AFSTUDEERVERSLAG

Behorend bij de afstudeeropdracht “Release Notes 2.0” bij Topicus Healthcare te Deventer

Student	Jelmer Duzijn
Studentnummer	325409
Hogeschool	Saxion Deventer
Opleiding	HBO-ICT Software Engineering
Afstudeerdocent	Jan Stroet
Bedrijf	Topicus Healthcare BV
Bedrijfsbegeleiders	Rens Toonen & Lisette Heerink
Kwartiel	4.3 & 4.4
Datum	17 juni 2019
Plaats	Deventer

Versiebeheer

Versies gemarkeerd met een ster (*) zijn reviewmomenten.

Versie	Datum	Wijzigingen	Auteur
0.1	18-03-2019	Opzetten template	Jelmer Duzijn
0.2	22-03-2019	Invoegen onderdelen uit Plan van aanpak	Jelmer Duzijn
0.3	25-03-2019	Opzet hoofdstuk 4 Uitvoering	Jelmer Duzijn
0.4	26-03-2019	Toevoegen keuze “Systeemarchitectuur”	Jelmer Duzijn
0.5*	26-03-2019		Jan Stroet
0.6	01-04-2019	Toevoegen Jira koppeling proces	Jelmer Duzijn
0.7	18-04-2019	Toevoegen uiteindelijke architectuur	Jelmer Duzijn
0.8	20-04-2019	Uitbreiden hoofdstuk 4	Jelmer Duzijn
0.9	21-04-2019	Brainstormsessies toevoegen Thema cyclus Bijlagen	
1.0*	22-04-2019		Jan Stroet
1.1	01-05-2019	Jira koppeling bijwerken	Jelmer Duzijn
1.1	08-05-2019	Architectuur bijwerken	Jelmer Duzijn
1.2	20-05-2019	Brainstormsessie Joliene toegevoegd Web Component integratie toegevoegd	Jelmer Duzijn
1.3	21-05-2019		Jan Stroet
1.4	31-05-2019	Themacyclus uitgewerkt Volgorde afbeeldingen tekst bij architectuurvoorstellen	Jelmer Duzijn
1.5	01-06-2019	Scopeverkleining verklaard Opnemen leeswijzer	Jelmer Duzijn
1.6*	05-06-2019		Jan Stroet Rogier Hommels
1.7	06-06-2019	Spelfouten verbeteren Toevoegen onderdeel Testen	Jelmer Duzijn
1.8*	12-06-2019		Rens Toonen Lisette Heerink
1.9	12-06-2019	Verwerken reviewpunten Brainstormsessies verplaatst naar de bijlagen Feedback uit onderzoek gehaald Instructies schrijven implementatie Invulling technische inhoud van de opdracht Schrijven conclusie en aanbevelingen	Jelmer Duzijn
2.0	13-06-2019	Samenvatting Opnemen technische invulling Bijschriften figuren en tabellen bijwerken	Jelmer Duzijn

SAMENVATTING

De ontwikkelteams van Topicus HAP beschikken niet over de mogelijkheid om hun eindgebruikers direct te kunnen informeren over nieuwe functionaliteiten en belangrijke functionele wijzigingen van het software pakket bij een nieuwe release. Ook is het voor hen momenteel niet mogelijk om feedback te ontvangen op deze wijzigingen. Daarnaast kost het opstellen van de release notes in de huidige vorm veel tijd.

Het doel van het project is om de teams van Topicus HAP in staat te stellen hun release notes bij elke release bij de juiste eindgebruiker te krijgen. Naast het gericht kunnen verspreiden van de release notes moet het ook mogelijk worden om eindgebruikers op eenvoudige wijze feedback te kunnen laten geven op nieuwe en gewijzigde functionaliteiten. Deze feedback moet ook inzichtelijk zijn voor de teams. Hierbij is de volgende hoofdvraag geformuleerd: “*Hoe kan het ontwikkelen van een webapplicatie de ontwikkelteams van HAP helpen om eindgebruikers beter te kunnen informeren over veranderingen in Topicus HAP?*”

Om antwoord te kunnen geven op deze vraag zijn gesprekken gevoerd met teamleden om een inventarisatie te maken van de benodigde functionaliteiten en in te zetten technieken zodat zij een set release notes en gebruikersinstructies bij een nieuwe release te kunnen verwerken, publiceren en presenteren.

Op basis van deze analyse is met inzet van onderdelen uit de Scrum methodiek een werkend Proof of Concept ontwikkeld. Het Proof of Concept bestaat uit drie onderdelen: een beheerapplicatie, een inzageapplicatie en een back-end applicatie. De back-end applicatie maakt verbinding met een extern systeem, Jira, om release notes op te halen die door de teams zijn geschreven. De beheerapplicatie biedt de mogelijkheid voor de teams om gebruikersinstructies te schrijven, voorzien van afbeeldingen en instructievideo’s. Bij het uitbrengen van een nieuwe versie van Topicus HAP kunnen zij via de beheerapplicatie release notes en gebruikersinstructies publiceren. De inzageapplicatie is het laatste onderdeel van het Proof of Concept. Dit is een in Topicus HAP geïntegreerde applicatie waarmee eindgebruikers van Topicus HAP de publicaties kunnen inzien, waardoor zij zich kunnen informeren over functionele wijzigingen.

Vanwege de complexiteit van de opdracht en de beperkte tijd van de afstudeerperiode is in overleg besloten de scope van de opdracht bij te stellen. Hierdoor is het onderdeel feedback, genoemd in de doelstelling, komen te vervallen ten gunste van onderwerpen met meer prioriteit: presenteren van release notes in Topicus HAP en de koppeling met Jira.

Met het afronden van de afstudeeropdracht beschikken de teams over een Proof of Concept waarmee aangetoond wordt dat zij release notes en gebruikersinstructies op een vernieuwde manier kunnen uitleveren aan de eindgebruikers van Topicus HAP.

Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

INHOUDSOPGAVE

INLEIDING

Bij het uitleveren van een nieuwe versie van een softwarepakket horen release notes. Een document waarin staat beschreven welke vernieuwingen en wijzigingen er zijn gemaakt aan het product en welke problemen met de software zijn verholpen in de nieuwe versie. Dit wordt enerzijds gedaan om de klant en de eindgebruikers in te lichten over de wijzigingen, zodat zij weten wat ze kunnen verwachten bij het gebruiken van de software. Anderzijds is het een manier om hun te laten zien dat het product wordt verbeterd, zodat eindgebruikers er meer binding mee krijgen.

Dit geldt ook voor Topicus Healthcare waar onder andere gewerkt wordt aan Topicus HAP. Een softwarepakket waarmee werknemers van een huisartsenpost worden ondersteund in hun werkzaamheden. Bij iedere nieuwe versie die zij uitleveren worden gedetailleerde release notes geschreven om hun eindgebruikers zo goed mogelijk in te lichten over de verbeteringen die gemaakt zijn aan het product.

Echter blijkt uit gesprekken met werknemers van Topicus Healthcare dat deze release notes, en daarmee de kennis over nieuwe en gewijzigde functionaliteiten, de eindgebruikers niet altijd bereiken (Lisette Heerink, Hubert Flisijn, persoonlijke communicatie, 16 januari 2019). Dit heeft als gevolg dat eindgebruikers de wijzigingen pas opmerken wanneer zij de applicatie weer openen, wat voor onduidelijkheid kan zorgen en de werkzaamheden kan vertragen. De huisarts binnen een huisartsenpost rouleert en maakt daarmee minder gebruik van Topicus HAP dan de vaste medewerkers. Hierdoor is de arts minder behendig met het systeem en zal hij of zij minder goed bij kunnen houden wat nieuw is in Topicus HAP en wat er veranderd is.

Tijdens mijn afstudeerperiode in het laatste half jaar van de opleiding HBO-ICT met uitstroomprofiel Software Engineering, heb ik onderzoek gedaan met de hoofdvraag “*Hoe kan het ontwikkelen van een webapplicatie de ontwikkelteams van HAP helpen om eindgebruikers beter te kunnen informeren over veranderingen in Topicus HAP?”.* Op basis van de onderzoeksresultaten heb ik een Proof of Concept ontwikkeld in de vorm van een webapplicatie, waarmee de teams die Topicus HAP ontwikkelen hun release notes direct aan de eindgebruiker kunnen communiceren.

Dit afstudeerverslag beschrijft het verloop en de procesmatige uitvoering van de afstudeeropdracht “Release Notes 2.0”. Daarnaast wordt ingegaan op de totstandkoming van de inhoud van de afstudeeropdracht en de gemaakte keuzes tijdens het afstuderen.

LEESWIJZER

Dit verslag bestaat uit zes hoofdstukken en een selectie aan bijlagen.

Hoofdstuk één gaat in de op de context van de opdracht en de aanleiding hiervoor. Hierin wordt gekeken naar de productcontext, de aanleiding voor de opdracht en de huidige release notes situatie van Topicus HAP.

Hoofdstuk twee vertelt over de opdrachtgever Topicus en de unit Topicus Healthcare.

Hoofdstuk drie beschrijft de opdracht. Er wordt ingegaan op de volgende onderwerpen: de probleemstelling, de doelstelling, de projectgrenzen, het onderzoek gekoppeld aan de opdracht en het Proof of Concept.

Hoofdstuk vier beschrijft de uitvoering van de opdracht. Hierbij wordt gekeken naar het tot stand komen van de inhoud van de opdracht, de functionele en technische invulling van de opdracht en hoe het Proof of Concept is ontwikkeld.

Hoofdstuk vijf beschrijft de voornaamste keuzes die tijdens het ontwikkelen van het Proof of Concept zijn gemaakt. Hierbij komen de volgende onderwerpen aan bod: systeemarchitectuur, de koppeling met Jira, het verkleinen van de scope van de opdracht, het schrijven van instructies met een Rich Text Editor en de integratie van een Web Component in een bestaande applicatie.

In hoofdstuk zes is de conclusie te lezen en zijn aanbevelingen opgenomen om het Proof of Concept te verbeteren en door te kunnen ontwikkelen.

De bijlagen bestaan uit: requirements, verantwoordingen van de brainstormsessies, dashboard mockups en een sprintverslag van sprint 3.

BEGRIPPENLIJST

Apache Wicket – Een framework voor het bouwen van webapplicaties in Java. Hierbij worden pagina’s op de server opgebouwd en naar de browser verzonden.

Bitbucket – Een versiebeheersysteem voor broncode op basis van Git, ontwikkeld door de Australische firma Atlassian. Onderdeel van het pakket tools die door Topicus worden gebruikt.

Confluence – Een webapplicatie ontwikkeld door de Australische firma Atlassian. Dit pakket stelt bedrijven in staat gestructureerd te kunnen documenteren, kennis te delen en samen te werken. Onderdeel van het pakket tools die door Topicus worden gebruikt.

Eindgebruiker – De persoon die gebruik maakt van de software, bijvoorbeeld de triagist of huisarts.

Huisartsenpost (HAP) – Een plek waar patiënten buiten de openingstijden van de reguliere huisartsenpraktijk terecht kunnen voor medische (spoedeisende) hulp.

Jira – Een webapplicatie ontwikkeld door de Australische firma Atlassian. Jira is een pakket waarmee het ontwikkelproces wordt ondersteund en agile projecten worden bewaakt. Onderdeel van het pakket tools die door Topicus worden gebruikt.

Monoliet – Een applicatie die niet modulair is opgebouwd, maar één groot geheel is. Het is een architectuurprincipe waarbij één applicatie verantwoordelijk is voor alle taken die het systeem moet uitvoeren.

Proof of Concept (PoC) – Een methode om te demonstreren of een idee of concept haalbaar is en een oplossing biedt voor een probleem door het ontwikkelen van een testproduct.

Release notes – Schriftelijke samenvatting van de wijzigingen die zijn gedaan aan een softwareproduct bij een nieuwe release. De release notes van Topicus HAP bevatten ook screenshots en instructievideo’s.

Requirements – Functionele en technische eisen waaraan een softwareproduct moet voldoen en waarop getoetst kan worden of het product naar wens is opgeleverd.

Topicus HAP – Een softwarepakket van Topicus Healthcare waarmee werkprocessen op een huisartsenpost ondersteund worden.

Triagist – Een gespecialiseerde doktersassistent voor de spoedeisende hulp op de huisartsenpost. De triagist beoordeelt in korte tijd de urgentie van de klachten wanneer een patiënt contact opneemt met de huisartsenpost. Daarnaast bepaalt de triagist het vervolgtraject (Nederlandse Vereniging van Doktersassistenten, 2019).

User story – Een in tekst uitgedrukte gewenste functionaliteit van een softwareproduct vanuit het oogpunt van de eindgebruiker. De user story beschrijft een functionaliteit die door een softwareontwikkelaar kan worden gebouwd.

Web Component – Een ingekapseld custom HTML element met een template, styling en javascript dat geen onderdeel uitmaakt van de HTML standaardtaal. In plaats daarvan wordt het element door een ontwikkelaar gedefinieerd en aan de DOM toegevoegd.

1. CONTEXT EN AANLEIDING

In dit hoofdstuk wordt de context waarbinnen de opdracht valt toegelicht en wordt de aanleiding tot het vormen van deze opdracht beschreven.

1.1 PRODUCTCONTEXT

Topicus HAP is een softwarepakket en totaaloplossing voor de huisartsenpost. Met dit pakket kunnen triagisten, verpleegkundigen, baliemedewerkers, huisartsen en administratief medewerkers volledig worden ondersteund in de werkprocessen op de huisartsenpost.

Door het invullen van de klachten van de patiënt in Topicus HAP en het beantwoorden van een aantal vragen kan door het systeem een urgentie worden vastgesteld. Topicus HAP biedt een triagist hiermee ondersteuning bij het opstellen van een toestandsbeeld wanneer een patiënt contact opneemt met de huisartsenpost. Naar aanleiding van de triage wordt door de triagist een actie geselecteerd en geadministreerd in Topicus HAP. Deze actie is bijvoorbeeld een telefonisch consult of een consult op de huisartsenpost (A. Schepen, persoonlijke communicatie, 21 februari 2019).

Een patiënt wordt bij binnenkomst op de huisartsenpost door een baliemedewerker binnen gemeld. Vanaf dat moment kan de patiënt door de huisarts worden opgeroepen vanuit de wachtkamer. Tijdens het consult registreert de huisarts informatie over dit contact in Topicus HAP.

1.2 AANLEIDING

Topicus HAP ondersteunt alle werkprocessen op de huisartsenpost en biedt daarmee een groot aantal functionaliteiten. Er wordt continu gewerkt aan het verbeteren van Topicus HAP om de eindgebruiker zo goed mogelijk van dienst te zijn. Bij iedere release worden release notes opgeleverd, waarin staat beschreven welke nieuwe functionaliteiten of functionele wijzigingen er zijn gemaakt in Topicus HAP.

Het blijkt uit gesprekken met werknemers van Topicus Healthcare dat deze release notes, en daarmee de kennis van de nieuwe of gewijzigde functionaliteiten, de eindgebruiker niet altijd bereiken. De verwachting is dat de eindgebruikers door het management worden ingelicht over nieuwe functionaliteiten of veranderingen in Topicus HAP, maar dit gebeurt niet altijd (Lisette Heerink, Hubert Flisijn, persoonlijke communicatie, 16 januari 2019). Het komt dus voor dat de eindgebruiker de wijzigingen pas opmerkt wanneer hij of zij de applicatie weer opent, wat voor onduidelijkheid kan zorgen en de werkzaamheden kan vertragen. De huisarts binnen een huisartsenpost rouleert en maakt daarmee minder gebruik van Topicus HAP dan de vaste medewerkers. Hierdoor is de arts minder behendig met het systeem en zal hij of zij minder goed bij kunnen houden wat nieuw is in Topicus HAP en wat er veranderd is.

Bij alle huisartsenposten wordt met het management en een werkgroep bepaald hoe Topicus HAP wordt geconfigureerd. Deze werkgroep bestaat uit triagisten, artsen, financieel medewerkers en de applicatiebeheerder van de HAP. Deze configuratie wordt gedaan om het systeem af te meten op het werkproces van de HAP, zodat het systeem aansluit bij de werkwijze die de huisartsenpost hanteert (Rens Toonen, persoonlijke communicatie, 15 februari 2019). Wanneer met het configureren bepaalde functionaliteiten worden uitgeschakeld betekent het dat eindgebruikers bepaalde functionaliteiten niet kunnen gebruiken terwijl ze daar wel baat bij hebben. Ook komt het voor dat het management door bijvoorbeeld tijdgebrek nieuwe functionaliteit niet activeert (A. Schepen, persoonlijke communicatie, 21 februari 2019).

Daarnaast is het momenteel niet mogelijk om eenvoudig feedback te verzamelen van eindgebruikers over de features die in de release notes staan beschreven. Deze feedback is waardevol voor de doorontwikkeling van Topicus HAP.

Als laatste kost het opstellen van de release notes in de huidige vorm veel tijd (Lisette Heerink, persoonlijk communicatie, 15 februari 2019). De release notes worden gedetailleerd uitgewerkt en waar nodig aangevuld met schermafbeeldingen of instructievideo’s om de eindgebruikers goed te informeren.

Voor de genoemde problemen zal er een nieuwe oplossing ontwikkeld moeten worden. Deze oplossing richt zich op het gericht verspreiden van de release notes en het verzamelen van eindgebruikers feedback over de wijzigingen die zijn beschreven in de release notes.

1.3 HUIDIGE SITUATIE RELEASE NOTES

De huidige vorm van het opstellen en uitleveren van release notes bestaat uit één pagina per release, op een voor klanten toegankelijke Confluence omgeving. Een link naar deze pagina wordt per mail naar de beheerder van de huisartsenpost gestuurd. De naamgeving van de pagina is ‘Release notes THAP’ gevolgd door het versienummer van de release.

De indeling van de release notes pagina’s is grotendeels gelijk en de pagina’s bevatten tenminste de volgende onderdelen:

- Versiebeheer
- Inhoudsopgave
- Managementsamenvatting
- Kop per feature met uitleg en één of meerdere afbeeldingen en in sommige gevallen een instructievideo.

Ook zijn er twee kopjes opgenomen met tabellen waarin de Jira issues staan die in de release zijn opgeleverd. Deze kopjes zijn:

- Wensen – Issues met betrekking tot klantwensen. Dit kunnen wensen zijn van een specifieke huisartsenpost, maar ook wensen vanuit de productontwikkeling van Topicus om HAP te verbeteren.
- Problemen – Issues met betrekking tot problemen met THAP. Dit zijn problemen (bugs) met THAP die zich voordoen bij één of meerdere huisartsenposten.

Uit deze indeling en uit gesprekken met Joliene Brouwer is gebleken dat de release notes die bij een release worden verspreid bestaan uit twee onderdelen: instructies en release notes (persoonlijke communicatie, 14 maart 2019).

Instructies

De instructies zijn bedoeld om eindgebruikers te informeren over functionele wijzigingen in Topicus HAP. Deze zijn gedetailleerd geschreven met opgemaakte tekst en voorzien van afbeeldingen en/of instructievideo’s. Ze geven stapsgewijs aan hoe een eindgebruiker een bepaalde actie moet voltooien in de nieuwe release zodat deze niet voor verrassingen komt te staan wanneer Topicus HAP wordt geopend na een nieuwe release.

Release notes

Dit zijn de stukken tekst die na afronding van een issue in Jira worden geschreven in het issue zelf. Het is een beknopte beschrijving van de functionele wijzigingen. Deze release notes worden in een tabel weergegeven op de Confluence pagina voor de klanten.

2. OPDRACHTGEVER

In dit hoofdstuk wordt uitleg gegeven over het bedrijf en de divisie waar de afstudeeropdracht wordt uitgevoerd.

2.1 TOPICUS

Topicus is een softwarebedrijf uit Deventer met een informele bedrijfscultuur dat zich richt op zes vlakken uit de maatschappij: zorg, financiën, onderwijs, overheid, recht en beveiliging. Ieder vlak wordt bedient door een aparte divisie.

Het bedrijf specialiseert zich in keten integratie en SaaS (Software as a Service), dat houdt in dat Topicus met haar producten de schakel wil zijn tussen verschillende bedrijfsprocessen binnen de zes vlakken. Met haar producten probeert Topicus een zinnige impact te hebben op de maatschappij en de burger te ondersteunen in eigen zeggenschap (Topicus, 2019).

Missie

Topicus wil met producten en diensten mensen, platforms en data met elkaar verbinden. Om zo, naar eigen zeggen, “de IT-eilandjes af te breken”. Hierin wordt de burger vooropgesteld, zodat deze inzicht krijgt in zijn situatie op gebied van finance, zorg, onderwijs en de overheid. Dit vergroot de zelfredzaamheid en maakt de burger minder kwetsbaar (Topicus, 2019).

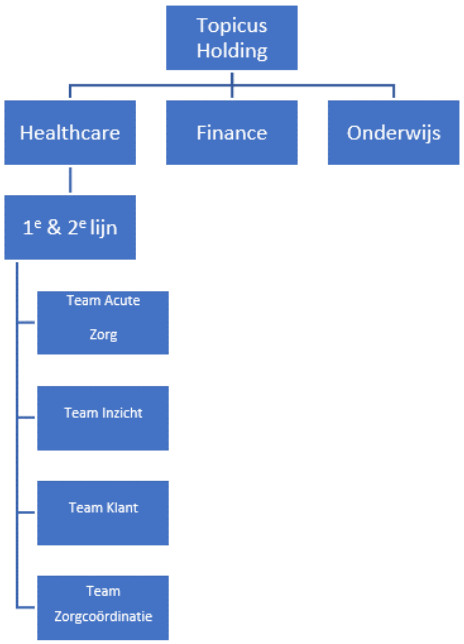
2.2 TOPICUS HEALTHCARE

Topicus Healthcare specialiseert zich in het ontwikkelen van software voor huisartsen, spoedzorg, publieke gezondheid en verzekeraars.

De afstudeeropdracht zal worden uitgevoerd bij de Business Line 1e en 2e lijn in het team Acute Zorg. Op deze afdeling werken vijf teams aan Topicus HAP.

- Team Acute Zorg bestaat uit drie teamleden.
- Team Inzicht bestaat uit drie teamleden.
- Team Klant bestaat uit zeven teamleden.
- Team Zorgcoördinatie bestaat uit zeven teamleden.

In Figuur 1 is een vereenvoudigde organisatiestructuur van de Business Line afgebeeld.



Figuur 1 Vereenvoudigde organisatiestructuur Business Line 1e & 2e lijn

3. OPDRACHTOMSCHRIJVING

Dit hoofdstuk beschrijft de inhoud van de opdracht “Release Notes 2.0”. In dit hoofdstuk komt het volgende aan bod: de probleemstelling, de doelstelling, de projectgrenzen, de inhoud van het onderzoek en de eisen aan het Proof of Concept.

3.1 PROBLEEMSTELLING

De ontwikkelteams van Topicus HAP beschikken niet over de mogelijkheid om hun eindgebruikers direct te kunnen informeren over nieuwe functionaliteiten en belangrijke functionele wijzigingen van het software pakket bij een nieuwe release. Ook is het voor hen momenteel niet mogelijk om feedback te ontvangen op deze wijzigingen. Daarnaast kost het opstellen van de release notes in de huidige vorm veel tijd (Lisette Heerink, persoonlijk communicatie, 15 februari 2019).

3.2 DOELSTELLING

Het doel van het project is om de teams van Topicus HAP in staat te stellen hun release notes bij elke release bij de juiste eindgebruiker te krijgen. Naast het gericht kunnen verspreiden van de release notes moet het ook mogelijk worden om eindgebruikers op eenvoudige wijze feedback te kunnen laten geven op nieuwe en gewijzigde functionaliteiten. Deze feedback moet ook inzichtelijk zijn voor de teams.

Dit doel wordt bereikt door een intern onderzoek uit te voeren om de functionele en technische requirements in kaart te brengen en op basis hiervan een webapplicatie te ontwikkelen als Proof of Concept.

3.3 PROJECTGRENZEN

Om het project af te kunnen bakenen is het belangrijk dat er projectgrenzen worden opgesteld. Ofwel zaken die niet worden opgepakt tijdens de afstudeerperiode. Deze projectgrenzen zijn opgesteld op basis van gesprekken met de bedrijfsbegeleiders.

- Het doel van het Proof of Concept met betrekking tot release notes en feedback is om de teams van Topicus HAP in staat te stellen hun eindgebruikers te informeren. Andere teams binnen Topicus Healthcare vallen buiten de scope van het project.
- Het ontwerp van de applicatie wordt afgemeten op het probleem van de Topicus HAP teams. Andere producten binnen Topicus Healthcare worden niet meegenomen in de oplossing.
- De back-end van het Proof of Concept wordt ontwikkeld in Java en/of Kotlin vanwege de kennis van de ontwikkelteams en de wens om te kunnen doorontwikkelen.
- De verantwoordelijkheid voor het schrijven van release notes ligt bij de Topicus HAP teams.
- De afstudeerperiode begint op **maandag 11 februari 2019** en eindigt op **vrijdag 12 juli 2019**.

3.4 HET ONDERZOEK

Het doel van het onderzoek is om de inhoud van de functionele en technische thema’s van het Proof of Concept vast te kunnen stellen. Het resultaat van het onderzoek zijn antwoorden op de gestelde vragen. Op basis van deze antwoorden is een functioneel en technisch ontwerp gemaakt. Een uitgebreide onderzoeksopzet waarin het doel van de vragen en de gebruikte methoden worden beschreven is opgenomen in het plan van aanpak.

3.4.1 Hoofdvraag

Hoe kan het ontwikkelen van een webapplicatie de ontwikkelteams van HAP helpen om eindgebruikers beter te kunnen informeren over veranderingen in Topicus HAP?

3.4.2 Deelvragen

Hieronder zijn de deelvragen opgenomen. Deelvragen 1 en 2 zijn functionele vragen. Deelvraag 3 is een technische vraag. De uitwerkingen van deze vragen zijn te lezen in hoofdstukken 4.2, 4.3 en 5.4.

1. ***Welke functionaliteiten moet het beheerpaneel in de webapplicatie bevatten om een set release notes te kunnen verwerken?***

2. ***Hoe kan de eindgebruiker vanuit Topicus HAP de release notes benaderen?***

3. ***Met welke frameworks wordt de webapplicatie ontwikkeld?***

In het plan van aanpak is te lezen dat een deel van het onderzoek is gewijd aan het verzamelen van feedback. Echter is gedurende het afstudeertraject de scope van de opdracht iets verkleind en is besloten dit onderdeel achterwege te laten. Dit wordt beschreven in hoofdstuk 5.3 van dit verslag.

3.5 HET PROOF OF CONCEPT

Het Proof of Concept is een werkende webapplicatie, die een oplossing biedt voor het probleem zoals beschreven in hoofdstuk 3.1 Probleemstelling. Het functioneel en technisch ontwerp die uit de onderzoeksresultaten voortkomen vormen de basis voor het Proof of Concept.

Eisen aan het Proof of Concept

Aan het Proof of Concept zijn vanuit de teams enkele eisen gesteld (Rens Toonen, persoonlijke communicatie, 15 februari 2019):

- Het Proof of Concept is een webapplicatie.

Bij Topicus Healthcare worden voornamelijk webapplicaties gebouwd. De aanwezig kennis is direct toepasbaar bij het uitvoeren van de opdracht en bij de doorontwikkeling.

- De applicatie dient een opzichzelfstaande oplossing te zijn.

Topicus Healthcare wil vernieuwen in het technisch ontwerp van de applicaties. Topicus HAP is een monoliet die niet meer met grote modules moet worden uitgebreid. In plaats daarvan wordt er vanuit de architectuur gestuurd op kleinere, onafhankelijke blokken. Daarnaast heeft het als voordeel dat de applicatie mogelijk in de toekomst ingezet kan worden voor andere producten.

- De applicatie wordt onder andere ontwikkeld met technieken die door de ontwikkelteams te begrijpen zijn ten behoeve van doorontwikkeling.

De ontwikkelteams hebben aangegeven de ontwikkeling van het Proof of Concept door te willen zetten. Het is daarom wenselijk om de applicatie te bouwen met technieken die begrepen worden door de teams. Dat wil zeggen dat de applicatie bijvoorbeeld niet wordt gebouwd in ASP.NET of Node.js omdat de teams hier niet mee bekend zijn. Echter staat deze eis geen vernieuwingen in de weg.

- De applicatie is zowel functioneel als technisch gedocumenteerd ten behoeve van doorontwikkeling.

Vanwege de wens om de ontwikkeling te kunnen doorzetten is het van belang een oplossing te bouwen die is voorzien van documentatie.

Functionele thema’s

In Tabel 1 zijn de functionele thema’s van de webapplicatie opgenomen die op basis van de onderzoeksresultaten worden verfijnd. De eerste kolom beschrijft het functionele thema van de webapplicatie. De tweede kolom geeft aan met welke deelvraag het thema verder kan worden uitgewerkt tot requirements.

Tabel 1 Koppeltabel functionele thema's en deelvragen

Thema	Deelvraag
Invoeren instructies	1. *Welke functionaliteiten moet het beheerpaneel in de webapplicatie bevatten om een set release notes te kunnen verwerken?*
Koppelen met bestaande bron release notes	
	2. *Hoe kan de eindgebruiker vanuit Topicus HAP de release notes benaderen?*
Verwijzen naar gepubliceerde releases vanuit Topicus HAP	

4. DE UITVOERING

Tijdens de afstudeerperiode worden een aantal fases doorlopen met ieder hun eigen doel en resultaat. De invulling van deze fases is op basis van de vragen uit het onderzoek zoals beschreven in hoofdstuk 3.4 Het onderzoek. De fases zijn:

- De totstandkoming van de inhoud van de opdracht

In deze oriënterende fase is het probleem verkend en is samen met de afstudeerbegeleiders bepaald welke onderdelen de opdracht moet bevatten om een oplossing te kunnen bouwen die een antwoord biedt voor de probleemstelling zoals beschreven in hoofdstuk 3.1 Probleemstelling. Het resultaat van deze fase is het plan van aanpak.

- Invulling functionele inhoud van de opdracht

Om tot een goed functioneel ontwerp te komen van het Proof of Concept is het belangrijk dat de functionele wensen met betrekking tot release notes van de Topicus HAP teams in kaart worden gebracht. Daarnaast is het van belang om het ontwerp op basis van deze wensen te toetsen bij de teams om te valideren of het overeenkomt met de wensen. Hiervoor worden brainstormsessies gehouden met UI/UX-designer Lisette Heerink, product owner van team Acute Zorg Joliene Brouwer en Tim Dieperink UI-designer bij team Zorgcoördinatie.

Ook worden productdemo’s georganiseerd waar de teamleden de voortgang van de ontwikkeling van het Proof of Concept gepresenteerd krijgen, deze momenten worden ook gebruikt voor het verzamelen van feedback.

- Invulling technische inhoud van de opdracht

De invulling van de technische inhoud van de opdracht bestaat uit een technisch ontwerp. Het technisch ontwerp wordt gedurende de afstudeerperiode wordt uitgebreid met gemaakte technische keuzes en ontwerpen van het Proof of Concept. Het doel van dit document is om de technische werking van de applicatie vast te leggen door uitleg te geven over de gebruikte technieken, de systeem- en softwarearchitectuur en gemaakte keuzes. Dit ontwerp wordt periodiek getoetst bij ontwikkelaar en begeleider Rens Toonen om te valideren of het aan de wensen voldoet en waar nodig aangepast. Ook worden andere ontwikkelaars van Topicus HAP geraadpleegd voor het verzamelen van input over te gebruiken technieken.

- Ontwikkeling van het Proof of Concept

De afstudeerperiode bestaat voor het grootste gedeelte uit het ontwikkelen van het Proof of Concept op basis van de gemaakte ontwerpen. Het Proof of Concept wordt volgens de Scrum methodiek ontwikkeld, zoals beschreven in het plan van aanpak. Het verloop van de ontwikkeling en het proces wordt beschreven in hoofdstuk 4.4.

4.1 TOTSTANDKOMING VAN DE INHOUD VAN DE OPDRACHT

Op maandag 12 februari, de eerste dag van de afstudeerperiode, was een gesprek gepland met begeleiders Lisette Heerink en Rens Toonen om de opdracht door te spreken. Op basis van dit gesprek is een eerste versie van het plan van aanpak gemaakt.

In deze eerste versie bestond de opdracht uit een groot onderzoek naar de informatiebehoeften van de eindgebruikers met betrekking tot release notes en de werkwijze voor het opstellen van release notes door de teams van Topicus HAP.

Deze invalshoek had als gevolg dat het onderzoek geen directe invulling kon geven aan de ontwikkeling van een applicatie die een oplossing zou bieden voor het probleem van de afstudeeropdracht. Dit zou betekenen dat naast het achtergrondonderzoek een tweede onderzoek moet worden gedaan om tot een functioneel en technisch ontwerp te kunnen komen. Daarnaast zou de afhankelijkheid van eindgebruikers voor het onderzoek voor veel vertraging kunnen zorgen.

In het eerste gesprek gaf schoolbegeleider Jan Stroet aan dat de afstudeeropdracht voor ongeveer driekwart uit ontwikkelen zou moeten bestaan (persoonlijke communicatie, 26 februari 2019). Op basis van dit gesprek is een herziene versie van het plan van aanpak geschreven waarbij de invalshoek van het onderzoek is aangepast. Deze versie laat interviews met de eindgebruiker buiten beschouwing en richt zich op functionele en technische vragen die de basis vormen voor het functioneel en technisch ontwerp. De wensen van de eindgebruiker over de presentatie van de release notes worden bepaald op basis van de kennis die aanwezig is binnen de teams en mag voor de opdracht als waarheid worden beschouwd (Lisette Heerink, persoonlijke communicatie, 4 maart 2019).

Gedurende de uitvoering van de opdracht zal tijdens de voortgangsgesprekken worden bepaald of er nog onderdelen aan de opdracht worden toegevoegd. Mocht het zo zijn dat bepaalde thema’s groter uitvallen dan verwacht is er de mogelijkheid de scope iets bij te stellen op basis van de prioriteit die aan de thema’s is toegekend.

4.2 INVULLEN FUNCTIONELE INHOUD VAN DE OPDRACHT

In dit hoofdstuk worden de functionele deelvragen beantwoord. Om tot deze antwoorden te komen zijn brainstormsessies ingepland met verschillende teamleden van Topicus HAP. De resultaten van deze sessies zijn in dit hoofdstuk opgenomen. De uitkomsten van de brainstormsessies wordt gebruikt bij het opstellen van de user stories. De volledige verantwoording van de brainstormsessies is te vinden in Bijlage 2 – Brainstormsessies.

4.2.1 Brainstormsessie: inhoud beheerpaneel

Deze eerste brainstormsessie is gehouden op donderdag 7 maart 2019 met Lisette Heerink, UI/UX designer bij Team Zorgcoördinatie.

Het doel van deze sessie was om een begin te maken aan het antwoord op de eerste vraag: ***Welke functionaliteiten moet het beheerpaneel in de webapplicatie bevatten om een set release notes te kunnen verwerken?***

Het beheerpaneel is de webapplicatie waarmee de teamleden van Topicus HAP hun release notes kunnen verwerken.

Het resultaat van de sessie is dat het beheerpaneel de volgende functionaliteiten moet bevatten:

- **Aanmaken/bewerken release**

Een teamlid van Topicus HAP moet in het beheerpaneel een release van Topicus HAP kunnen aanmaken en bewerken zodat hieraan release notes kunnen worden toegevoegd.

- **Weergeven releases**

De aangemaakte releases moeten inzichtelijk zijn in het beheerpaneel.

- **Aanmaken/bewerken release notes**

De teamleden moeten de mogelijkheid hebben om in het beheerpaneel hun release notes te schrijven en te bewerken bij een release.

- **Weergeven release notes**

De aangemaakte release notes moeten inzichtelijk zijn in het beheerpaneel.

- **Categoriseren van release notes**

De teamleden moeten de mogelijkheid hebben om de release notes te verdelen in de categorieën: triage (bestemd voor triagisten), medicatie (bestemd voor huisartsen), beheer (bestemd voor beheerders van Topicus HAP).

- **Verwijderen van een release/release note**

De aangemaakte releases en release notes moeten verwijderd kunnen worden.

4.2.2 Brainstormsessie: verwerken release notes en instructies

Deze tweede brainstormsessie is gehouden op donderdag 14 maart 2019 met Joliene Brouwer, product owner bij Team Acute Zorg. Joliene schrijft release notes voor Topicus HAP en heeft daarmee inhoudelijke kennis over dit proces. Met haar inzicht wordt het mogelijk om de functionaliteiten die het beheerpaneel moet bevatten verder te definiëren.

Het doel van deze sessie was enerzijds om de resultaten van de eerste brainstormsessie te valideren. Anderzijds om tot een definitief antwoord te komen op de eerste vraag: ***Welke functionaliteiten moet het beheerpaneel in de webapplicatie bevatten om een set release notes te kunnen verwerken?***

Conclusie

Op basis van de uitkomsten van dit gesprek moet een deel van de schermontwerpen en geplande functionaliteiten worden aangepast. De beheerapplicatie was een goed idee, maar de invulling hiervan moet worden aangepast.

Er moet in de release note applicatie worden onderscheid gemaakt tussen release notes en instructies.

Release notes

Release notes zijn onderdeel van een issue en bevinden zich in Jira en moeten daar blijven. Om deze release notes aan eindgebruikers te kunnen presenteren zal er een koppeling moeten worden gelegd met Jira. Release notes worden onderverdeeld in twee categorieën: wensen en opgeloste problemen. Het schermontwerp voor release notes uit de eerste brainstormsessie is daarmee aangepast.

Het is nu niet meer mogelijk om release notes aan te maken. Deze worden rechtstreeks uit Jira gehaald en gepresenteerd in de beheerapplicatie.

Instructies

Instructies zijn handgeschreven stukken tekst, soms voorzien van afbeeldingen of YouTube video’s die een gebruiker uitleg geven over hoe bepaalde functionaliteiten in HAP werken. Deze instructies moeten over de tijd bewerkt kunnen worden totdat ze gepubliceerd worden. Daarnaast worden instructies gegroepeerd op een functioneel thema. De instructies worden geschreven in een tekstverwerker.

4.3 INVULLING TECHNISCHE INHOUD VAN DE OPDRACHT

In dit hoofdstuk worden antwoord gegeven op de vraag: ***Met welke frameworks wordt de webapplicatie ontwikkeld?***

Een belangrijk uitgangspunt bij deze vraag is dat de applicatie voor een groot gedeelte ontwikkeld wordt met technieken die bij de teams bekend zijn. Op deze manier wordt het voor hen eenvoudiger om de ontwikkeling in de toekomst voort te zetten.

De technische invulling van de opdracht is gedaan op basis van eigen voorstellen die vervolgens getoetst zijn bij Rens Toonen.

4.3.1 Back-end

De back-end van het Proof of Concept is een gelaagde applicatie met een duidelijke ‘separation of concerns’ (zie hoofdstuk 5.1). Ook moet de back-end applicatie een REST API kunnen aanbieden. Vanwege de gestelde projectgrens dat er in Java of Kotlin ontwikkeld moet worden, wordt de keuze voor een webframework beperkt tot deze talen.

Kotlin of Java

Voor het ontwikkelen van de back-end applicatie had ik de keuze tussen twee talen voor de JVM: Kotlin of Java. Tijdens de studie is Java ruimschoots aan bod gekomen, waardoor hier minder over te leren valt. Daarnaast biedt de afstudeeropdracht de laatste mogelijkheid om tijdens de studie te experimenteren met nieuwe technieken onder professionele begeleiding. Ook biedt Kotlin volledige interoperabiliteit met Java, waardoor gebruik kan worden gemaakt van alle features uit de Java SDK. Hierdoor was de keuze snel gemaakt en is Kotlin ingezet als taal voor de back-end applicatie.

Back-end webframework

Het back-end framework moet gebruikt kunnen worden met Kotlin en de volgende functionaliteiten bieden:

- Eenvoudige setup en goede documentatie
- Learning curve die past binnen het tijdsbestek van de afstudeerperiode
- Dependency injection voor het vereenvoudigen van een scheiding van verantwoordelijkheden
- REST API
- Veelgebruikt zodat er informatie over te vinden is op bijvoorbeeld Stack Overflow
- Integratie met een ORM voor een database

De frameworks die zijn meegenomen in de overweging zijn: KTor, Javalin, Spring, Spring Boot en Spark.

Van deze frameworks bieden enkel Spring en Spring Boot out-of-the-box dependency injection. De andere frameworks hebben hiervoor een library nodig zoals Koin.

Zowel Spring als Spring Boot werken goed met Kotlin en worden veel gebruikt volgens de star statistieken op Github. Spring Boot is de evolutie van Spring waarbij weinig configuratie nodig is om een applicatie te bouwen die gewoon werkt (Pivotal, sd). Daarnaast biedt Pivotal, de makers van Spring Boot, een eenvoudige tool aan waarmee snel een Spring Boot project kan worden opgezet.

Een bijkomend voordeel is dat er binnen de business line kennis is over Spring Boot waardoor ik met vragen bij collega’s terecht kan. Daarmee is de keus gevallen op Spring Boot.

Database en ORM

Voor het opslaan van data is een database nodig. In de applicatie wordt gewerkt met relationele data, waardoor behoefte is aan een relationele database. De twee voor de hand liggende opties zijn: MySQL

en PostgeSQL. Binnen de business line wordt enkel gewerkt met PostgreSQL, waardoor de keuze hierop gevallen is.

Om de interactie met de database te vereenvoudigen wordt gebruik gemaakt van Hibernate. Hibernate is een ORM waarmee een in code gedefinieerd model kan worden gepersisteerd naar relationele data en andersom. Hibernate is eenvoudig te integreren met Spring Boot en biedt database drivers voor een PostgreSQL database.

Een onderdeel van Java is de Java Persistence API (JPA), die ook in Kotlin kan worden gebruikt. JPA is een ORM onafhankelijke set annotaties waarmee data relaties in een object georiënteerd model kunnen worden beschreven. De met JPA beschreven relaties worden door Hibernate geïnterpreteerd en vertaald naar tabellen in de database.

Ter illustratie is een Customer entiteit opgenomen uit een JPA tutorial van Spring de die voorzien is van JPA annotaties (Pivotal).

```
@Entity
public class Customer {

    @Id
    @GeneratedValue(strategy=GenerationType.AUTO)
    private Long id;
    private String firstName;
    private String lastName;
    …
}
```

4.3.2 Front-end

De front-end van het Proof of Concept is een webapplicatie. In mijn afstudeervoorstel heb ik aangegeven graag Angular in te zetten om de front-end te ontwikkelen, zodat ik mijn ervaring hiermee nog verder kan uitbreiden. Daarnaast wil ik met het inzetten Angular bijdragen aan vernieuwing binnen het front-end landschap van Topicus applicaties.

Tijdens de specialisatie Web & Hybrid app development in kwartielen 4.1 en 4.2 heb ik mogen experimenteren met Vue.js, maar dit heeft mij niet overtuigd om het in te zetten voor de afstudeeropdracht. Vue.js biedt naar mijn mening te weinig structuur voor het opzetten van een grootschalige applicatie, iets waar Angular juist goed op scoort met het module concept.

4.4 ONTWIKKELING VAN HET PROOF OF CONCEPT

Het beantwoorden van de vragen uit het onderzoek is gelijktijdig met het ontwikkelen van het Proof of Concept opgepakt. Hiermee wordt iteratief aan het product gewerkt en vindt de analyse van een thema plaats wanneer het wordt opgepakt. Deze thema’s worden ook wel epics genoemd.

De voortgang van de userstories wordt vastgelegd in Jira.

In Figuur 2 zijn de epics opgenomen die tijdens de afstudeerperiode zijn uitgewerkt.

Figuur 2 Epics release note applicatie

- Koppelen met externe bron release notes

Hier wordt een koppeling gelegd met Jira voor het ophalen van de release notes voor Topicus HAP.

- Beheren van releases

Dit onderwerp gaat in op het aanmaken van een release in de beheerapplicatie.

- Publiceren van releases

Dit onderwerp gaat in op het publiceren van een release, release notes en instructies vanuit de beheerapplicatie.

- Weergeven publicaties

Hierbij wordt gewerkt aan het beschikbaar maken en presenteren van de gepubliceerde releases, release notes en instructies in de inzage webapplicatie.

- Verwijzen naar release notes vanuit Topicus HAP

Dit onderwerp gaat in op de integratie van de inzage webapplicatie in Topicus HAP en de manier waarop eindgebruikers naar deze applicatie kunnen navigeren.

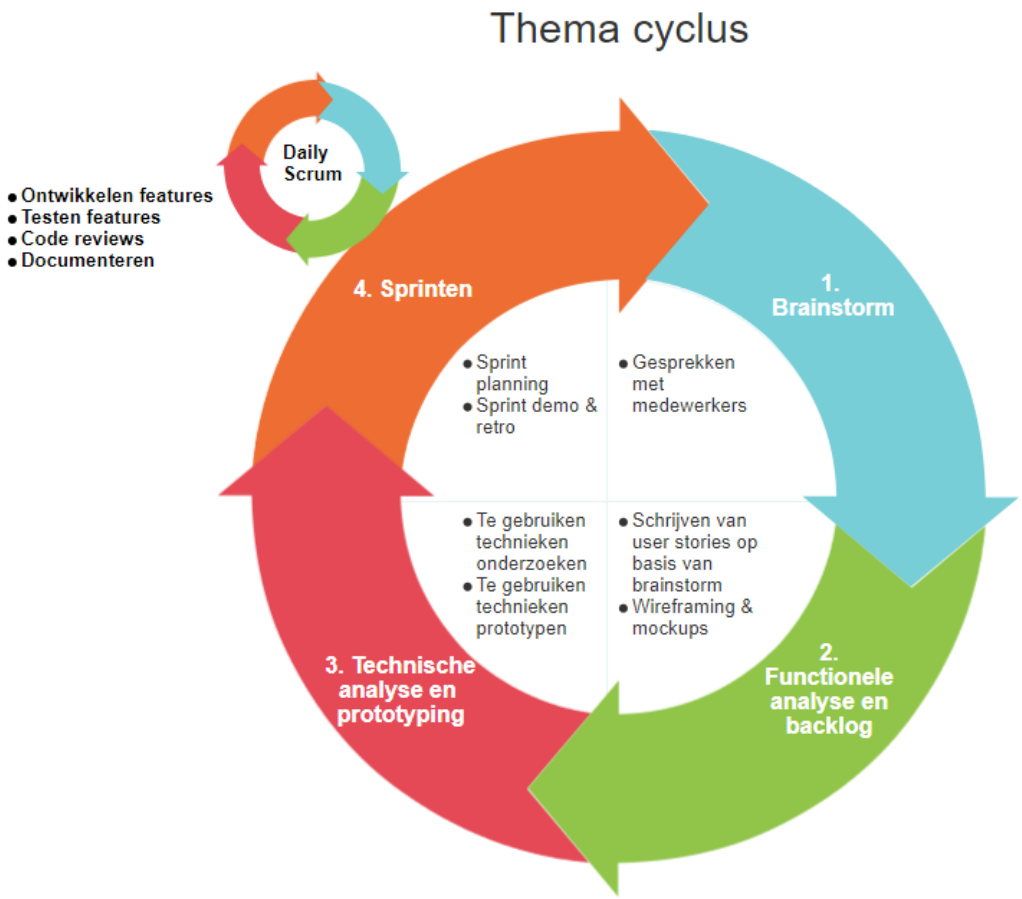
- Beheren van instructies

Hierbij wordt gewerkt aan het schrijven en bewerken van instructies in de beheerapplicatie.

- Instructies bijlagen

Dit voegt bijlage functionaliteit toe aan instructies in de vorm van plaatjes en embedded YouTube filmpjes.

In Figuur 3 is een procesdiagram opgenomen dat visualiseert hoe de verschillende thema’s zijn opgepakt.



Figuur 3 Thema cyclus

1. **Brainstorm**

Tijdens de brainstormfase wordt gesproken met medewerkers van Topicus HAP om de verschillende functionele en technische vragen te beantwoorden. De uitkomsten van deze gesprekken worden meegenomen in de functionele analyse en het schrijven van de user stories.

2. **Functionele analyse en backlog**

Na de brainstormsessies is een goed beeld gevormd van de benodigde functionaliteiten. Deze functionaliteiten worden gegroepeerd per epic en omgezet tot user stories. Daarnaast wordt gewerkt aan wireframing en het maken van mockups. Deze worden daarna getoetst bij de werknemers. De user stories worden met de begeleiders doorgenomen om te controleren of ze helder en toereikend zijn.

3. **Technische analyse en prototyping**

In sommige gevallen wordt gewerkt met technieken die nog niet eerder zijn gebruikt. Bijvoorbeeld de integratie van een tekstverwerker voor het schrijven van instructies, het uploaden van afbeeldingen of het koppelen met de REST API van Jira.

Om deze technische uitdagingen op te lossen wordt gebruik gemaakt van prototyping om tot de meest geschikte oplossing te komen.

Bij de integratie van een tekstverwerker zijn meerdere prototypes gebouwd met verschillende tekstverwerkers om te kijken welke aan de eisen van de teams voldoen.

De uitkomsten van deze technische analyse wordt in de user stories en het technisch ontwerp opgenomen.

4. Sprinten

Om dit iteratieve proces te kunnen ondersteunen is een methodiek nodig die hierop aansluit. Hiervoor zijn een aantal opties, waarvan Kanban en Scrum het meest toepasbaar zijn voor dit project. Een uitgebreide opzet van de inrichting van het scrumproces is te lezen in het projectdossier.

Tijdens dit project wordt een sprintlengte gehanteerd van twee weken waarbij een het aantal ingeplande punten tussen de 11 en 19 ligt afhankelijk van de complexiteit van de user stories. Het aantal punten van de stories is ingeschat op basis van een baseline van **5** punten die tijdens de eerste sprint is ingeschat op basis van één duidelijke user story.

Voor het bewaken van de voortgang van de sprints, het schrijven van de user stories en het bijhouden van de backlog is gebruik gemaakt van Jira.

De sprintplanning heeft altijd aan het begin van iedere sprint plaatsgevonden en is samen met de begeleiders gehouden.

Aan het eind van iedere sprint is een demo en retrospective gehouden. Ter illustratie is sprint 3 opgenomen in Bijlage 4 – Sprintverslag. De overige sprintverslagen zijn te lezen in het projectdossier.

4.4.1 Testen

Om een uitspraak te kunnen doen over de werking en betrouwbaarheid van de software is het noodzakelijk enige zaken te testen. In de applicatie onderkennen we drie testniveaus:

- Schermtesten
- Integratietesten
- Unittesten

Verder zijn er enkele uitgangspunten die voor de gehele applicatie gelden:

- De back-end code moet compileren
- De webapplicaties moeten zonder warnings of errors builden

De integratietesten en unittesten hebben betrekking tot de back-end van het systeem en worden met behulp van unittest framework JUnit5 en het Spring Boot framework uitgevoerd. Waar nodig worden afhankelijkheden van repositories of dataservices verzorgd door Mockito, een tool voor het mocken van classes en methode aanroepen.

Om de tests te draaien wordt gebruik gemaakt van de *MockitoJUnitRunner* in plaats van de *SpringBootRunner* omdat dit mogelijkheden biedt om dependencies die via Mockito worden gemockt te injecteren.

Als naslagwerk is het artikel Testing in Spring Boot van Baeldung gebruikt (baeldung, 2019).

Schermtesten

Schermtesten zijn bedoeld om de functionaliteit in de webapplicaties te valideren. Hiervoor zijn een aantal mogelijkheden, waaronder:

- End-to-end testen met Protractor
- Handmatig uitvoeren van testscenario's

Het schrijven van end-to-end tests voor Angular applicaties is behoorlijk arbeidsintensief en vergt veel code om relatief eenvoudige scenario's te testen.

Om toch de werking van de gebouwde functionaliteit in de webapplicaties te valideren worden testscenario's handmatig uitgevoerd voor de aanwezige functionaliteiten. Tijdens de sprintdemo worden de ontwikkelde functionaliteiten gepresenteerd. De functionaliteiten worden gezamenlijk uitgevoerd en beoordeeld op functioneren.

Integratietesten

Om te valideren of de volledige stack naar behoren geconfigureerd is worden integratietests ingezet. De integratietests roepen API end-points aan in de service laag van de back-end applicatie en worden uitgevoerd op een draaiende applicatie in de ontwikkelomgeving.

Spring Boot biedt hiervoor de class *TestRestTemplate* aan waarmee HTTP verzoeken kunnen worden verzonden.

Er zijn integratietests toegevoegd voor de volgende acties in het systeem:

- Aanmaken nieuwe Release
- Ophalen lijst met ReleaseReference
- Ophalen aangemaakte Release

Unittesten

Om het gedrag van de business rules die in het systeem aanwezig zijn te kunnen testen wordt gebruik gemaakt van unittests. Deze unittests testen individuele validator classes, waarbij eventuele dependencies van de class worden gemockt met Mockito.

Naast het testen van de business rules zijn er unittests opgenomen die bepaalde ontwerpkeuzes in modellen afdwingen.

Validators

De meeste unittests zijn geschreven voor de validators die in de business laag aanwezig zijn. De validators voeren de business rules uit en moeten daarom naar behoren functioneren. Een uitgebreide uitleg over de aanwezige business rules en door welke validators deze worden uitgevoerd is te lezen in de technische documentatie.

Models

In een aantal gevallen is er in een model class een duidelijke keuze gemaakt. Deze volgende keuzes worden door een unittest gevalideerd:

- In alle model classes in de *management-web-api* module geldt dat properties van het type LocalDateTime voorzien moeten zijn van een @JsonFormat annotatie.

Om de LocalDateTime properties naar een correct format te serialiseren is het noodzakelijk deze van een format te voorzien. Ondanks de vele opties die hiervoor worden voorgeschreven in de Spring Boot documentatie, is de het annoteren van de property de enige oplossing die werkt.

Wanneer een nieuwe class wordt toegevoegd met een LocalDateTime property zal de unittest falen wanneer dit veld niet voorzien is van een annotatie.

- De class JiraIssueFieldDefinitionEntity moet voorzien zijn van een *objectDataFieldName* indien het fieldtype een OBJECT is. Dit wordt gevalideerd bij het instantiëren van de class.

Vanwege de manier waarop Jira omgaat met verschillende datatypes in het response model is het noodzakelijk om aan te geven in welke property een gegeven opgeslagen ligt.

5. GEMAAKTE KEUZES

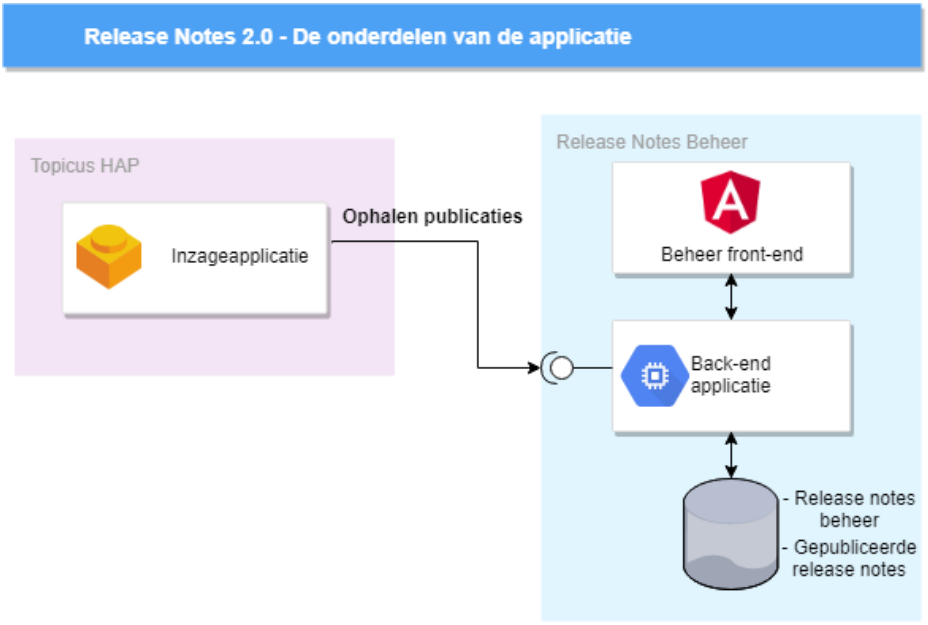
In dit hoofdstuk wordt ingegaan op de meest interessante keuzes die tijdens de afstudeerperiode gemaakt zijn. Dit hoofdstuk is opgenomen om inzicht te geven in hoe ik technische, functionele of procesmatige vraagstukken heb opgepakt.

5.1 HET KIEZEN VAN EEN GESCHIKTE SYSTEEMARCHITECTUUR

Om tot een geschikt ontwerp te komen heb ik vijf voorstellen gemaakt hoe het Proof of Concept technisch vormgegeven kon worden. Het doel hiervan was om het gesprek aan met ontwikkelaars van Topicus HAP aan te gaan om tot een gedegen besluit te komen over de meest relevante en haalbare architectuur voor de afstudeeropdracht. Deze voorstellen zijn getoetst bij Rens Toonen en Hubert Flisijn ontwikkelaars bij team Acute Zorg.

Van de vijf voorstellen worden de drie voornaamste in dit hoofdstuk besproken. Ieder voorstel is voorzien van een architectuurplaat met daarbij een uitleg en een conclusie.

Op basis van gesprekken met werknemers is besloten dat Het Proof of Concept in ieder geval zal bestaan uit drie onderdelen weergegeven in *Figuur 4*. In het blauwe blok is de release note applicatie afgebeeld. Het roze blok is Topicus HAP, waarin een applicatie zal worden geïntegreerd die release notes kan presenteren aan de eindgebruikers.



Figuur 4 Onderdelen release notes applicatie

Het Proof of Concept bestaat uit drie onderdelen:

- **Beheer front-end**

In deze applicatie, ofwel het beheerpaneel, kunnen de teams van Topicus HAP een release aanmaken. Onder deze release vallen enerzijds release notes die direct uit Jira gehaald worden. Anderzijds kan de release gedetailleerde instructies bevatten, voorzien van afbeeldingen of video’s. Deze instructies worden in het beheerpaneel geschreven.

- **Back-end applicatie**

De back-end applicatie is enerzijds verantwoordelijk voor het beheer van de release notes en bedient daarmee via een REST API de beheerapplicatie. Anderzijds bedient deze applicatie via een tweede REST API de webapplicatie voor inzage van de release notes die door eindgebruikers van Topicus HAP te benaderen is.

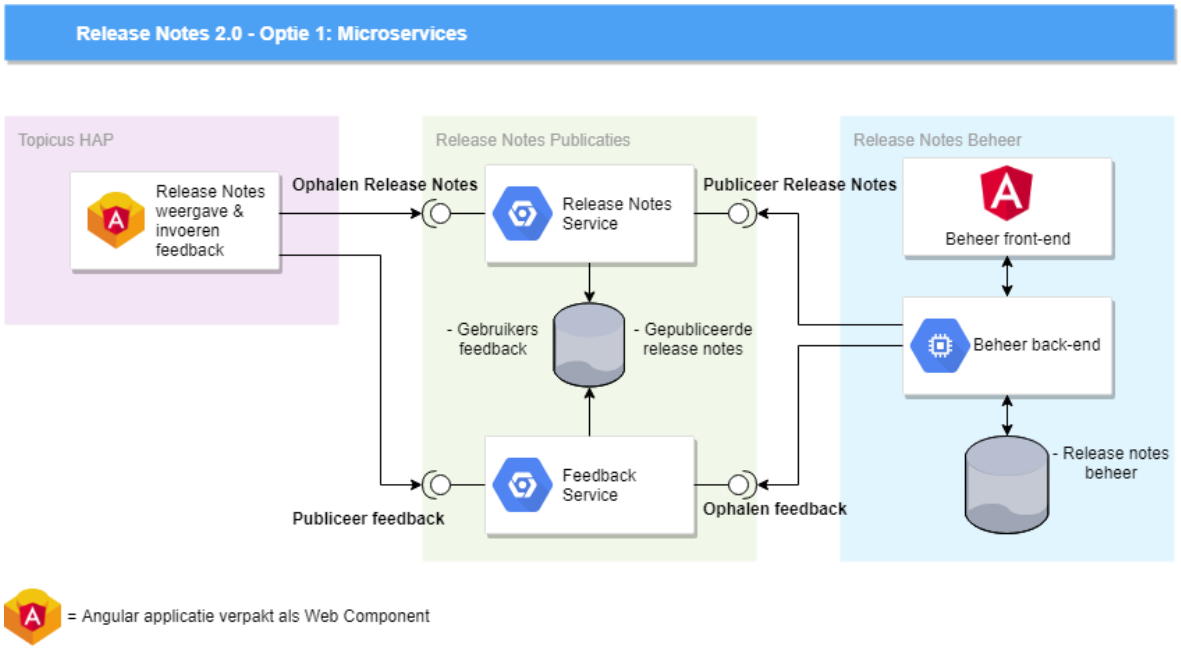
De back-end bevat de businesslogica, web api’s, een koppeling met Jira en een koppeling met een eigen database voor het vastleggen van data die voortkomt uit het beheerpaneel.

- **Webapplicatie voor de inzage van release notes**

Deze webapplicatie is te gebruiken door de eindgebruikers van Topicus Hap en kan gepubliceerde release notes opvragen bij de beheer back-end en tonen op het scherm voor eindgebruikers van Topicus HAP.

5.1.1 Optie 1 – Microservices met Web Component

In Figuur 5 is een microservices architectuur van het Proof of Concept afgebeeld. De voor en nadelen van deze architectuur zijn samengevat in Tabel 2.



Figuur 5 Architectuur optie 1: Microservices

Het idee achter deze oplossingsrichting is om gepubliceerde release notes en ontvangen feedback los te trekken van het beheer hiervan. De voornaamste redenen hiervoor zijn high-availability voor het bevragen van de publicaties vanuit Topicus HAP en scheiding van verantwoordelijkheden op applicatieniveau.

Echter bleek uit gesprekken dat high-availability van release notes geen hoge prioriteit heeft en dat het best voor mag komen dat deze informatie een paar dagen niet beschikbaar is (Lisette Heerink, persoonlijke communicatie, 4 maart 2019). Daarnaast brengt een dergelijke architectuur veel beheer en configuratie op netwerkniveau met zich mee. Er moeten meerdere Virtual Machines worden opgezet, waarbij ook rekening moet worden gehouden met onder andere load-balancing en

redundantie van systemen. Daarnaast speelt geld een rol, bij het optuigen en inrichting van iedere Virtual Machine zijn kosten gemoeid (Hubert Flisijn, persoonlijke communicatie, 28 februari 2019).

De webapplicatie ter inzage van de gepubliceerde release notes bestaat uit een Angular applicatie die verpakt wordt als Web Component en in Topicus HAP wordt geplaatst. Er bestaat wel enige onzekerheid met betrekking tot de integratie van een Web Component omdat dit een onbekende techniek is. Om dit te kunnen implementeren moet eerst onderzoek worden gedaan hoe deze techniek werkt.

Enerzijds is dit om de release notes in Topicus HAP te kunnen presenteren, zonder dat de applicatie uitgebreid hoeft te worden met uitgebreide kennis voor het ophalen en weergeven van release notes. Een bijkomend voordeel van deze opzet is dat het eenvoudiger toe te passen is binnen andere applicaties waar release notes getoond moeten worden.

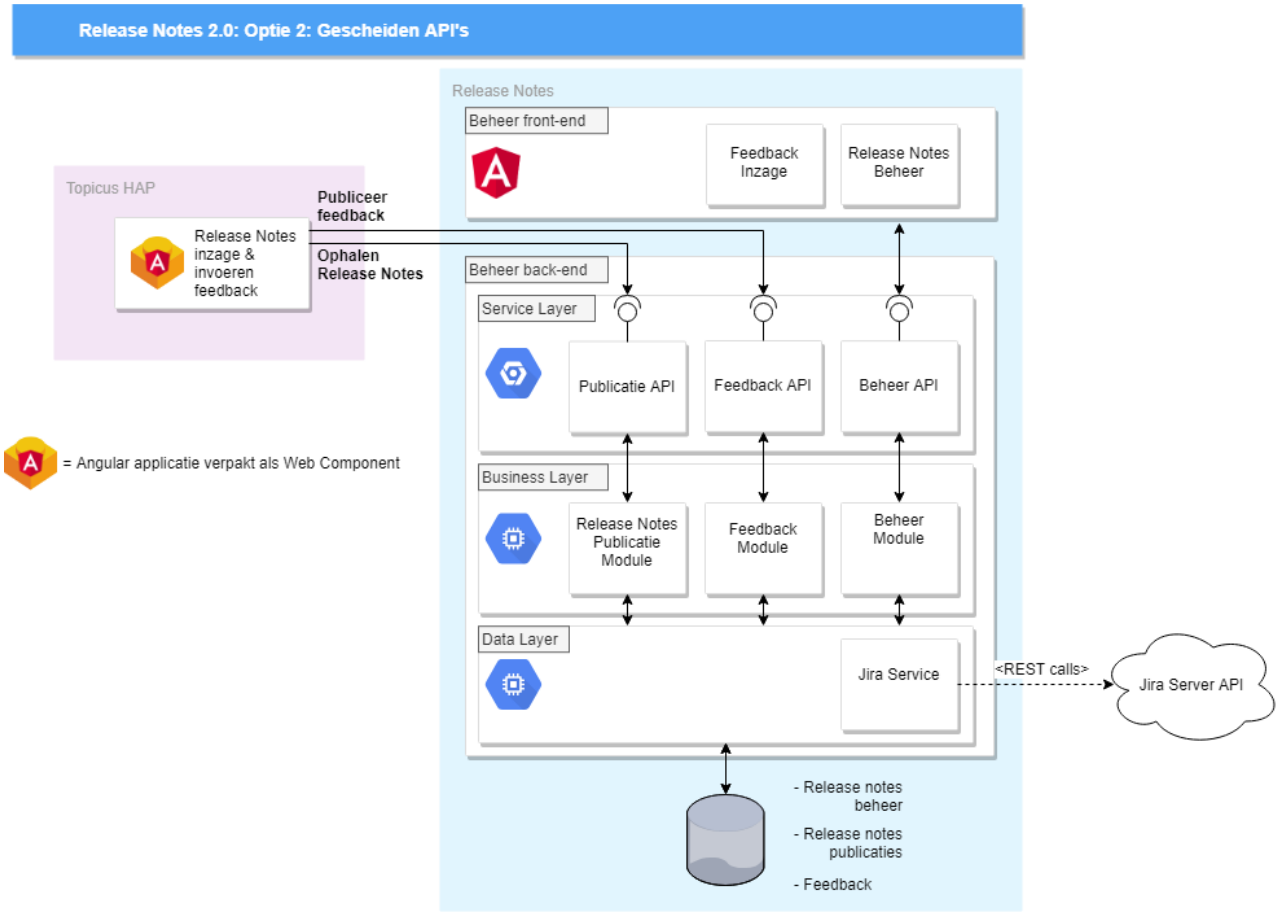
Anderzijds heeft het als voordeel dat een eindgebruiker van Topicus HAP niet naar een externe applicatie hoeft te navigeren om de release notes in te kunnen zien. De eindgebruiker blijft daarmee dus in de vertrouwde omgeving van de applicatie waar hij of zij mee werkt.

Voordelen	Nadelen
• Scheiding van publicaties en beheer, waarbij enkel de publicaties high-available hoeven te zijn	• Infrastructuur overhead (verschillende VMs, netwerkconfiguratie, loadbalancing & redundancy)
• Onafhankelijk kunnen onderhouden van de applicaties zolang het contract tussen de applicaties gelijk blijft.	• Tijd investeren in techniek betekent minder functionaliteit
• Technisch moderne architectuur	• Beveiliging
• Inzet van Web Components voor de inzageapplicatie houdt de eindgebruiker binnen de HAP omgeving	• Onzekerheid Web Component

Tabel 2 Voor-/nadelen microservices

5.1.2 Optie 2 – Gelaagde applicatie met inzage in Web Component

In Figuur 6 is een gelaagde architectuur binnen de applicatie afgebeeld. De voor- en nadelen van deze oplossingsrichting zijn samengevat in Tabel 3.



Figuur 6 Architectuur optie 2: Gelaagde applicatie met gescheiden APIs

Op basis van de uitkomst van het gesprek over een microservices architectuur is gekozen om de scheiding van verantwoordelijkheden binnen de applicatie op te lossen. Een groot voordeel is dat er slechts één Virtual Machine hoeft worden ingericht voor de applicatie, wat het beheren hiervan eenvoudiger en goedkoper maakt. Daarnaast is het opzetten van de afgebeelde architectuur relatief minder complex waardoor er meer tijd overblijft voor functionaliteit. Functionaliteit telt zwaarder voor de teams, vandaar deze concessie.

Daarnaast wordt in dit scenario een koppeling met Jira geïntroduceerd. Deze koppeling maakt het mogelijk de geschreven release notes uit Jira op te kunnen halen.

Er wordt een duidelijke scheiding gemaakt in de REST end-points voor het beheer en de inzage in de Service laag. Deze keuze heeft als voordeel dat het beheer enkel op het interne netwerk hoeft worden neergezet en daarmee wordt afgeschermd van het internet. Daarnaast worden er verschillende lagen gehanteerd met ieder hun eigen doelen en verantwoordelijkheden.

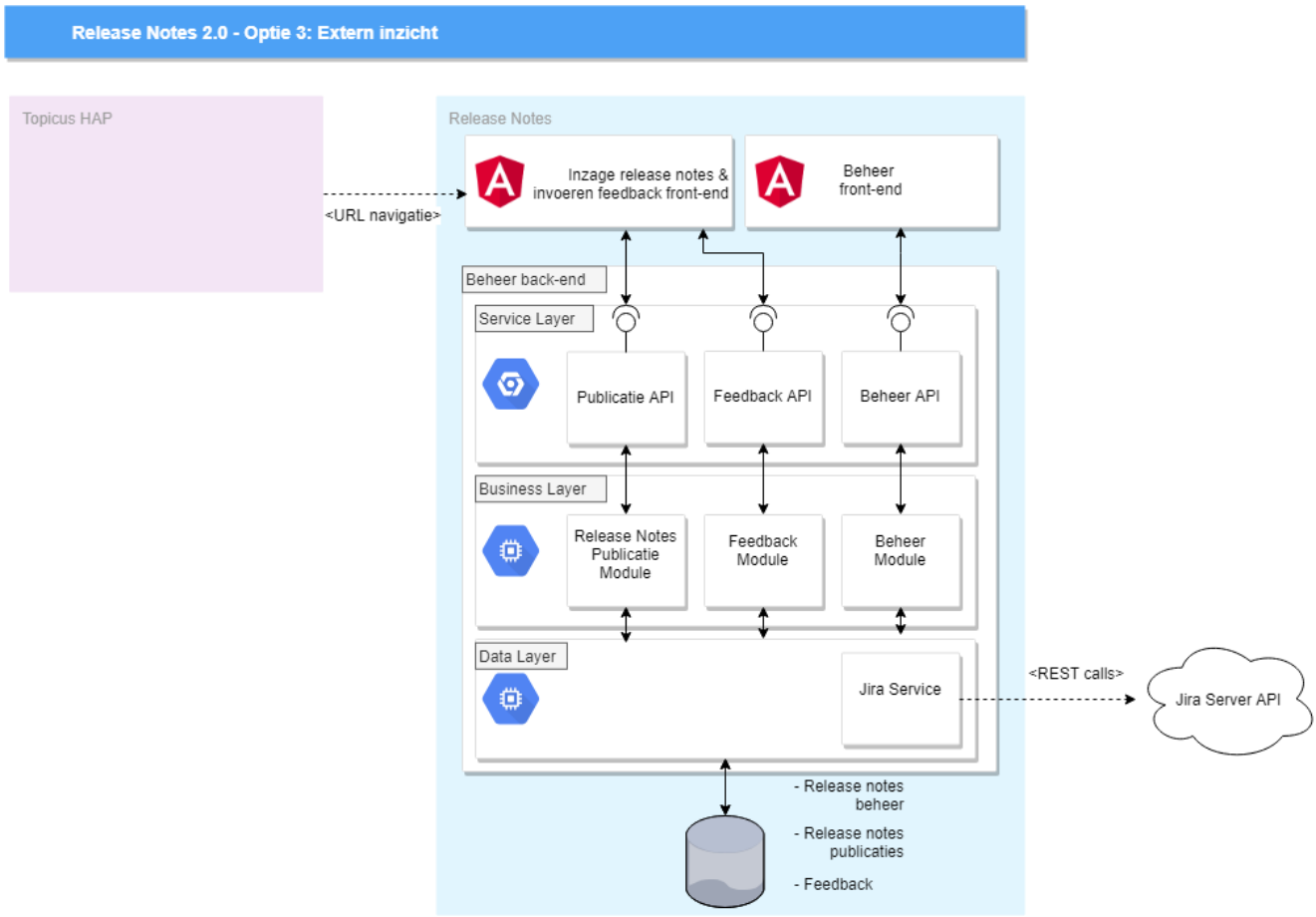
De webapplicatie ter inzage van de release notes blijft een Web Component dat in Topicus HAP kan worden geplaatst. De tijdsinvestering die benodigd is om deze techniek te onderzoeken wordt meegenomen in het scrumproces.

Voordelen	Nadelen
• Eenvoudiger op te zetten dan optie 1 (minder infrastructuur, beter lokaal te testen, meer tijd voor functionaliteit)	• Publicaties en beheer in één applicatie
• Scheiding van verantwoordelijkheden op applicatieniveau	• Nieuwe deploy betekent alles down (geen onafhankelijke bron voor publicaties)
• Inzet van Web Components voor de inzageapplicatie houdt de eindgebruiker binnen de HAP omgeving	• Onzekerheid Web Components
• Koppeling met Jira	

Tabel 3 Voor-/nadelen gelaagde applicatie

5.1.3 Optie 3 – Gelaagde applicatie met inzage in externe applicatie

In Figuur 7 is net als bij optie twee een gelaagde architectuur binnen de applicatie afgebeeld. De voor- en nadelen van deze oplossingsrichting zijn samengevat in Tabel 4.



Figuur 7 Architectuur optie 3: Gelaagde applicatie met inzage in extern applicatie

Het verschil met optie twee is echter dat de webapplicatie ter inzage van de release notes nu als losse webapplicatie wordt neergezet in plaats van als Web Component. Deze keuze maakt integratie met Topicus HAP eenvoudiger doordat in Topicus HAP nu enkel een hyperlink hoeft te worden geplaatst waarmee de gebruiker naar de webapplicatie navigeert.

Dit heeft als voordeel dat in Topicus HAP geen ruimte gemaakt hoeft te worden voor een Web Component wat minder ontwikkeltijd kost. Het nadeel hiervan is dat een gebruiker bij het navigeren naar de release notes in een nieuw venster of tabblad van de browser terechtkomt. Dit kan als storend kan worden ervaren en is niet erg gebruikersvriendelijk (Lisette Heerink, persoonlijke communicatie, 4 maart 2019).

Voordelen	Nadelen
• Geen onzekerheid Web Component, er wordt immers niets geïntegreerd in Topicus HAP	• Bij navigeren naar de inzageapplicatie vanuit Topicus HAP, ‘verlaat’ de eindgebruiker de HAP omgeving wat als storend kan worden ervaren.
• Minimale aanpassing Topicus HAP	

Tabel 4 Voor-/nadelen externe inzage

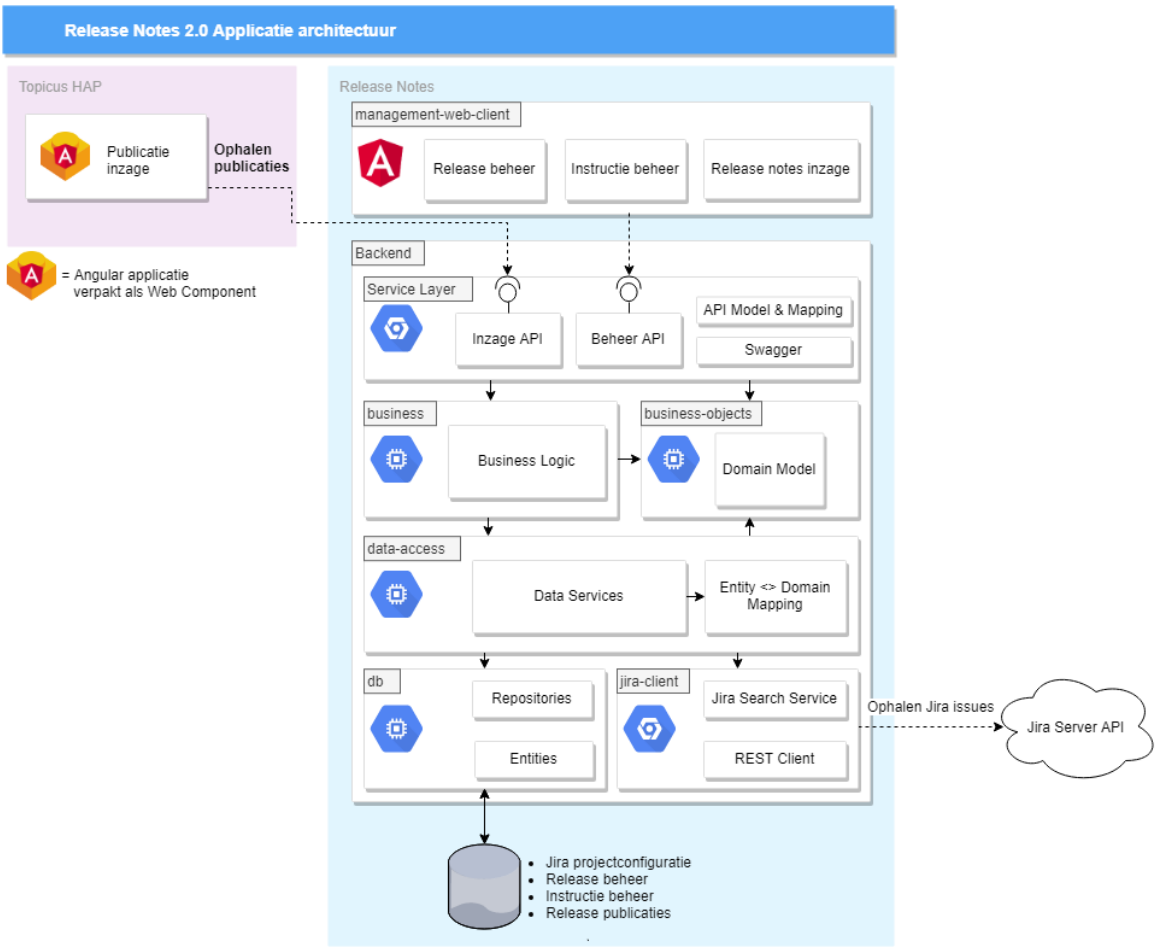
5.1.4 Conclusie – Gelaagde applicatie met inzage in Web Component

Een micro-services architectuur kost tijd en geld om goed op te zetten met als gevolg dat er minder tijd besteed kan worden aan het ontwikkelen van functionaliteit en dat het in verhouding met de andere opties duur is. Het ontwikkelen van Het voorstel om de applicatie ter inzage van release notes als Web Component op te zetten werd zeer positief ontvangen.

Wanneer de scheiding van verantwoordelijkheden op applicatie niveau worden opgelost betekent dit een vermindering in complexiteit ten opzichte van een micro-services architectuur. Dit heeft als voordeel dat er meer tijd overblijft om functionaliteit te ontwikkelen. Door de scheiding van verantwoordelijkheden binnen de applicatie op te lossen behoudt de applicatie een modulaire structuur die overeenkomt met de gewenste architectuurprincipes van de teams van Topicus HAP.

Door de inzage in de release notes naar een losstaande applicatie te verplaatsen is de aanpassing die moet worden gedaan aan Topicus HAP minimaal, wat positief is. Echter heeft dit wel tot gevolg dat de eindgebruiker de release notes niet in HAP gepresenteerd krijgt maar in een externe applicatie. Dit werkt afleidend en heeft niet de voorkeur vanuit de teams.

Op basis hiervan is gekozen om de architectuur uit optie twee verder uit te werken. Een wijziging ten opzichte van optie twee is een nieuwe data-access laag. Deze laag is toegevoegd om de implementatie van de data laag en de manier waarop het domeinmodel wordt gevuld af te schermen van de business laag. De data waarmee het domeinmodel wordt gevuld bestaat immers uit twee onderdelen: data uit de database en data uit Jira.



Figuur 8 - Gekozen applicatiearchitectuur

Service Layer

In de service laag bevinden zich de web APIs die de business laag ontsluiten. De APIs bevatten de endpoints waarmee externe applicaties via het HTTP protocol met de backend kunnen communiceren. De service laag bevat een eigen model dat door middel van mapping wordt gevuld op basis van het domeinmodel. Dit model komt grotendeels overeen met het domeinmodel, maar is specifiek afgestemd op de databehoefte van de front-end applicaties.

Daarnaast bevat de service laag Swagger functionaliteit waarmee een OpenAPI 2.0 specificatie kan worden gegenereerd die het publieke contract van de APIs beschrijft (zie hoofdstuk 4.3). Deze specificatie wordt gebruikt om code te genereren zodat externe applicaties zonder handwerk één op één aansluiten op de API.

Business Layer

De business laag bevat alle business logica die in de applicatie aanwezig is. Hieronder valt:

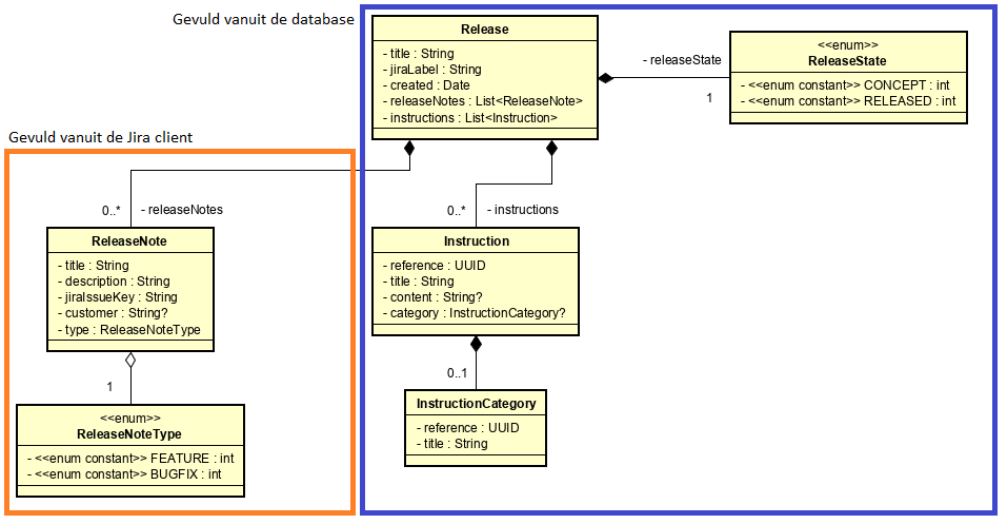
- Aanmaken van releases
- Koppelen van een Jira label aan een release
- De status van een release: concept of gepubliceerd
- Aanmaken en bewerken van instructies
- Publiceren van releases met bijbehorende release notes en instructies.

Verder vindt hier businessvalidatie van gegevens plaats die via de Service Layer de applicatie binnenkomen. Bijvoorbeeld bij het aanmaken van een release wordt gevalideerd of de opgegeven titel uniek is.

Het domeinmodel is vastgelegd in een losse module (business-objects in Figuur 8) zodat de afhankelijkheden tussen de verschillende modules binnen de applicatie eenvoudiger kunnen worden beheerd.

Data-access Layer

De data-access laag is de schakel tussen enerzijds de database module en de Jira client en anderzijds de business laag. Het domeinmodel wordt namelijk vanuit deze twee bronnen gevuld (zie Figuur 9). De business laag heeft enkel kennis van het domeinmodel en de data-access laag.



Figuur 9 Domeinmodel release en release notes

Met welke bronnen het model verrijkt wordt is een los concern dat nu in de data-access laag kan worden opgelost. Daarnaast is de data-access laag verantwoordelijk voor de mapping tussen data uit Jira, de database entiteiten en het domeinmodel.

Het bestaansrecht van deze laag is af te leiden uit de eis dat Jira altijd als waarheid wordt beschouwd voordat een release is gepubliceerd. Bij het publiceren van een release wordt een snapshot gemaakt van de release notes die in Jira aanwezig zijn. Deze snapshot wordt gepersisteerd naar de database waarna de release notes beschikbaar zijn voor inzage.

Hiermee voorkom je dat er een verschil kan bestaan tussen de staat in Jira en de release notes applicatie wanneer een release nog niet gepubliceerd is.

Data Layer

De data laag bestaat uit twee modules: *db* en *jira-client*.

De db module bevat de database entiteiten, de ORM en de repositories waarmee de data-access laag data uit de database kan ophalen.

De jira-client module bevat de services waarmee issues uit Jira kunnen worden opgehaald.

Presentation Layer

De presentatie laag bestaat uit twee webapplicaties: de beheerapplicatie en de inzageapplicatie. Met de beheerapplicatie kunnen de teamleden van Topicus HAP hun releases, release notes en instructies inzien, verwerken en publiceren.

De inzageapplicatie is een Web Component dat gepubliceerde releases, release notes en instructies bij de back-end kan opvragen en tonen aan de eindgebruiker.

5.2 KOPPELING MET JIRA

In dit hoofdstuk wordt gekeken naar het technische aspect van de deelvraag “***Welke functionaliteiten moet het beheerpaneel in de webapplicatie bevatten om een set release notes te kunnen verwerken?”*** met betrekking tot het leggen van een koppeling met Jira.

De ontwikkelteams van Topicus HAP, Team Acute Zorg en Team Zorgcoördinatie, werken net als de rest van het bedrijf met Jira om de voortgang van het softwareproject te bewaken. Dit gebeurt door middel van een product backlog en Kanban of Scrum borden met de gebruikelijke user stories voor nieuwe features en het oplossen van bugs. Deze user stories worden in Jira issues genoemd. Naast het bijhouden van de voortgang van de issues worden hieraan ook release notes toegevoegd die in begrijpelijke taal uitdrukken welke functionele wijzigingen er zijn gemaakt.

Om te voorkomen dat de teams van Topicus HAP op twee plekken release notes moeten schrijven en belast worden met extra administratie is het noodzakelijk om een koppeling te leggen met Jira. Het uitgangspunt voor release notes uit een externe bron is dat de externe bron altijd de waarheid bevat. Om deze reden is gekozen deze data niet in de database vast te leggen, maar altijd uit Jira op te halen. Dit voorkomt een dubbele boekhouding waarbij verschil kan ontstaan tussen de externe bron en de database van de beheerapplicatie. De enige uitzondering hierop is bij het publiceren van een release. Wanneer een release wordt gepubliceerd zal de data uit Jira worden opgehaald en vastgelegd in de database van de beheerapplicatie, zodat deze ter inzage beschikbaar kan worden gesteld.

De koppeling met Jira betekent dat issues in Jira door de beheerapplicatie omgezet kunnen worden in release notes. Deze release notes kunnen vervolgens gepubliceerd worden en beschikbaar worden gesteld aan de eindgebruikers van Topicus HAP.

Bij het aanmaken van een release in de beheerapplicatie wordt door de gebruiker een label aan de release gekoppeld. Dit label wordt gebruikt om een zoekopdracht in Jira uit te kunnen voeren naar de bij de release behorende issues. Hiervoor wordt gebruik gemaakt van de Jira Server REST API.

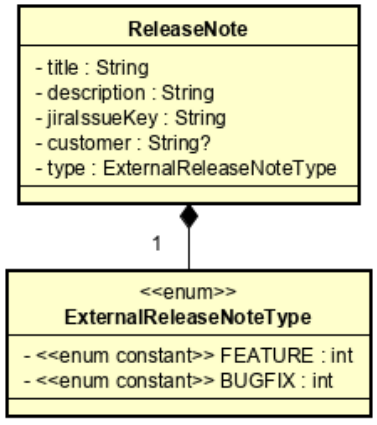
5.2.1 Verhouding Jira ticket tegenover het businessmodel uit de beheerapplicatie

In Figuur 10 is een issue uit Jira afgebeeld. Met oranje kaders is aangegeven welke gegevens uit het issue gebruikt worden in de beheerapplicatie. Op basis van het label (**5**) worden issues aan een release verbonden.

Figuur 10 Voorbeeld gegevens Jira issue

Binnen de beheerapplicatie wordt voor release notes het model gebruikt zoals afgebeeld in Figuur 11. Per Jira issue wordt een nieuw release note object aangemaakt.

De benodigde velden van de release notes zijn alsvolgt (bij ieder veld is aangegeven welk gegeven het bevat van het Jira issue in Figuur 10):



Figuur 11 Classdiagram externe release notes

- **Title** – De titel van het issue uit Jira. Dit is ook de titel van de release note. Bevat gegeven **3.**
- **Description** – De geschreven release note behorend bij het issue uit Jira. Bevat gegeven **6.**
- **JiraIssueKey** – De unieke Jira sleutel waarmee het issue kan worden geïdentificeerd. Bevat gegeven **2.**
- **Customer** – Optioneel veld waarin de klant is opgenomen die de wijzigingsaanvraag of bug heeft ingediend. Bevat gegeven **7.**
- **Type** – Het type van de release note. In de beheerapplicatie wordt onderscheid gemaakt tussen release notes voor features en bugfixes. Bevat gegeven **4.**

De data die uit Jira wordt gehaald zal door middel van een mapping kunnen worden omgezet tot dit model.

5.2.2 Het dynamische response model van Jira

In de web omgeving van Jira is te zien dat een issue allerlei data bevat (zie Figuur 10). Uit deze data zijn de volgende velden van toepassing zijn voor de beheerapplicatie:

- **Key** – In dit veld staat de unieke referentie waaraan het issue kan worden herkend.
- **Issuetype** – In dit veld is opgenomen van welk type het issue is (Story, Bug of . Op basis van deze informatie kan worden herleid of het om een feature of een bug gaat.
- **Summary** – De summary bevat de titel van het issue.
- **Release notes** – In dit veld worden release notes opgenomen die bij het issue horen.
- **Label** – In dit veld wordt een label toegewezen. Dit label veld wordt gebruikt om stories te groeperen die bij een nieuwe release worden uitgeleverd.
- **Klant** – In dit optionele veld is de klant opgenomen die deze wijzigingsaanvraag of bug heeft ingediend.
- **Zichtbaar voor klant** – In dit veld wordt aangegeven of de inhoud van het issue inzichtelijk is voor klanten.

In Figuur 12 Voorbeeld Jira issue response model is te zien hoe Jira een issue teruggeeft wanneer deze wordt opgevraagd via de REST API. Hierbij valt op dat bij een groot aantal velden de functionele naam ontbreekt. In plaats van *release notes* heet het veld *customfield_41402* zoals te zien in Figuur 12.

```
{
   "id":"611818",
   "key":"AZ-25",
   "fields":{
      "summary":"<<Titel van het issue>>",
      "customfield_41402":"<<Release note veld>>",
      "issuetype":{
         "name":"<<Type van het issue>>",
         "subtask":false,
      },
      "customfield_15800":{
         "value":"<<Klantnaam>>",
         "id":"22400"
      },
      "customfield_43700":{
         "value":"<<Zichtbaar voor klant>>",
         "id":"46100"
      }
   }
}
```

Figuur 12 Voorbeeld Jira issue response model

Dit dynamische response model heeft als gevolg dat er niet één op één kan worden gemapt tussen de velden uit het model van Jira en de velden uit het businessmodel van de beheerapplicatie. Bovendien kunnen de custom fields per Jira project en instantie verschillen omdat deze velden in eigen beheer zijn en het numerieke id door Jira wordt gegenereerd. Om dit dynamische model aan te kunnen sluiten op het model uit de beheerapplicatie moet onderzocht worden hoe de velden zich tegenover elkaar verhouden.

5.2.3 Mapping van dynamisch naar vast model

Om te achterhalen welke issuetypes en velden uit het response model aanwezig zijn is handmatig met Postman (een API tool) een aantal requests naar de Jira REST API uitgevoerd naar issues uit de projecten voor Team Acute Zorg (AZ) en Team Zorgcoördinatie (ZCT). Door middel van het uitvoeren van dit onderzoek kan een tabel worden opgesteld die de koppeling tussen de velden vastlegt.

In Tabel 5 is het resultaat van het onderzoek te zien naar de issue velden die benodigd zijn om het model uit de beheerapplicatie te kunnen vullen.

Jira issue field	JSON Data type		Businessmodel property	Beschrijving
Algemeen (gelijk tussen beide projecten)				
key	String	→	jiraIssueKey	De unieke sleutel van het issue.
summary	String	→	title	De titel van het issue.
issuetype	Object	→	type	Het type van het issue.
customfield_15800	Object	→	customer	Optioneel veld met de klantnaam die de wijziging heeft aangevraagd.
AZ (Jira project van Team Acute Zorg)				
customfield_41402	String	→	description	Het release note veld van issues binnen het AZ project.
ZCT (Jira project van Team Zorgcoördinatie)				
customfield_41400	String	→	description	Het release note veld van issues binnen het ZCT project.

Tabel 5 Mappingtabel issue velden

In Tabel 6 is het resultaat van het onderzoek te zien naar de verschillende issuetypes die binnen de Jira projecten aanwezig zijn en naar welk type release note deze wijzen.

Jira issue type	Type release note
Story	Feature
Bug	Bugfix
Zct-feature	Feature
Zct-beheer	Feature
Zct-improvement	Feature
Zct-bug	Bugfix

Tabel 6 Mappingtabel issuetypes

Hierbij valt op dat team AZ enkel gebruik maakt van de standaard Jira issuetypes Story en Bug. Team ZCT heeft eigen issuetypes aangemaakt maar maakt ook gebruik van de standaard issuetypes.

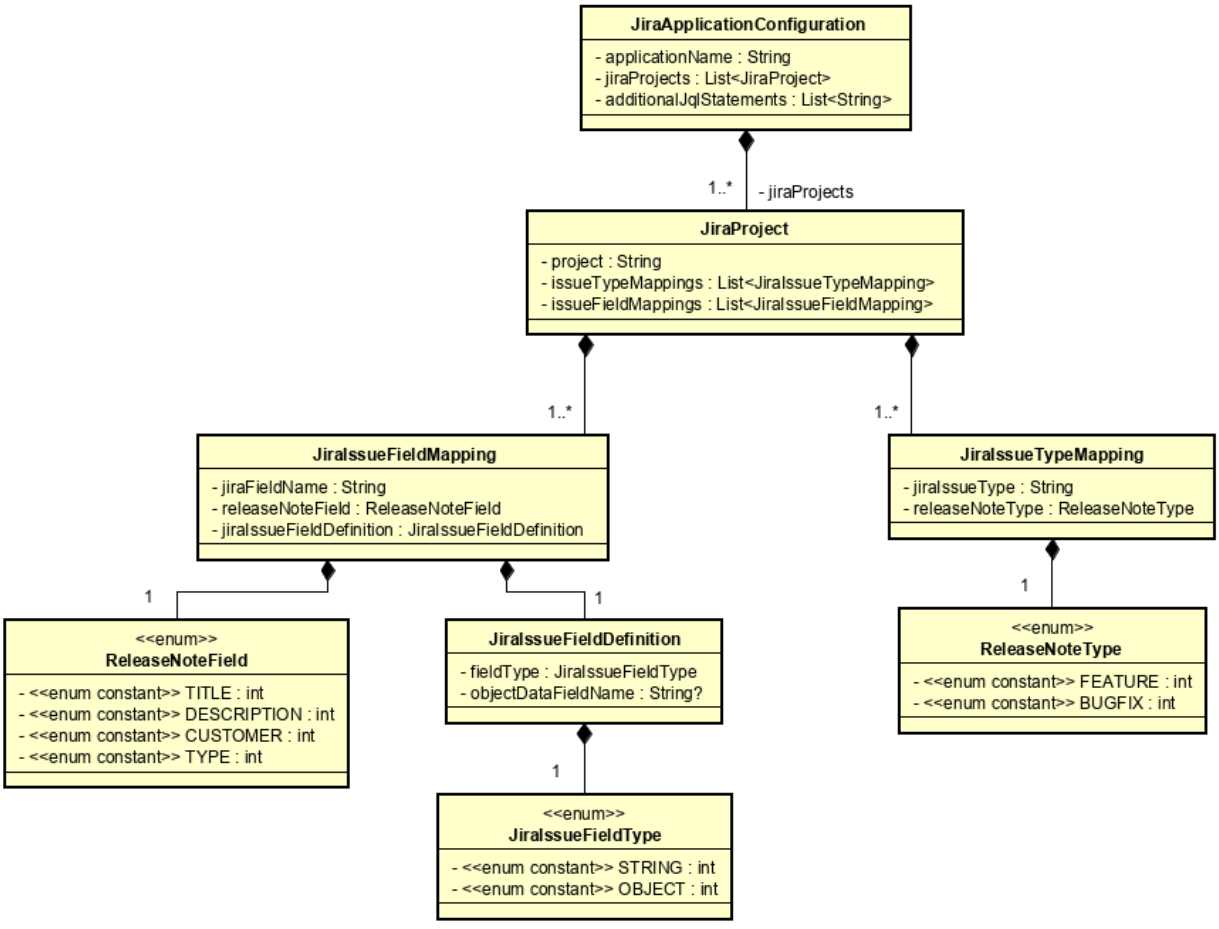
5.2.4 Mapping configuratie

Vanwege het dynamische response model dat Jira hanteert en het feit dat projecten in Jira verschillende velden en issuetypes kunnen bevatten is het noodzakelijk om voor deze zaken een configuratie te maken. Het doel van deze configuratie is om een brug te kunnen vormen tussen enerzijds Jira issues en anderzijds release notes in de beheerapplicatie. Door deze configuratie abstract te maken en vast te leggen in een model is het ook mogelijk om met enig uitzoekwerk andere Jira projecten aan te sluiten.

Het opstellen van de benodigde configuratie voor Topicus HAP is gedaan op basis van de resultaten weergegeven in 5.2.3 Mapping.

5.2.5 Configuratiemodel

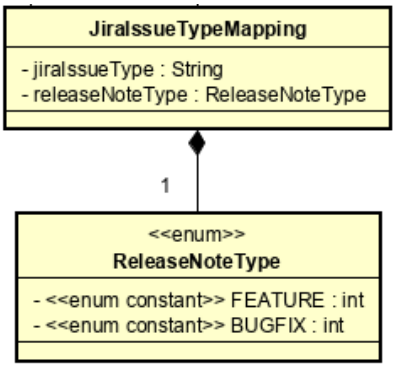
In Figuur 13 is het configuratiemodel opgenomen dat wordt gehanteerd in de beheerapplicatie en bevindt zich in de gradle module *data-access* van het softwareproject. Dit model wordt in de gradle module *data-access-filler* van het softwareproject gevuld met een configuratie die aansluit op de context van Topicus HAP. De configuratie wordt vervolgens gepersisteerd naar de database. Deze configuratie maakt het mogelijk dat issues uit de Jira projecten AZ en ZCT met de juiste velden kunnen worden bevraagd en gemapt naar het businessmodel van de beheerapplicatie. Deze generieke opzet maakt het mogelijke andere projecten aan te sluiten in de toekomst. Op de volgende pagina wordt het model puntsgewijs toegelicht.



Figuur 13 Classdiagram Jira configuratie

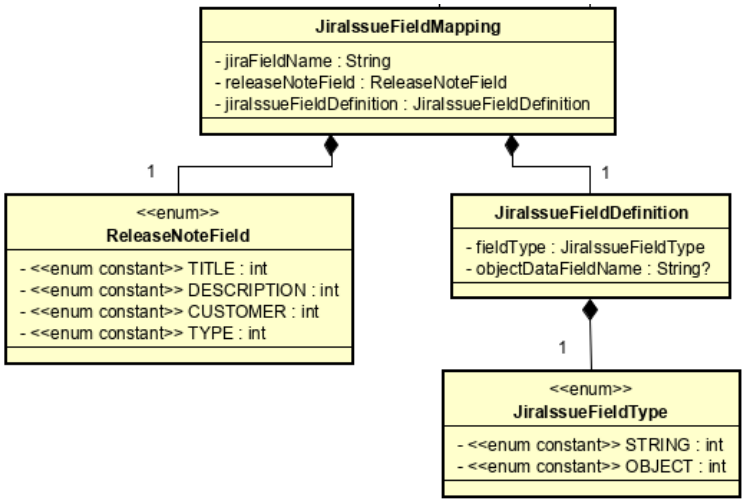
Het configuratiemodel bestaat uit de volgende onderdelen:

- **JiraIssueTypeMapping** – Deze class geeft de mapping aan tussen een issuetype uit Jira en een release note type in de beheerapplicatie.
    - **ReleaseNoteType** – Deze enum bevat de type release notes die in de beheerapplicatie worden ondersteund.



Figuur 14 Mapping tussen Jira issuetype en release note type

- **JiraIssueFieldMapping** – Deze class geeft de mapping aan tussen een issue veld uit Jira en het bijbehorende veld van een release note in de beheerapplicatie.
    - **JiraIssueFieldDefinition** – Deze class bevat de kennis of een veld uit het Jira model een JSON property van het type String of Object is. Bij een object is het verplicht de naam van de property mee te geven die het gegeven bevat.
    - **ReleaseNoteField** – Deze enum bevat de verschillende velden die in het model van de beheerapplicatie worden ondersteund.

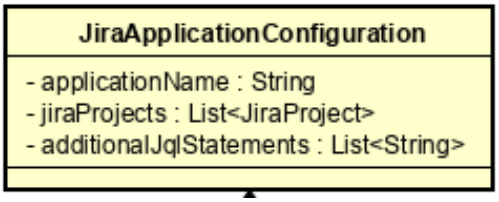


Figuur 15 Mapping tussen Jira issuefield en release note veld

- **JiraProject** – Deze class bevat de mappings voor zowel issuetypes als issuefields die voor één specifiek Jira project gelden. In de property *project* wordt de naam van het project in Jira opgenomen.

Figuur 16 Jira projectspecifieke configuratie

- **JiraApplicationConfiguration** – Deze class bevat één of meer Jira projecten en eventueel extra JQL statements die tijdens het uitvoeren van de zoekopdracht moeten worden gebruikt.



Figuur 17 Overkoepelende applicatie configuratie

5.2.6 Issues zoeken met behulp van JQL

De teams van Topicus HAP groeperen issues die bij een nieuwe release worden uitgeleverd door middel van het Label veld. Het is daarmee een logische keuze om issues op basis van dit gegeven op te vragen via de Jira Server REST API.

Hoofdonderdeel van een zoekopdracht in Jira is JQL, oftewel Jira Query Language. JQL is een taal die sterk lijkt op SQL waarmee zeer efficiënt issues kunnen worden opgehaald die voldoen aan de opgegeven query (Radigan, 2015).

Een voorbeeld van een JQL query die alle issues met het label *THAP9.*7, uit project *AZ* en met issue types *Story* en *Bug* ophaalt ziet er als volgt uit:

```
labels in ("THAP9.7") and project in ("AZ") and issuetype in ("Story",
"Bug")
```

Figuur 18 Voorbeeld JQL query string

Eventueel kunnen er nog JQL statements aan de query worden toegevoegd waarmee de zoekopdracht nog verder kan worden verfijnd. In de huidige configuratie worden issues ook gefilterd op het veld “Zichtbaar voor klant”, zodat release notes die niet voor de klant bestemd zijn niet worden meegenomen in het resultaat.

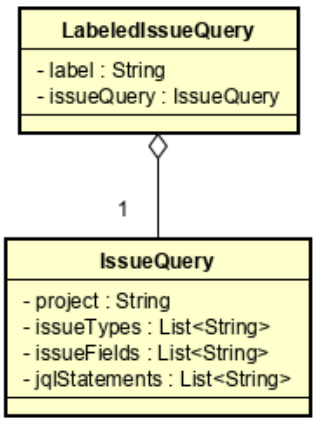
```
labels in ("THAP9.7") and project in ("AZ") and issuetype in ("Story",
"Bug") and "zichtbaar voor klant" is "ja"
```

Figuur 19 JQL query string met extra parameter

Querymodel

Wanneer een gebruiker in de beheerapplicatie een release opvraagt, zal deze worden opgehaald uit de database. Als er aan deze release een label gekoppeld is, wordt in de *data-access* laag een request opgesteld om de issues met dit label op te halen uit Jira.

Op basis van de aanwezige configuratie in de database en het opgegeven label wordt per project bij het uitvoeren van een zoekopdracht het model gevuld dat is afgebeeld in Figuur 20. Dit model kan vervolgens worden omgezet in een zoekopdracht die door de Jira REST API wordt herkend.



Figuur 20 Querymodel

Jira search request

Het uitvoeren van een zoekopdracht via de Jira Server REST API via HTTP kan op twee manieren (Atlassian, sd):

- GET request – Bij het uitvoeren van een zoekopdracht via de HTTP GET methode worden de zoekparameters als query parameters in de URL meegegeven.
- POST request – Bij het uitvoeren van een zoekopdracht via de HTTP POST methode worden de zoekparameters als JSON object naar de API verstuurd.

In Tabel 7 staan de parameters die worden gebruikt bij het uitvoeren van een zoekopdracht.

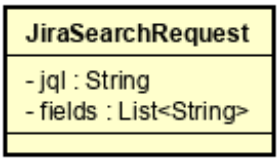
Parameter	Type	Beschrijving
jql	String	De JQL query string die de basis van de zoekopdracht vormt.
fields	String[]	De velden van een issue die in het zoekresultaat moeten worden meegenomen. Via deze property kan de grootte van de response worden beperkt. Bij het ontbreken van deze property worden alle velden teruggestuurd.

Tabel 7 Jira search request parameters

Er is voor gekozen om de POST methode in te zetten voor het uitvoeren van de zoekopdracht om twee redenen:

- Eenvoudige serialisatie – Het is met de ingebouwde JSON serialisatie van Spring Boot eenvoudig om een Kotlin class naar JSON om te zetten. Het encoden van een URL is foutgevoelig.
- Leesbaarheid van de code – Door de query in een Kotlin class vast te leggen is eenvoudiger op te maken welk request naar de Jira REST API wordt gestuurd. Een URL bestaande uit query parameters is lastiger te lezen.

Op basis van het model zoals afgebeeld in Figuur 21 kan het object worden aangemaakt dat als JSON object naar de Jira REST API verstuurd wordt (zie Figuur 21).



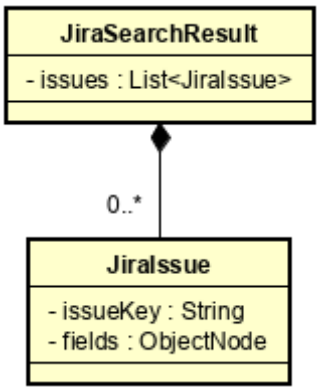
Figuur 21 Jira search request model

Hierbij wordt de property *jql* gevuld met de JQL query zoals beschreven in 5.2.6 Issues zoeken met behulp van JQL. De property *fields* wordt gevuld met alle issue velden die in de configuratie aanwezig zijn. Het opgeven van de velden heeft als voordeel dat enkel de gevraagde data wordt teruggegeven in plaats van het volledige issue waardoor de response grootte beperkt blijft.

Het uitvoeren van het HTTP request wordt gedaan met behulp van RestTemplate uit Spring Boot (Pivotal, sd). Een instantie van RestTemplate wordt aangemaakt in de *RestClient* class die daadwerkelijk het HTTP request verstuurd.

Mappen van het resultaat

Nadat het request succesvol is uitgevoerd wordt de JSON response van de Jira REST API door Spring Boot gedeserialiseerd naar het model afgebeeld in Figuur 22.



Figuur 22 Jira search result model

De property *fields* in *JiraIssue* is een ObjectNode. Een ObjectNode is een class uit de Jackson JSON parser library en representeert een JSON object in Kotlin. Een ObjectNode maakt het mogelijk om properties op te vragen uit het JSON object en om te controleren of deze properties aanwezig zijn (ObjectNode, sd).

Door middel van de geconfigureerde field mappings (Figuur 15) kan nu door de ObjectNode worden gebladerd op zoek naar de geconfigureerde jira fields, waarna de waarde kan worden gemapt naar het juiste veld in het businessmodel (Figuur 11).

Op basis van de geconfigureerde issue mappings (Figuur 14) kan de release note van het juiste type worden voorzien (Feature of Bugfix).

Het resultaat hiervan is dat alle in Jira aanwezig issues voor een gegeven release kunnen worden omgezet naar bruikbare data in de beheerapplicatie.

5.3 SCOPEVERKLEINING

In eerste instantie was een onderdeel van de opdracht dat eindgebruikers feedback moeten kunnen geven op functionaliteiten. Het geven van feedback zou via de inzageapplicatie verlopen. De feedback kon vervolgens worden weergegeven in de beheerapplicatie.

De vragen uit het onderzoek die bij dit onderdeel horen zijn:

Op welke manier kan de eindgebruiker feedback geven op nieuwe of gewijzigde functionaliteiten?

Hoe moet de verkregen feedback inzichtelijk worden gemaakt voor de Topicus HAP teams in de webapplicatie?

Echter bleek na drie sprints dat de beoogde functionaliteiten niet haalbaar waren gezien de planning en de complexiteit van andere onderdelen met meer prioriteit: presenteren van release notes in Topicus HAP en de koppeling met Jira. Deze koppeling kreeg de hoogste prioriteit en bleek groter dan vooraf gedacht.

Na gesprekken met Lisette Heerink en Rens Toonen is besloten de scope van de opdracht iets bij te stellen en het onderdeel feedback achterwege te laten. De functionaliteiten zijn wel op de product backlog opgenomen, zodat deze op een later moment kunnen worden opgepakt.

5.4 HET SCHRIJVEN VAN INSTRUCTIES MET EEN RICH TEXT EDITOR

Zoals aangeven in hoofdstuk 4.2 is een onderdeel van de beheerapplicatie het kunnen schrijven van gebruikersinstructies met een tekstverwerker, ofwel Rich Tekst Editor.

5.4.1 Waarom een Rich Text Editor?

Bij het uitleveren van een release worden functionele wijzigingen of vernieuwingen vaak in de vorm van instructies in een handleiding aan eindgebruikers aangeleverd.

Deze instructies worden nu op Confluence geschreven en geplaatst, maar dit is niet toereikend voor de nieuwe oplossing omdat er volledige controle nodig is over de gegevens om deze aan de eindgebruikers te kunnen presenteren. Daarnaast is het niet mogelijk om de tekst uit Confluence uit te lezen. Het format van de data wanneer deze via de Confluence API wordt opgevraagd is te specifiek afgestemd op de Confluence front-end, waardoor het niet eenvoudig te gebruiken is in de beheerapplicatie. Dit is handmatig getest door het uitvoeren van verzoeken naar de Confluence API.

Het is daarom het noodzakelijk een Rich Text Editor aan te bieden waarmee de teams van Topicus HAP de instructies in de beheerapplicatie kunnen schrijven.

Aan de editor zijn de volgende eisen gesteld (Joliene Brouwer, persoonlijke communicatie, 25 april 2019):

- Opgemaakte tekst:
  - **Dikgedrukt**
  - *Cursief*
  - Onderstreept
  - Koppen (h1, h2, h3)
- Tabellen
- Invoegen afbeeldingen
- Plakken van afbeeldingen vanaf het klembord
- Embedded YouTube

Vergelijking verschillende opties

Voor de implementatie van een editor zijn enkele opties bekeken, die op basis van de eisen met elkaar zijn vergeleken (zie Tabel 8).

Naast de gestelde functionele eisen spelen er nog drie belangrijke criteria:

- Kosten van het product
- Integratie in een Angular applicatie

Voor elk criterium waar de optie aan voldoet wordt **1** punt toegekend, voor elke **$** wordt 1 punt in mindering gebracht.

Editor	Opgemaakte tekst	Tabellen	Invoegen afbeeldingen	Plakken afbeeldingen	Embedded Youtube	Kosten	Integratie met Angular	Totaal
TinyMCE	X	X	X	X	X	$12 p/maand	X	**-6**
QuillJS	X		X		X	Geen	X	**5**
CKEditor5	X	X	X	X	X	0$ p/maand	X	**7**

Tabel 8 Vergelijkingstabel tekstverwerkers

Prototyping

Om inzicht te krijgen in hoe de verschillende tekstverwerkers zijn in gebruik en om ze zij aan zij te kunnen vergelijken zijn prototypes gebouwd.

Deze prototypes zijn eenvoudige Angular applicaties met een tekstverwerker implementatie op basis van elk van de beschreven editors.

CKEditor5

Op basis van de vergelijking en de eenvoud in gebruik is CKEditor5 de meest geschikte keus voor het implementeren van een Rich Text Editor.

De editor voldoet aan alle gestelde eisen, is te integreren met Angular en biedt een commerciële licentie voor 0$ per maand.

Deze licentie biedt de volgende mogelijkheden:

- 5 actieve gebruikers
- 2 ontwikkelaars
- Te gebruiken voor 1 interne applicatie

5.4.2 Drag and Drop interface voor instructies

Uit gesprekken is gebleken dat het groeperen van instructies op thema en het ordenen van de instructies een proces is dat vooraf aan de release nogal eens veranderd. Dit betekent dat bij het schrijven van de instructies de categorieën mogelijk nog niet bekend zijn (Joliene Brouwer, persoonlijke communicatie, 14 maart 2019).

Bij het schrijven van instructies in Confluence is het mogelijk om deze te knippen en op de juiste plek op de pagina te plakken met een koptekst. Om deze functionaliteit in de beheerapplicatie na te bootsen en gebruikersvriendelijk te houden wordt een drag-and-drop interface ingezet voor categoriseren van aangemaakte instructies. Het aanpassen van de volgorde van de instructies en categorieën en het bewerken van categorietitels zijn functionaliteiten die op een later moment opgepakt wordt. Deze zijn als user stories opgenomen op de product backlog.

Prototyping

Om te beoordelen of het inzetten van een drag-and-drop interface een haalbare optie is wordt een prototype in Angular gebouwd. Voorwaarde is dat de interface in één sprint moet kunnen worden gebouwd.

Aan het prototype zijn de volgende eisen gesteld:

- Drag-and-drop interface wordt gebouwd op basis van bestaande en geteste techniek.
- De techniek moet geneste drag-and-drop ondersteunen (instructies in categorieën en de categorieën zelf)
- De techniek moet voorzien zijn van duidelijke documentatie.
- De techniek mag niet end of life zijn.

Vergelijking

De technieken die ingezet kunnen worden zijn in Tabel 9 opgenomen en tegenover de eisen gezet. Voor elke eis waar de techniek aan voldoet wordt één punt toegekend, waarbij een hoger aantal punten beter is.

Techniek	Getest	Geneste drag-and-drop	Duidelijke documentatie	Niet end of life	Totaal
NgxDragDrop		X		X	2
Angular Drag & Drop					0
Angular Material Drag and Drop	X	X	X	X	4

Tabel 9 Vergelijkingstabel drag and drop technieken

Op basis van deze vergelijking is de Drag-Drop functionaliteit uit de Angular Material package de beste optie. Een bijkomend voordeel is dat Angular Material al in de applicatie wordt gebruikt, wat een extra afhankelijkheid scheelt.

Prototype

Het prototype bestaat uit twee categorieën met elk drie instructies. Deze instructies kunnen tussen de twee categorieën worden verplaatst en ook de volgorde van de categorieën kan worden aangepast.

Het implementeren van de functionaliteit in het prototype heeft slechts één dag gekost. Vanwege dit succes is besloten om de drag-and-drop interface in te zetten voor het instructieoverzicht.

Uitwerking

De uiteindelijke uitwerking van de drag-and-drop interface bevindt zich in het instructieoverzicht in de beheerapplicatie. Zie hiervoor de functionele documentatie.

Via deze interface kunnen instructies worden gecategoriseerd.

5.5 INTEGRATIE VAN EEN WEB COMPONENT IN EEN APACHE WICKET APPLICATIE

In dit hoofdstuk wordt gekeken naar het technische aspect van de vraag ***Hoe kan de eindgebruiker vanuit Topicus HAP de release notes benaderen?*** De wens van de teams van Topicus HAP is dat de eindgebruikers de release notes in Topicus HAP kunnen inzien. Er moet dus op een manier geïntegreerd worden met bestaande software. Omdat zowel de inzageapplicatie als Topicus HAP webapplicaties zijn kan gebruik worden gemaakt van Web Components. Dit hoofdstuk gaat in op de integratie van een Web Component in Topicus HAP een Apache Wicket applicatie. Een uitgebreide uitleg over deze integratie met code voorbeelden is te lezen in de technische documentatie.

De inzageapplicatie is een Angular 7 applicatie die gepubliceerde releases, instructies en release notes aan de eindgebruikers van Topicus HAP kan presenteren.

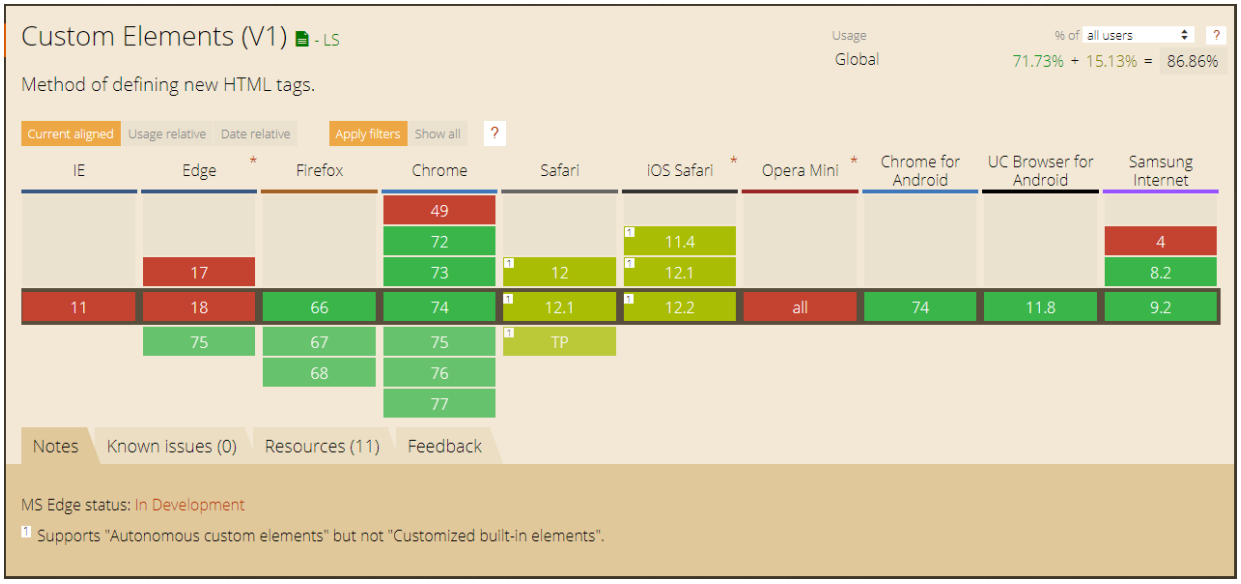
Deze applicatie wordt verpakt als Web Component en geïntegreerd in Topicus HAP op een daarvoor ingerichte pagina. Topicus HAP is een Apache Wicket applicatie.

5.5.1 Wat zijn Web Components

Web Components zijn HTML elementen die niet in de HTML specificatie voorkomen, maar door ontwikkelaars gebouwd kunnen worden met behulp van een set API's en zijn onderdeel van de W3C standaard (W3C, 2019).

Deze componenten zijn volledig ingekapseld, herbruikbaar en worden door de evergreen browsers standaard ondersteund. Met inzet van polyfills kunnen ze ook in andere browsers worden gebruikt, bijvoorbeeld Internet Explorer en Edge. Een component bestaat uit hoofdzakelijk drie componenten: HTML, CSS en Javascript. Het is dus mogelijk om zonder inzet van een front-end framework een Web Component te bouwen (webcomponents.org, sd).

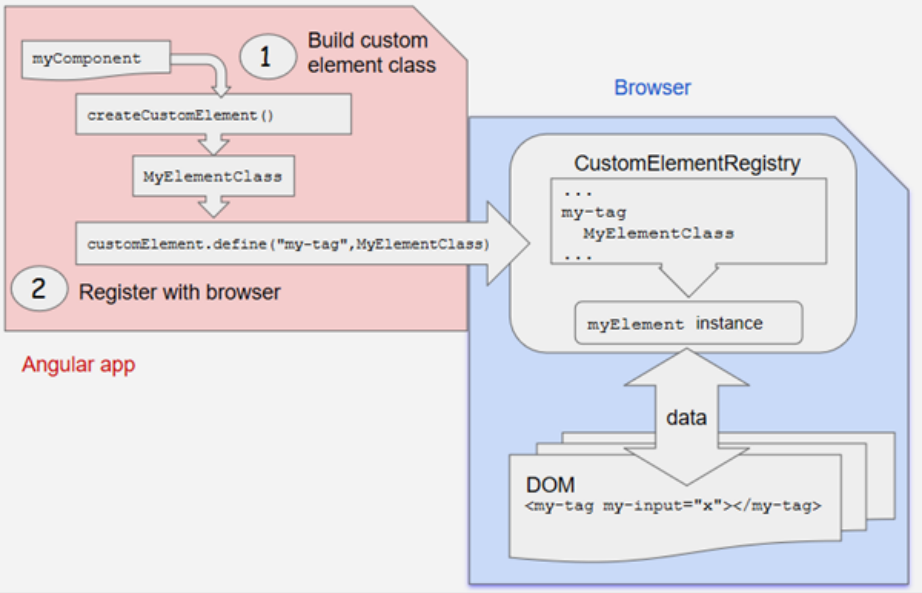
In Figuur 23 is te zien dat Firefox, Chrome en Safari de Custom Elements (V1) spec ondersteunen. Internet Explorer en Edge ondersteunen geen Web Components zonder een EcmaScript 5 polyfill.



Figuur 23 CanIUse Web Components (Deveria, 2019)

5.5.2 Angular en Web Components

Angular biedt de mogelijkheid om componenten of zelfs volledige applicaties tot Web Component te verpakken vanaf de introductie van Angular Elements in versie 6. Angular Elements is de schakel tussen het Angular framework en de Web Component APIs. De manier waarop dit gebeurd is afgebeeld in Figuur 24.



Figuur 24 Angular Elements (Angular Team, 2019)

5.5.3 Integratie in Topicus HAP

Bij het integreren van een Web Component in Topicus HAP zijn enkele stappen gemaakt die hieronder worden beschreven.

Inzageapplicatie verpakken als Web Component

Dependencies *installeren*

Om een Angular applicatie als Web Component te kunnen inzetten moeten allereerst een aantal dependencies worden geïnstalleerd

Package	Taak
@angular/elements	**Angular Elements** is de schakel tussen het Angular framework en de Web Component APIs.
document-register-element	**document-register-element** is een standalone implementatie van de Web Components specificatie die door Angular Elements gebruikt wordt om de applicatie in de DOM te registreren.
@webcomponents/custom-elements	**@webcomponents/custom-elements** is een polyfill voor browsers die de Web Component v1 specificatie nog niet ondersteunen.
ngx-build-plus	**ngx-build-plus** verrijkt de standaard buildtooling van de Angular CLI voor het builden van het Web Component en maakt het mogelijk om te packagen naar een enkele bundle file. Daarnaast biedt het de mogelijkheid om bepaalde gedeelde dependencies naar een los bestand te transpileren zodat meerdere Angular Web Components hier gebruik van kunnen maken.

Tabel 10 Dependencies voor Angular Web Component

Aanpassen root module

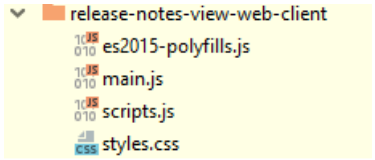
Vervolgens moet de root module van de inzageapplicatie worden aangepast. De applicatie wordt namelijk niet meer door Angular opgestart in de browser, maar gedefinieerd als custom HTML element.

Builden

Daarna kan dankzij *ngx-build-plus* de applicatie worden verpakt tot drie Javascript bestanden en één CSS bestand.

Resultaat

Het resultaat van de build zijn de volgende minified files:

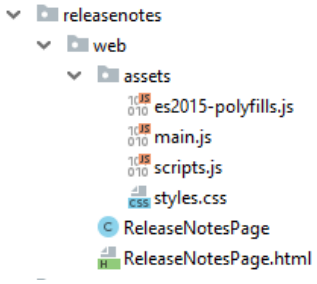
	File	Bevat
	es2015-polyfills.js	Polyfills voor EcmaScript2015 (ES6). Deze zijn nodig om te zorgen dat het element werkt in browsers die nog geen ES6 ondersteunen.
	main.js	De Angular inzageapplicatie.
	scripts.js	Gedeelde dependencies die hergebruikt kunnen worden door andere Angular Web Components.
	styles.css	Alle stylesheets van de applicatie

5.4.4 Pagina aanmaken in Topicus HAP waar het Web Component gehost kan worden

Om het Web Component te kunnen tonen in Topicus HAP moet er een nieuwe Wicket pagina worden aangemaakt die de files van het component inlaadt en het HTML element aan de DOM toevoegt.

Release notes package

In het HapTotaal project is de volgende package toegevoegd met de build files van het Web Component in de assets package en een ReleaseNotesPage in de web package.



ReleaseNotesPage

De ReleaseNotesPage is een Wicket pagina die vanuit het menu van een ingelogde gebruiker wordt geopend. Wanneer de paginaheader wordt opgebouwd door Wicket worden de assets van het Web Component toegevoegd. De ReleaseNotesPage.html bevat het Web Component HTML element.

Content Security Policy

De inzageapplicatie haalt data op via de REST end-points van de release-notes backend applicatie en is daarmee in grote mate geïsoleerd van de applicatie die het component host.

Echter staat HAP niet standaard toe dat er verbinding mag worden gemaakt met externe APIs, dit moet expliciet aan de configuratie van HAP worden meegegeven. Hiervoor wordt gebruik gemaakt van de Content Security Policy header op uitgaande HTTP verzoeken.

Deze configuratie is vastgelegd in de database van HAP en verschilt per omgeving (DEVELOPMENT, TEST, ACCEPTATIE en PRODUCTIE). In dit geval moet het voor de DEVELOPMENT omgeving worden bijgewerkt met de API end-points van de release note backend, om het Proof of Concept in een ontwikkelomgeving aan te kunnen tonen.

De volgende CSP header directives moeten worden aangepast om alle content te kunnen presenteren (Foundeo Inc, 2016):

- connect-src - Nodig om HTTP requests uit te kunnen voeren naar externe services
- img-src - Nodig om instructieafbeeldingen op te kunnen halen
- child-src - Nodig om iframes te kunnen renderen (embedded YouTube)

Hiertoe is het volgende SQL script opgesteld:

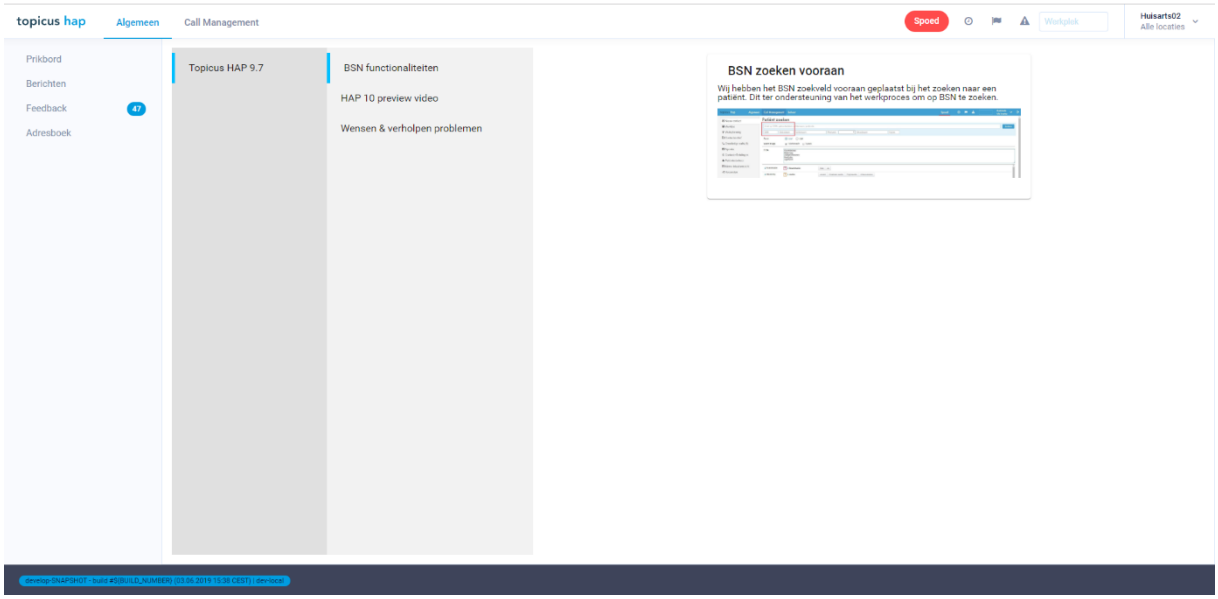
```
UPDATE hap.hapt_jettyconfiguratie

SET value = CONCAT(value, 'connect-src http://localhost:8080/api/view/; img-src http://localhost:8080/api/images/; child-src https://www.youtube.com;')

WHERE omgeving = 'DEVELOPMENT' AND key = 'header/Content-Security-Policy'
```

5.5.5 Presenteren van de release notes

Na deze wijzigingen is het mogelijk om gepubliceerde release notes en instructies inclusief afbeeldingen en embedded YouTube video’s in Topicus HAP te presenteren via het Web Component.



6. CONCLUSIE EN AANBEVELINGEN

In dit hoofdstuk wordt antwoord gegeven op de hoofdvraag ***Hoe kan het ontwikkelen van een webapplicatie de ontwikkelteams van HAP helpen om eindgebruikers beter te kunnen informeren over veranderingen in Topicus HAP?***

Daarnaast worden een aantal punten aangedragen waarop het Proof of Concept in de toekomst verbeterd zou kunnen worden.

6.1 CONCLUSIE

Het doel van de afstudeeropdracht was om een nieuwe oplossing te vinden voor het verspreiden van release notes aan de eindgebruikers van Topicus HAP, omdat deze niet altijd goed geïnformeerd zijn over wijzigingen in Topicus HAP. Na gesprekken met werknemers bleek dat release notes uit twee delen bestaan: release notes en gebruikersinstructies die via een pagina op Confluence beschikbaar worden gesteld.

De oplossing voor het probleem is uiteengezet in een Proof of Concept met drie onderdelen: een beheerapplicatie voor de teams van Topicus HAP, een inzageapplicatie voor de eindgebruikers van Topicus HAP en een back-end applicatie die de verbinding verzorgd tussen de twee applicaties. Elk onderdeel is voorzien van technische en functionele documentatie zodat de teams er in de toekomst verder aan kunnen ontwikkelen.

De beheerapplicatie is een webapplicatie gebouwd in Angular. Met de beheerapplicatie kunnen de teams van Topicus HAP hun releases, release notes uit Jira en gebruikersinstructies beheren en bij een nieuwe release kunnen publiceren.

De inzageapplicatie is een webapplicatie gebouwd in Angular en als Web Component geïntegreerd in Topicus HAP. Met deze inzageapplicatie kunnen eindgebruikers van Topicus HAP alle gepubliceerde release notes en gebruikersinstructies in hun vertrouwde omgeving kunnen lezen en zich kunnen informeren over de wijzigingen.

De back-end applicatie is een Spring Boot applicatie verdeeld in verschillende modules met ieder hun eigen taak. De back-end applicatie ontsluit de functionaliteiten uit het systeem via REST API’s aan de twee webapplicaties. Daarnaast legt de back-end applicatie een verbinding met Jira om de release notes die hierin geschreven zijn op te halen. Ook beschikt de back-end over een eigen database waarin de gegevens uit de beheerapplicatie worden opgeslagen. Deze gegevens worden na publicatie beschikbaar gesteld aan de inzageapplicatie.

Met het afronden van de afstudeeropdracht beschikken de teams over een Proof of Concept waarmee aangetoond wordt dat zij release notes en gebruikersinstructies op een vernieuwde manier kunnen uitleveren aan de eindgebruikers van Topicus HAP.

6.2 AANBEVELINGEN

Omdat de afstudeerperiode een einddatum heeft is tijdens de sprintplanningen nauwkeurig gekeken naar de prioriteit van alle beoogde functionaliteit. De functionaliteiten die niet zijn afgerond tijdens de afstudeerperiode zijn als user stories op de product backlog opgenomen en kunnen in toekomstige sprints worden opgepakt.

Ook is in verband met de tijd niet altijd de meest optimale technische oplossing gekozen voor een probleem. In dit hoofdstuk worden een aantal punten uiteengezet waar het Proof of Concept in de toekomst op kan worden verbeterd.

6.2.1 Afhandelen van geüploade afbeeldingen

In de meest ideale situatie moet het uploaden van afbeeldingen en het serveren hiervan worden afgehandeld door een dienst die hierin gespecialiseerd is. Het opslaan van afbeeldingen bij een cloud-dienst als S3 van Amazon heeft als voordeel dat de afbeeldingen losstaan van de applicatieserver. Wanneer de applicatieserver verplaatst, of opnieuw wordt uitgerold hoeven niet alle bestaande afbeeldingen mee verplaatst te worden.

Echter moet ook gekeken worden naar de haalbaarheid van de verschillende implementaties in het kader van een Proof of Concept bij een afstudeerproject. Hierdoor valt een Cloud Storage optie af vanwege de kosten die er bij komen kijken en de onzekerheid van de toekomst van het Proof of Concept. Maar de applicatie moet wel mogelijkheden bieden voor het uploaden en serveren van afbeeldingen. Om dit te kunnen aanbieden is een minimale implementatie geschreven die overweg kan met geüploade afbeeldingen en deze serveren vanaf het bestandssysteem. Echter wordt geen rekening gehouden met het maximale aantal bestanden in een directory of andere beperkingen die zich voor kunnen doen.

In de toekomst is het aan te raden de upload implementatie te herzien en om te zetten naar een gespecialiseerde dienst cloud-dienst.

6.2.2 Beveiligen van de API end-points

Voor het Proof of Concept is besloten het onderwerp security buiten scope te laten. De enige plek waar beveiliging is toegepast is de verbinding tussen de back-end applicatie en de Jira REST API, omdat anders geen verbinding kan worden gemaakt.

Er is wel een inventarisatie gemaakt van de verschillende verbindingspunten in de applicatie die in ieder geval met authenticatie beveiligd moeten worden.

Back-end ↔ Jira REST API

De Jira REST API biedt twee opties om verbinding te kunnen maken:

- Basic Authentication (https://developer.atlassian.com/server/jira/platform/basic-authentication/)
- OAuth 1.0a (https://developer.atlassian.com/server/jira/platform/oauth/)

OAuth is veiliger, er worden immers geen encoded credentials met de HTTP verzoeken meegestuurd, maar de keerzijde is dat het implementeren van deze authenticatie flow zeer uitgebreid is en aanpassingen vergt in de configuratie van Jira.

Een externe applicatie (in dit geval de back-end applicatie) die via deze weg verbindt moet namelijk worden aangemeld in Jira waar beheerrechten voor nodig zijn. Voor het Proof of Concept gekozen om Basic Authentication in te zetten vanwege de eenvoudige implementatie en omdat het om een interne applicatie gaat waarbij de HTTP verzoeken binnen het Topicus netwerk blijven.

Basic Authentication

Om basic authentication toe te kunnen passen moet in Jira een account bestaan met rechten om projecten in te zien waarvoor queries worden uitgevoerd. In dit geval zijn dat de projecten AZ en ZCT.

De gebruikersnaam en het wachtwoord moeten in dit formaat *username:password* Base64 encoded worden. Tijdens de ontwikkeling heb ik hiervoor mijn eigen Topicus account gebruikt.

Aan ieder uitgaand HTTP verzoek naar de Jira REST API wordt de volgende header toegevoegd:

```
Authorization: Basic <encoded credentials>
```

Deze beveiliging is voor een interne applicatie voldoende om verbinding te leggen met Jira.

Beheer webapplicatie ↔ Beheer API

De beveiliging tussen de beheerapplicatie en de beheer API kan eenvoudig worden opgezet met Bearer Authentication in de vorm van OAuth 2.0 en OpenIDConnect (OIDC) met bijvoorbeeld Keycloak. Keycloak is een open source Identity Provider die het mogelijk maakt om beveiliging voor applicaties centraal te regelen.

Keycloak is eenvoudig te integreren in Spring Boot via de adapter die Keycloak beschikbaar stelt voor Spring Boot applicaties.

Integratie met Angular applicaties is mogelijk met behulp van de beschikbare JavaScript OIDC client.

Beheer webapplicatie ↔ Image API

De POST endpoint wordt door de beheerapplicatie gebruikt om afbeelding te uploaden. Deze kan ook via Keycloak worden beveiligd met Bearer Authentication.

Inzage webapplicatie ↔ Inzage API

De beveiliging tussen de inzage webapplicatie en de Inzage API kan worden opgelost met behulp van API keys.

De inzage webapplicatie wordt met Topicus HAP gecompileerd en kan worden voorzien van een API key. Deze API key kan gebruikt worden om de endpoints van de Inzage API aan te roepen. Deze API ondersteunt enkel GET requests en biedt geen mogelijkheid om gegevens in het systeem aan te passen. Echter is de API key in de browser waar Topicus HAP draait uit te lezen, waardoor het mogelijk is om buiten de inzageapplicatie om requests uit te voeren. Ondanks dat de API key enkel uit een browser van een huisartsenpost uit de browser worden uitgelezen is deze oplossing niet waterdicht.

Het voordeel van deze opzet is wel dat de applicatie die het component host slechts minimaal aangepast hoeft te worden en toch een manier biedt om de Inzage API af te schermen van verzoeken die niet van een API key zijn voorzien.

Om de API key implementatie dicht te maken zonder dat er API keys uitgelezen kunnen worden in de browser moeten aanpassingen in zowel de inzageapplicatie als de host applicatie worden gedaan. Dit kan alsvolgt:

- Ieder HTTP verzoek vanuit het Web Component wordt via een interceptor als event naar de host applicatie gestuurd.
- De events worden door de host applicatie ontvangen.
- De host applicatie is server-side voorzien van een API key waarmee met de API end-points kan worden gecommuniceerd.
- De host applicatie voert namens het Web Component de HTTP requests uit en voorziet deze van de aanwezige API key.
- Het resultaat van de requests wordt via de host applicatie aan het Web Component teruggegeven.

Op deze manier verlaat de API key nooit de server en is het niet mogelijk om buiten de inzageapplicatie om requests uit te voeren.

De functionaliteit voor het uitvoeren van de HTTP requests die voorzien worden van een API key in de host applicatie kan worden verpakt in SDK’s voor verschillende programmeertalen. Dit maakt het mogelijk een generieke oplossing aan te kunnen bieden die door meerdere applicaties gebruikt zou kunnen worden.

Inzage webapplicatie ↔ Image API

De GET endpoint wordt door de inzageapplicatie gebruikt om instructieafbeeldingen te downloaden. Ook dit endpoint kan met een API key worden beveiligd.

BIJLAGEN

BIJLAGE 1 – REQUIREMENTS

***** *Deze functionaliteiten zijn enkel mogelijk indien de bovenliggende release de status CONCEPT heeft.*

1.1 Beheren van releases

#	Requirement	MoSCoW	Done
Release-F01	Er kan een concept release worden aangemaakt in de beheerapplicatie	**M**ust	**X**
Release-F02*	De volgende velden van een concept release kunnen worden bewerkt in de beheerapplicatie: • Titel • Jira label	**C**ould	
Release-F03*	Een concept release kan worden verwijderd in de beheerapplicatie	**S**hould	
Release-F04	Er kan een lijst met aangemaakte concept releases worden getoond in de front-end applicatie	**M**ust	**X**
Release-F05	De details van een concept release kan worden ingezien in de front-end applicatie	**M**ust	**X**

1.2 Schrijven van instructies

#	Requirement	MoSCoW	Done
Instructies-F01*	Er kan in de beheerapplicatie een nieuwe instructie worden aangemaakt onder een concept release met de volgende velden: • Titel • Tekst	**M**ust	**x**
Instructies-F02*	De instructie kan in de beheerapplicatie worden voorzien van opgemaakte tekst, die over de tijd bewerkt kan worden.	**M**ust	**x**
Instructies-F03*	De titel van een instructie kan worden bewerkt in de beheerapplicatie.	**M**ust	**x**
Instructies-F04*	Er kunnen afbeeldingen worden toegevoegd in de tekst van een instructie in de beheerapplicatie.	**S**hould	**x**
Instructies-F05*	Er kunnen embedded YouTube filmpjes worden toegevoegd in de tekst van een instructie in de beheerapplicatie.	**C**ould	**x**
Instructies-F06*	Er kunnen instructiecategorieën worden aangemaakt in de beheerapplicatie met het veld: • Titel	**M**ust	**x**
Instructies-F07*	Instructies kunnen aan een categorie worden toegevoegd in de beheerapplicatie.	**M**ust	**x**
Instructies-F08*	De titel van een instructiecategorie kan worden bewerkt in de beheerapplicatie.	**C**ould	
Instructies-F09*	De volgorde van de instructies in een categorie kan worden aangepast in de beheerapplicatie.	**C**ould	
Instructies-F10*	De volgorde van de instructiecategorieën kan worden aangepast in de beheerapplicatie.	**C**ould	

Instructies-F11*	Een aangemaakte instructie kan worden verwijderd in de beheerapplicatie.	**W**on’t	
Instructies-F12*	Een aangemaakte instructiecategorie kan worden verwijderd in de beheerapplicatie.	**W**on’t	
Instructies-F13*	Opgemaakte tekst bestaat uit: • Dikgedrukt, cursief en onderstreept • Koppen • Opsommingstekens • Nummering	**M**ust	**x**

1.3 Publiceren van releases

Functioneel

#	Requirement	MoSCoW	
Publiceren-F01*	Een concept release kan in de beheerapplicatie worden gepubliceerd waarmee alle hieraan gekoppelde instructies en release notes beschikbaar worden voor inzage.	**M**ust	**x**
Publiceren-F02	Een gepubliceerde release is voor maximaal een half jaar inzichtelijk voor eindgebruikers van Topicus HAP.	**C**ould	**x**
Publiceren-F03	Alle gepubliceerde releases zijn in een lijst beschikbaar in de beheerapplicatie.	**M**ust	**x**
Publiceren-F04	Een gepubliceerde release is in te zien in de beheerapplicatie.	**M**ust	**x**
Publiceren-F04	De gegevens van een gepubliceerde release zijn niet te bewerken.	**M**ust	**x**
Publiceren-F05	Instructies die bij een gepubliceerde release horen zijn niet te bewerken.	**M**ust	**x**
Publiceren-F06	Bij het publiceren van een release worden alle release notes zoals deze aanwezig zijn in Jira opgehaald en opgeslagen in de database van de beheerapplicatie.	**M**ust	**x**

Technisch

#	Requirement	MoSCoW	
Publiceren-T01	De gepubliceerde releases zijn via een aparte web-api te benaderen.	**M**ust	**x**
Publiceren-T02	De endpoints in de web-api voor inzage zijn middels bearer token authenticatie beveiligd.	**W**on’t	

1.4 Inzage gepubliceerde releases

#	Requirement	MoSCoW	
Inzage-F01	Gepubliceerde releases zijn tot een half jaar na publicatie te benaderen via de inzageapplicatie.	**M**ust	**x**
Inzage-F02	Gepubliceerde instructies en categorieën zijn onder een release beschikbaar in de inzageapplicatie	**M**ust	**x**
Inzage-F03	Gepubliceerde release notes zijn onder een release beschikbaar in de inzageapplicatie	**M**ust	**x**

1.5 Koppelen met bestaande bron release notes

Functioneel

#	Requirement	MoSCoW	Done
Release-Notes-F01	Er kan een Jira label worden gekoppeld aan een concept release in de beheer applicatie	**M**ust	**X**
Release-Notes-F02	Jira issues kunnen automatisch worden opgehaald op basis van een jira project configuratie en het label van een concept release	**M**ust	**X**
Release-Notes-F03	De informatie uit een Jira Issue kan worden overgezet naar het release note model zoals gedefinieerd in de beheerapplicatie	**M**ust	**X**
Release-Notes-F04	Er wordt bij het aanmaken van een release note onderscheid gemaakt tussen een feature en een bugfix op basis van het Jira Issue Type	**M**ust	**X**
Release-Notes-F05	De release notes die op basis van het zoekresultaat uit Jira zijn aangemaakt kunnen in de front-end applicatie worden getoond in het detailscherm van de release waarbij deze release notes horen.	**M**ust	**X**
Release-Notes-F06	De jira project configuratie uit de database kan worden aangepast in de front-end applicatie.	**W**on’t	
Release-Notes-F07	De configuratie die gebruikt wordt voor het ophalen van de issues uit Jira is te kiezen in de front-end applicatie.	**C**ould	

Technisch

#	Requirement	MoSCoW	Done
Release-Notes-T01	Er kan in de database een applicatie geconfigureerd worden waaronder de jira projectconfiguraties worden opgenomen.	**M**ust	**X**
Release-Notes-T02	Er kan in de database worden geconfigureerd in welke Jira projecten naar issues moet worden gezocht	**M**ust	**X**
Release-Notes-T03	Er kan in de database worden geconfigureerd welke Jira Issue Types moeten worden meegenomen in het zoekresultaat	**M**ust	**X**
Release-Notes-T04	Er kan in de database worden geconfigureerd welke velden uit een Jira Issue worden overgezet naar velden uit het Release Note model	**M**ust	**X**
Release-Notes-T05	Er kan in de database worden geconfigureerd welke type Release Note hoort bij een Jira Issue Type	**M**ust	**X**

BIJLAGE 2 – BRAINSTORMSESSIES

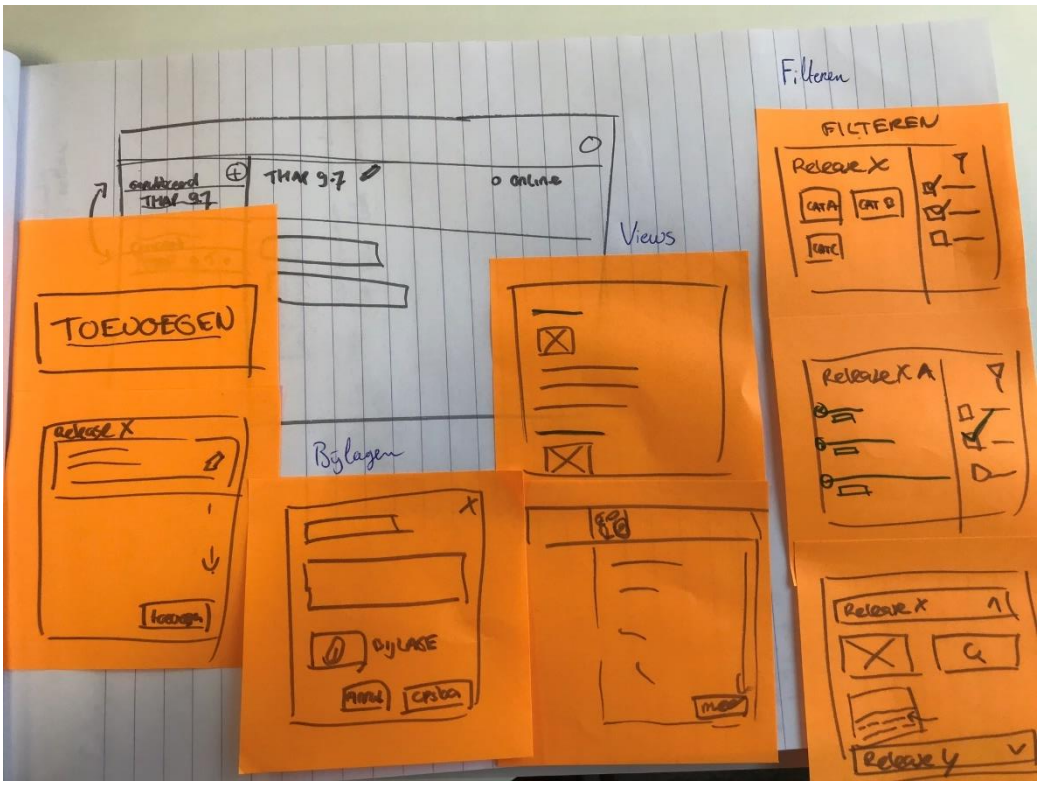
Brainstormsessie: inhoud beheerpaneel

Deze eerste brainstormsessie is gehouden op donderdag 7 maart 2019 met Lisette Heerink, UI/UX designer bij Team Zorgcoördinatie.

Het doel van deze sessie was om een begin te maken aan het antwoord op de eerste vraag: ***Welke functionaliteiten moet het beheerpaneel in de webapplicatie bevatten om een set release notes te kunnen verwerken?***

Het beheerpaneel is de webapplicatie waarmee de teamleden van Topicus HAP hun release notes kunnen verwerken.

Op aanraden van Lisette hebben we gebruik gemaakt van ‘paper prototyping’ (Figuur 25) een techniek waarmee je met pen, papier en post-its snel een aantal verschillende interface designs in elkaar zet. Dit heeft als voordeel dat je niet gebonden bent aan de beperkingen van een ontwerptool bij de eerste wireframes. Wanneer je werkt met een ontwerptool voor de eerste versies van je ontwerp is het makkelijk om hier veel tijd aan te besteden terwijl het juist de bedoeling is om snel te kunnen beoordelen wat werkt en wat niet.



Figuur 25 Paper prototyping

Het resultaat van de sessie is dat het beheerpaneel de volgende functionaliteiten moet bevatten:

- **Aanmaken/bewerken release**

Een teamlid van Topicus HAP moet in het beheerpaneel een release van Topicus HAP kunnen aanmaken en bewerken zodat hieraan release notes kunnen worden toegevoegd.

- **Weergeven releases**

De aangemaakte releases moeten inzichtelijk zijn in het beheerpaneel.

- **Aanmaken/bewerken release notes**

De teamleden moeten de mogelijkheid hebben om in het beheerpaneel hun release notes te schrijven en te bewerken bij een release.

- **Weergeven release notes**

De aangemaakte release notes moeten inzichtelijk zijn in het beheerpaneel.

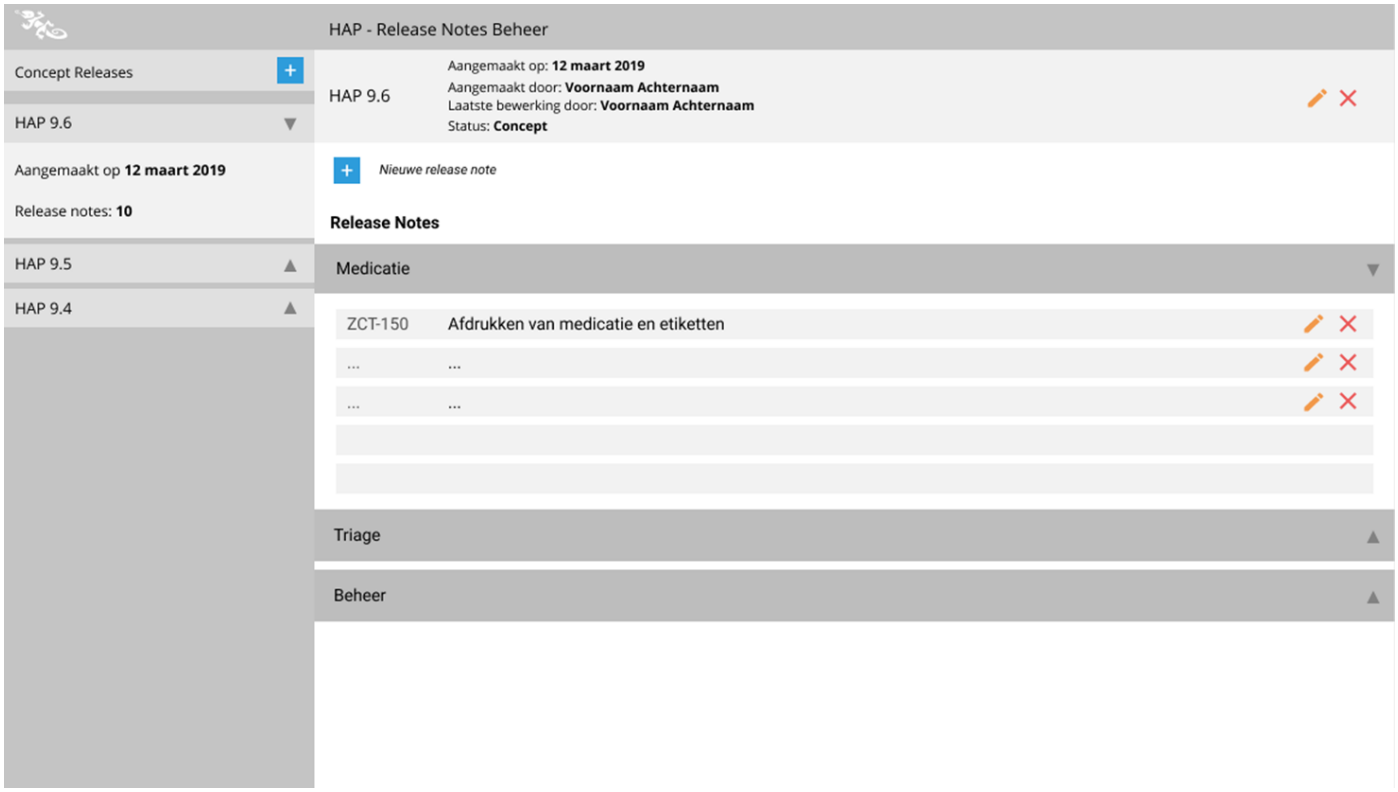
- **Categoriseren van release notes**

De teamleden moeten de mogelijkheid hebben om de release notes te verdelen in de categorieën: triage (bestemd voor triagisten), medicatie (bestemd voor huisartsen), beheer (bestemd voor beheerders van Topicus HAP).

- **Verwijderen van een release/release note**

De aangemaakte releases en release notes moeten verwijderd kunnen worden.

Na afloop van de sessie heb ik met behulp van Figma, een online ontwerp tool, de schetsen vertaald naar een mockup van het beheerpaneel (zie Bijlage 3 – Dashboard mockups). De lijst met functionaliteiten heb ik omgezet naar stories die ik aan de backlog in Jira heb toegevoegd. De requirements die hierbij horen zijn te vinden in Bijlage 1 – Requirements.



Figuur 26 Beheerpaneel mockup

Brainstormsessie: verwerken release notes en instructies

Deze tweede brainstormsessie is gehouden op donderdag 14 maart 2019 met Joliene Brouwer, product owner bij Team Acute Zorg. Joliene schrijft release notes voor Topicus HAP en heeft daarmee inhoudelijke kennis over dit proces. Met haar inzicht wordt het mogelijk om de functionaliteiten die het beheerpaneel moet bevatten verder te definiëren.

Het doel van deze sessie was enerzijds om de resultaten van de eerste brainstormsessie te valideren. Anderzijds om tot een definitief antwoord te komen op de eerste vraag: ***Welke functionaliteiten moet het beheerpaneel in de webapplicatie bevatten om een set release notes te kunnen verwerken?***

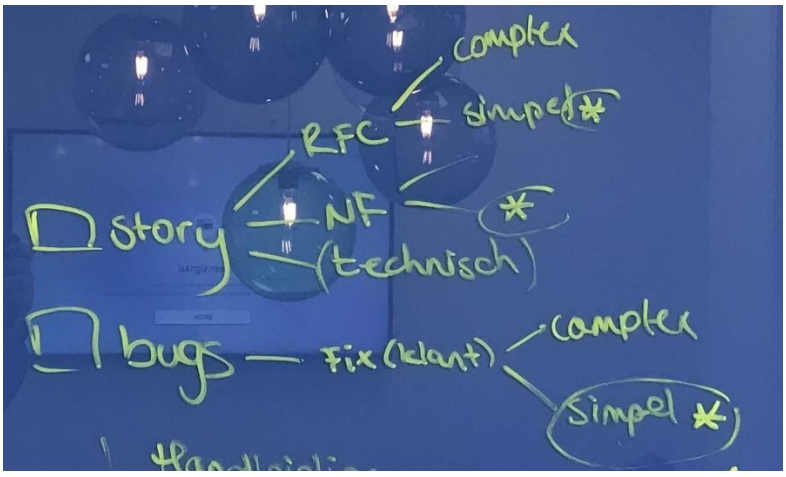
In de vorige sessie met Lisette Heerink is besloten dat release notes in de beheerapplicatie worden geschreven. Nadat ik de schermontwerpen en functionaliteiten aan Joliene had voorgelegd was haar oordeel dat dit niet overeenkwam met de wensen van de teams.

Daarna schetste ze hoe release notes tot stand komen (Figuur 27). Hierbij wordt duidelijk onderscheid gemaakt tussen release notes voor stories (nieuwe functionaliteiten of functionele wijzigingen) en bugs.

Daarnaast wordt er onderscheid gemaakt tussen simpele (aangegeven met een *) en complexe release notes. De simpele release notes worden bij een issue in Jira vastgelegd en worden niet gecategoriseerd op bijvoorbeeld triage, medicatie of beheer.

De complexe release notes, ofwel instructies, worden op een aparte pagina op Confluence geschreven. Deze pagina is inzichtelijk voor klanten. Het doel van instructies is om eindgebruikers te informeren over hoe bepaalde functionaliteiten in gebruik moeten worden genomen. De instructies bestaan uit opgemaakte tekst en eventueel plaatjes of YouTube video’s. Opgemaakte tekst moet tenminste bestaan uit: kopjes, dikgedrukt, cursief, bullet-lijsten, genummerde lijsten en tabellen.

De instructies moeten wel gecategoriseerd kunnen worden, maar dan op functioneel onderwerp in plaats van op gebruikersrol. Bijvoorbeeld alle instructies die over inloggen gaan groeperen onder ‘Inloggen’.



Figuur 27 Totstandkomen release notes

Joliene gaf ook aan dat het schrijven van de instructies een continu proces is, waarbij de titels, inhoud en categorieën van de instructies kunnen wijzigen voordat ze gepubliceerd worden.

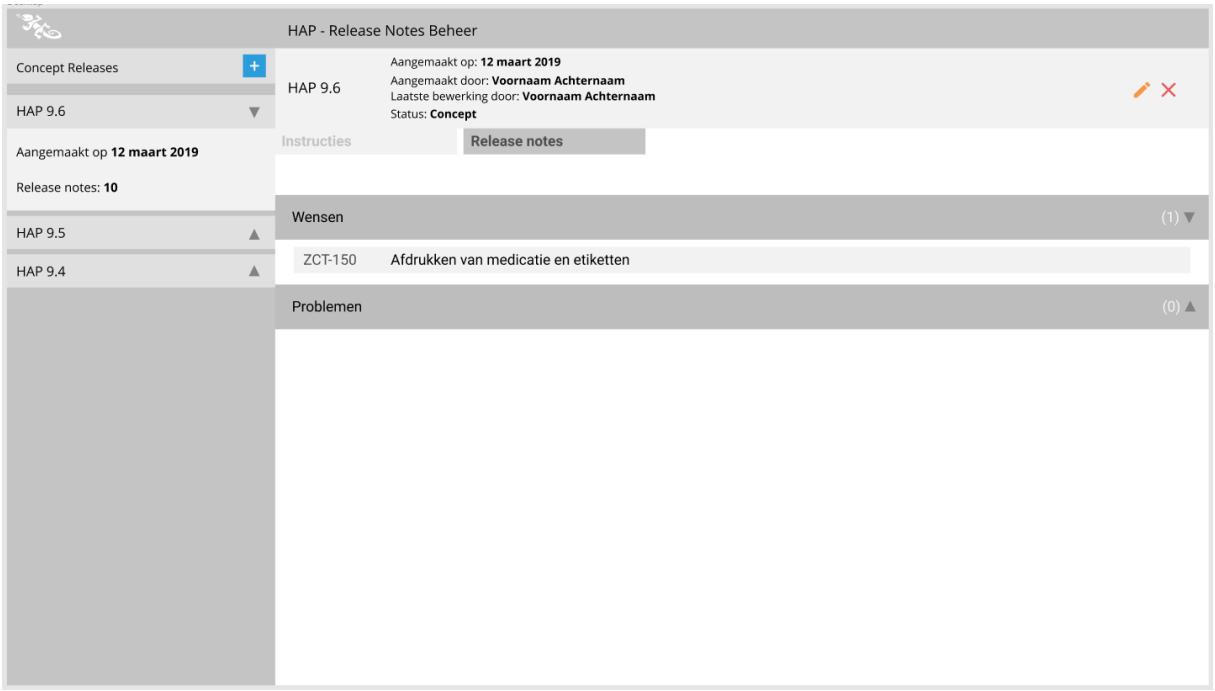
Conclusie

Op basis van de uitkomsten van dit gesprek moeten de schermontwerpen en geplande functionaliteiten worden aangepast om tot een oplossing te komen die aansluit bij de wensen van de teams.

Er moet in de release note applicatie worden onderscheid gemaakt tussen release notes en instructies.

Release notes

Release notes zijn onderdeel van een issue en bevinden zich in Jira en moeten daar blijven. Om deze release notes aan eindgebruikers te kunnen presenteren zal er een koppeling moeten worden gelegd met Jira. Release notes worden onderverdeeld in twee categorieën: wensen en opgeloste problemen. Het schermontwerp voor release notes uit de eerste brainstormsessie is daarmee aangepast.

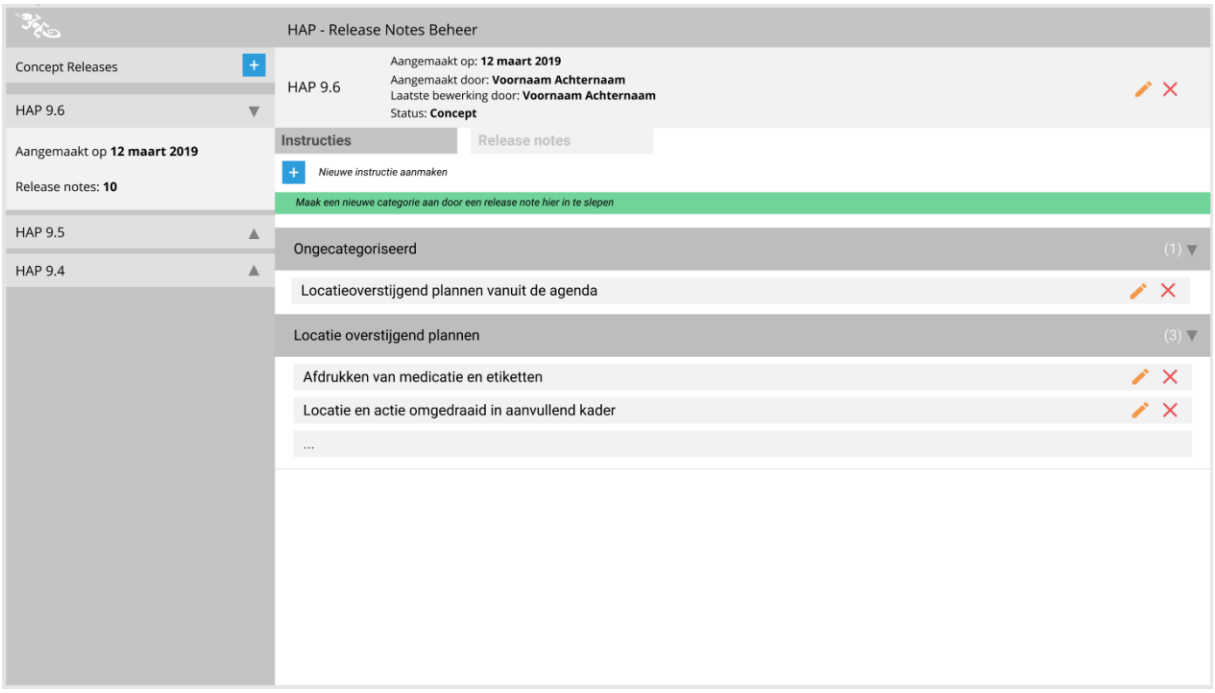


Figuur 28 Mockup release notes

Het is nu niet meer mogelijk om release notes aan te maken. Deze worden rechtstreeks uit Jira gehaald en gepresenteerd in de beheerapplicatie.

Instructies

Instructies zijn handgeschreven stukken tekst, soms voorzien van afbeeldingen of YouTube video’s die een gebruiker uitleg geven over hoe bepaalde functionaliteiten in HAP werken. Deze instructies moeten over de tijd bewerkt kunnen worden totdat ze gepubliceerd worden. Daarnaast worden instructies gegroepeerd op een functioneel thema.



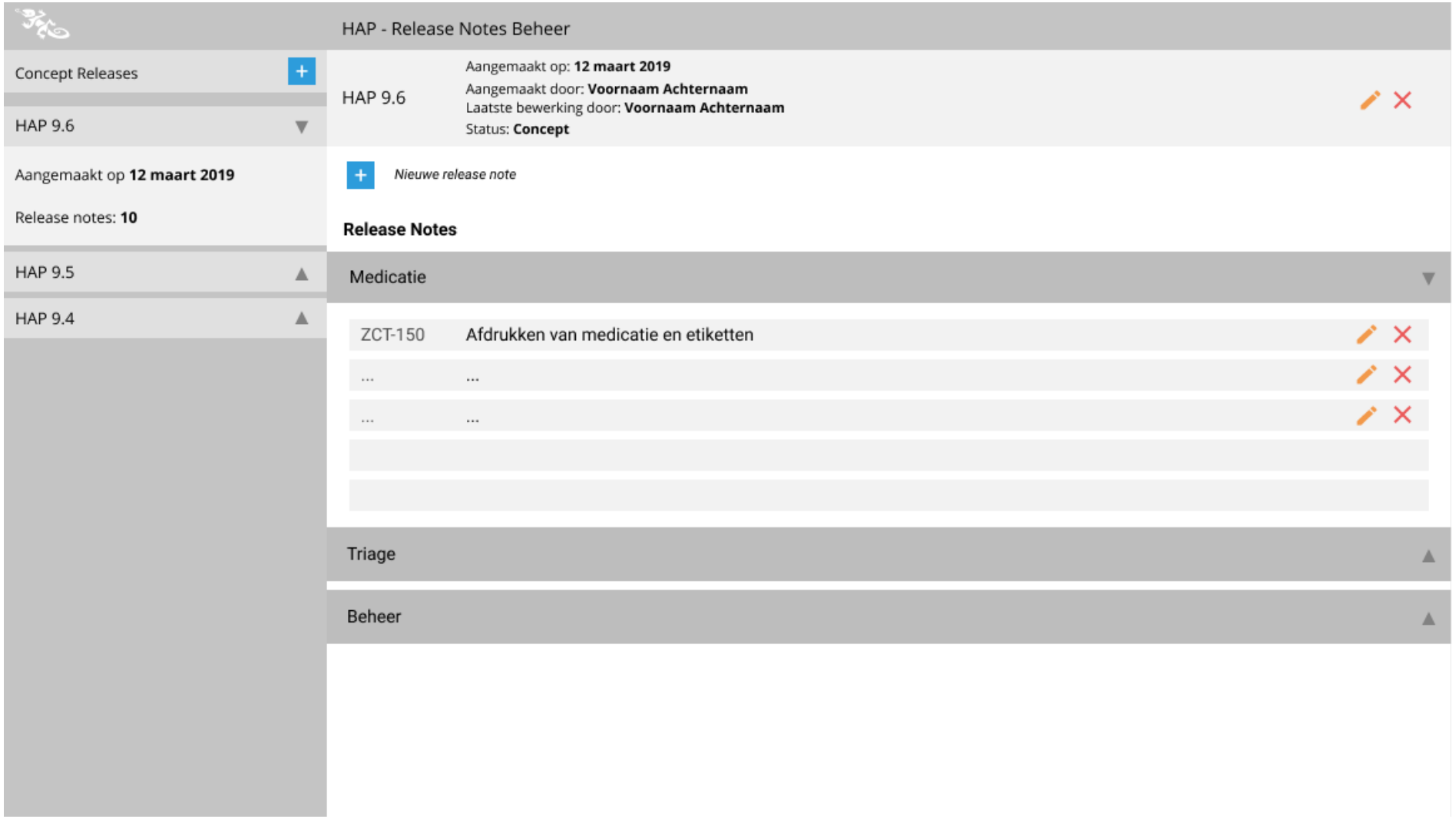
Figuur 29 Mockup instructieoverzicht

In Figuur 29 is het instructieoverzicht te zien. Hier kunnen nieuwe instructies worden aangemaakt, en via een drag en drop interface worden gecategoriseerd. De instructies worden geschreven in een tekstverwerker (Figuur 30).

Figuur 30 Mockup tekstverwerker instructie

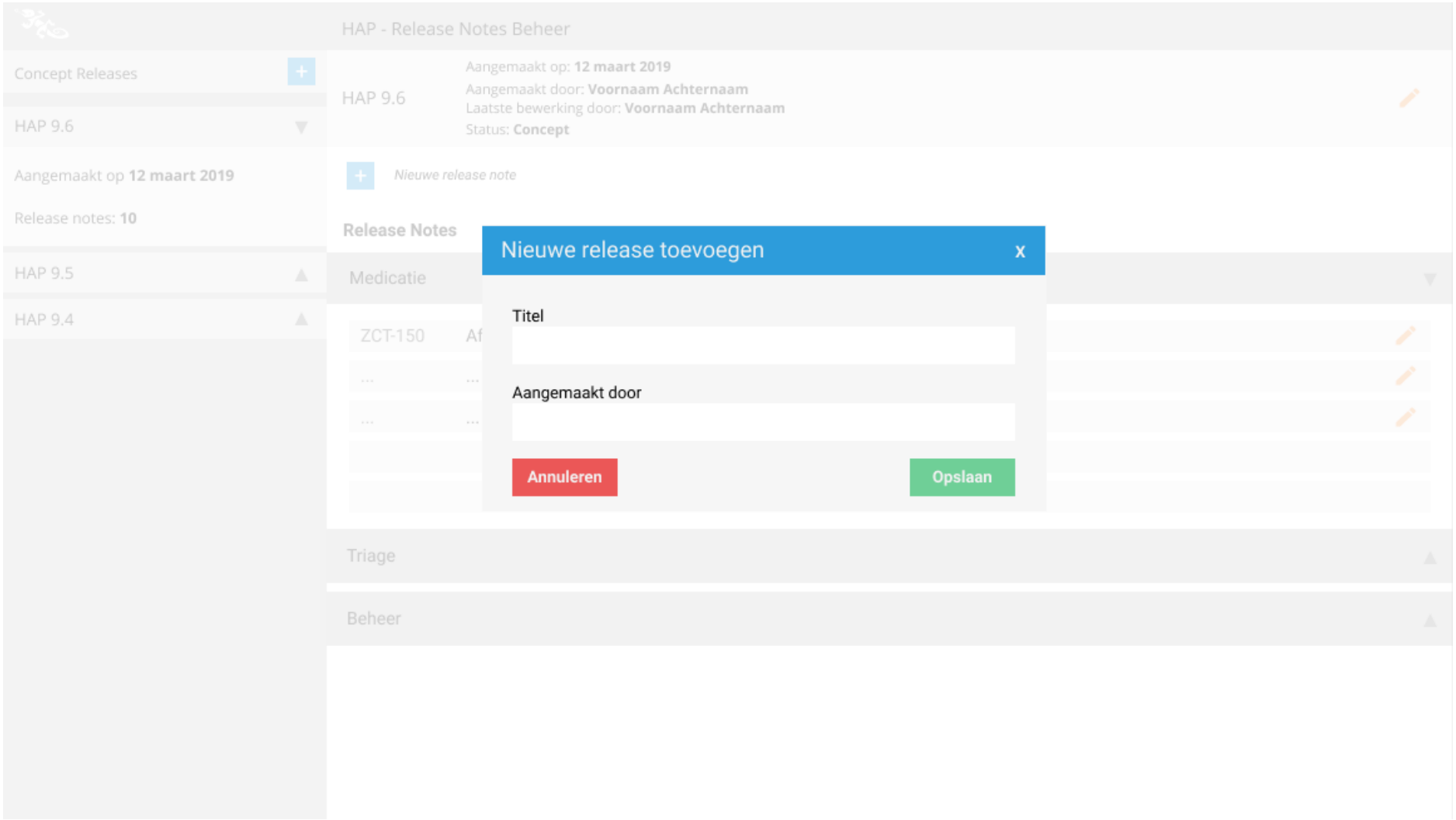
BIJLAGE 3 – DASHBOARD MOCKUPS

1. Dashboard



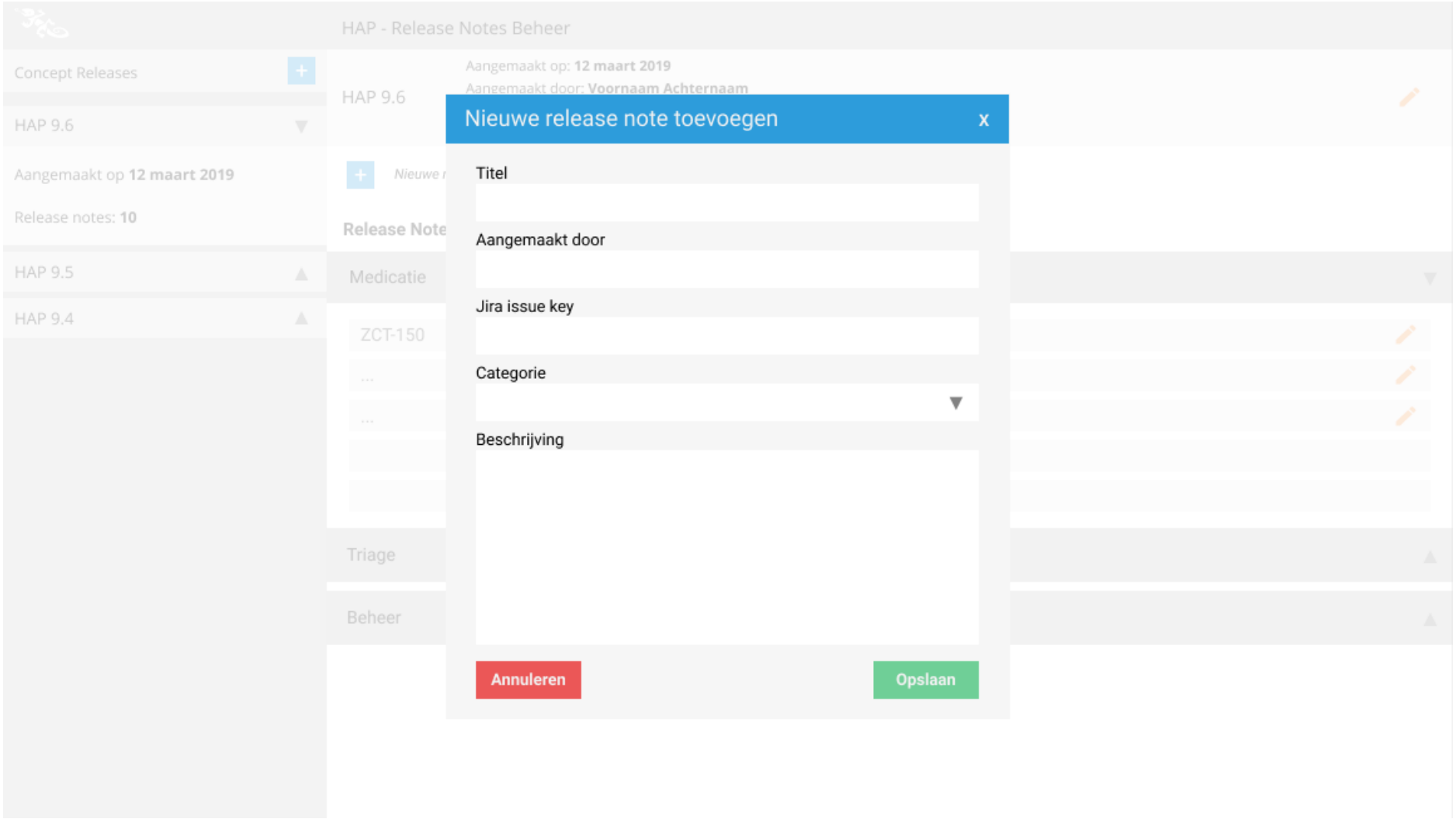
Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

2. Nieuwe release toevoegen



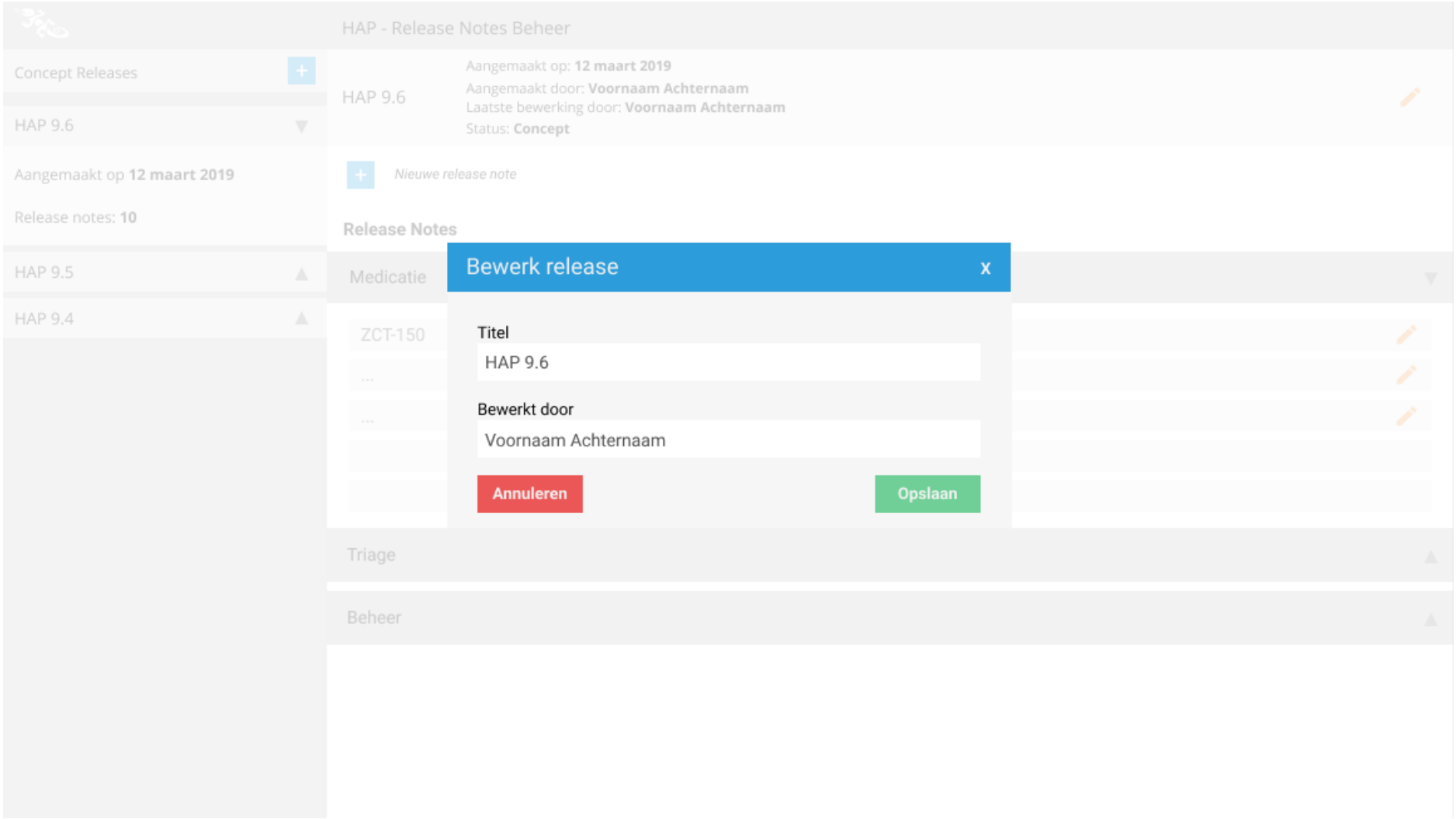
Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

3. Nieuwe release note toevoegen



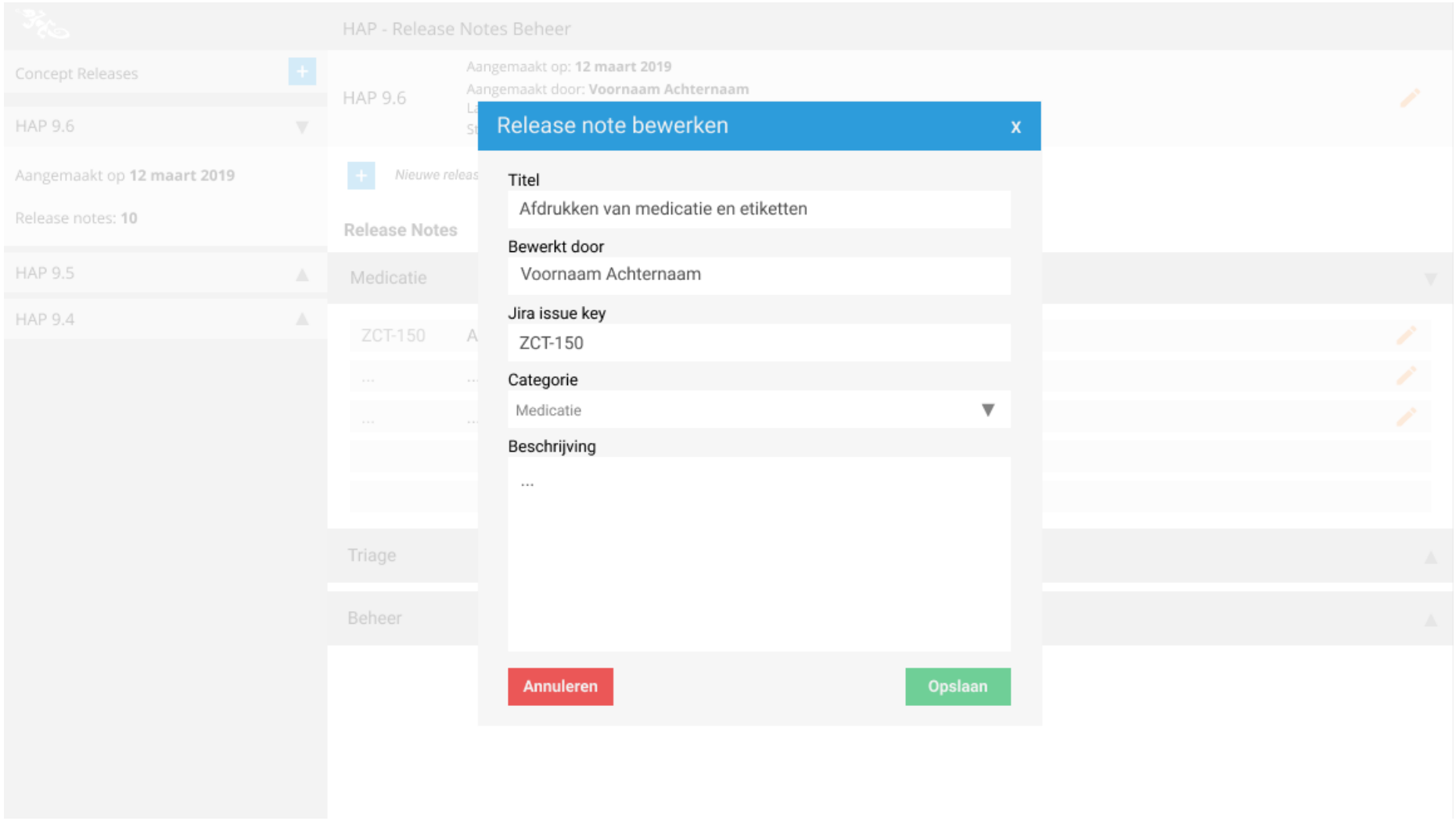
Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

4. Release bewerken



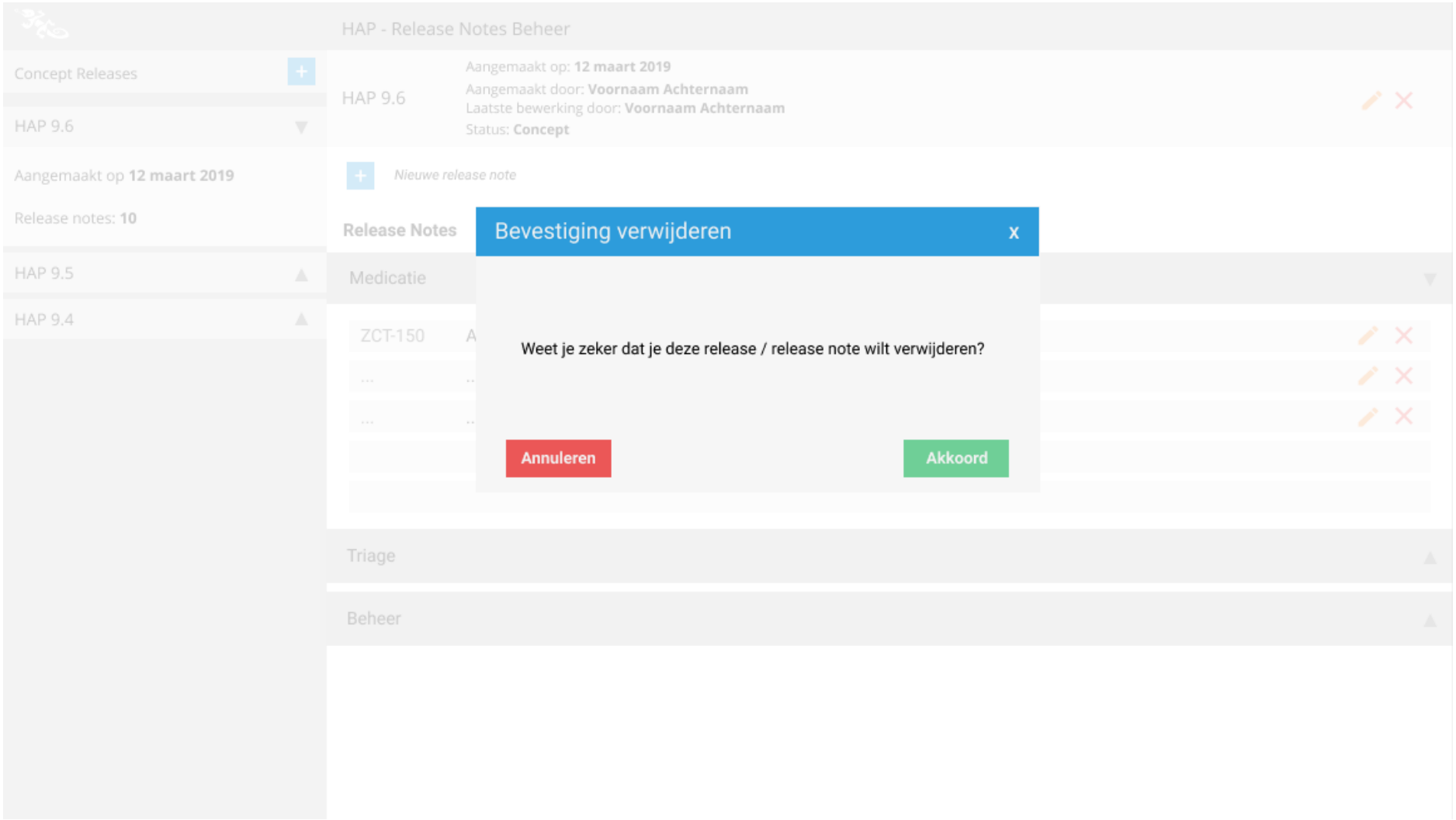
Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

5. Release note bewerken



Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

6. Release/release note verwijderen



Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

BIJLAGE 4 – SPRINTVERSLAG VAN SPRINT 3

Deze bijlage bevat een opname uit het projectdossier en beschrijft één van de sprints die tijdens de afstudeerperiode heeft plaatsgevonden.

Sprint 3 liep van maandag 8 april 2019 tot en met dinsdag 23 april 2019 in verband met Goede Vrijdag en het paasweekend. Deze sprint heeft een lengte van twee weken.

Backlog en planning

Tijdens deze sprint heb ik een begin gemaakt aan het onderwerp ‘Beheren van releases’. Dit onderwerp gaat in op het kunnen aanmaken van een release in de beheerapplicatie en hier een Jira label aan kunnen koppelen.

Key	Summary	Issue Type	Status	Story Points (- → 9)
RELEASE-9	Aanmaken nieuwe concept release	Story	DONE	- → 3
RELEASE-11	Inzage aangemaakte concept releases (lijst)	Story	DONE	- → 2
RELEASE-25	Inzage aangemaakte concept release (detail)	Story	DONE	- → 2
RELEASE-35	Inzage release notes onder concept release	Story	DONE	- → 1
RELEASE-37	Refactor: ExternalReleaseNote hernoemen naar ReleaseNote	Task	DONE	- → 1
RELEASE-39	Themaplanning maken	Task	DONE	-

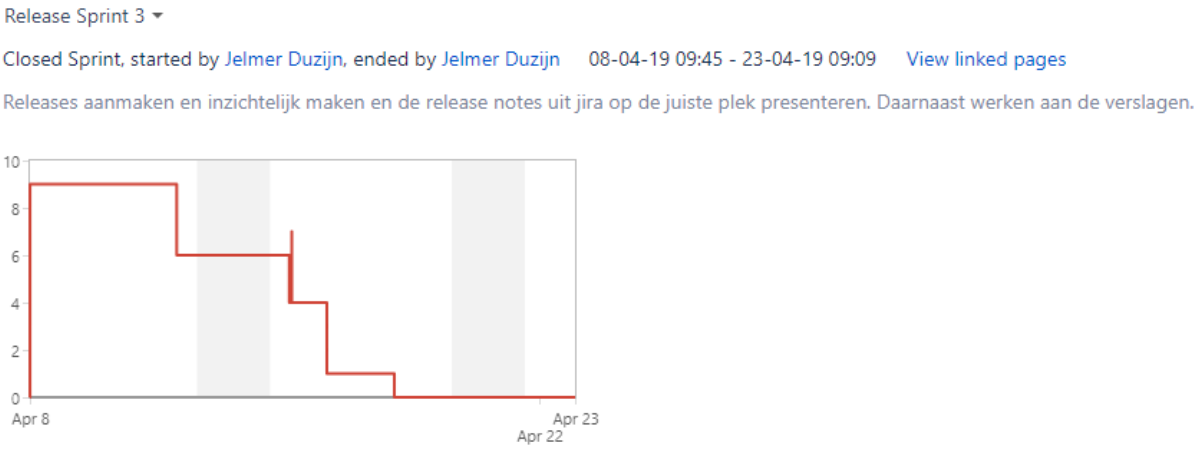
Figuur 31 Backlog sprint 3

Het doel van de sprint was om releases aan te kunnen maken en inzichtelijk te maken in de beheerapplicatie. Ook moeten de release notes uit jira op de juiste plek gepresenteerd worden. Daarnaast werken aan de verslagen.

In totaal zijn in deze sprint **9** punten gepland. Dit is nog minder punten in verhouding tot de vorige sprint. Maar dit gaf mij wel de mogelijkheid om te werken aan alle verslagen en om veel analyse werk te kunnen doen voor de toekomstige sprint waarin gewerkt wordt aan het publiceren van release notes en het weergeven hiervan in de inzageapplicatie.

Na afronding van deze sprint is het mogelijk om releases aan te maken in de beheerapplicatie en deze in te zien. Ook kunnen release notes die bij een release horen in de beheerapplicatie worden getoond.

Burndown en retrospective



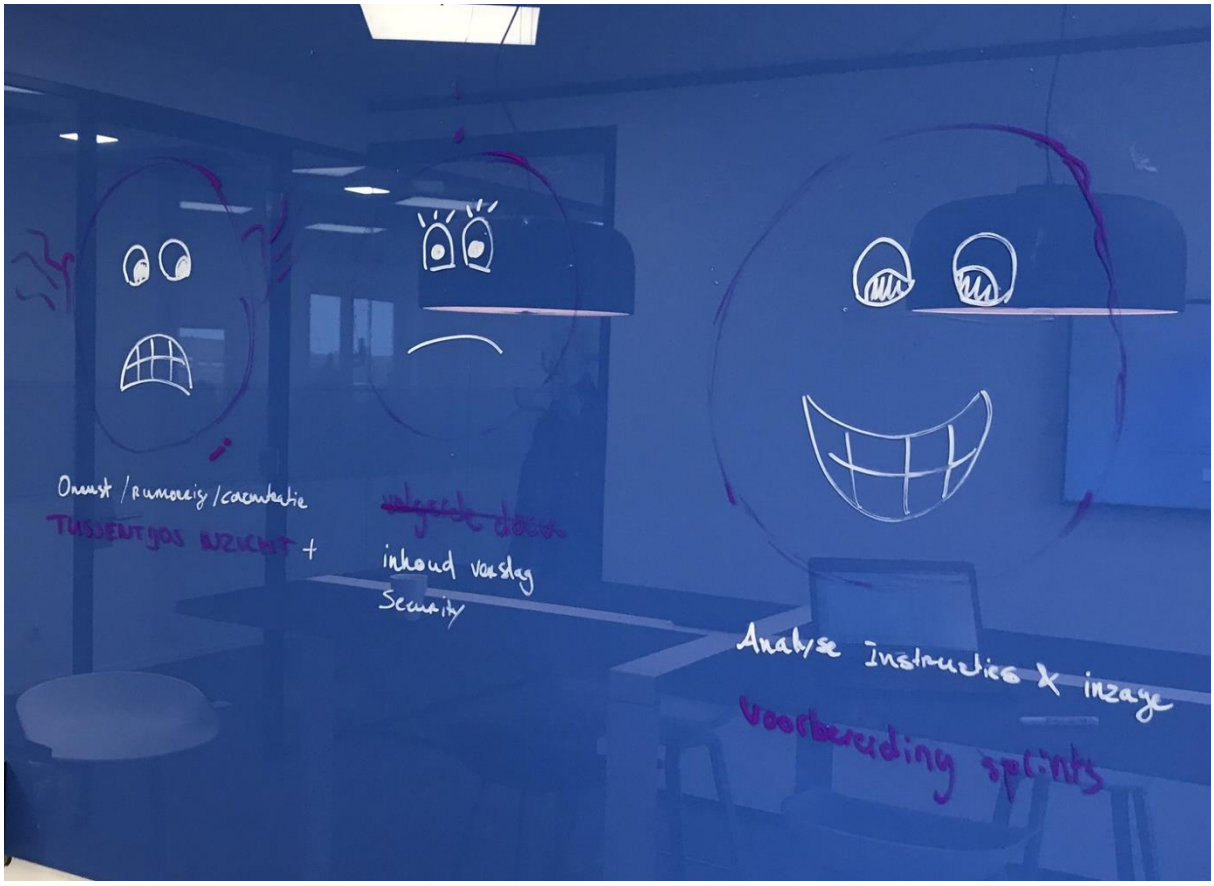
Figuur 32 Burndown chart sprint 3

Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0

In sprint 3 zijn net als in sprint 2 alle geplande functionaliteiten direct in de sprint backlog geplaatst. In tegenstelling tot sprint 2 zijn stories nu wel gedurende de sprint na afronding gesloten. Wat opvalt is dat de scope van de sprint halverwege met één punt is verhoogd. Dit komt doordat aan de refactor taak *RELEASE-37* (Figuur 31) aanvankelijk geen punten waren toegekend. Achteraf bleek dat deze taak ongeveer één punt waard was en is het issue geüpdatet. De functionaliteiten waren twee dagen voor het einde van de sprint afgerond. De tijd die overbleef heb ik gebruikt om het onderwerp ‘Weergeven publicaties’ verder te analyseren.

De retrospective van sprint 3 werd gehouden op 23 april met de bedrijfsbegeleiders en tijdens deze sessie is gebruik gemaakt van de techniek ‘mad/sad/glad’ (Figuur 33). Deze retrospective techniek helpt je om gebeurtenissen tijdens de sprint te verdelen over drie smileys:

- Mad (links) – Dingen waar je ontevreden over bent
- Sad (midden) – Dingen waar je minder blij mee bent
- Glad (rechts) – Dingen waar je blij mee bent



Figuur 33 Sprint 3 retrospective

Mad	Sad	Glad
Onrust/rumoerig/concentratie	Inhoud verslag	Analyse instructies & inzage
Tussentijds inzicht (+)	Security	Voorbereiding sprints

Tabel 11 Uitkomsten sprint 3 retrospective

Lisette gaf aan dat ze een gebrek aan tussentijds inzicht heeft. Dit komt doordat zij niet in Team Acute Zorg werkt, maar in Team Zorgcoördinatie aan de andere kant van de vleugel. Hierdoor mist zij discussies die ik met Rens voer over de opdracht. Om dit te verbeteren ga ik meer functionele vragen

bij haar neerleggen en haar vaker dan tijdens de voortgangsgesprekken bijpraten over de vorderingen.

Ik gaf aan veel onrust en rumoer te ervaren in Team Acute Zorg vanwege de vele (werk)gesprekken die aan de bureaus gevoerd worden. Helaas kan hier niet direct iets aan gedaan worden omdat er nou eenmaal gesprekken moeten worden gevoerd om goed samen te kunnen werken. Om de drukte te kunnen ontsnappen werk ik per week één dag thuis wat erg prettig is.

De inhoud van het verslag blijf ik een lastig punt vinden, omdat ik vaak terugval naar proberen een groot document aan te vullen in plaats van het schrijven van losse stukken. Door stukken te laten reviewen door Rens of Lisette en hier gesprekken over te voeren krijg ik meer inzicht in hoe het verslag vormgegeven kan worden.

Wat betreft security heb ik aangegeven dat de requirements hierover niet duidelijk zijn. Rens gaf aan dat security voor het Proof of Concept geen hoge prioriteit heeft, maar dat we er wel over moeten sparren (persoonlijke communicatie, 23 april 2019). Om dit te doen hebben we een meeting ingepland waarin we alle onduidelijke technische requirements gaan uitzoeken.

De laatste twee punten vallen samen. Ik was erg tevreden over de analyse die ik heb kunnen doen tijdens deze sprint naar het onderwerp ‘Beheren van instructies’ en ‘Weergeven publicaties’. Rens en Lisette gaven aan erg tevreden te zijn over de manier waarop ik sprints voorbereid. Duidelijke user stories met goede acceptatiecriteria. Dit zorgt dat de sprintplanning sessies soepel verlopen en er gericht kan worden ontwikkeld tijdens de sprint zonder veel onduidelijkheid.

EXTERNE BIJLAGEN

Naast de bijlagen die aan dit document zijn toegevoegd, zijn er ook nog externe bijlagen die tot het afstudeerdossier behoren:

- Functionele documentatie – Afstudeeropdracht Release Notes 2.0 – Jelmer Duzijn (325409)
- Technische documentatie – Afstudeeropdracht Release Notes 2.0 – Jelmer Duzijn (325409)
- Projectdossier – Afstudeeropdracht Release Notes 2.0 – Jelmer Duzijn (325409)
- Reflectierapport – Afstudeeropdracht Release Notes 2.0 – Jelmer Duzijn (325409)

Deze documenten zijn als losse bestanden opgeleverd.

Naast de genoemde documenten wordt ook de geschreven broncode als .zip file opgeleverd. In de technische documentatie is een installatiehandleiding opgenomen.

VERWIJZINGEN

Angular Team. (2019, Mei 5). *Angular Elements Overview.* Opgehaald van Angular: https://angular.io/guide/elements

Atlassian. (sd). *JIRA Server platform REST API reference*. Opgehaald van Atlassian Developers: https://docs.atlassian.com/software/jira/docs/api/REST/7.13.1/#api/2/search

baeldung. (2019, April 6). *Testing in Spring Boot*. Opgehaald van Baeldung: https://www.baeldung.com/spring-boot-testing

Deveria, A. (2019, Mei 20). *Customer Elements (V1).* Opgehaald van Can I Use: https://caniuse.com/#feat=custom-elementsv1

Foundeo Inc. (2016). *Content Security Policy (CSP) Quick Reference Guide*. Opgehaald van Content Security Policy: https://content-security-policy.com/

Nederlandse Vereniging van Doktersassistenten. (2019, 2 13). *Triage opleidingen - Nvda*. Opgehaald van Nederlandse vereniging van doktersassistenten: https://www.nvda.nl/opleiding-scholing/triage-opleidingen/

ObjectNode. (sd). Opgehaald van jackson-databind: https://fasterxml.github.io/jackson-databind/javadoc/2.5/com/fasterxml/jackson/databind/node/ObjectNode.html

Pivotal. (sd). *Accessing Data with JPA.* Opgehaald van Spring.io: https://spring.io/guides/gs/accessing-data-jpa/

Pivotal. (sd). *Class RestTemplate*. Opgehaald van Spring Boot Framework: https://docs.spring.io/spring/docs/5.1.5.RELEASE/javadoc-api/org/springframework/web/client/RestTemplate.html

Pivotal. (sd). *Spring Boot*. Opgehaald van Spring.io: https://spring.io/projects/spring-boot

Radigan, D. (2015, 7). *Search JIRA like a boss with JQL*. Opgehaald van Atlassian Support: https://confluence.atlassian.com/jiracore/blog/2015/07/search-jira-like-a-boss-with-jql

Schwaber, K., & Sutherland, J. (2017, November). *The Scrum Guide.* Opgehaald van Scrum Guides: https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

Topicus. (2019, Februari 14). *Over Topicus*. Opgehaald van Werken bij Topicus: https://www.werkenbijtopicus.nl/over-topicus

W3C. (2019, April 4). *Custom Elements*. Opgehaald van World Wide Web Consortium: https://w3c.github.io/webcomponents/spec/custom/

webcomponents.org. (sd). *Introduction*. Opgehaald van webcomponents.org: https://www.webcomponents.org/introduction#what-are-web-components-

Auteur: Jelmer Duzijn – 325409@student.saxion.nl
Versie: 2.0