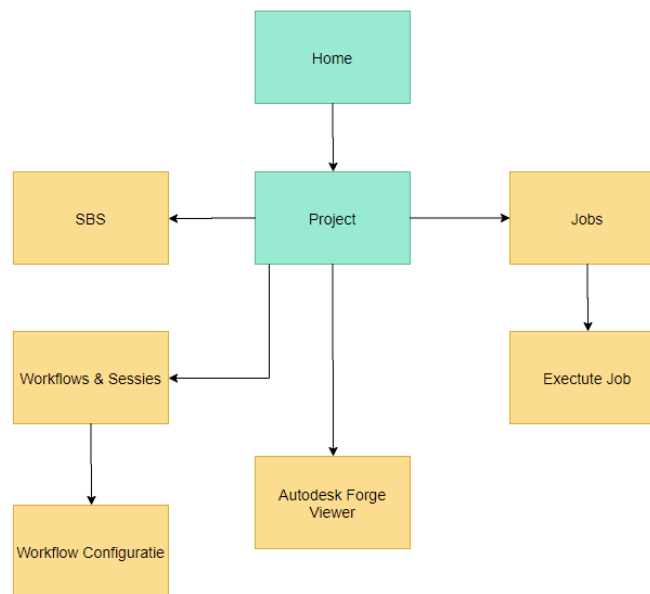
Een gebruiker met dezelfde rol als de node kan deze ook zelf oppakken of toewijzen aan een andere gebruiker binnen zijn team. De gebruiker kan de job uitvoeren en als de job slaagt wordt de job gesloten voor de gebruiker en kan de volgende job worden opgepakt.

Het laatste is het recursieve gedeelte wat de flow van een workflow definieert waar gebruikers opeenvolgend en parallel gezamenlijk door de workflow kunnen heenlopen.

6.4 DESIGN

Voor het maken van de Mockups is er gebruik gemaakt van het material design principe (Vuetify, 2021). Material design is over de huidige ANT applicatie gebruikt als design principe en om consistentie voor de eind gebruiker te behouden zal dit ook zo zijn voor de nieuwe componenten. Naast de keuze voor material design heeft ANT een vaste huisstijl gedefinieerd voor de gebruikte kleuren binnen de applicatie. Deze zullen worden aangehouden binnen de nieuw ontwikkelde pagina's.

Om een beter overzicht te geven van alle ontwikkelde pagina's volgt hier een overzicht hoe gebruikers navigeren binnen de applicatie:



Figuur 3 pagina flow

ANT wordt door veel projecten via de API aangesproken. Voor de eerste iteratie van de workflows zijn de bovengenoemde schermen (geel) toegevoegd. Deze schermen zorgen ervoor dat gebruikers workflows kunnen configureren en doorlopen via de UI. Het bouwen van de workflow zal voor nu via de API worden gedaan.

6.4.1 MOCKUPS

Voor het ontwerp van de workflow pagina's, taken en configuratie is er gebruik gemaakt van het design in Adobe XD van het ANT platform. Hierdoor konden er veel componenten hergebruikt worden wat niet alleen consistentie voor de eindgebruiker biedt, maar ook het design proces versnelde.

Hieronder volgt een overzicht van de mockups voor de pagina's die in figuur 3 zijn weergegeven:



De sidebar die zich aan de linkerkant binnen het ANT platform bevindt is de weg hoe gebruikers navigeren binnen de applicatie. Dit geeft de gebruiker een overzicht en controle over waar hij in de applicatie naar toe wil navigeren op één consistente plek binnen de applicatie.

In de bovenkant kan de gebruiker de gewenste content presenteren voor binnen in de sidebar.

In de onderkant bevinden zich de navigatie opties om tot de schermen te komen die in figuur 3 zijn weergegeven.

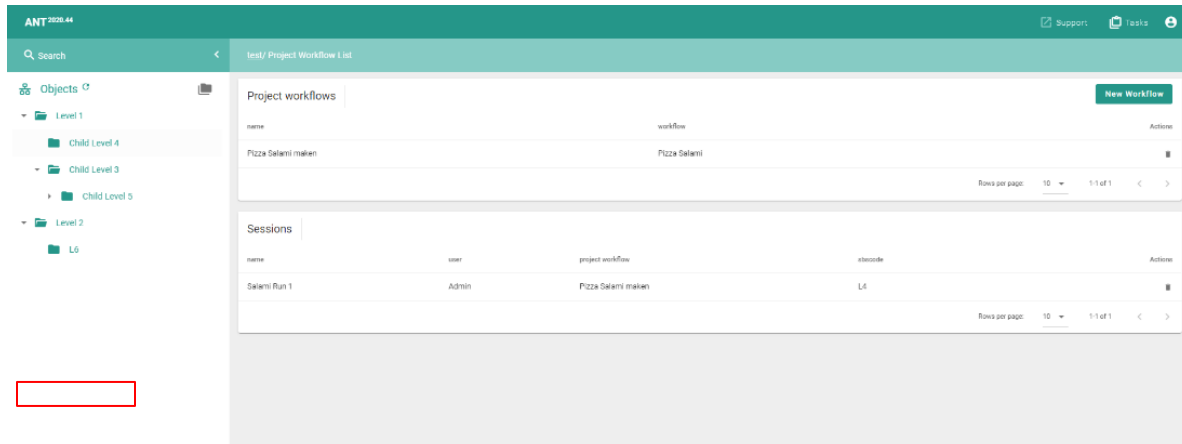
Deze items, in het menu, worden op basis van de rechten van de gebruiker weergegeven.

Zo is de workflows-navigatie net zoals de project-settings-navigatie alleen beschikbaar voor project-admins.

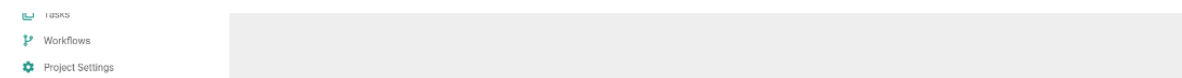
Om voor de gebruikers de navigatie binnen de ANT applicatie zo eenvoudig mogelijk te behouden is ervoor gekozen om de navigatie van de nieuwe functionaliteiten op deze plek te zetten. Zodat dit voor de gebruikers consistent blijft en er niet vanuit twee verschillende manieren genavigeerd kan worden. (Neil, 2009)

6.4.1.1 CONFIGUREREN VAN EEN WORKFLOW

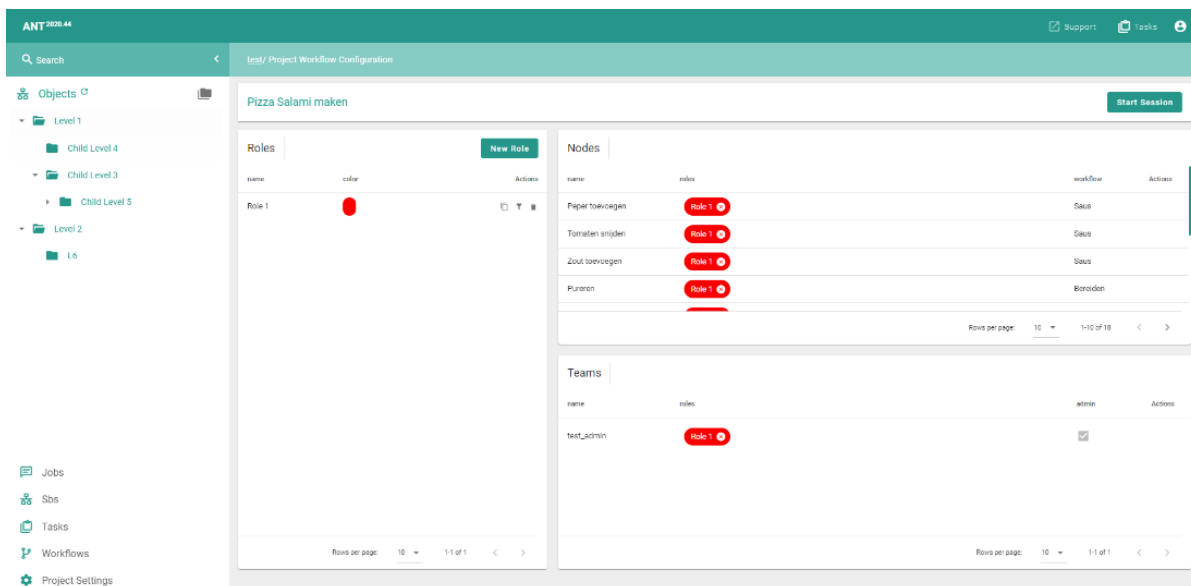
Hieronder volgende mockups voor het configureren van een workflow. Deze pagina's worden alleen beschikbaar voor de project admin.



Figuur 4 project workflows pagina



In de menubalk linksonder bevindt zich de knop naar het eerste configuratiescherm van een workflow. Hier kan een project-admin een workflow toekennen aan zijn of haar project wanneer deze binnen de licentie valt.

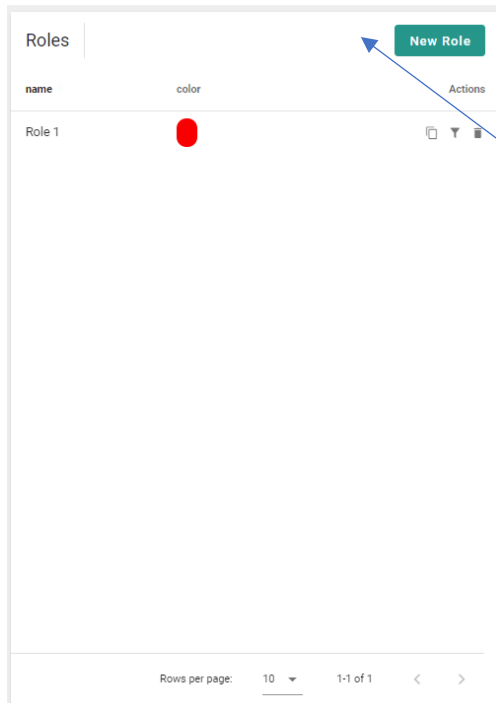


Figuur 5 workflow configuratie

Het configureren van een workflow splitst zich op in drie delen: de gedefinieerde rollen, de rollen koppeland aan de nodes van de workflow en de rollen koppeland aan de teams van het project.

Om het overzicht voor de eindgebruiker te behouden is dit allemaal zichtbaar binnen hetzelfde venster. Alle data wordt weergegeven in data tabel UI component. Om consistentie te behouden over de gehele applicatie werd er een vaste structuur aangehouden voor de tabel.

Deze vaste structuur is als volgt:



The screenshot shows a web interface for a table titled 'Roles'. The table has two columns: 'name' and 'color'. The first row contains the text 'Role 1' and a red circle. To the right of the table is an 'Actions' column with icons for adding, filtering, and deleting. Above the table is a 'New Role' button. At the bottom, there is a pagination control showing 'Rows per page: 10' and '1-1 of 1'.

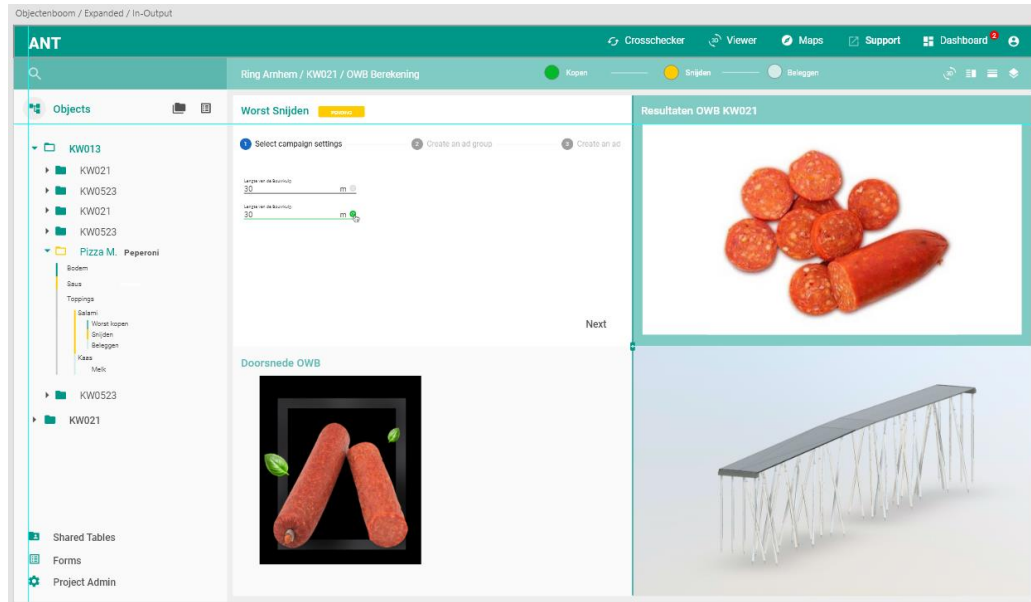
name	color	Actions
Role 1		  

De tabel start met de titel van het data object links bovenin. Vervolgens kunnen nieuwe elementen aan de tabel toegevoegd worden rechtsboven met de knop. Eventuele filters die aan- en uitgezet kunnen worden zijn weergegeven links van de knop. Vervolgens komen de kolommen van de data tabel en alle records. De laatste kolom is altijd de acties kolom waarmee er verschillende acties uitgevoerd kunnen worden op de geselecteerde record.

Onderin vindt de paginering plaats voor elke tabel waar een gebruiker zich mee kan navigeren in de tabel.

Deze structuur van een data tabel is vastgesteld en wordt aangehouden over de gehele ANT applicatie. (Neil, 2009)

6.4.1.2 UITVOEREN VAN EEN WORKFLOW

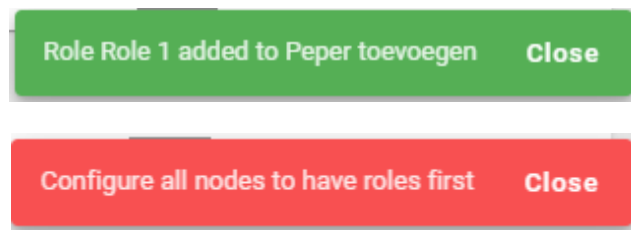


Figuur 6 workflow stap

Hierboven is een mockup weergegeven van een stap binnen een workflow (netwerk). Wat kenmerkend is voor deze pagina is dat het over 4 panelen beschikt waarbij de workflow bouwer zelf kan indelen welke informatie er in deze panelen wordt weergegeven. Denk hierbij aan data uit tabellen, afbeeldingen, 3D modellen en het invoer formulier voor nieuwe data van de stap. Zo zijn gebruikers vrij om deze panelen te configureren naar hun wens. Daarnaast moeten deze panelen schaalbaar zijn zodat de gebruiker deze kan vergroten of verkleinen naar wens.

6.4.2 GEBRUIKER FEEDBACK

Om gebruikers acties terug te koppelen naar de gebruiker is er een notificatie component ontwikkeld waar gebruikers responses terugkrijgen vanuit de applicatie. De groene responses zijn acties die succesvol zijn uitgevoerd door de gebruiker en de rode responses geven aan- dat de gebruiker iets probeert te doen wat nog niet mogelijk is. De errors die vanuit de server terugkomen worden automatisch als rode response getoond aan de gebruikers (Nielsen, 1994).

A form titled 'start session for workflow' is shown. It has two input fields: 'name' with a character count '0 / 250' and 'sbs code' which is a dropdown menu. The 'sbs code' field is highlighted with a red rectangular box. At the bottom right of the form are two buttons: 'Cancel' and 'Create'. Below the form, the words 'name' and 'roles' are partially visible.

Om fouten bij de gebruiker tegen te gaan wordt er gebruik gemaakt van drop down menu's waar de gebruiker kan selecteren welke optie hij/zij wilt. Hieronder volgt een voorbeeld waarbij een gebruiker een sbscode kan selecteren. Als hij deze zelf zou kunnen invullen zou dit veel foutgevoeliger zijn. Door in de UI zo veel mogelijke fouten af te vangen merkt de gebruiker hier het minst van. (Duggirala, 2016)

6.5 AUTODESK FORGE

De Forge viewer, van Autodesk, is de 3D viewer die wordt gebruikt binnen het ANT platform. In de viewer kunnen 3D modellen worden weergegeven. Deze 3D modellen zijn opgesplitst in objecten. Binnen een project kunnen deze objecten worden gekoppeld aan een SBS code. Deze code wordt gekoppeld aan data van bijvoorbeeld berekeningen. Via het ANT platform wordt zo alle data en objecten gekoppeld aan elkaar.

Binnen de Forge viewer wordt al deze data weergegeven zodat gebruikers het in een overzicht kunnen zien. Er wordt door gebruikers steeds meer gevraagd om extra Forge functies. De viewer krijgt een nieuw design waarmee er rekening wordt gehouden met de nieuwe en toekomstige functionaliteiten.

De Forge viewer is zichtbaar met ANT componenten erover heen. Deze stijl zal worden aangehouden met een nieuw design. Wat uit gesprekken met onder andere Jan Willem (CollaborAll) en Gert Karrenbelt (W+B) kwam met de huidige Forge viewer is dat er teveel zwevende elementen over de viewer hangen waardoor het veel klikwerk is voor de gebruiker om deze componenten te weergeven. Dit is een van de hoofdpunten voor het nieuwe design van de Forge viewer.

Het nieuwe design van de Forge viewer heeft als basis het oude design. De gebruiker heeft drie knoppen (zie rode pijlen in figuur 7) om zelf zijn zichtbare elementen in te delen. Bovenin kan de gebruiker de detail tabel en zoekbalk in en uit schakelen.

In eerste instantie was het de bedoeling dat de Forge viewer dezelfde uitstraling kreeg als het oude design. Dit betekende dat de gebruiker, door middel van knoppen, views aan en uit kon zetten boven op de Forge viewer. Nadat de nieuwe functionaliteiten bovenop de Forge de viewer waren vastgesteld. Leek dit niet de juiste oplossing. De gebruiker zou dan veel meer handelingen moeten verrichten om het gewenste resultaat te krijgen. ANT laat gebruikers navigeren via de linker sidebar die applicatie breed wordt ingezet. Als inspiratie is de Forge sidebar bedacht. Een centrale plek waar alle Forge functionaliteiten door de gebruiker kunnen worden geselecteerd. Dit zorgt niet alleen voor consistentie in de ANT applicatie, maar zorgt er ook voor dat gebruikers beter geleid worden naar de nieuwe functionaliteiten.

Deze specifieke sidebar kan worden opengeklapt aan de rechterzijde van de viewer. In het begin is er overwogen om de sidebar links te vervangen voor de forge sidebar om met name ruimte te besparen. Maar na enkele afwegingen is er besloten om de sidebar rechts doen. De gedachte hierachter is dat gebruikers altijd een manier moeten hebben om te navigeren binnen de ANT applicatie. Dit kan via de linker sidebar. De sidebar rechts is om de content binnen de view te manipuleren. In het kort: Links is de navigatie binnen de applicatie en rechts is de content binnen de view manipuleren of functies op toepassen.

In deze zijkant zijn de oude en nieuwe functionaliteiten ondergebracht. Hier kan de gebruiker alles over zijn ingeladen model terugvinden:

- De ingeladen modellen
- Alle objecten met eigenschappen van een model
- Alle eigenschappen van alle objecten samen kunnen worden ingeladen

Deze zijkant zorgt voor het overzicht voor alle nieuwe functies binnen een model. Zonder dat het nieuwe elementen laat zweven boven op de viewer.

7 TECHNISCH ONTWERP

In het technisch ontwerp wordt er door middel van database diagrammen en API documentatie de gerealiseerde functionaliteiten nader toegelicht. Daarnaast wordt er dieper inzicht gegeven in de logica achter de workflow functionaliteiten

7.1 STACK

De huidige ANT backend maakt gebruik van het php Laravel framework. Aangezien de workflows een van de grootste kenmerken wordt voor het ANT platform en veel connecties maakt met het huidige systeem is er gekozen om het workflow component binnen de huidige stack, van de ANT applicatie, te laten vallen.

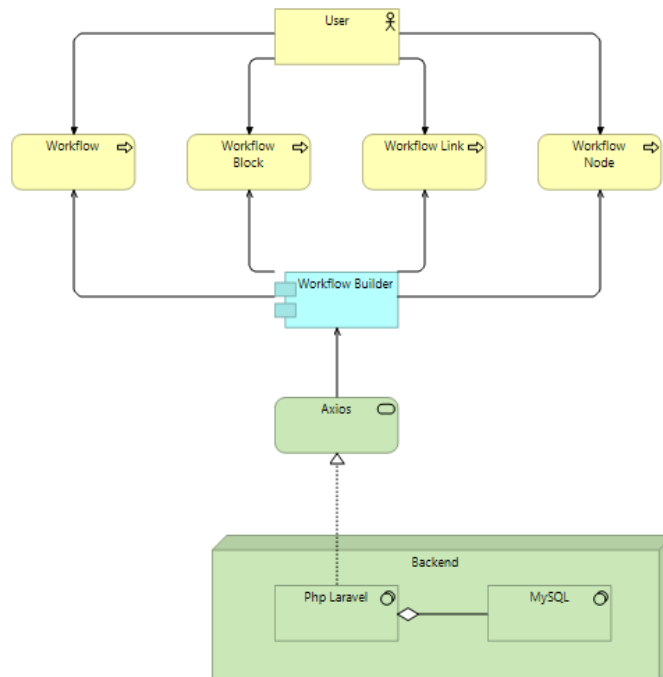
De UI van ANT gebruikt het Vue framework in combinatie met het Material Design framework Vuetify. Vuetify is een Vue UI library voor material design componenten. De ANT applicatie gebruikt deze componenten vanuit Vuetify over de gehele applicatie. Om consistentie te behouden voor de eindgebruiker en de applicatie als een geheel te laten aanvoelen is de keuze gemaakt om de Vuetify componenten ook te gebruiken voor de workflow functionaliteit. De UI van de workflows valt daarom binnen dezelfde stack als de ANT applicatie. Het wordt gebruikt binnen het ANT platform zelf en is een eis vanuit de stakeholders.

7.2 ARCHITECTUUR MODEL

De volgende modellen geven de architectuur van de drie workflow componenten: bouwen, configureren en uitvoeren binnen de ANT applicatie weer:

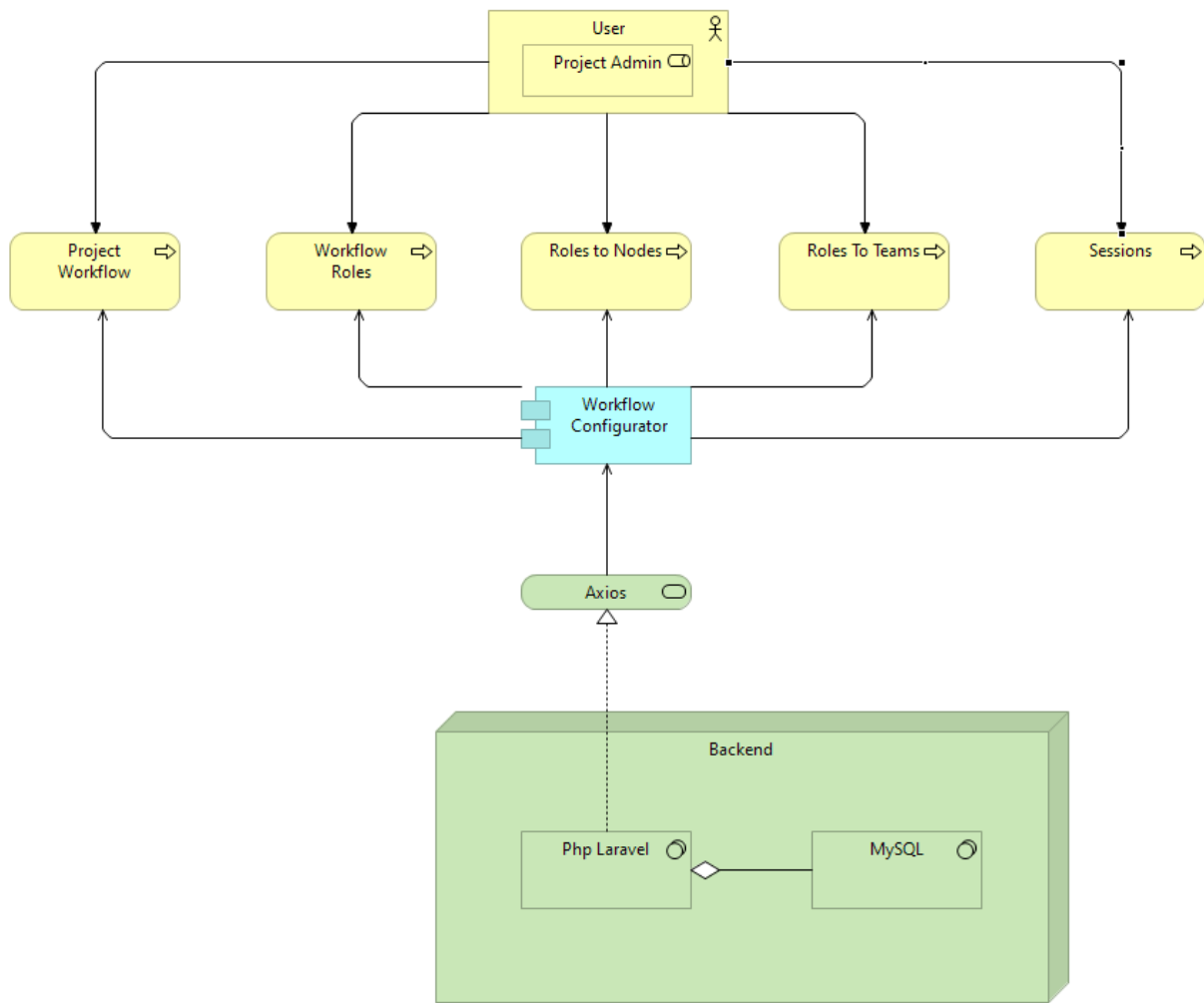
7.2.1 WORKFLOW BUILDER

Hieronder volgt het architectuur model van de workflow builder binnen de ANT omgeving. Gebruikers kunnen hier workflows, workflow blokken, nodes en links aanmaken.



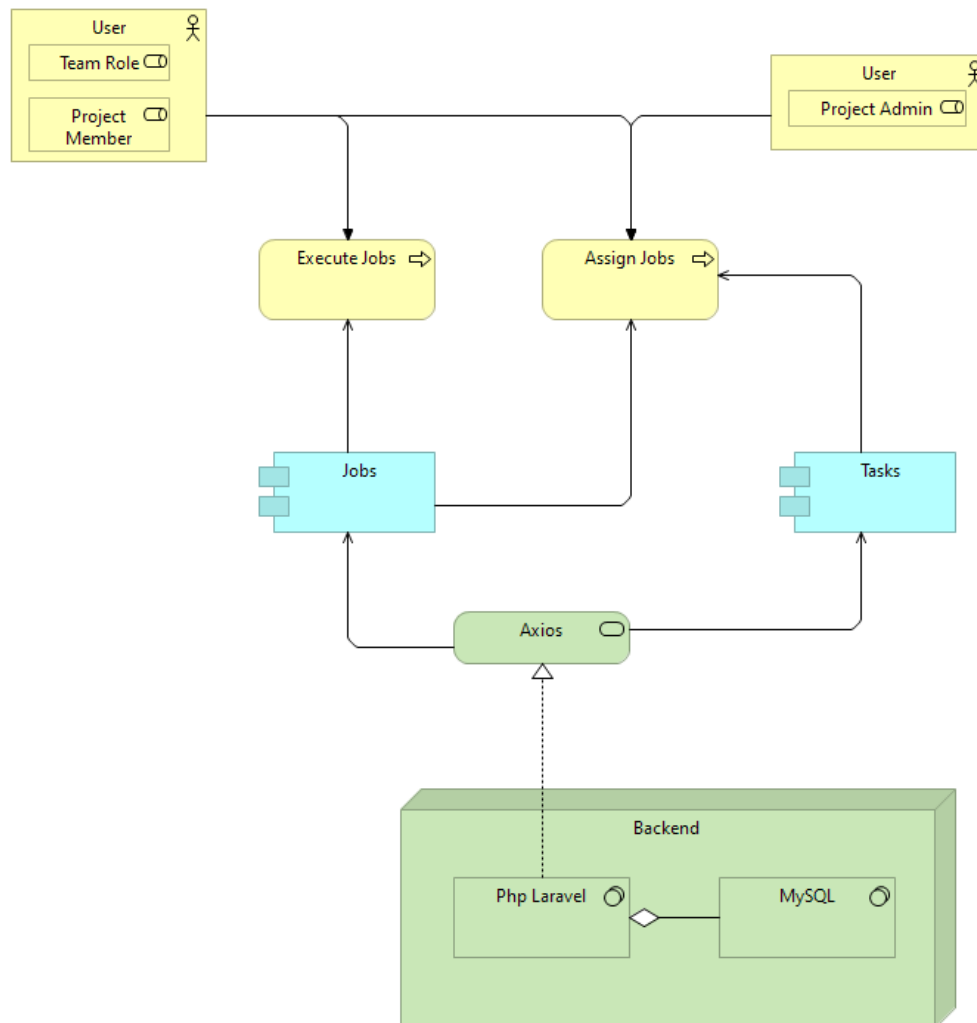
7.2.2 WORKFLOW CONFIGUREREN

Het volgende architectuur model geeft de workflow configuratie weer waar een gebruiker (project admin) workflows kan toevoegen binnen zijn/haar project en rollen kan toevoegen en toekennen aan de nodes en teams. Daarnaast kan er een sessie gestart worden met de geconfigureerde workflow.



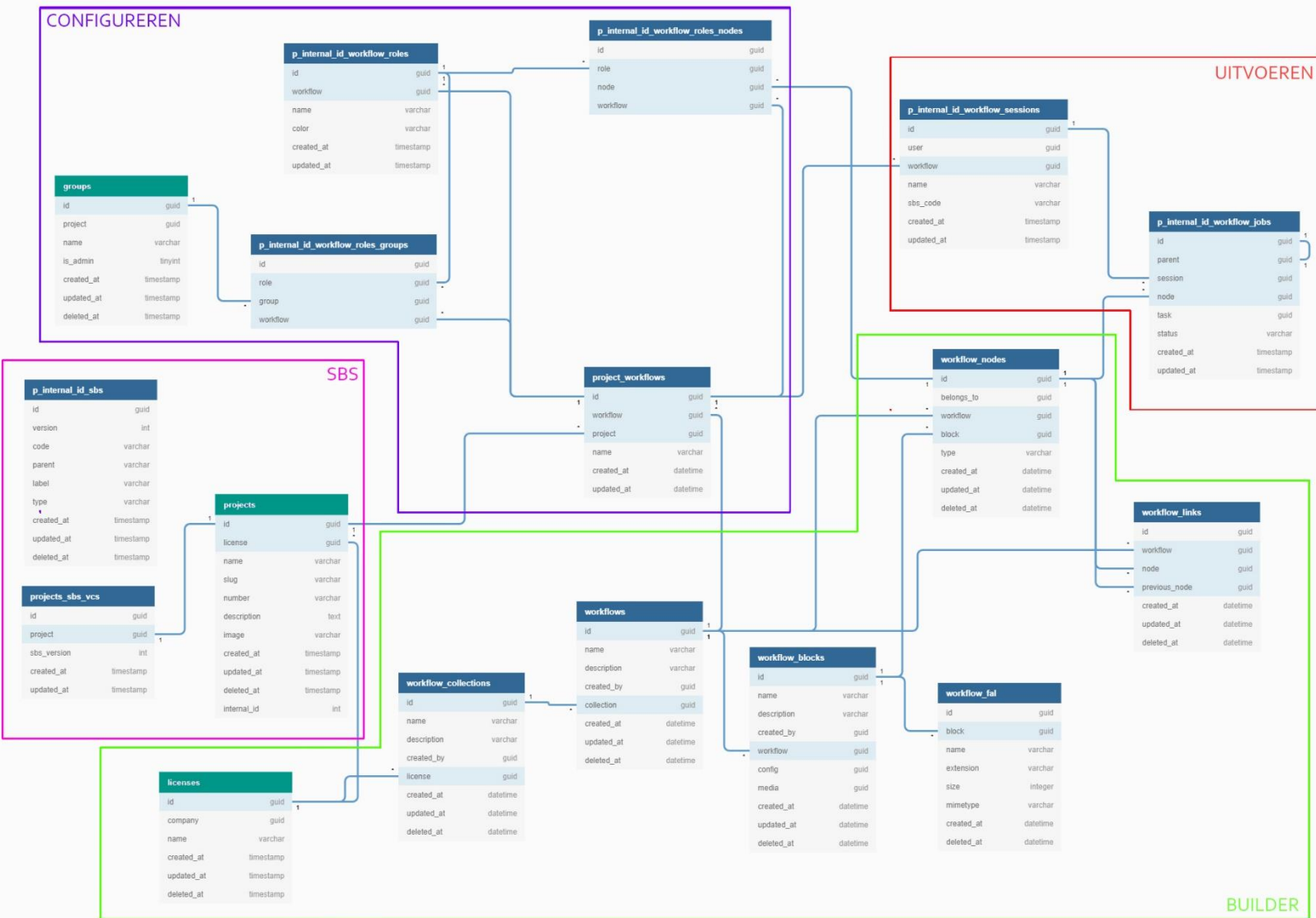
7.2.3 WORKFLOW UITVOEREN

Het laatste architectuur model heeft betrekking op het uitvoeren van een workflow. Zowel een user als project admin kan een job oppakken en een user kan daarnaast deze job uitvoeren. Workflow jobs maken, op de achtergrond, gebruik van het ANT taken systeem waar voor elke job een taak wordt aangemaakt die toegekend is aan de gebruiker die de job mag uitvoeren.



7.3 DATABASE DIAGRAMMEN

Het workflow systeem maakt gebruik van de interne project tabellen als systeem brede tabellen. Hieronder volgt eerst het systeem brede diagram met de tabellen voor workflows en de relatie tabellen met al bestaande tabellen in het huidige ANT systeem. De groene tabellen zijn al bestaande tabellen binnen het ANT systeem die een directe correlatie hebben met de Workflow tabellen (blauw).



Het database model van de workflows kan worden opgesplitst in de vier delen: bouwen, configureren, uitvoeren en het SBS onderdeel.

7.3.1 HET BOUWEN VAN EEN WORKFLOW

Voor het bouwen van een workflow zijn de volgende tabellen en relaties bedacht:

Workflows:

De workflow tabel legt de basis voor het gehele workflow systeem. Alle relaties worden teruggeleid naar deze tabel. De complexe logica wordt vastgelegd in de andere tabellen. De workflow tabel is van zichzelf erg statisch en houdt alleen een naam en beschrijving bij.

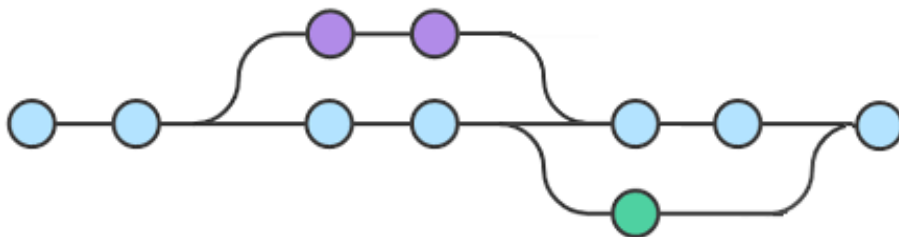
Workflow Blocks:

De workflow blok tabel heeft betrekking op de configuratie van een individuele stap binnen een workflow. Een blok maakt gebruik van de workflow_fal tabel. Hierin worden alle relaties tot configuratie bestanden en media opgeslagen. Via deze tabel kunnen de juiste bestanden worden gevonden op de server.

Workflow Nodes & Links:

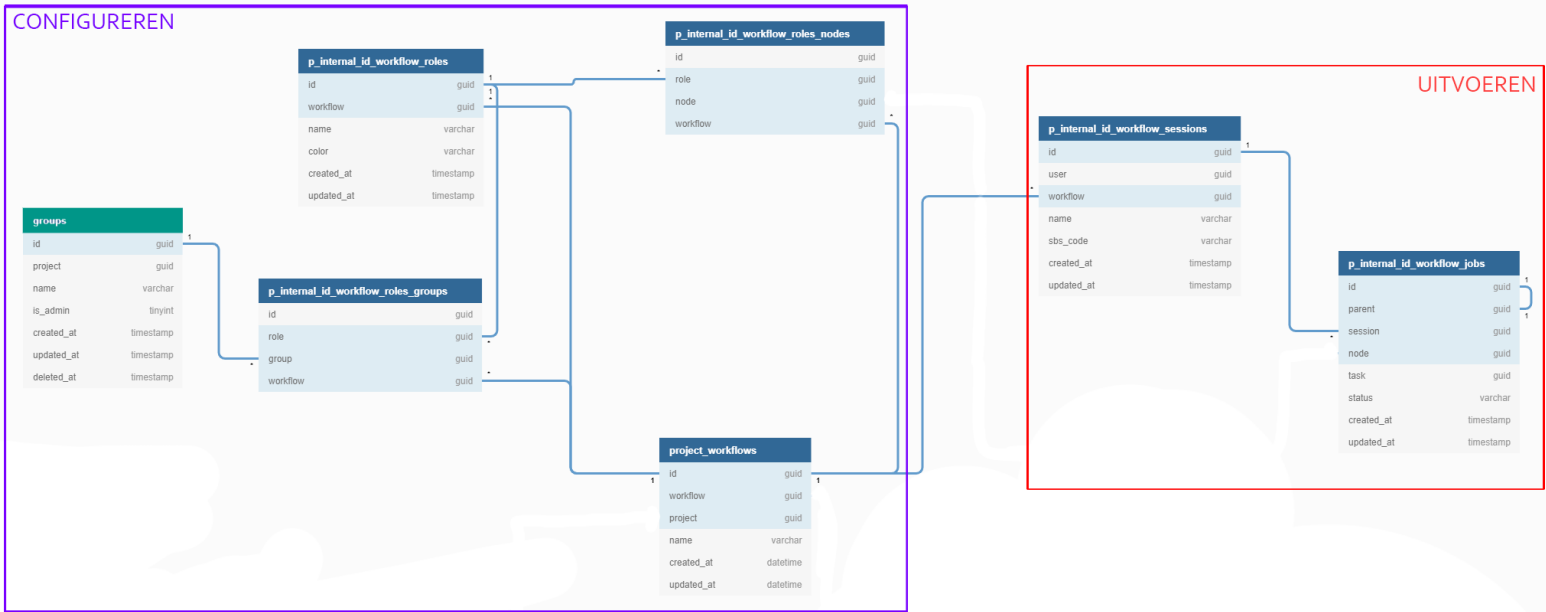
Om een netwerk te kunnen opzetten van verschillende workflows en individuele blokken zijn er twee tabellen opgezet die deze relaties kunnen vastleggen. Een node is niets anders dan een block of een andere workflow en de links leggen door middel van een node en previous_node de relatie af tussen verschillende nodes binnen een workflow. Een van de belangrijke eisen is dat een workflow of block parallel naast elkaar kunnen lopen waardoor er meerdere taken tegelijk gedaan kunnen worden.

Via de nodes en de links kan een heel netwerk aan blokken en workflows worden teruggeven die parallel kunnen splitsen (recursief) en samen kunnen komen. Dit is vergelijkbaar met het git branching systeem waar je vanuit één punt naar verschillende vertakkingen kan splitsen en deze ook weer samen kunnen komen. Dit systeem heeft ook inspiratie gegeven voor het uitdenken van het workflow netwerk.



7.3.2 PROJECT SPECIFIEKE TABELLEN

Het huidige ANT systeem maakt gebruik van een intern project id om project specifieke tabellen aan te maken. Voor het configureren en uitvoeren van de workflows worden vijf tabellen voor ieder project aangemaakt. In het volgende diagram zijn deze tabellen en de relaties met elkaar weergegeven:



Omdat workflows meerdere keren kunnen worden ingezet op een project is er een mapping tabel opgezet waar een workflow wordt gekoppeld aan een project (project_workflows tabel). Het id van deze tabel legt de relatie tussen de project specifieke workflow tabellen en de systeem brede workflow tabellen.

Workflow roles:

De workflow rollen zijn project specifiek en worden met twee mapping tabellen gekoppeld aan de nodes, van een workflow, en de teams binnen een project. Deze mapping wordt gedaan op basis van het id van de rol.

Workflow sessions & workflow jobs:

De workflow sessies tabel houdt samen met de jobs de progressie bij van het uitwerken van de nodes, die het uitvoeren van een workflow bijhoudt.

7.3.3 SBS

De SBS ofwel System Breakdown Structure wordt in het ontwerpen gebruikt om een code te koppelen aan een object binnen een model. Dit kan gezien worden als een ID van een object. Met de sbscode kan ANT een boom maken van het model en alle onderliggende objecten en hun kinderen. De sbscode wordt gebruikt om alle data binnen ANT te koppelen aan objecten van een model. De volgende tabellen zijn aangemaakt om dit kunnen realiseren:



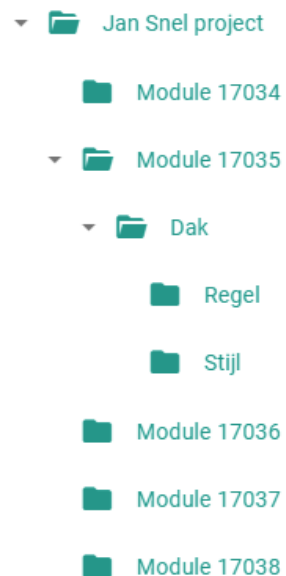
Voor het regelen van de SBS structuur zijn twee tabellen bedacht waarvan één systeem brede tabel waar het project wordt gekoppeld aan de huidige SBS versie die wordt gebruikt.

Daarnaast is er één project specifieke tabel waar alle SBS records die kunnen worden omgevormd tot een boom worden opgeslagen.

De SBS boom in ANT ziet er als volgt uit:

Elk object heeft een x aantal kinderen die van zichzelf ook kinderen kunnen hebben. Deze recursieve manier van kinderen kan in een boom worden weergegeven zoals hier rechts is aangegeven.

Op deze manier zijn alle objecten opgedeeld en door middel van de SBS code kan er data gekoppeld worden aan ieder object binnen de boom en dus binnen het model.



7.4 SERVICE REPOSITORY PATTERN

Wanneer we de huidige ANT omgeving bekijken zien we dat het hele backend systeem is opgebouwd vanuit het MDD (model driven development). Dit zorgt ervoor dat de modellen allemaal apart van elkaar gescheiden zijn en de logica van een model/object binnen het object valt (seperation of concern) (Warmer, 2018).

Deze trend ga ik doorzetten binnen het workflow gedeelte. Een van de dingen die mij opviel is dat het huidige ANT systeem alle logica binnen de controller doet van een API call. Dit botst heel erg met het MDD en het separation of concern model (Naumov, 2018).

Om deze botsing te voorkomen voor de nieuwe componenten heb ik onder andere het repository patroon, DAO patroon en het service repository patroon onderzocht.

Het repository patroon focust zich op het wegschrijven en ophalen van data uit een database ten opzichte van een model. Dit patroon principe leidt dat alle database transacties voor een model vanuit een plek komen (Petrakovich, n.d.).

Het DAO patroon, ofwel data access object patroon, is een patroon wat dichtbij de onderliggende opslag (database) ligt. DAO's komen in vele gevallen overeen met databasetabellen (Bansal, 2020).

Het service repository patroon is in feite het repository patroon met een extra service layer toegepast voor elk model waarin de business logica op de data kan worden afgehandeld (Vodovnik, 2015).

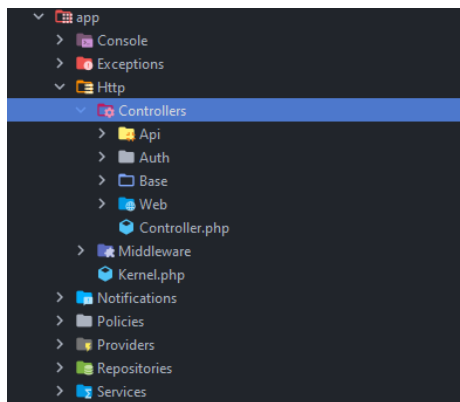
Wanneer we deze drie patronen met elkaar vergelijken en daarmee meenemen dat het development team graag de code zo leesbaar mogelijk wil houden. Kan er volgens Vodonik en Basal gesteld worden dat het service-repository patroon als beste naar voren komt. Het ANT development team wil graag de code zo leesbaar mogelijk en gescheiden houden, omdat zij hier in het verleden een slechte ervaring mee hadden gehad tijdens het uitbesteden van een component aan een derde partij.

De workflows hebben een dusdanige complexiteit dat het scheiden van de logica en database transacties een must is. Naast dat de meeste modellen binnen de workflows een directe correlatie hebben met elkaar kan deze logica in de services worden vastgesteld. Het service-repository patroon is voor deze use case de beste oplossing zodat de code leesbaarder is en duidelijker te begrijpen waar de logica zich bevindt binnen de workflow modellen.

Het service repository pattern zorgt er ook voor dat dezelfde database transacties over verschillende controllers gebruikt kunnen worden. In de service wordt de validatie gedaan en vanuit de service wordt de repository class aangeroepen waar alle database transacties plaatsvinden. Zo is de logica verdeeld over de controller -> service -> repository per model. Omdat alle workflow modellen zo erg met elkaar verbonden zijn scheidt dit patroon de logica, wat het ontwikkelen niet alleen praktischer maakt maar ook leesbaarder. Dit zorgt ervoor dat het debuggen en aanvullen sneller en makkelijker gaat waardoor er minder fouten ontstaan. Hierdoor wordt de kwaliteit van de code verhoogd.

Een controller van het huidige ANT systeem heeft vaak minimaal 500+ regels, waarbij alle controllers omtrent workflows rond de 200 regels code schommelen. Dit is ongeveer 60% minder regels code in de controllers (Zie bijlage 7).

Binnen het project ziet dit er als volgt uit:



alle controllers bevinden zich in de Http folder en voor elk model is in de services en repositories folder een class aangemaakt waar alle validaties en database transacties voor dat model plaatsvinden.

7.5 API DOCUMENTATIE

De huidige ANT applicatie wordt gedocumenteerd in Swagger Open API 3. Alle nieuwe API calls zijn toegevoegd aan de Swagger documentatie, van het ANT platform op: <https://developer.anticde.io/>.

Alle API calls geven data terug in JSON formaat met de juiste response codes. (Sahni, n.d.)

De API calls zijn opgedeeld in de volgende hoofd- en subroutes:

- /workflow-collections
- /builder
 - /workflows
 - /blocks+
- - /nodes
 - /links
- /project
 - /workflows
 - /roles
 - /teams
 - /nodes
 - /sessions
 - /jobs
 - /sbs

De routes zijn op gedeeld in drie hoofdroutes: workflow-collections, builder en project. De workflow collecties vallen buiten alle al bestaande routes en staan los van de builder of project. De collecties hebben daarom een eigen end-point voor de benodigde CRUD acties.

De /builder hoofdroute is bedacht met het inzicht dat ANT in de toekomst niet alleen workflows door de gebruiker wil laten bouwen. Dit is naar voren gekomen onder andere in de gesprekken met Jan Willem Kattouw. De /builder/workflows route is daarom op een logische manier genest achter de builder. De blokken, nodes en links zijn genest achter de workflow route. Zij hebben alle drie een correlatie met de workflow zelf, waardoor deze nesting logisch kan worden verklaard.

Een groot deel van de workflows logica wordt gedaan voor project specifieke tabellen. /project is daarom een hoofdroute waar de logische nested routes elk een eigen project specifieke tabel aanspreken. (Sahni, n.d.)

Door deze manier van nesting, van de subroutes, is in één oog opslag duidelijk wat de API call globaal doet. ANT wordt veel gebruikt zonder UI door bijvoorbeeld een berekening server. Voor de mensen die deze servers configureren maakt dit een groot verschil omdat de API call verheldert wat de request doet en wat het verwachte response is. (Au-Yeung & Donovan, 2020)

Hieronder volgen twee voorbeelden van hoe een route met logische geneste routes een groot verschil maken:

Type	Route	Body	Verwachte response t.o.v. route
POST	/role/{role}	Bevat data voor het toevoegen van een rol, maar ook de data van het project en workflow waar de rol voor geldt.	Kijkend naar de route is het onduidelijk wat rol in houdt. Daarnaast bevat de body dusdanig meer informatie die juist in de route inbegrepen had kunnen worden voor meer overzicht
POST	/project/{project}/workflow/{workflow}/role/{role}	Bevat data van de rol	De route laat direct zien door de nesting dat het gaat om een bepaald project en een workflow waar een rol aan toegevoegd moet worden. De data in de body heeft ook alleen betrekking op de rol en bevat geen extra informatie om de request af te handelen. Dat kan uit de route zelf gehaald worden.

Kijkend naar de twee voorbeelden is er veel meer overzicht wat de request doet in voorbeeld 2 ten opzichte van voorbeeld 1. Deze manier van nesting wordt ook aangeraden door Au-Yeun & Donovan in hun best practices voor REST API design. De hoofd- en subroutes zijn met deze gedachtegang opgesteld. (Au-Yeung & Donovan, 2020)

Alle API calls zijn gemaakt ten opzichte van het database model. Een volledig overzicht van de API calls is te vinden in de bijlage (Zie Bijlage 1). Hieronder wordt dieper ingegaan op de logica van de complexere API calls voor het importeren van een workflow en het doorlopen van een workflow. De meeste andere API calls zijn de CRUD acties voor de modellen die gebruikt worden om de logica van de workflow te voorzien.

7.6 WORKFLOW IMPORT

Het maken van een workflow kan op twee verschillende manieren. Met de ene manier kunnen gebruikers een workflow opbouwen door individueel workflows, blokken, nodes en links aan te maken met de CRUD acties die voor elk model beschikbaar zijn in de API.

Een tweede manier is een gehele workflow importeren vanuit één enkele API call:
`/builder/workflow/import`

Een gebruiker kan met een JSON format een workflow maken met alle benodigde blokken, nodes en links. Een voorbeeld van een simpele JSON is te vinden in de bijlage (zie Bijlage 2). De JSON wordt meegestuurd in de body van request. Waarbij alle nodes staan gedefinieerd met wat voor type het is: 'block' of 'workflow'. Daarnaast wordt voor elke node ook de vorige node(s) gedefinieerd om zo de links op te kunnen maken.

De importer maakt recursief de workflows aan en de benodigde blokken daarin die zijn gedefinieerd in de JSON. Voor elke blok of workflow wordt een node gemaakt en aan het eind van elke workflow worden de nodes aan elkaar gekoppeld via de links. Van het voorbeeld in bijlage 2 is bijlage 3 de response die de gebruiker terugkrijgt van het importeren van een workflow (zie Bijlage 3).

7.7 FLOW VAN EEN WORKFLOW

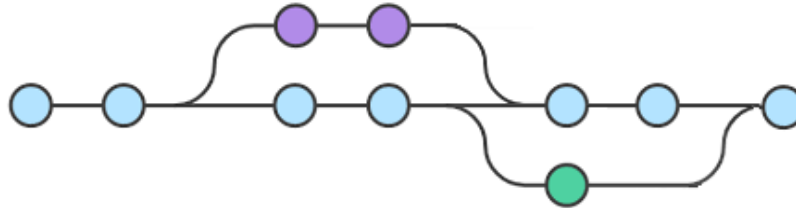
Een van de grootste uitdagingen voor de workflows is dat workflows en blokken sequentieel en parallel naast elkaar moeten kunnen lopen. Dit noemen we de "flow" van de workflow. Vanwege het hergebruiken van juiste workflows moest er een manier worden gevonden om workflows en blokken samen te laten werken binnen dezelfde flow. Om dit te kunnen realiseren is er een soort shell object bedacht genaamd een Node. Een node is een blok of heeft een hele workflow in zichzelf.

Om nodes aan elkaar te knopen zijn er links bedacht. Voor de links zijn er drie iteraties geweest om de flow van een workflow op te zetten.

1. Een link heeft een huidige node en een next node.
2. Een link heeft een huidige node en een next node en een previous node.
3. Een link heeft een huidige node en een previous node.

Omdat een node een workflow kan zijn (die van zichzelf nodes bevat) werkte de eerste iteratie niet. Een node die een workflow is, is van zichzelf agnostisch. De stappen binnen de workflow weten zelf niet dat ze bij een groter geheel horen (parent workflow). Deze iteratie liep stuk op het feit dat de laatste node in de kinder workflow niet de stap kon doorzetten naar de parent workflow, omdat deze niet van zijn bestaan afweert.

De tweede iteratie van links bevatten een next node en een previous node. Door de uitkomst van de eerste iteratie was dit de logische vervolg stap. Als we kijken naar een stappenplan, met vertakkingen:



Een node kan alleen uitgevoerd worden als alle nodes voor hem zijn afgerond. Vandaar dat de previous node van belang is. Bij het afsluiten van een node kan er worden gekeken of alle nodes voor de huidige node zijn afgesloten. Als dit zo is dan kan de node uitgevoerd worden.

Tijdens de tweede iteratie werd de flow van een workflow gerealiseerd en wat opviel was dat de next node overbodig was. De derde en laatste iteratie heeft dit bevestigd door een flow te kunnen opzetten tussen de nodes door alleen de previous node te gebruiken. Dit lukte zowel sequentieel als parallel naast elkaar.

7.8 DOORLOPEN VAN EEN WORKFLOW

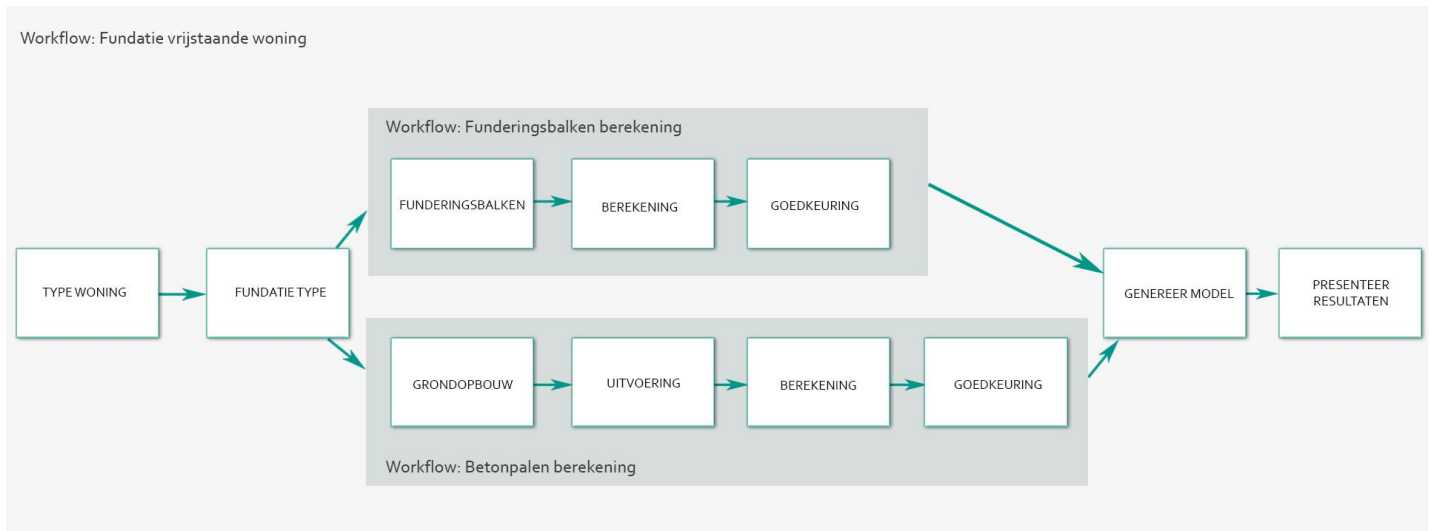
Met de flow van een workflow kan voor de gebruikers een stappenplan worden gegenereerd op basis van de nodes in een workflow. Het systeem bekijkt elke node en wanneer een node een blok is wordt hiervoor een job aangemaakt. Wanneer het systeem een kind workflow als node tegenkomt wordt er voor elke node, van de kind workflow, opnieuw gekeken welk type node dit is en als dit een blok is worden hier ook jobs voor aangemaakt. Dit recursieve proces leidt tot een lijst aan uitvoerbare stappen voor gebruikers of berekening servers. (Zie bijlage 4)

Deze lijst van jobs wordt gegenereerd wanneer er een sessie wordt gestart van een workflow. De eerste job binnen de workflow wordt open gezet. Deze kan door een gebruiker of berekening server worden opgepakt en uitgevoerd.

Het ANT takensysteem wordt gebruikt om tussen gebruikers en servers taken uit te wisselen. Dit systeem wordt door de jobs gebruikt om de stap van de workflow uit te voeren. Wanneer een job wordt opgepakt door een gebruiker of server wordt er automatisch een taak toegewezen aan deze gebruiker. Met de taak kan de gebruiker de job uitvoeren en daarmee ook afsluiten als deze voltooid is.

Met de flow van de workflow kan na het afsluiten van een job de volgende job worden geopend door middel van de flow in de nodes. Wanneer een job afgesloten is wordt er gekeken wat de volgende job in de reeks is. Als voor de volgende alle voorgaande jobs succesvol zijn afgesloten wordt deze open gezet om opgepakt kunnen worden. Dit proces zorgt ervoor dat gebruikers sequentieel of parallel kunnen samenwerken met andere gebruikers, maar ook met berekening servers aan dezelfde workflow. (Zie bijlage 5)

De complexiteit van de workflows zit het met name in het agnostische en herbruikbare gedeelte van een workflow terwijl deze sequentieel maar ook oneindig parallel kunnen lopen. Wanneer we kijken naar de voorbeeld workflow:



In dit voorbeeld zijn alle uitvoerbare stappen gedefinieerd. Omdat een workflow van zichzelf agnostisch is was het technisch lastig om de flow bij te kunnen houden. Een volgende stap mag alleen open worden gezet om uitgevoerd worden als alle voorgaande stappen zijn voltooid. Het agnostische gedeelte maakt dit lastig omdat de laatste stap van een kinder workflow dus niet van zichzelf weet wat zijn volgende stap is, omdat deze agnostisch is. In dit voorbeeld is dat dus de connectie tussen de goedkeuring stap naar genereermodel stap.

Om dit probleem op te lossen wordt er voor elke uitvoerbare stap in de workflow een job aangemaakt waaraan de node van de workflow is gekoppeld. De nodes en links bepalen de volgorde over welke jobs open worden gezet als een job wordt afgesloten. Om het bovengenoemde probleem op te lossen wordt er voor de hele workflow kinder workflow een job aangemaakt die niet zichtbaar is voor de gebruiker. Deze job houdt bij of alle kinder jobs van hem zijn afgesloten. Wanneer dit zo is weet hij dat de kinder workflow is afgerond en dat de volgende stap kan worden open gezet. De volgende job bekijkt op zichzelf of hij open mag gaan als al zijn voorgaande stappen zijn voltooid.

Een ander probleem was de communicatie tussen gebruiker en berekening server om het geautomatiseerde deel van een workflow te kunnen uitvoeren. Elke job geeft een taak weg aan de gebruiker die de job krijgt toegewezen. Door dit proces kan een server, wie binnen ANT als een gebruiker wordt gezien, wachten totdat hij een taak wordt toegewezen. Wanneer de server een taak binnen krijgt kan hij vervolgens de informatie ophalen vanuit de configuratie van het blok die gekoppeld is aan de taak. Met deze informatie kan hij de berekening uitvoeren door de juiste data op te halen binnen ANT, deze te verwerken en wegschrijven naar de geconfigureerde tabel in ANT.

In het voorbeeld zou dit zo eruit zien. Voor de workflow 'Funderingsbalken berekening' wordt in de eerste stap data ingevoerd van bijvoorbeeld het type balk, de lengte, breedte en diepte. Deze data wordt weggeschreven naar een tabel binnen ANT. De server krijgt een taak binnen en ziet in de configuratie welk soort berekening hij moet uitvoeren en waar hij de parameters kan vinden voor deze berekening. Deze parameters haalt hij op uit de tabel en voert de berekening uit. De uitkomst van de berekening wordt weggeschreven en kan in de volgende stap door de gebruiker worden gevalideerd en goedgekeurd. Met al deze data kan een model worden gegenereerd.

Deze workflow kan door zijn agnostische en herbruikbare eigenschappen. Op zichzelf draaien, maar ook ingezet worden in een grotere workflow zoals in het voorbeeld. Deze eigenschappen samen met het automatiseren van stappen zorgde voor de grote complexiteit van de functionaliteiten.

8 TESTEN

Om de kwaliteit van de code te bewaren en te zorgen dat de gewenste functionaliteit werkt zoals bedoeld is er bijna elke week een sessie gehouden met Danny Lamarti (externe vanuit CollaborAll). Samen met hem zijn in de backend de functies vastgesteld waar unit tests voor nodig waren en deze zijn uitgewerkt (Zie Bijlage 6). Voor het testen van sommige functies zijn er lambda functies geschreven, zodat de logica van de functie getest kon worden zonder dat daar bijvoorbeeld database toegang voor nodig zou zijn (Potencier, 2009). De geschreven unit tests hadden als uitgangspunt om zowel good weather als bad weather scenarios te testen (Deursen, 21). Een test was geslaagd wanneer er minimaal 2 good weather scenarios en 2 bad weather scenarios werden getest.

Het ANT platform heeft naast zijn live omgeving twee test omgevingen: alpha en beta. Deze test omgevingen zijn gebruikt om de nieuwe functionaliteiten te testen. Op alpha kunnen de developers in een soortgelijke omgeving, zoals de productie omgeving, de functionaliteiten testen.

Beta komt dichterbij de productie-versie. Hier worden de functionaliteiten als een release klaargezet en op beta worden deze functionaliteiten getest door onder andere Gert Karrenbelt (W + B) en Jan Willem Kattouw (CollaborAll). Wanneer de nieuwe functionaliteiten zijn getest op beta wordt om de week een release vanuit beta naar de productie omgeving gezet. Alle functionaliteiten zijn in Gitlab opgenomen als tickets en deze tickets bevatte de informatie om de functie te testen. Een test was succesvol wanneer Gert Karrenbelt of Jan Willem akkoord heeft gegeven en daarmee kon de ticket worden gesloten.

De workflow functionaliteit is aan het einde van de periode na succesvolle beta tests op de productie omgeving gezet. Het configuratie overzicht van de workflow werd goed ontvangen. Wat gebruikers opviel was de kleur codes die gegeven konden worden aan rollen. Dit was iets waar de stakeholders niet aan gedacht hadden en ik vanuit mijn input heb gerealiseerd.

De gebruikers konden de workflows doorlopen via de tabel. Een van de verbeter punten die zij aangaven was dat ze graag de workflow visueel voor zich wilden hebben en de status van de blokken hierin terug kunnen zien. Dit wordt opgenomen in de aanbevelingen en zal de eerst volgende stap zijn voor de workflows.

Als laatste feedback punt vonden de gebruikers de panels die ze zelf kunnen configureren erg goed werken. Juist door deze functie gaven ze aan nog meer met de panels te willen doen, zoals meerdere media bestanden, een markdown panel. Dit zal ook worden opgenomen in de aanbevelingen als

Door de complexiteit van de workflows zijn er op alpha een aantal test projecten aangemaakt waar eindgebruikers kunnen afkijken en een feeling kunnen krijgen over hoe dit in hun live omgeving straks zichtbaar is, nadat de workflows voor hun projecten zijn ingeregeld en geconfigureerd.

Het realiseren en testen van de requirements werd tijdens de ontwikkelperiode opnieuw geprioriteerd tijdens de wekelijkse sessies met de stakeholders. Dit resulteerde in dat sommige should of would requirements eerder gerealiseerd en getest werden ten opzichte van andere must requirements.

In het volgende overzicht zijn alle functionele requirements opgenomen en of ze succesvol zijn gerealiseerd zowel benaderbaar via de REST API als via de UI:

- De groene requirements zijn succesvol gerealiseerd en getest
- De oranje zijn gedeeltelijk gerealiseerd en getest
- De rode zijn niet gerealiseerd

Bouwen van een workflow		API	UI
MUST	Als een gebruiker wil ik mijn eigen workflows kunnen bekijken en beheren		
MUST	Als een gebruiker wil ik een workflow blok kunnen bekijken, beheren en configureren		
MUST	Als een gebruiker wil ik workflows aan elkaar kunnen knopen		
MUST	Als een gebruiker wil ik workflow blokken aan elkaar kunnen knopen		
MUST	Als een systeem wil ik alle project specifieke workflow tabellen aanmaken tijdens het aanmaken van een project		
MUST	Als een gebruiker wil ik configuratie bestanden, in JSON, kunnen uploaden en toevoegen aan een workflow blok		
SHOULD	Als een gebruiker wil ik een workflow kunnen importeren		
SHOULD	Als een gebruiker wil ik een media bestand kunnen uploaden en toevoegen aan een workflow blok configuratie		
SHOULD	Als een gebruiker wil ik een Autodesk Forge configuratie toevoegen aan een workflow blok configuratie		
SHOULD	Als een gebruiker wil ik ANT tabellen kunnen toevoegen aan een workflow blok configuratie		
Configureren van een workflow		API	UI
MUST	Als een licentie houder wil ik een workflow binnen mijn licentie kunnen publiceren en beheren		
MUST	Als een project admin wil ik een workflow binnen mijn licentie kunnen toevoegen aan mijn project		
MUST	Als een project admin wil ik mijn project specifieke workflows kunnen bekijken en beheren		
MUST	Als een project admin wil ik voor een workflow rollen kunnen bekijken en beheren		
MUST	Als een project admin wil ik de relatie tussen workflow rollen en project teams kunnen bekijken en beheren		
MUST	Als een project admin wil ik de relatie tussen workflow rollen en workflow nodes kunnen bekijken en beheren		
MUST	Als een project admin wil ik sbs objecten kunnen bekijken en beheren		

MUST	Als een project admin wil ik een sbscode toekennen aan een workflow sessie		
SHOULD	Als een licentie houder wil ik mijn workflow collecties kunnen bekijken en beheren		
SHOULD	Als een licentie houder wil ik een workflow kunnen toevoegen aan een licentie binnen een collectie		
COULD	Als een project admin wil ik een overzicht hebben van alle rollen, teams en blokken		
WOULD	Als een project admin wil ik een versie bijhouden van mijn sbs objecten		
Uitvoeren van een workflow		API	UI
MUST	Als een project admin wil ik een sessie kunnen starten		
MUST	Als een project admin wil ik alle workflow sessies kunnen bekijken en beheren		
MUST	Als een project admin wil ik gebruikers kunnen toekennen aan een bepaalde stap binnen een workflow		
MUST	Als een gebruiker wil ik andere gebruikers binnen mijn team kunnen toekennen aan een bepaalde stap binnen een workflow		
MUST	Als een gebruiker wil ik al mijn openstaande stappen zien binnen een workflow		
MUST	Als een gebruiker wil ik parameters kunnen invullen binnen een workflow stap		
SHOULD	Als een systeem wil ik workflow stappen koppelen aan ANT taken		
SHOULD	Als een gebruiker wil ik mijn stappen kunnen filteren op sessie en status		
WOULD	Als een gebruiker wil ik zelf kunnen indelen welke delen ik zie van een workflow stap		
Autodesk Forge		API	UI
MUST	Als een gebruiker wil ik alle objecten van een model kunnen weergeven	n.v.t.	
MUST	Als een gebruiker wil ik de eigenschappen van een object kunnen weergeven	n.v.t.	
MUST	Als een gebruiker wil ik data van een object kunnen updaten	n.v.t.	
MUST	Als gebruiker wil ik een overzicht kunnen genereren van alle eigenschappen van het model	n.v.t.	
SHOULD	Als een gebruiker wil ik een geselecteerd sbs object kunnen highlighten in de viewer	n.v.t.	
SHOULD	Als een gebruiker wil ik een geselecteerd model object kunnen highlighten in de viewer	n.v.t.	
SHOULD	Als gebruiker wil ik de Viewer componenten kunnen aan en uit schakelen	n.v.t.	

9 CONCLUSIE

Het ANT platform geeft, met de workflows, haar gebruikers de mogelijkheid om ontwerp processen te automatiseren. Het bouwen van de workflow kan een gebruiker naar eigen wens doen. Elk blok heeft zijn eigen configuratie, zodat hier de gevraagde informatie voor bijvoorbeeld een berekeningsserver in kan worden gezet. Met de nodes en de links kan een gebruiker al zijn blokken aan elkaar knopen en hiermee de “flow” van de workflow definiëren.

Als projectleider kunnen deze workflows geconfigureerd worden naar eigen wens. Wie wat mag doen of zien kan worden vastgezet in rollen die worden toegekend aan de projectteams. Op die manier kan er worden vastgesteld welke blokken voor gebruikers zijn toegekend, maar ook welke blokken worden uitgevoerd door een berekening server.

Het uitvoeren van een workflow wordt gedaan in sessies. Een sessie wordt gekoppeld aan de sbscode van een object. Hierdoor wordt alle data vanuit gebruikers input, berekening servers of vanuit het ontwerp gekoppeld aan objecten. Dit noemt men het parametrisch ontwerpen.

Met de workflows kunnen projecten hun ontwerpproces niet alleen in kaart brengen, maar ook automatiseren. De vrijheid die de workflows biedt zorgt ervoor dat gebruikers hun eigen creativiteit en inbreng kunnen geven aan het ontwerpproces. Daarnaast kunnen projectleiders ook kritischer naar hun proces kijken.

ANT geeft gebruikers de mogelijkheid om creatieve, innovatieve en geautomatiseerde processen toe te passen op het ontwerpen. Met de workflows zijn de mogelijkheden eindeloos.

10 DISCUSSIE

Naar aanleiding van de vraag om ontwerpprocessen te kunnen automatiseren en configureren binnen het ANT platform door de eindgebruikers is dit onderzoek uitgevoerd om deze workflows te analyseren, ontwerpen en realiseren. Tijdens het opstellen van de requirements werd de functionaliteit onderzocht en opgesplitst in de drie kernonderdelen van de workflows: het bouwen, configureren en uitvoeren. Dit bleek een goede keuze te zijn om de complexiteit te scheiden en overzichtelijk te houden. Desondanks door deze scheiding werden de functionaliteiten niet goed op elkaar afgestemd. Dit leidde tot aanpassingen aan requirements die voorzien hadden kunnen worden door eerst het geheel te schetsen met de stakeholders en vervolgens de diepte in te gaan en de scheiding te definiëren tussen de drie kernonderdelen.

De aanpak voor het ontwerpen en realiseren werd geschreven vanuit de ervaring van mij als software engineer. Dit hield in dat voorgaande projecten werden opgeleverd aan het eind. Door de feedback loop die ANT heeft met gebruikers ging deze aanpak niet zoals gewenst. De functionaliteiten moesten zo snel mogelijk en bruikbaar ontwikkeld worden, zodat gebruikers er al mee aan de slag konden en hun feedback op konden geven. Deze loop elke week zorgde ervoor dat er veel requirements en extra's wekelijks bij op kwamen waardoor het overzicht van de requirements die aan het begin werden opgesteld steeds meer verwaterden en bijgewerkt moesten worden. Deze inschatting van hoe een project wordt gerealiseerd had van te voren beter door mij ingeschat moeten worden. Waardoor er beter geanticipeerd kon worden op het realisatie proces. ANT wilde gebruikers zo snel mogelijk toegang geven voor de ontwikkelde functionaliteit ook al was dit nog maar een klein deel van het grotere geheel, zodat ze zoveel mogelijk feedback en aanpassingen konden doen naar de wensen van de gebruiker.

Het analyseren en ontwerpen van de gevraagde functionaliteit is iets wat dit onderzoek naar de workflows goed heeft gedaan. Door de vele contact momenten, met de stakeholders, en presentaties over hoe het design er uit gaat zien, maar ook het database model werd vastgesteld zorgde ervoor dat in een korte tijd een heel groot deel van de functionaliteit gerealiseerd kon worden. Deze voorbereiding zorgde ervoor dat er weinig verrassingen waren tijdens het realiseren van de functionaliteiten. Dit hielp voor niet alleen het development team, maar ook voor de stakeholders om te begrijpen hoe de functionaliteit zou worden gerealiseerd en of dit dan ook hun eisen beantwoorde door kritische vragen te stellen.

Kijkend naar de drie HBO competenties: analyseren, ontwerpen en realiseren. Het analyseren van de gevraagde functionaliteit met de hoge complexiteit door het agnostische en herbruikbare eis van een workflow, maar ook het geautomatiseerd stappen kunnen doorlopen zowel sequentieel als parallel en dit kunnen omzetten naar requirements, modellen en diagrammen toont aan dat ik een complex probleem kan vertalen naar een functioneel ontwerp voor de gevraagde functionaliteit. Het ontwerpen van dit complexe probleem en met name kunnen toelichten, van het ontwerp, en verklaren aan de desbetreffende stakeholders. Toont aan dat ik zelfstandig een ontwerp voor een complex vraagstuk kan realiseren, maar dit ook juist kan presenteren en onderbouwen aan de vele stakeholders.

Juist door het correct analyseren en ontwerpen van dit vraagstuk was het realiseren niet moeilijk. Dit kwam niet omdat de complexiteit niet hoog genoeg was, maar juist omdat dit in de voorgaande fases al goed was uitgedacht. De stakeholders waren dan ook verrast dat hun idee nu al door gebruikers kan worden ingezet en de volgende fases van dit idee vroegtijdig konden worden gestart om hun product nog beter te maken. Met het workflow vraagstuk heb ik kunnen aantonen dat ik een complex probleem zelfstandig kan analyseren, ontwerpen en realiseren als een software engineer zou moeten kunnen.

11 AANBEVELINGEN

Met de workflows is de eerste stap gezet naar het automatiseren van het ontwerpproces. Een van de vervolg stappen voor ANT is om de workflow met een intuïtief UI aan de gebruikers te presenteren. Dit kan bijvoorbeeld door de workflow om te zetten naar een flowchart waardoor gebruikers inzicht krijgen waar zij zich bevinden binnen de workflow.

De drie kernfunctionaliteiten van de workflow zijn uitgewerkt en toegankelijk via de API van ANT. Het configureren en uitvoeren van een workflow zijn ook vertaald binnen de UI van het ANT platform. Voor de UI/UX voor het bouwen van een workflow raad ik ANT aan om te kijken naar een drag en drop systeem waar gebruikers hun blokken kunnen plaatsen, configureren en aan elkaar knoppelen. Met als resultaat als het ware een “flowchart” die overeenkomt qua design hoe gebruikers de workflow doorlopen binnen het ANT platform.

Na de gebruikersfeedback op de productie omgeving wordt er gevraagd om meerdere soorten configuratie panels binnen een workflow blok. Zo vragen gebruikers om een markdown paneel, waar zij extra informatie kunnen geven aan de blok stap. Ook het toevoegen van meerdere mediabestanden en/of video bestanden aan een paneel was een van de wensen die naar voren kwam bij de eindgebruikers.

Wanneer we kijken naar het technische aspect van de workflows zijn daar ook nog twee vervolgstappen naar voren zijn gekomen vanuit de gebruikersfeedback van de eerste versie. Enerzijds willen gebruikers een conditionele flow maken. Hiermee kunnen ze de workflows nog meer configureren naar eigen wens. Dit is in de huidige versie van de workflows niet mogelijk. Daarnaast is er vraag naar versiebeheer omtrent het bouwen van de workflows. Configuraties kunnen veranderen. Er komen extra stappen bij in de workflow. Gebruikers willen een workflow kunnen bijwerken, maar ook de oude versie behouden.

ANT kan met de bovengenoemde verbeteringen een tweede versie van de workflows realiseren wat hun product nog toegankelijker maakt voor gebruikers.

12 BIBLIOGRAFIE

- Au-Yeung, J., & Donovan, R. (2020, maart 2). *best-practices-for-rest-api-design*. Opgehaald van stackoverflow.blog: <https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/>
- Bansal, A. (2020, september 10). *java-dao-vs-repository*. Opgehaald van baeldung: <https://www.baeldung.com/java-dao-vs-repository>
- CollaborAll. (2021). Opgehaald van <https://collaborall.net/>
- Deursen, A. v. (21, december 2012). *unit-testing*. Opgehaald van avandeursen: <https://avandeursen.com/tag/unit-testing/>
- Duggirala, S. (2016, Augustus 18). *10-usability-heuristics-with-examples-4a81ada920c*. Opgehaald van prototypr: <https://blog.prototypr.io/10-usability-heuristics-with-examples-4a81ada920c>
- Naumov, A. (2018). *separation-of-concerns*. Opgehaald van nalexn: <https://nalexn.github.io/separation-of-concerns/>
- Neil, T. (2009, Mei 17). *6-tips-for-a-great-flex-ux-part-5*. Opgehaald van designingwebinterfaces: <http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5>
- Nielsen, J. (1994, april 24). *ten-usability-heuristics*. Opgehaald van nngroup: <https://www.nngroup.com/articles/ten-usability-heuristics/>
- parametrisch-ontwerpen-voor-dummies*. (2018, 11 19). Opgehaald van paotm: <https://paotm.nl/nl/nieuws/parametrisch-ontwerpen-voor-dummies/#:~:text=Parametrisch%20ontwerpen%20is%20een%20proces,laatste%20jaren%20een%20enorme%20vlucht.&text=Bij%20parametrische%20software%20kan%20je%20zelf%20expliciet%20de%20parameters%20te%20defini>
- Petrakovich, J. (sd). *why-should-you-use-the-repository-pattern*. Opgehaald van makingloops: <https://makingloops.com/why-should-you-use-the-repository-pattern/#:~:text=The%20repository%20pattern%20is%20a,list%20items%20in%20a%20table>.
- Potencier, F. (2009, april 16). *on-php-5-3-lambda-functions-and-closures*. Opgehaald van fabien.potencier: <http://fabien.potencier.org/on-php-5-3-lambda-functions-and-closures.html#:~:text=To%20sum%20up%2C%20a%20lambda,aware%20of%20its%20surrounding%20context>.
- Rutten, N. (2016, mei 4). *op-naar-netwerkorganisatie*. Opgehaald van nieuworganiseren: <https://www.nieuworganiseren.nu/actueel/op-naar-netwerkorganisatie/>
- Sahni, V. (sd). *best-practices-for-a-pragmatic-restful-api*. Opgehaald van vinaysahni: <https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>

Visual Paradigm. (sd). *what-is-activity-diagram*. Opgehaald van visual-paradigm: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>

Vodovnik, A. (2015, August 26). *repository-and-services-pattern-in-a-multilayered-architecture*. Opgehaald van vodovnik: <https://www.vodovnik.com/2015/08/26/repository-and-services-pattern-in-a-multilayered-architecture/>

Voort. (2017, november 9). *wat-is-civiele-techniek*. Opgehaald van voort: <https://www.voort.com/vakgebied/wat-is-civiele-techniek/>

Vuetify. (2021, mei 31). *ui-kits*. Opgehaald van vuetifyjs: <https://vuetifyjs.com/en/resources/ui-kits/>

Warmer, J. (2018). *mdd*. Opgehaald van openmodeling: <https://www.openmodeling.nl/nl/mdd.html>

13 BIJLAGES

13.1 BIJLAGE 1, API CALLS

Aan alle routes gaat vooraf de host, /api en het versie nummer

Type	Route	Beschrijving
Workflow Collecties		
GET	/workflow-collections	Haal alle collecties op voor de gebruiker
GET	/workflow-collection/{collection} • {collection} = collectie id	Haal specifieke collectie op
POST	/workflow-collection	Maak collectie aan
PUT	/workflow-collection/{collection} • {collection} = collectie id	Update specifieke collectie
DELETE	/workflow-collection/{collection} • {collection} = collectie id	Verwijder specifieke collectie
Workflows		
GET	/builder/workflows	Haal alle workflows van gebruiker op
GET	/builder/workflow/{workflow} • {workflow} = workflow id	Haal specifieke workflow van gebruiker op
POST	/builder/workflow	Maak workflow aan
PUT	/builder/workflow/{workflow} • {workflow} = workflow id	Update specifieke workflow
DELETE	/builder/workflow/{workflow} • {workflow} = workflow id	Verwijder specifieke workflow
POST	/builder/workflow/import	Importeer workflow inclusief alle blokken, nodes en links

Workflow Blocks		
GET	/builder/workflow/{workflow}/blocks • {workflow} = workflow id	Haal alle blokken op van een specifieke workflow
GET	/builder/workflow/{workflow}/block/{block} • {workflow} = workflow id • {block} = block id	Haal een specifiek block op van een workflow
POST	/builder/workflow/{workflow}/block • {workflow} = workflow id	Voeg een block toe aan een workflow
PUT	/builder/workflow/{workflow}/block/{block} • {workflow} = workflow id • {block} = block id	Update een block van een workflow
DELETE	/builder/workflow/{workflow}/block/{block} • {workflow} = workflow id • {block} = block id	Verwijder een block uit een workflow
GET	/builder/workflow/{workflow}/block/{block}/document/{document} • {workflow} = workflow id • {block} = block id • {document} = document id	Haal document op van een workflow block
Workflow Nodes		
GET	/builder/workflow/{workflow}/nodes {workflow} = workflow id	Haal alle nodes op van een specifieke workflow
GET	/builder/workflow/{workflow}/node/{node} • {workflow} = workflow id • {node} = node id	Haal een specifiek node op van een workflow
POST	/builder/workflow/{workflow}/node • {workflow} = workflow id	Voeg een node toe aan een workflow
PUT	/builder/workflow/{workflow}/node/{node} • {workflow} = workflow id • {node} = node id	Update een node van een workflow

DELETE	/builder/workflow/{workflow}/node/{node} • {workflow} = workflow id • {node} = node id	Verwijder een node uit een workflow
Workflow Links		
GET	/builder/workflow/{workflow}/links {workflow} = workflow id	Haal alle links op van een specifieke workflow
GET	/builder/workflow/{workflow}/link/{link} • {workflow} = workflow id • {link} = link id	Haal een specifiek link op van een workflow
POST	/builder/workflow/{workflow}/link • {workflow} = workflow id	Voeg een link toe aan een workflow
PUT	/builder/workflow/{workflow}/link/{link} • {workflow} = workflow id • {link} = link id	Update een link van een workflow
DELETE	/builder/workflow/{workflow}/link/{link} • {workflow} = workflow id • {link} = link id	Verwijder een link uit een workflow
Project Workflows		
GET	/project/{project}/workflows • {project} = project id	Haal alle workflows binnen een project
GET	/project/{project}/workflow/{workflow} • {project} = project id • {workflow} = workflow id	Haal specifieke workflow op binnen een project
POST	/project/{project}/workflow • {project} = project id	Voeg een workflow toe aan een project
PUT	/project/{project}/workflow/{workflow} • {project} = project id • {workflow} = workflow id	Update een workflow binnen het project
DELETE	/project/{project}/workflow/{workflow} • {project} = project id • {workflow} = workflow id	Verwijder een workflow uit het project

GET	/project/{project}/workflows/inLicense • {project} = project id	Haal alle workflows op die binnen de project licentie vallen
Project Workflow Roles		
GET	/project/{project}/workflow/{workflow}/roles • {project} = project id • {workflow} = workflow id	Haal alle rollen op van een specifieke workflow binnen een project
GET	/project/{project}/workflow/{workflow}/role/{role} • {project} = project id • {workflow} = workflow id • {role} = role id	Haal een specifieke rol op van een specifieke workflow binnen een project
POST	/project/{project}/workflow/{workflow}/role • {project} = project id • {workflow} = workflow id	Voeg een rol toe aan een workflow binnen een project
PUT	/project/{project}/workflow/{workflow}/role/{role} • {project} = project id • {workflow} = workflow id • {role} = role id	Update een rol van een workflow binnen een project
DELETE	/project/{project}/workflow/{workflow}/role/{role} • {project} = project id • {workflow} = workflow id • {role} = role id	Verwijder een rol van een workflow binnen een project
GET	/project/{project}/workflow/{workflow}/roles/teams • {project} = project id • {workflow} = workflow id	Haal alle teams op van een project met hun toegekende rollen voor een specifieke workflow
GET	/project/{project}/workflow/{workflow}/roles/nodes • {project} = project id • {workflow} = workflow id	Haal alle nodes op van een workflow met hun toegekende rollen binnen een project
POST	/project/{project}/workflow/{workflow}/role/{role}/team • {project} = project id • {workflow} = workflow id • {role} = role id	Voeg een rol toe aan een team binnen het project voor een specifieke workflow

POST	/project/{project}/workflow/{workflow}/role/{role}/node • {project} = project id • {workflow} = workflow id • {role} = role id	Voeg een rol toe aan een node van een workflow binnen een project
DELETE	/project/{project}/workflow/{workflow}/role/{role}/team/{team} • {project} = project id • {workflow} = workflow id • {role} = role id • {team} = team id	Verwijder een rol van een team binnen het project voor een specifieke workflow
DELETE	/project/{project}/workflow/{workflow}/role/{role}/node/{node} • {project} = project id • {workflow} = workflow id • {role} = role id • {node} = node id	Verwijder een rol van een node binnen een workflow binnen het project
Workflow Sessions		
GET	/project/{project}/sessions • {project} = project id	Haal alle sessies op binnen een project
GET	/project/{project}/session/{session} • {project} = project id • {session} = session id	Haal specifieke sessie op binnen een project
POST	/project/{project}/session • {session} = session id	Start een nieuwe sessie voor een workflow
PUT	/project/{session}/session/{session} • {project} = project id • {session} = session id	Update een workflow sessie
DELETE	/project/{project}/session/{session} • {project} = project id • {session} = session id	Verwijder een workflow sessie uit een project
Workflow Job		
GET	/project/{project}/jobs • {project} = project id	Haal alle jobs op binnen een project
GET	/project/{project}/job/{job} • {project} = project id • {job} = job id	Haal een specifieke job op binnen een project

GET	/project/{project}/job/{job}/node • {project} = project id • {job} = job id	Haal de workflow node op van een job binnen een project
GET	/project/{project}/job/{job}/users • {project} = project id • {job} = job id	Haal alle users op die de job mogen uitvoeren binnen project
POST	/project/{project}/job/{job} • {project} = project id • {job} = job id	Voeg een gebruiker toe aan de job die uitgevoerd moet worden
POST	/project/{project}/job/{job}/finish • {project} = project id • {job} = job id	Sluit de job af door de toegewezen gebruiker
GET	/project/{project}/task/{task}/job • {project} = project id • {task} = task id	Haal alle benodigde informatie op vanuit een taak voor een specifieke job
SBS		
GET	/project/{project}/sbs • {project} = project id	Haal alle sbs codes op binnen een project
GET	/project/{project}/sbs/{sbs} • {project} = project id • {sbs} = sbs id	Haal specifieke sbs code op binnen een project
POST	/project/{project}/sbs • {project} = project id	Voeg een nieuwe sbs code toe binnen een project
PUT	/project/{session}/sbs/{sbs} • {project} = project id • {sbs} = sbs id	Update een specifieke sbs code binnen een project
DELETE	/project/{project}/sbs/{sbs} • {project} = project id • {sbs} = sbs id	Verwijder een sbs code uit een project
GET	/project/{project}/sbs-tree • {project} = project id	Haal alle sbs codes op met een boom structuur

13.2 BIJLAGE 2, WORKFLOW IMPORT JSON

```
{
  "id": 0,
  "name": "ANT Queuer test",
  "description": "ANT Queuer test",
  "prev": [],
  "nodes": [
    {
      "id": 1,
      "type": "block",
      "name": "Input the values",
      "description": "Voer alle waardes in",
      "prev": [],
      "config": {
        "data": "ewoidHlwZTogInF1ZXVlc190ZXN0IiwKImNvbW11bnQiOiAiaG",
        "name": "config",
        "extension": "json"
      }
    },
    {
      "id": 2,
      "type": "block",
      "name": "Calculate",
      "description": "Rekenen!",
      "prev": [1],
      "config": {
        "data": "ewoidHlwZTogInF1ZXVlc190ZXN0IiwKImNvbW11bnQiOiAiaG",
        "name": "config",
        "extension": "json"
      }
    }
  ]
}
```

13.3 BIJLAGE 3, WORKFLOW IMPORT RESPONSE

```

{
  "message": "Workflow ANT Queuer test imported 1 workflows, 2 blocks, 2 nodes and 1 links in 0.66673183441162 seconds.",
  "status": "ok",
  "errors": [],
  "workflow": {
    "name": "ANT Queuer test",
    "description": "ANT Queuer test",
    "created_by": "9473154a-f175-4a39-afb9-0b30d6c59a76",
    "id": "4065764c-119d-4bcb-b651-b0b392960bfb"
  },
  "created": {
    "workflows": [
      {
        "name": "ANT Queuer test",
        "description": "ANT Queuer test",
        "created_by": "9473154a-f175-4a39-afb9-0b30d6c59a76",
        "id": "4065764c-119d-4bcb-b651-b0b392960bfb"
      }
    ],
    "blocks": [
      {
        "id": "fddcc35c-6671-45bd-aab3-4fea1dae0414",
        "name": "Input the values",
        "description": "Altijd belangrijker, en zeer ingewikkeld...",
        "created_by": "9473154a-f175-4a39-afb9-0b30d6c59a76",
        "workflow": "4065764c-119d-4bcb-b651-b0b392960bfb",
        "config": "c53ab465-db1d-4c43-aae8-084defe5c0ba",
        "media": null
      },
      {
        "id": "0b8d1abb-6dc2-414d-9957-0766a1fba7de",
        "name": "Calculate",
        "description": "Rekenen!",
        "created_by": "9473154a-f175-4a39-afb9-0b30d6c59a76",
        "workflow": "4065764c-119d-4bcb-b651-b0b392960bfb",
        "config": "17f24488-9da1-4573-812e-8c48fc713fe2",
        "media": null
      }
    ]
  },
  "links": [
    {
      "id": "fddcc35c-6671-45bd-aab3-4fea1dae0414",
      "name": "Input the values",
      "description": "Altijd belangrijker, en zeer ingewikkeld...",
      "created_by": "9473154a-f175-4a39-afb9-0b30d6c59a76",
      "workflow": "4065764c-119d-4bcb-b651-b0b392960bfb",
      "config": "c53ab465-db1d-4c43-aae8-084defe5c0ba",
      "media": null
    },
    {
      "id": "0b8d1abb-6dc2-414d-9957-0766a1fba7de",
      "name": "Calculate",
      "description": "Rekenen!",
      "created_by": "9473154a-f175-4a39-afb9-0b30d6c59a76",
      "workflow": "4065764c-119d-4bcb-b651-b0b392960bfb",
      "config": "17f24488-9da1-4573-812e-8c48fc713fe2",
      "media": null
    }
  ]
}

```

```
"nodes": [  
  {  
    "belongs_to": "4065764c-119d-4bcb-b651-b0b392960bfb",  
    "block": "fddcc35c-6671-45bd-aab3-4fea1dae0414",  
    "type": "start",  
    "id": "b929b02a-1693-44aa-a338-65e5ba4bfd1d"  
  },  
  {  
    "belongs_to": "4065764c-119d-4bcb-b651-b0b392960bfb",  
    "block": "0b8d1abb-6dc2-414d-9957-0766a1fba7de",  
    "type": null,  
    "id": "5f05207f-0593-4a59-897f-042108891aaf"  
  }  
],  
"links": [  
  {  
    "node": "5f05207f-0593-4a59-897f-042108891aaf",  
    "previous_node": "b929b02a-1693-44aa-a338-65e5ba4bfd1d",  
    "workflow": "4065764c-119d-4bcb-b651-b0b392960bfb",  
    "id": "1c6c2e7d-791d-4bc5-8760-54491436fc4a"  
  }  
]  
}
```

13.4 BIJLAGE 4, MAAK JOBS VOOR ELKE NODE MET BLOK TYPE

```
/**
 * Create job for each node in workflow recursively
 * @param $nodes
 * @param $internalProjectId
 * @param $session
 * @param null $parentJobId
 */
private function createJobsForNodes($nodes, $internalProjectId, $session,
$parentJobId = null)
{
    foreach ($nodes as $node) {
        // check if node is workflow
        if (isset($node->workflow)) {
            $job = $this->workflowJobRepository-
>storeWorkflowJob($internalProjectId, $node, $session, null, $parentJobId);
            $this->createJobsForNodes($this->workflowNodesService-
>getWorkflowNodes($node->workflow), $internalProjectId, $session, $job->id);
        } else {
            $this->workflowJobRepository-
>storeWorkflowJob($internalProjectId, $node, $session, null, $parentJobId);
        }
    }
}
```

13.5 BIJLAGE 5, AFSLUITEN EN OPENEN VAN EEN JOB

```
/**
 * @param $userId
 * @param $projectId
 * @param $job
 * @param $isParentJob
 */
public function finishJobWithTask($userId, $projectId, $job, $isParentJob)
{
    if (!$isParentJob) {
        // finish block job
        $finishedJob = $this->workflowJobRepository->finishJob($projectId,
        $job->id);
        Log::channel('workflow')->info($userId . " - Finished Job : " .
        $finishedJob->id . " in project: " . $projectId);

        // finish task for job
        $this->taskService->finishTaskByJob($userId, $job);

        $this->findNextNodeJobsAndSetOpen($finishedJob, $projectId, $userId);
    } else {
        if (!empty($job->parent)) {
            // get all incomplete child jobs
            $childJobs = $this->workflowJobRepository-
            >getIncompleteChildJobs($job->parent, $projectId);

            // check if there are no incomplete child nodes
            if (count($childJobs) === 0) {
                // get parent job
                $parent = $this->workflowJobRepository-
                >getParentJobById($projectId, $job->parent);

                $this->finishJobWithTask($userId, $projectId, $parent, true);
            } else {
                $this->findNextNodeJobsAndSetOpen($job, $projectId, $userId);
            }
        } else {
            $this->findNextNodeJobsAndSetOpen($job, $projectId, $userId);
        }
    }
}
```

```
private function findNextNodeJobsAndSetOpen($job, $projectId, $userId)
{
    // get next nodes
    $nextNodes = $this->workflowNodesService->getNextNodesInWorkflow($job->node);

    // check if next node exists
    if (count($nextNodes) === 0) {
        if (isset($job->parent)) {
            // get all incomplete child jobs
            $childJobs = $this->workflowJobRepository->getIncompleteChildJobs($job->parent, $projectId);

            // check if there are no incomplete child nodes
            if (count($childJobs) === 0) {
                // get parent job
                $parent = $this->workflowJobRepository->getParentJobById($projectId, $job->parent);

                $this->workflowJobRepository->finishJob($projectId, $parent->id);

                $this->finishJobWithTask($userId, $projectId, $parent, true);
            }
        } else {
            $this->setNextJobStatusToOpen($nextNodes, $projectId, $job->session);
        }
    }
}

private function setNextJobStatusToOpen($nodes, $projectId, $sessionId)
{
    // open all jobs for next nodes
    foreach ($nodes as $node) {
        if (isset($node->workflow)) {
            // find first node of workflow and open job
            return $this->setNextJobStatusToOpen($this->workflowNodesService->getWorkflowStartNodes($node->workflow), $projectId, $sessionId);
        } else {
            // get next jobs from node
            $nextJobs = $this->workflowJobRepository->getPlannedJobsByNodeIdAndSessionId($projectId, $node->id, $sessionId);

            // get incomplete previous jobs
            $incompletePreviousJobs = $this->workflowJobRepository->getIncompletePreviousJobs($projectId, $node->id, $sessionId);

            foreach ($nextJobs as $job) {
                if (count($incompletePreviousJobs) === 0) {
                    $this->openOrAssignJob($projectId, $job->id);
                }
            }
        }
    }
}
```

```
private function openOrAssignJob($projectId, $jobId) {
    $assignableUsers = $this->workflowJobRepository->assignableUsersByJobId($projectId, $jobId);
    if (count($assignableUsers) == 1) {
        $userId = $assignableUsers[0]->user_id;
        // auto assign job
        $this->assignJobToUser($userId, $jobId, $projectId);
        Log::channel('workflow')->info("System - Auto assigned job: $jobId to user: $userId in project: $projectId");
    } else {
        // open job to pick up
        $this->workflowJobRepository->openJob($projectId, $jobId);
        Log::channel('workflow')->info("System - Job opened : $jobId in project: $projectId");
    }
}
```

13.6 BIJLAGE 6, UNIT TESTS VOOR HET VALIDEREN VAN EEN BODY

Hier volgen 3 van de 5 unit tests voor het valideren van de required velden van een request body.

```
/**
 * test when data and keys are empty
 * @runInSeparateProcess
 * @return void
 */
public function testWhenDataAndKeysAreEmpty()
{
    $data = [];
    $required_keys = [];

    $this->assertNotNull($data);
    $this->assertNotNull($required_keys);

    $this->validateRequiredBodyFields($data, $required_keys);
}

/**
 * test when data and keys are empty
 * @runInSeparateProcess
 * @return void
 */
public function testWhenDataIsEqualToKeys()
{
    $data = ['key' => 'value', 'key2' => 'value2'];
    $required_keys = ['key', 'key2'];

    $this->assertNull($this->validateRequiredBodyFields($data,
    $required_keys));
}

/**
 * test when data and keys are empty
 * @runInSeparateProcess
 * @return void
 */
public function testWhenDataNotEqualToKeys()
{
    $data = ['key' => 'value', 'key2' => 'value2'];
    $required_keys = ['key1', 'key2'];

    $this->assertExceptionCode(1612799459, function () use ($data,
    $required_keys) {
        $this->validateRequiredBodyFields($data, $required_keys);
    });
}
```

13.7 BIJLAGE 7, SERVICE REPOSITORY PATTERN

Oude situatie:

```
/**
 * Store the new object and perform necessary actions
 *
 * @param \Illuminate\Http\Request $request
 * @return \Illuminate\Http\JsonResponse
 */
public function store(Request $request)
{
    $this->authorize('store', new Table(
        array_merge(
            $this->getFieldsFromRequest(),
            $this->getAdditionalFields()
        )));

    DB::beginTransaction();
    try {
        $name = strtolower($this->request->input('name'));
        $this->validateTableName($name);

        $internalName = $this->createModelName();
        $itemRow = $this->class::create(
            array_merge(
                $this->getFieldsFromRequest(),
                ['internal_name' => $internalName],
                $this->getAdditionalFields()
            )
        );

        $this->buildModel($internalName, $name);
        DB::commit();

        return response($itemRow, 201);
    } catch (Exception $e) {
        DB::rollback();

        throw new BadRequestHttpException($e->getMessage(),
            null, intval($e->getCode()));
    }
}
```

Nieuwe situatie:

```
/**
 * @param Request $request
 * @return Application|ResponseFactory|JsonResponse|Response
 * @throws AuthorizationException
 * @throws Exception
 */
public function store(Request $request)
{
    $this->authorize('store', new Workflow());

    $data = $this->getFieldsFromRequest();

    $this->validateRequiredColumnNames($data, ['name']);

    $workflow = $this->workflowService->storeWorkflow($this->request->user('api')->id, $data);

    return response($workflow, 201);
}
```

Beide voorbeelden zijn van een create api call. Wat direct opvalt is dat de nieuwe situatie een stuk schoner oogt qua code en leesbaarder is.