Afstudeerscriptie
Ontwikkelen van een flexibele workflow-module voor AscMe

Project in gebruik bij:

VU
VRIJE UNIVERSITEIT AMSTERDAM

Afstudeerproject voor de bacheloropleiding Informatica
Opleidingsinstituut: De Haagse Hogeschool, locatie Zoetermeer
Afstudeerbedrijf: Vorsen B.V.
Bedrijfsbegeleider: Jeroen van Schagen
Afstudeercoördinatoren: A.A.A.M. Jacobs en A.A. Nederend
Afstudeerder: A.P.G. Vlek (Arjan)
Studentnummer: 13052888
Datum: 2 juni 2017
Voorwoord

Mijn dank gaat uit naar iedereen die mij gedurende mijn afstudeerstage heeft geholpen.

In het bijzonder bedank ik mijn opdrachtgever, Steven Losekoot, voor het aanbieden van deze afstudeeropdracht.

Daarnaast bedank ik Jeroen van Schagen en Marcel Noordzij voor de begeleiding en de input gedurende deze afstudeeropdracht.

Tot slot wil ik mevrouw A.A.A.M Jacobs en de heer A.A. Nederend bedanken voor de begeleiding vanuit De Haagse Hogeschool.

Zoetermeer, 2 juni 2017

Arjan Vlek
Inhoudsopgave

1. INLEIDING .................................................................................................................. 7
   1.1 LEESWIJZER ................................................................................................................. 7

2. DE ORGANISATIE ........................................................................................................ 11
   2.1 GESCHIEDENS ........................................................................................................... 11
   2.2 DOELSTELLING EN VISIE ......................................................................................... 12
   2.3 INTERNE ORGANISATIE .......................................................................................... 13
   2.4 ORANOGRAM ............................................................................................................ 13

3. DE OPDRACHT ............................................................................................................. 15
   3.1 ACHTERGROND ....................................................................................................... 15
   3.2 PROBLEMEMSTELLING ............................................................................................ 15
   3.3 DOELSTELLINGEN .................................................................................................. 16
   3.4 RESULTAAT ............................................................................................................... 16
   3.5 OP TE LEVEREN PRODUCTEN ................................................................................. 16

4. DIRECTE WIJZIGINGEN TEN OPZICHTE VAN HET AFSTUDEERPLAN .................... 17
   4.1 REDENEN ................................................................................................................ 17
   4.2 AANPASSINGEN AAN OPDRACHTOMSCHRIJVING .............................................. 17

5. GEBRUIKTE TECHNIEKEN EN METHODEN ............................................................. 19
   5.1 ONTWIKKELING VOLGENGS SCRUM .................................................................... 19
       5.1.1 Afspraken voor het gebruik van Scrum ............................................................ 19
       5.1.2 Definition of Done ............................................................................................ 20
       5.1.3 Afwijkingen bij het gebruik van Scrum ................................................................ 20
   5.2 FRONT-END ONTWIKKELING MET ANGULARJS .................................................. 22
   5.3 BACK-END ONTWIKKELING MET SPRING EN HIBERNATE .................................. 24
       5.3.1 Database ........................................................................................................... 24
       5.3.2 Overige frameworks / libraries ........................................................................ 25
   5.4 VERSIEBEHEER ....................................................................................................... 26
   5.5 OTAP ........................................................................................................................ 26
   5.6 PLANNING ................................................................................................................ 27

6. SITUATIE BIJ AANVANG AFSTUDEERPERIODE .............................................................. 29
   6.1 STAAT ....................................................................................................................... 29
   6.2 NOTIFICATIES ......................................................................................................... 29
   6.3 WORKFLOWS ......................................................................................................... 29
   6.4 VOORTGANGSOVERZICHTEN ............................................................................... 30
       6.4.1 In het menu “Rapportages” ............................................................................... 30
       6.4.2 Op de pagina’s van onderdelen ..................................................................... 31

7. START VAN HET PROJECT (ORIËNTATIEFASE, SPRINT 0) ...................................... 35
   7.1 VERKENNEN VAN DE APPLICATIE EN VASTSTELLEN VAN DE REQUIREMENTS.... 35
7.1.1 Workflows .................................................................................................................. 36
7.1.2 Notificaties .................................................................................................................. 36
7.3 EERSTE ONTERPEN VAN DE GEBRUIKERSINTERFACE (GUI) ........................................... 37
7.4 REQUIREMENTS VANUIT DE VU ..................................................................................... 39

8. SPRINT 1 - VERNIEUWDE NOTIFICATIE-MODULE............................................................. 41
8.1 PLANNEN VAN DE SPRINT ............................................................................................ 41
8.2 KLASSENDIAGRAM NOTIFICATIE-MODULE .................................................................. 42
8.3 ONTERPEN GEBRUIKERSINTERFACE (GUI) .................................................................... 43
8.4 REALISATIE NOTIFICATIE-MODULE ................................................................................ 44
  8.4.1 Versturen van automatische e-mailnotificaties .............................................................. 44
  8.4.2 Beheren en gebruiken van sjablonen voor het sturen van e-mailnotificaties ............... 45
  8.4.3 Inloggen met een link vanuit een e-mailbericht .......................................................... 47
  8.4.4 Notificaties en notificatie-instellingen in de applicatie ................................................. 49
8.5 TESTEN VAN DE NOTIFICATIE-MODULE ....................................................................... 50
8.6 SPRINT DEMO ............................................................................................................... 50
8.7 SPRINT RETROSPECTIVE ................................................................................................ 51

9 SPRINT 2: AFRONDEN NOTIFICATIE-MODULE / BEGIN WORKFLOW-MODULE ................. 53
9.1 PLANNEN VAN DE SPRINT ............................................................................................ 53
9.2 TESTEN VAN DE BACK-END CODE VAN SPRINT 1 ....................................................... 53
  9.2.1 Verschillende soorten unittests .................................................................................. 53
  9.2.2 Bugs in notificatie-module ....................................................................................... 55
9.3 VERWERKEN CODE REVIEW NOTIFICATIE-MODULE ................................................. 56
9.4 KLASSENDIAGRAM INSTELLEN WORKFLOWS PER FACULTEIT .................................. 57
9.5 ONTERPEN GRAFISCHE USERINTERFACE ..................................................................... 57
9.6 DATABASE-DIAGRAM WORKFLOW INSTELLINGEN PER FACULTEIT .............................. 58
9.7 IMPLEMENTEREN WORKFLOW-INSTELLINGEN PER FACULTEIT ................................. 58
9.8 SPRINT DEMO ............................................................................................................... 59
9.9 SPRINT RETROSPECTIVE .............................................................................................. 59

10 SPRINT 3: WORKFLOWS CONFIGUREERBAAR ................................................................ 61
10.1 WIJZIGINGEN AAN DE REQUIREMENTS ...................................................................... 61
10.2 PLANNEN VAN DE SPRINT .......................................................................................... 61
10.3 KLASSENDIAGRAM EN GEWIJZIGD DATAMODEL ...................................................... 62
10.4 IMPLEMENTEREN VERSTUREN HANDMATIGE NOTIFICATIES ................................... 63
10.5 VERBETERINGEN NOTIFICATIE-EVENTS (CODE REVIEW SPRINT 1) ......................... 64
10.6 Hervatten van het instelbaar maken van workflows ..................................................... 65
10.7 TESTEN .......................................................................................................................... 70
  10.7.1 Handmatige notificaties .......................................................................................... 70
  10.7.2 workflow-instellingen per faculteit ......................................................................... 70
10.8 SPRINT DEMO ............................................................................................................... 71
10.9 SPRINT RETROSPECTIVE ................................................................. 71

11 SPRINT 4: AANPASbare WORKFLOW-PAGINA’S ..................................... 73
  11.1 VASTSTELLEN EN VERIFIëREN VAN DE REQUIREMENTS ...................... 73
  11.2 PLANNEN VAN DE SPRINT .......................................................... 73
  11.3 KLASSENDIAGRAM AANGEPASTE WORKFLOW-PAGINA’S ...................... 74
  11.4 ONTWERPEN VAN DE GEBRUIKERSINTERFACE .................................. 74
  11.5 ONTWERPEN VAN EEN NIEUW GEGEVENSMODEL ................................ 75
  11.6 REALISATIE ................................................................................. 76
    11.6.1 Modelleren van de database in Java ............................................. 76
    11.6.2 Samenstellen van de pagina ....................................................... 77
    11.6.3 Tonen van gegevens op de pagina ............................................... 77
  11.7 SPRINT DEMO ................................................................................ 80
  11.8 SPRINT RETROSPECTIVE ............................................................. 80

12 SPRINT 5: WIZARD VOOR SAMENSTELLEN WORKFLOW ........................... 81
  12.1 VASTSTELLEN REQUIREMENTS ....................................................... 81
  12.2 PLANNEN VAN DE SPRINT .......................................................... 82
  12.3 ONTWERPEN VAN DE GEBRUIKERSINTERFACE ................................ 83
  12.4 REALISATIE WORKFLOW-WIZARD ................................................... 86
    12.4.1 Wizard stap 1: Basisgegevens voor de workflow ........................... 86
    12.4.2 Wizard stap 2: Pagina’s aan de workflow toevoegen ..................... 86
    12.4.3 Wizard stap 3: Pagina’s configureren ......................................... 86
  12.5 REALISATIE BEHEREN VAN AANGEPASTE WORKFLOW-PAGINA’S ........ 87
  12.6 AANPASSINGEN ONDERDELEN VORIGE SPRINT A.D.H.V. CODE REVIEW 87
  12.7 TESTEN AANGEPASTE (WORKFLOW-)PAGINA’S ................................ 88
  12.8 TESTEN INRICHTEN VAN WORKFLOWS ....................................... 90

13 SPRINT 6: VOORTGANGSOVERZICHT, BEHEERPAGINA EN AFRONDEN ........ 91
  13.1 REQUIREMENTS .......................................................................... 91
  13.2 PLANNEN VAN DE SPRINT .......................................................... 92
  13.3 SYSTEEMONTWERPEN ................................................................. 93
  13.4 ONTWERP GEBRUIKERSINTERFACE .............................................. 95
  13.5 REALISATIE NIEUWE ONDERDELEN ............................................. 96
    13.5.1 Voortgang workflows in rapportage met vakken ............................ 96
    13.5.2 Beheerpagina voor workflows .................................................. 98
    13.5.3 Nieuwe code voor workflow-importer ......................................... 101
    13.5.4 Afkeuren van workflows .......................................................... 101
    13.5.5 Automatisch opstarten (triggeren) van workflows ......................... 102
    13.5.6 Bijhouden audits bij aangepaste workflow-pagina’s ....................... 103
    13.5.7 Workflow-notificaties ............................................................... 104
  13.6 SPRINT DEMO ................................................................................ 105
  13.7 SPRINT RETROSPECTIVE ............................................................. 106
13.8 Afronding en retrospectieve over het gehele project ................................................................. 106

14 Procesevaluatie ................................................................................................................................. 107
  14.1 Ontwikkeling volgens Scrum ........................................................................................................ 107
  14.2 Technologieën en programmeerваardigheden ............................................................................ 107
  14.3 Meetings bij de VU ........................................................................................................................... 107
  14.4 Verbeterpunten voor een volgend project ................................................................................... 108

15 Productevaluatie ............................................................................................................................... 109
  15.1 Vernieuwde notificatie-module ...................................................................................................... 109
  15.2 Vernieuwde workflow-module ...................................................................................................... 109
  15.3 Voortgangsoverzichten .................................................................................................................. 110

16 Evaluatie van de beroepstaken ........................................................................................................ 111
  16.1 Voorbereiden en opstarten software-ontwikkeltraject ............................................................... 111
  16.2 Uitvoeren analyse door definitie van requirements ..................................................................... 111
  16.3 Ontwerpen systeemdeel ................................................................................................................ 111
  16.4 Bouwen applicatie ........................................................................................................................ 111
  16.5 Uitvoeren en rapporteren over het testproces .......................................................................... 112
    16.5.1 Unitests ..................................................................................................................................... 112
    16.5.2 Aanbevelingen op het gebied van testen ............................................................................... 112

Bijlagen .................................................................................................................................................. 117

Referaat

Administratiesysteem, Academische structuur, Web development, AngularJS, Spring Framework
1. Inleiding

In dit verslag staan de activiteiten beschreven die ik bij Vorsen B.V. heb uitgevoerd ten behoeve van mijn afstudeerstage voor De Haagse Hogeschool. Dit verslag beschrijft de activiteiten die ik gedurende mijn afstudeerperiode heb uitgevoerd op procesmatige wijze. Het verslag is bedoeld voor eenieder die hierin is geïnteresseerd.

Deze activiteiten betreffen het ontwikkelen van een flexibele notificatie- en workflowmodule voor de applicatie AscMe, een product waarmee onderwijsinstellingen een belangrijk onderdeel van hun administratieve takenpakket kunnen uitvoeren. Meer informatie over het bedrijf Vorsen B.V. staat in [hoofdstuk 2]; daarnaast staat de applicatie AscMe uitvoeriger beschreven in [hoofdstukken 3 en 6].

1.1 Leeswijzer

Het document bestaat uit verschillende onderdelen. De eerste hoofdstukken, hoofdstuk 1 t/m 6, beschrijven alle informatie die nodig is om de daadwerkelijke uitvoering van het project, hoofdstukken 7 t/m 13, te begrijpen. Onder andere opdractomschrijving, aanpak en planning staan hierin omschreven.

De daadwerkelijke uitvoering van het project is per periode van twee weken (Scrum sprint, zie hoofdstuk 5) beschreven. Elke periode had een bepaald thema, waarna deze hoofdstukken zijn genoemd.

De laatste hoofdstukken, hoofdstuk 14 t/m 16, vormen de afronding van deze afstudeeropdracht en bevatten evaluaties over het proces, het product en de beroepstaken (volgens naamgeving van De Haagse Hogeschool) die ik heb uitgevoerd.

De figuren die in de diverse hoofdstukken staan, zijn opgenomen in de lijst met figuren [hoofdstuk 17]

Daarnaast bevat dit document, zoals u wellicht reeds heeft vernomen, technische termen. Omdat deze lastig te begrijpen kunnen zijn, heb ik in [hoofdstuk 18] een lijst met termen plus uitleg opgenomen. Termen staan, de eerste keer dat ze voorkomen, cursief gedrukt.

Tot slot bevatten sommige hoofdstukken verwijzingen naar requirements. Deze worden met aanduiding genoemd (voorbeeld: WFE-F001). Requirements zijn te vinden in het requirements document, opgenomen als [bijlage 11].
I. ACHTERGRONDINFORMATIE
2. De organisatie

Deze afstudeeropdracht is uitgevoerd voor het bedrijf Vorsen B.V. In dit hoofdstuk volgt achtergrondinformatie over dit bedrijf. Er wordt hierbij ingegaan op enkele aspecten: De geschiedenis van het bedrijf, de doelstellingen van het bedrijf en de organisatiestructuur binnen het bedrijf.

2.1 Geschiedenis

De oprichting van het bedrijf Vorsen komt voort uit het AscMe-project waar deze opdracht betrekking op heeft. In deze paragraaf staat beschreven hoe het bedrijf is ontstaan.

Het project AscMe (Academic Structure & Content Management Environment) werd oorspronkelijk door het bedrijf 42 B.V. uitgevoerd onder de naam UAS (Uitvragen Academische Structuur). Er waren echter plannen om dit project uit te breiden en om een tweede project voor de onderwijsmarkt op te starten. Deze projecten hadden zeer veel potentie, maar deze potentie werd niet volledig bereikt. Dit kwam omdat de projecten binnen 42 B.V. niet goed op de markt gepositioneerd konden worden. 42 B.V. was namelijk een bedrijf wat zich richtte op het ontwikkelen van diverse IT-producten, terwijl het binnen de onderwijsmarkt gebruikelijk is dat een product door een bedrijf wat zich specifiek op deze markt richt wordt verkocht.

Daarnaast was Steven Losekoot inmiddels ook bij het UAS-project betrokken geraakt. Hij was voorheen bij de VU in dienst en werkte daar aan nieuwe onderwijsoplossingen. Tijdens deze werkzaamheden kwam hij ook met UAS in aanraking. Steven had goede ideeën om UAS te verbeteren en kwam hierbij steeds meer in contact met de ontwikkelaars.

De toegenomen betrokkenheid van Steven en de behoefte aan een betere marktpositie voor het verkopen van de applicaties voor het onderwijs vormden samen de basis voor het oprichten van het bedrijf Vorsen. In december 2015 was het zover: Vorsen B.V. werd door Eric Meijer, de eigenaar van 42 B.V., en Steven Losekoot opgericht.

Hoewel er met de oprichting van Vorsen een nieuw bedrijf was geboren, bleef Vorsen voor haar werkzaamheden gebruik maken van de faciliteiten van 42. Zo is Vorsen gehuisvest in hetzelfde gebouw, gebruikt het bedrijf de IT-faciliteiten en de HR-faciliteiten van 42. Maar ook de technische kennis en software-ontwikkelmethoden komen overeen. Door al deze diensten bij 42 af te nemen bespaart Vorsen een hoop overhead, waardoor het zich op haar eigen activiteiten kan concentreren.
2.2 Doelstelling en visie

De doelstelling van Vorsen B.V. is om op maat gemaakte software-oplossingen voor het beheren en uitvoeren van de onderwijslogistiek te verkopen. Onder onderwijslogistiek valt bijvoorbeeld het inrichten / samenstellen van een nieuw academisch jaar, het inroosteren van colleges en het houden van evaluaties over gegeven cursussen / vakken. Dit zijn allen mogelijkheden van de producten die Vorsen verkoopt. Vorsen richt zich bij het verkopen van haar producten op alle Nederlandse hogescholen en universiteiten. Vorsen heeft vooralsnog geen plannen om ook producten aan buitenlandse onderwijsinstellingen te leveren.


2.3 Interne organisatie

Vorsen bestaat uit vier werknemers, die allen mede-eigenaar van het bedrijf zijn. Deze vier werknemers vervullen een gedeelte van de taken binnen het bedrijf. De rest van de taken wordt door ingehuurde werknemers van 42 B.V. verricht. In deze paragraaf volgt een omschrijving van de rollen en taken van de vaste medewerkers van Vorsen. De faciliteiten die door 42 worden geleverd, worden in dit hoofdstuk niet beschreven.

**Steven Losekoot:**
Steven Losekoot is verantwoordelijk voor de sales en de communicatie met de klanten. Hierbij staat hij in contact met diverse onderwijsinstellingen in Nederland en bezoekt hij regelmatig nieuwe onderwijsinstellingen. Bovendien is hij meestal bij meetings bij bestaande klanten aanwezig, omdat hij -gezien zijn verleden in de onderwijsmarkt - de wensen vanuit de klant goed in kaart kan brengen.

**Marcel Noordzij:**
Marcel Noordzij is, zoals hij zelf zegt, het manusje van alles. Hij is requirements analist en software-architect, maar houdt zich soms ook bezig met communicatie naar de klant. Als software-architect specialiseert hij zich in de globale architectuur. Deze is met name van belang bij het koppelen van de producten van Vorsen aan bestaande systemen van onderwijsinstellingen. Bovendien maakt Marcel soms een demoversie of een prototype van een nieuw onderdeel in een product, zodat dit reeds voordat het wordt ontwikkeld kan worden getoond. Marcel houdt zich echter niet bezig met de daadwerkelijke ontwikkeling van de producten van Vorsen.

**Jeroen van Schaegen:**
Jeroen van Schaegen is *lead developer* en projectleider bij Vorsen. Als projectleider zorgt hij voor het verdelen van (ontwikkel)taken over de verschillende (bij 42 B.V. ingehuurde) medewerkers. Als *lead developer* ontwikkelt hij zelf in razend tempo aan de producten van Vorsen mee. Bovendien heeft hij het leeuwendeel van AscMe ontwikkeld, waardoor hij over uitgebreide kennis van dit product beschikt.

**Eric Meijer:**
Eric Meijer is verantwoordelijk voor de financiële en de administratieve kant van Vorsen. Hij regelt de contracten en overeenkomsten met klanten. Bovendien vertegenwoordigt hij het bedrijf en houdt hij op afstand toezicht op de ontwikkeling binnen de lopende projecten, zodat hij kan ingrijpen indien er een probleemsituatie ontstaat.

2.4 Organogram

Onderstaand organogram geeft een globaal overzicht van de organisatiestructuur binnen Vorsen. Alle faciliteiten die door 42 worden geleverd, staan in de rechterkant van het organogram.

---

*Figuur 1: Organogram Vorsen B.V. en 42 B.V. Overige afdelingen van 42 B.V. zijn niet getoond.*
3. De opdracht

In dit hoofdstuk staat de uitgevoerde afstudeeropdracht nader omschreven. Onder andere de probleemstelling en doelstelling komen hierbij aan bod. Deze opdrachtomschrijving komt niet volledig overeen met het voorafgaand ingeleverde Afstudeerplan. In hoofdstuk 4 (Directe wijzigingen ten opzichte van het afstudeerplan) worden de ontstane wijzigingen nader toegelicht.

3.1 Achtergrond

In de afgelopen twee jaar hebben in eerste instantie 42, en later Vorsen een applicatie ontwikkeld voor het hoger onderwijs, voor het ondersteunen van het vaststellen van het onderwijsprogramma van een instelling.

Deze applicatie, AscMe, biedt hiervoor een procesmatige ondersteuning aan de diverse betrokkenen binnen het proces. Dit omvat o.a. de ondersteuning van de opleidingsverantwoordelijke die de opbouw van een vak moet vaststellen, een examencommissie die advies geeft, de vakcoördinator die de opbouw van het vak voorstelt en de studiegidsteksten invoert. Elk van de betrokkenen wordt ondersteund bij het uitvoeren van hun deeltaak, waarbij AscMe het volledige proces monitor en inzichtelijk maakt. Alle wijzigingen doorlopen een proces van indienen tot af-/goedkeuring, voordat ze effectief worden. In het kader van de accreditatie wordt een volledig digitaal logboek (audit) bijgehouden van alle gemaakte wijzigingen.

3.2 Probleemstelling

AscMe is initieel ontwikkeld voor één onderwijsinstelling, waarbij vanaf het begin rekening gehouden is met het feit dat de applicatie inzetbaar moet zijn voor meerdere onderwijsinstellingen. Echter zijn er hierbij op een aantal plekken keuzes gemaakt die tijdens het implementeren bij nieuwe onderwijsinstellingen minder gelukkig bleken te zijn.

De opdracht richt zich op het probleem dat de momenteel aanwezige ondersteunende processen ‘hard geconfigureerd’ zijn binnen het systeem, en dat er meestal geen goede flow tussen de verschillende processen te vinden is. Op sommige van de schermen die bij het uitvoeren van een proces langskomen staat te veel en / of niet relevante informatie.

Wijzigingen in het proces zijn niet te configureren en vragen ontwikkelwerk en een uitrol van een nieuwe versie. In het verlengde hiervan geldt dat de deeltaken binnen een proces idealiter via een Wizard doorlopen moeten kunnen worden, wat nu slechts beperkt het geval is. Ook zitten er nog geen harde deadlines en voortgangsrapportages gekoppeld aan deeltaken, waardoor de eindverantwoordelijke van de instelling het hele proces lastig kan overzien.
3.3 Doelstellingen

Deze opdracht is opgesplitst in vier doelstellingen, die samen tot een flexibeler en intuitiever systeem moeten leiden.

Als eerste moet het mogelijk worden, om met minder ontwikkelwerk het systeem voor nieuwe onderwijsinstellingen in te kunnen richten. Dit zal gebeuren door het configureerbaar maken van processen (workflows) en deeltaken.

Als tweede moet het systeem betrokkenen beter op de hoogte houden door middel van notificaties. Dit zal gebeuren door het aanbrengen van een uitgebreide notificatie-module, waarbij e-mailteksten, frequenties en ontvangers configureerbaar zijn.

Als derde moet het systeem beter kunnen omgaan met (menselijke) fouten die tijdens het uitvoeren van een proces kunnen optreden. Het systeem moet de mogelijkheid bieden om in bepaalde, te configureren gevallen, naar een vorige fase binnen een proces terug te keren.

Als vierde moet het systeem een overzicht aan de eindverantwoordelijke van een proces kunnen bieden met daarin de voortgang van alle betrokkenen binnen het proces.

3.4 Resultaat

Indien de afstudeeropdracht succesvol is afgerond, is de applicatie AscMe beter configureerbaar voor nieuwe onderwijsinstellingen. Bovendien worden betrokkenen beter op de hoogte gehouden van nog uit te voeren taken en kunnen eindverantwoordelijken beter inzicht verkrijgen in de voortgang van de betrokkenen binnen hun processen.

3.5 Op te leveren producten

Als onderdeel van deze afstudeeropdracht zullen er verschillende producten worden opgeleverd aan Vorsen B.V. en aan De Haagse Hogeschool. Deze staan in deze paragraaf gespecificeerd.

De op te leveren producten aan Vorsen B.V. zijn:
- Requirements document
- Sprint demo presentaties
- Broncode van de vernieuwde AscMe-applicatie
- Documentatie om de vernieuwde applicatie uit te kunnen rollen

De op te leveren producten aan De Haagse Hogeschool zijn:
- Afstudeerscriptie
- Plan van Aanpak
- Requirements document
- Overzicht van de Scrum backlog (met uitgevoerde en niet-uitgevoerde items)
4. Directe wijzigingen ten opzichte van het Afstudeerplan

Bij aanvang van de afstuderingperiode bleek dat een gedeelte van de opdrachtomschrijving en een van de doelstellingen uit het Afstudeerplan niet in overeenstemming met de daadwerkelijke opdracht waren. Dit betrof twee onderdelen: Het onderbrengen van een aantal workflows in wizards, en het uitvoeren van praktijkgericht onderzoek naar het zo goed mogelijk inrichten van processen / workflows.

4.1 Redenen

Er waren twee redenen waarom deze onderdelen uit het Afstudeerplan niet in overeenstemming met de afstuderingopdracht en de wensen vanuit Vorsen B.V. waren. De eerste reden was dat de VU vlak voordat de afstuderingperiode begon (in januari, reeds na het goedkeuren van het Afstudeerplan) bij Vorsen de wens had neergelegd om de ontbrekende workflows direct in wizards te laten plaatsen. Deze wens was ontstaan, omdat de bij deze workflows behorende bedrijfsprocessen reeds waren vastgesteld en het uitvoeren hiervan zeer spoedig moest worden opgestart. Hierdoor is deze taak komen te vervallen - Het inbrengen van de workflows in wizards was reeds door de medewerkers van Vorsen uitgevoerd.

Het praktijkgericht onderzoek naar het inrichten en verbeteren van de workflows kon niet worden uitgevoerd, omdat dit niet aansloot bij de verwachtingen vanuit Vorsen over de uit te voeren opdracht. Dit kwam omdat de vanuit Vorsen aan mij gegeven informatie over de opdracht initieel niet duidelijk genoeg was, en bij mijn voorstel of praktijkgericht onderzoek binnen deze opdracht mogelijk was (aangeraden vanuit De Haagse Hogeschool) werd dit direct door de opdrachtgever bevestigd. Op basis van het eerste gesprek over de opdracht en de bevestiging dat praktijkgericht onderzoek mogelijk was, had ik het idee gekregen dat de opdracht (voor een deel) uit het inrichten en ontwikkelen van processen binnen de applicatie bestond, en dat het praktijkgericht onderzoek kon dienen om deze processen vast te stellen. Toen ik dit in de opdrachtomschrijving had vermeld, werd dit door het bedrijf goedgekeurd. Bij aanvang van de afstuderingperiode bleek echter direct dat de binnen de applicatie aanwezige processen (voor de VU) reeds waren vastgesteld, en dat het aan de klant is om deze te specificeren. Daardoor was het niet mogelijk om praktijkgericht onderzoek naar het inrichten van de processen / workflows te doen. De opdracht betrof het inrichten van een vernieuwde, beter configureerbare, workflow-module in de applicatie AscMe, maar dit was dus enkel het implementeren ervan.

4.2 Aanpassingen aan de opdrachtomschrijving

Vanwege de ontstane afwijkingen was het noodzakelijk om de opdrachtomschrijving aan te passen. De nieuwe opdrachtomschrijving staat in hoofdstuk 3 omschreven.

De aanpassingen betreffen het volgende:
In plaats van het inbrengen van de workflows in wizards en het uitvoeren van het onderzoek hiernaar is in overleg met het bedrijf besloten om een veelzijdige notificatie-module voor de applicatie te ontwikkelen. Deze moet de gebruikers in verschillende situaties en op verschillende manieren op de hoogte houden van nog uit te voeren taken.

Doordat er geen onderzoek werd uitgevoerd kon de opstartfase van het project worden verkort van vier naar twee weken. Op deze manier was er meer tijd beschikbaar voor het realiseren van de aan de opdracht toegevoegde notificatie-module.
5. Gebruikte technieken en methoden

Het uit te voeren project betreft het ontwikkelen van software. Voor het uitvoeren van dit project zijn verschillende technieken en methoden voor het ontwikkelen van software gebruikt. De gebruikte methoden en technieken worden in dit hoofdstuk besproken.

5.1 Ontwikkeling volgens Scrum

Voor het ontwikkelen van deze opdracht heb ik gebruik gemaakt van de agile software-ontwikkelmethode Scrum [1]. Er zijn meerdere redenen waarom ik voor deze ontwikkelmethode heb gekozen. Deze worden hieronder toegelicht.

- Scrum is een iteratieve ontwikkelmethode. Dit houdt in dat de eisen en de wensen vanuit de opdrachtgever periodiek kunnen worden bijgesteld. Omdat AscMe een product is waaraan momenteel actief wordt ontwikkeld, is de kans groter dat er wijzigingen aan de eisen vanuit de aanbieders en de opdrachtgever ontstaan. Het gebruiken van een iteratieve methode maakt het in deze situatie mogelijk om de requirements bij te stellen.
- De opdrachtomschrijving bevat niet alle requirements. Hierdoor is het noodzakelijk om de ontbrekende requirements tijdens het project uit te vragen en indien nodig bij te stellen. Een niet-iteratieve ontwikkelmethode zou ervoor gezorgd hebben dat onderdelen waarvan de requirements niet van tevoren bekend zijn niet kunnen worden gerealiseerd.
- Vorsen gebruikt Scrum voor het ontwikkelen van al haar softwareproducten. Het gebruik van Scrum sluit beter aan in de manier van werken binnen het bedrijf.

5.1.1 Afspraken voor het gebruik van Scrum

Bij het werken aan het project zijn verschillende afspraken tussen mij en het bedrijf gemaakt om succesvol te kunnen werken met Scrum.

Bij Scrum wordt er in iteratieve sprints (periodes) gewerkt, waarvan de duur hoogstens een maand is. Bij deze opdracht is er gewerkt in sprints van twee weken. Op deze manier is het werk op te delen in kleine gedeelten en kan er - indien nodig - regelmatig worden bijgesteld.

De backlog wordt opgesteld op basis van de requirements. Op deze manier zijn de backlog-items rechtstreeks te koppelen aan de requirements en is het duidelijk waar elk item voor dient.

Het Scrum bord wordt bij dit project digitaal bijgehouden in de applicatie JIRA. JIRA is een issue-managementapplicatie van Atlassian, welke Vorsen gebruikt voor het bijhouden van openstaande taken. Alle backlog items worden als taken in JIRA aangemaakt, in JIRA van Story Points voorzien en tijdens de sprint op het scrumbord gezet. Op deze manier is het Scrumboard overal beschikbaar en kan later worden teruggekomen welke items nog openstaan.

Als eerste onderdeel van de sprint volgt de Sprint Planning. Hierbij worden de backlog-items voor de komende sprint op de backlog geplaatst en voorzien van Story points. Story Points kunnen in relatieve eenheden of in tijdeenheden worden uitgedrukt. Bij dit project heb ik de Story Points in tijdeenheden uitgedrukt.
Na het vullen van de sprint backlog wordt eerst een ontwerp gemaakt, zowel van de grafische userinterface (GUI) als in UML, en vervolgens aan de stakeholders getoond. Op deze manier wordt gecontroleerd of ik de wensen van de stakeholders op een goede manier kan omzetten in software. Als deze ontwerpen zijn goedgekeurd, start de ontwikkeling.


Na elke demo-presentatie volgt de sprint-retrospective, waarin ik aan de stakeholders toelicht wat er gedurende sprint goed ging en wat de verbeterpunten waren.

Na de retrospective worden de requirements nagelopen en eventueel bijgewerkt. Er wordt ook een code review gehouden om de kwaliteit van de geschreven programmacode te controleren. Hierna wordt de volgende sprint ingepland.

5.1.2 Definition of Done

Bij Scrum is een backlog-item klaar indien aan de Definition of Done voldoet. Indien een item hier niet aan voldoet, wordt het terug op de backlog geplaatst en dient het opnieuw in een sprint te worden opgenomen.

De Definition of Done voor dit project is vastgesteld in overleg met het bedrijf en luidt als volgt:

Een backlog-item is klaar dan, en slechts dan als het betreffende onderdeel in de applicatie aanwezig is en werkt zoals in de omschrijving van het item staat omschreven. Hierbij moeten er voor de back-end unit tests zijn die de geschreven code volledig (streven is 100% code coverage) testen. Bovendien is het item door de ontwikkelaar (Jeroen) handmatig gecontroleerd op correcte werking en zijn eventuele verbeterpunten uit een code review verwerkt.

5.1.3 Afwijkingen bij het gebruik van Scrum

Scrum is vastgesteld volgens een set aan afspraken en regels, waarmee het een veelzijdig te gebruiken ontwikkelmethode is. Er zijn echter een aantal regels die niet of minder goed aansluiten bij de manier van werken aan dit project. Daarom heb ik op een aantal punten afgeweken bij het gebruik van Scrum. Deze afwijkingen staan hieronder toegelicht.

De eerste grote afwijking is de eerste sprint (oriëntatie-sprint). Dit is qua opzet een volledig andere sprint dan de rest van de sprints. Aan het begin van deze sprint is de backlog nog niet gevuld, omdat deze wordt opgesteld op basis van het oriënteren en vaststellen van de requirements. Dit betekent dat er nog geen items in sprint kunnen worden genomen. Bovendien is er tijdens de eerste sprint geen softwareontwikkeling gepland.

De tweede afwijking zit in de verdeling van de scrum-rollen. Bij Scrum zijn er namelijk verschillende rollen: Dit zijn de Product Owner, de ontwikkelaar en de Scrummaster. Hierbij is de Product Owner verantwoordelijk voor de backlog, de ontwikkelaar ontwikkelt de backlog items en de Scrummaster zorgt ervoor dat Scrum op de goede manier binnen de organisatie wordt toegepast.
Bij dit project is Steven Losekoot de Product Owner, maar hij is vanwege zijn beperkte aanwezigheid bij Vorsen (i.v.m. andere werkzaamheden) niet in staat om de backlog zelf bij te houden. Hierdoor moest ik een deel van de taken van de product owner op mijzelf nemen. Daaronder vallen het aanmaken van backlog items, het invoeren van items in JIRA.

Bovendien is er bij Vorsen geen scrummaster, waardoor ik zelf zowel ontwikkelaar als scrummaster was. Een aantal taken van de scrummaster, zoals het aanleren van Scrum in de organisatie, waren niet relevant en heb ik niet hoeven uitvoeren.

De derde afwijking betreft het ontbreken van de daily standup. Scrum schrijft normaalgesproken voor om een daily standup of daily scrum te houden om de stand van zaken binnen het team te bespreken. Omdat ik alleen aan het afstudeerproject werk (en dit werk los staat van het werk aan de applicatie wat door overige collega’s wordt uitgevoerd) was het niet mogelijk om deze stand-up te houden. Ik hield zelf wel in de gaten of het werken aan de backlog-items volgens schema verliep. Daarnaast verloopt de communicatie met de opdrachtgever in principe mondeling of via e-mail, en tijdens de demo-presentatie wordt de voortgang alsnog getoond.
5.2 Front-end ontwikkeling met AngularJS

De applicatie AscMe bestaat uit twee grote componenten. Dit zijn een front-end (website) en een back-end (applicatie op de server). Door middel van Client-Server communicatie (REST met JSON over HTTPS) staan de front-end en de back-end in verbinding.

De Front-end van dit project is met het web-framework AngularJS gebouwd. AngularJS is een JavaScript-framework voor het ontwikkelen van Single-Page webapplicaties. Single Page application is een term die inhoudt dat bij het navigeren naar de webpagina de basiscode slechts eenmalig hoeft te worden uitgevoerd. Vervolgens verloopt het navigeren tussen verschillende pagina’s supersnel, omdat alleen de data voor de nieuwe pagina hoeft te worden opgehaald.

Een webapplicatie gemaakt met AngularJS heeft een herkenbare structuur. De applicatie bestaat uit HTML (Templates) en CSS voor het tonen van de pagina, en uit JavaScript bestanden met de Angular-code. Deze code is verder te verdelen in verschillende onderdelen. Dit zijn: Controllers, Services, Resources, Directives, Filters en Components. Elk van deze onderdelen vervult een specifieke taak, samen vormen deze onderdelen de gehele applicatie. Onderstaande tabel toont de in AscMe gebruikte onderdelen van AngularJS:

<table>
<thead>
<tr>
<th>Naam AngularJS-component</th>
<th>Functie van component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template</td>
<td>HTML-code die kan worden gebruikt in een Controller of Component. Kan ook een apart HTML-bestand zijn, maar dan staat er een hele pagina op.</td>
</tr>
<tr>
<td>Component</td>
<td>Klein onderdeel of bouwsteen van de applicatie. Toont een gedeelte van een scherm, zoals een lijst met personen, een workflow of de basisgegevens van een opleiding</td>
</tr>
<tr>
<td>Service</td>
<td>Herbruikbaar onderdeel voor complexere logica. Is geschikt om data uit de back-end op te halen, berekeningen te maken of gegevens te sorteren</td>
</tr>
<tr>
<td>Directive</td>
<td>Herbruikbaar onderdeel wat veelal een achtergrondtaak doet, zoals het sorteren van een tabel, het controleren van twee wachtwoorden of het markeren of een veld vereist is. Met de komst van Components minder in gebruik.</td>
</tr>
<tr>
<td>Filter</td>
<td>Component voor het manipuleren van tekst die in de front-end wordt getoond. Bijvoorbeeld het omzetten van cijfers naar geldbedragen, het omzetten van datum naar het Nederlandse formaat.</td>
</tr>
<tr>
<td>Resource</td>
<td>Onderdeel van de module Angular-Resource (Wel officieel van AngularJS). Een resource is een service die geschikt is om CRUD-operaties naar de back-end te versturen en te ontvangen. Beschikt over enkele vooraf gedefinieerde methodes. Kan worden uitgebreid met eigen methodes voor aangepaste requests naar de back-end.</td>
</tr>
<tr>
<td>Route</td>
<td>Onderdeel van de Angular-library Angular UI-router *(dus NIET standaard bij AngularJS inbegrepen!) om de applicatie in verschillende subpagina’s op te delen. Elke pagina krijgt zijn eigen URL, template en eventueel controller. Wordt als hoofponderdeel een component gebruikt, dan is het template een html-tag met daarin het component.</td>
</tr>
<tr>
<td>Controller</td>
<td>Verouderd sinds Angular 1.5.x. Werd gebruikt om data voor een hele pagina te tonen, maar is moeilijker herbruikbaar. Nog wel in AscMe aanwezig bij de oudere stukken code</td>
</tr>
</tbody>
</table>

Figuur 2a: Tabel die de onderdelen van een AngularJS-applicatie omschrijft
Om data op een nette manier in de HTML te kunnen tonen beschikt AngularJS over een bijzondere eigenschap: het gebruik van 2-way binding. Dit is een mechanisme waarbij Angular er real-time voor zorgt dat data tussen de JavaScript en HTML wordt gesynchroniseerd. Dit betekent dat wanneer een waarde van een variable in het script (dynamisch) wordt aangepast, dat deze wijziging direct in de browser zichtbaar is.

Figuur 2b: Globale werking van een AngularJS-webapplicatie
5.3 Back-end ontwikkeling met Spring en Hibernate

Het tweede component van AscMe is de **back-end**. Dit onderdeel werkt op de server en is verantwoordelijk voor het verwerken en het tonen van data uit de database, authenticatie en communicatie met systemen van de onderwijsinstelling.

De back-end van AscMe is opgebouwd rondom het Java-framework Spring, in combinatie met Hibernate en Spring Data JPA.

Het *Spring* framework is een groot Java-framework wat geschikt is voor het bouwen van bedrijfsapplicaties in java. Hiervoor biedt het vele functionaliteiten op het gebied van webapplicaties, database-management (transacties) en Dependency Injection. Daarmee kunnen klassen door Spring worden geïnstantieerd, waardoor er van elk component slechts een instantie (singleton) actief is. Het injecteren van een dependency gebeurt met de Java-annotatie "@Autowired". Door deze op de constructor van een Java-klasse te plaatsen, "weet" het Spring framework dat de aanwezige dependencies (beans) kunnen worden geïnjecteerd.

![Spring Framework Runtime](image)

**Figuur 3: Officieel diagram van het Spring framework. Opmerking: Niet alle mogelijkheden worden voor de applicatie AscMe gebruikt.**

5.3.1 Database

5.3.2 Overige frameworks / libraries

Belangrijk om te vermelden (i.v.m. aflezen figuren en code) is dat AscMe gebruik maakt van Project Lombok [3]. Dit is een pakket voor het automatisch genereren van Getter en Setter-methodes, en voor het eenvoudig definiëren van de Java-methodes Equals en Hashcode. Dit werkt geheel met Java-annotaties, en voegt tijdens het compileren de daadwerkelijke methodes toe, zodat ze gewoon verschijnen in de byte-code die wordt gegenereerd. Bovendien is een plug-in voor de editor (IDE) nodig, omdat deze anders denkt dat de getter- en setter methodes niet bestaan. Deze plugin is echter eenvoudig te installeren.

De applicatie gebruikt een aantal libraries / pakketten van het bedrijf zelf. De belangrijkste hiervan zijn BeanMapper, Restzilla en JaRB.

BeanMapper [4] is een library om klassen naar een ander type te converteren (mappen), zodat een webresultaat (Result) niet alle velden met gevoelige informatie bevat. Ook kan bij het versturen van een request een formulier (Form) met een paar velden worden gemapt naar een database-entiteit. BeanMapper kan desgewenst een conversie uitvoeren door handmatig een convertor te schrijven.


JaRB (Java Repository Bridge [6]) is een library voor het uitvoeren van een aantal veelgebruikte database-taken. Dit omvat het ophalen van alle database constraints voor het eenvoudig maken van formulieren (om te zien welke velden vereist zijn, minimale en maximale lengte), Daarnaast kan het ook een testdatabase vullen bij het opstarten van de applicatie, en zet het foutmeldingen van de database om naar beter leesbare foutmeldingen.

Een interessant weetje is dat Restzilla en JaRB vrijwel volledig door Jeroen van Schagen zijn geschreven.
5.4 Versiebeheer

Vorsen maakt voor al haar projecten gebruik van versiebeheer via Git. Op deze manier staat de broncode veilig opgeslagen en kan er samen aan de applicatie worden gewerkt. De conventie is om per onderdeel een nieuwe branch aan te maken. Aan het eind van de sprint wordt een pull request aangemaakt, waarna een ander een code review uitvoert om te bepalen of het werk van voldoende kwaliteit geprogrammeerd is. Hieruit kunnen verbeterpunten naar voren komen, deze moeten dan door de ontwikkelaar worden opgelost. Pas als de reviewer akkoord gaat, wordt de gewijzigde code doorgevoerd (merge).

5.5 OTAP

Er wordt binnen Vorsen gebruik gemaakt van het OTAP-principe. Dit staat voor Ontwikkeling, Testen, Acceptatie en Productie en houdt in dat er vier verschillende omgevingen zijn waarop gewerkt en getest kan worden. Elke van deze omgevingen vervult een specifieke rol binnen de ontwikkelingscyclus van de applicatie.

De ontwikkelomgeving (O) bestaat enkel op de lokale machines van de ontwikkelaars en vormt het startpunt voor het werk aan de applicatie. Binnen deze omgeving worden nieuwe functies gebouwd en worden de unittests voor de back-end gedraaid. Deze tests dienen te slagen voordat een release naar de testomgeving (T) kan worden aangemaakt.

De test-omgeving (T) is een server (VPS) met daarop een exemplaar van de applicatie (zowel de back-end als de front-end) geïnstalleerd. Hierop kunnen ontwikkelaars zelf, door middel van Secure Shell (SSH) of continuous integration (Jenkins) deployments uitvoeren om te kijken of de applicatie goed op een server werkt. Dit is met name belangrijk om de applicatie te testen in combinatie met een echte Postgresql database (voor ontwikkeling wordt immers een in-memory HSQLDB-database gebruikt) en met een Apache Tomcat applicatieserver. Er wordt buiten het zelf testen door ontwikkelaars verder niets met deze server gedaan.

Voor dit project heb ik een eigen testomgeving (T2) gehad, zodat ik daar deployments op kon uitvoeren terwijl mijn collega’s de gewone testomgeving konden gebruiken. Deze testomgeving was in het begin identiek aan de gewone testomgeving (T). Vanwege de inrichting van Jenkins (via een speciale deployment-branch) kon ik echter geen gebruik maken van Continuous integration. Daarom heb ik al snel een Bash-script geschreven wat via SSH een deployment uitvoert. Dit werkte sneller dan via Jenkins, en voorkwam een hoop nutteloze “deploy” commits naar een aparte branch.

Zodra de ontwikkelaars de functionaliteit goed genoeg bevinden, kan via Operations (via 42 B.V.) een release naar de acceptatie-omgeving (A) worden aangemeld. De acceptatie-omgeving is identiek aan de test-omgeving, alleen kan de klant (in dit geval de VU) erbij om zelf tests uit te voeren. Dit zijn in het geval van de VU gebruikersacceptatietests. Ontwikkelaars kunnen dus niet op deze server inloggen, en kunnen ook niet rechtstreeks bij de database komen.

Als de klant de versie op acceptatie heeft goedgekeurd, wordt via Operations een release naar de productie-omgeving (P) aangemaakt. Dit is de omgeving waarin alle gebruikers de applicatie dagelijks gebruiken. Op deze omgeving worden uiteraard geen tests of ontwikkelwerkzaamheden uitgevoerd. Bovendien wordt deze omgeving regelmatig geback-uppt om in geval van problemen een oude versie terug te kunnen zetten.
5.6 Planning

Omdat ik voor deze opdracht Scrum gebruik en dit een iteratieve ontwikkelmethode is, was het niet mogelijk om van tevoren een volledige planning voor het project te maken. Omdat de doelstelling van deze opdracht uit verschillende delen bestaat, was het wel mogelijk om een globale planning voor dit project op te stellen. Hierbij heb ik de planning opgesplitst per onderdeel uit de doelstellingen van de opdracht.

Oriëntatiefase

Omdat de opdracht een project aan een grote applicatie betreft en de requirements nog niet bekend zijn, bestaan de eerste twee weken van het project uit een oriëntatiefase (sprint-0). Tijdens de oriëntatiefase zal de nadruk liggen op het voorbereiden en het opstarten van het software-ontwikkeltraject en het uitvoeren van een analyse door het verkrijgen van de requirements. Tijdens deze sprint zullen er de volgende werkzaamheden plaatsvinden:

- Verkennen van de bestaande applicatie door middel van het navigeren door de userinterface, inzicht in relevante delen van de code, doorlezen van de relevante documentatie en het maken van klassendiagrammen van relevante delen van de huidige situatie.
- Het maken van een voorstel en ontwerpen voor de dynamische workflow-engine. Dit omvat het maken van wireframes en een presentatie.
- Het houden van één of meer besprekingen bij de VU, waarbij de gemaakte voorstellen en de hieronder vermelde roadmap worden aangedragen.
- Het uitwerken van de uitkomsten van deze besprekingen, waarbij steeds de voortgekomen requirements in kaart worden gebracht en op basis hiervan de analyse kan worden uitgebreid en verbeterde voorstellen kunnen worden gedaan.

Nadat de oriëntatie-sprint is afgerond, is het de bedoeling dat de wensen in kaart zijn gebracht en dat er op basis van de gemaakte prioritering van de requirements een backlog kan worden opgesteld. Vanwege het iteratieve karakter van Scrum kon er vooraf er geen volledige planning worden gemaakt, maar is op basis van de inmiddels aangepaste opdrachtomschrijving een globale planning van uit te voeren taken gemaakt. Deze planning staat op de volgende pagina:
Sprint 1: **Vereiste onderdelen voor de workflow-module.** Uit de eerste bespreking met de VU is al gebleken dat ondersteuning voor notificaties eerst moeten worden ontwikkeld. Dit onderdeel wordt dus in de eerste sprint ontworpen, ontwikkeld en getest.

Sprint 2: **Workflow-module.** Ontwerpen van klassesdiagrammen van te wijzigen onderdelen, bouwen initiële versie workflow-module en testen van gebouwde workflow-module.

Sprint 3: **Uitbreiden workflow-module.** Verwerken feedback vorige sprint, aanvullen ontwerpen workflow-module, bouwen uitgebreidere versie workflow-module en testen hiervan.

Sprint 4: **Afronden workflow-module en starten rapportage-module.** Verwerken feedback vorige sprint, starten met ontwerpen van de rapportagomodule, starten met bouwen van rapportage-module en testen initiële versie rapportagomodule.

Sprint 5: **Afronden Rapportage-module en overige kleine verbeteringen.** Verwerken feedback vorige sprint, verbeteren en testen van de rapportagomodule, ontwerpen wizards voor ontbrekende workflows, bouwen wizards die nog ontbreken, testen wizards die nog ontbreken.

Sprint 6: **Afronden project.** Er zal worden gecontroleerd of de opgeleverde applicatie klaar is om in de productieomgeving van de VU te worden gebruikt. Afhankelijk van de kwaliteit en de bruikbaarheid zal er al dan niet direct een release worden uitgevoerd. Indien het systeem nog niet geschikt is voor gebruik in een productieomgeving, wordt een lijst van nog uit te voeren taken opgesteld, zodat Vorsen B.V. hiermee aan de slag kan om het nieuwgebouwde systeem alsnog geschikt voor productie te krijgen.
6. Situatie bij aanvang afstudeerperiode

Zoals reeds in de opdrachtomschrijving (hoofdstuk 3) is vermeld, betreft deze opdracht het aanbrengen van nieuwe functionaliteiten en verbeteringen aan de bestaande applicatie AscMe, zodat deze eenvoudiger bij nieuwe onderwijsinstellingen in gebruik kan worden genomen. Hiermee hoopt Vorsen de applicatie aan meer klanten te kunnen verkopen. Naar aanleiding van de directe wijzigingen ten opzichte van het Afstudeerplan (hoofdstuk 4) wordt er bij deze opdracht gericht op drie grote verbeterpunten: Notificaties, Workflows en Voortgangsoverzichten. In dit hoofdstuk wordt van elk van deze drie onderdelen de situatie bij aanvang van de afstudeerperiode geschetst.

6.1 Staat

De applicatie was bij aanvang van de periode van productiekwaliteit, maar bleek lastig uit te rollen naar nieuwe onderwijsinstellingen. De niet-configurerbare workflow-module, de zeer beperkt ondersteunde en niet-configurerbare notificaties en de beperkte voortgangsoverzichten waren hier de grootste oorzaak van. Omdat deze opdracht zich op deze punten richt, licht ik de situatie bij aanvang voor elk van deze onderdelen toe.

6.2 Notificaties

In de oude situatie beschikte de applicatie over enkele vooraf gedefinieerde gevallen waarbij er een notificatie werd verstuurd. Deze gevallen betroffen:

- Het reageren op een reactie (opmerking) bij een onderdeel waarop gereageerd kon worden
- Het afkeuren van een statuswijziging (Is een technisch onderdeel, maar dit omvatte mede het afkeuren van een workflow)
- Het afkeuren van een individuele wijziging (zoals het afkeuren van een aangepaste tekst, of het verhinderen van een wijziging aan de betrokkenen van een onderdeel).

Aan de notificaties was niets in te stellen en nieuwe gebeurtenissen konden niet eenvoudig worden toegevoegd. Dit kwam omdat de sjablonen met de teksten van de notificaties in het interne systeem van de VU stonden. De uitrol van AscMe naar nieuwe onderwijsinstellingen werd belemmerd, omdat dit notificatie-gedeelte te specifiek voor de VU was ontwikkeld. Bovendien waren de berichten die werden verstuurd niet duidelijk, waardoor de berichten de betrokkenen niet echt stimuleerden om hun taken op tijd uit te voeren. Deze opdracht richt zich erop om de configuratie van notificaties flexibeler te laten verlopen, wat ervoor moet zorgen dat de betrokkenen actief op de hoogte worden gehouden over de voor hun relevante taken.

6.3 Workflows

Er waren bij aanvang van de opdracht zes workflows in de applicatie aanwezig. Deze workflows waren specifiek voor de VU gebouwd - ze boden ondersteuning bij hun bedrijfssprocessen. Het was in de situatie bij aanvang zeer beperkt mogelijk om deze workflows in te stellen. Hoewel de titel, sommige teksten en de betrokkenen waren in te stellen, waren alle pagina’s ingebouwd in de programmacode van de applicatie. Het was enkel mogelijk om de volgorde van de pagina’s te kiezen, maar de inhoud van de pagina’s was niet aan te passen.

Bovendien zaten de workflows aan de momenteel actieve status van een object (dit kan een faculteit, opleiding of een vak zijn) gekoppeld, waardoor er altijd maar één workflow actief kon zijn. De enige
uitzondering hierop was het inroosteren van opleidingen, omdat dit buiten het workflow-systeem om was gebouwd. De harde koppeling met statussen van objecten, plus het feit dat de pagina’s ingebouwd waren, maakten het lastig om AscMe naar nieuwe onderwijsinstellingen uit te rollen. Deze opdracht richt zich erop om de workflows flexibeler en beter configureerbaar te maken, zodat de workflows beter aansluiten bij de processen waarover de onderwijsinstelling beschikt.

6.4 Voortgangsoverzichten

Er waren in de oude situatie op verschillende plekken voortgangsoverzichten aanwezig. Dit maakte het voor eindverantwoordelijken lastig om een goed inzicht in de voortgang van de door hun verantwoordelijken uit te voeren taken. Onderstaand screenshot toont hier een voorbeeld van: Er staan rapportages in het menu “Rapportages (1)”, maar het getoonde rapportage is via de faculteit op het tabblad “Status” (2) getoond. Vervolgens bevat het informatie over workflows (3), wat de titel “Status” niet zou doen vermoeden.

![Figuur 4: Het was onduidelijk waar voortgangsrapportages te vinden waren. De rapportages stonden op verschillende plekken.](image)

6.4.1 In het menu “Rapportages”

Als eerste waren er twee voortgangsoverzichten in de bovenbalk van de applicatie, onder het menu “Rapportages”, aanwezig.

Het eerste voortgangsoverzicht uit dit menu, genaamd “Voortgang roosteruitvraag” was – zoals de titel al suggereert – toegespitst op het monitoren van de voortgang van het roosteren van opleidingen. Dit overzicht bevatte alle faculteiten, welke uit te klappen waren zodat de bijbehorende opleidingen zichtbaar waren. Naast elk onderdeel stonden balkjes met de voortgang sven het roosteren. Er stond op de pagina van de rapportage helemaal geen uitleg wat de getoonde voortgangsindicatoren betekenen, maar het bleek om aantallen vakken die reeds ingeroosterd waren te gaan.

Het tweede voortgangsoverzicht uit het menu Rapportages, genaamd “Statistieken”, toonde enkel een grafiek van het aantal gebruikers van de applicatie per dag. Er waren er geen andere statistieken zichtbaar.
6.4.2 Op de pagina’s van onderdelen

Verder waren er voortgangsoverzichten verstopt op de pagina’s van de onderdelen: bij een faculteit stond er een voortgangsrapportage op het tabblad “status”, bij een opleiding stond er een voortgangsrapportage op het tabblad “workflow”.

De overzichten bij de pagina’s van de onderdelen toonden de voortgang van de huidige openstaande workflow. Bij een faculteit was dit per opleiding gegroepeerd. Naast elke opleiding stond een kolom met het uit te voeren onderdeel. Indien dit was voltooid, stond er een vinkje met wie het op welke tijd had afgerond.

Bij een opleiding was dit rapportage anders ingericht. Er was een verticaal overzicht met daarin alle fasen die binnen de huidige workflow doorlopen moesten worden. De vakken van de opleiding stonden in deze fasen ingedeeld.

Idealiter staan de voortgangsoverzichten op een centrale plek, en tonen deze meer relevante informatie over de voortgang van de verschillende processen binnen de applicatie.
II. UITVOERING VAN HET PROJECT
7. Start van het project (oriëntatiefase, sprint 0)

De eerste twee weken van de afstudeerperiode vormden de oriëntatiefase van het project. In deze twee weken heb ik de benodigde stappen uitgevoerd om het eigenlijke werk aan de deze afstudeeropdracht te kunnen starten. In dit hoofdstuk staan de in deze fase uitgevoerde activiteiten beschreven.

7.1 Verkennen van de applicatie en vaststellen van de requirements

De eerste week bestond uit het verkennen van de bestaande applicatie. Op basis van het verkennen en van overleg konden de eerste requirements worden opgesteld. Omdat de betreffende applicatie voor mij nog onbekend was, moest ik de architectuur en de werking van de applicatie eerst in kaart brengen. Door middel van het doornemen van de documentatie en het doorlopen van delen van de programmacode verkreeg ik inzicht in de architectuur en de werking van de applicatie.

Nadat ik basale kennis van de architectuur en de werking van de applicatie had verkregen, heb ik de voor deze opdracht relevante delen (Workflows en Voortgangsoverzichten) van de applicatie uitvoerig doorgenomen. Door middel van het volledig doornemen van de bijbehorende code heb ik van het workflow-geheugen en (later) de notificatie-geheugen een klassendiagram gemaakt. Een stukje van deze diagrammen staat hieronder, de volledige diagrammen zijn te vinden in [bijlagen 3 en 4].

![Diagram Workflow en Notificaties](image-url)
De afhankelijkheid van klassen als “StatusTypeName”, het gebrek van datumvelden bij “Workflow” en de beperkte set methoden in “NotificationFactory” bevestigden wat in de opdrachtomschrijving werd vermeld: De processen binnen de applicatie zijn ‘hard geconfigureerd’ in de code van de applicatie, er kunnen geen deadlines worden ingesteld en er worden onvoldoende meldingen aan betrokkenen gestuurd.

7.1.1 Workflows

Naast het doornemen van de achterliggende code heb ik ook de gebruikersinterface van de bestaande workflow-module grondig doorgenomen. Tijdens het doornemen vielen een aantal zaken op:

- De reeds aanwezige workflows hadden allen een summier omschrijving. De aanwezige omschrijvingen en teksten bevatten op meerdere plekken taalfouten en grammaticale onjuistheden. Hierdoor was het voor een buitenstaander niet duidelijk wat er precies in de workflow gebeurde.
- Alle workflows in de applicatie waren al in wizards ingedeeld. Dat viel op, omdat de opdrachtomschrijving vermeldde dat dit nog niet bij alle workflows het geval was. Volgens de opdrachtomschrijving moesten de workflows die nog niet in wizardvorm beschikbaar waren namelijk worden geanalyseerd en indien mogelijk in wizards worden geplaatst. Zie ook hoofdstuk 3 “Directe wijzigingen ten opzichte van het afstudeerplan”, waarin wordt toegelicht wat dit voor gevolgen voor de afstudeeropdracht had.
- Het was hierdoor minder duidelijk wat er op het gebied van workflows moest gebeuren. Daardoor konden er in de eerste week van de oriëntatiefase nog weinig requirements van de workflows worden opgesteld.

De requirements die wel bekend waren van de workflow, hadden met name betrekking op het dynamisch maken van de inhoud van de pagina’s. Zo was bekend dat er ingebouwde en dynamische componenten zouden zijn (WFE-P001, WFE-P006), en dat deze als sjablonen zouden worden opgeslagen, zodat de aangemaakte componenten hergebruikt kunnen worden. Pagina’s waren of een inleidings-, of een gegevens- of een afrondende pagina.

Tot slot was er een requirement voor het sturen van meldingen (notificaties) wanneer de deadline van een workflow naderde (N-014).

7.1.2 Notificaties

Nadat bekend was geworden dat er een nieuwe notificatie-module in de applicatie moest worden ingebouwd, heb ik de momenteel aanwezige functionaliteiten voor het versturen van notificaties nader bekeken. Ik zag hierbij dat er enkele notificaties naar een intern notificatiesysteem van de VU werden gestuurd. Het was bovendien niet mogelijk om de teksten van de notificaties aan te passen: Deze stonden in een intern systeem van de VU opgeslagen.

Ook waren de notificaties nog niet zichtbaar in de applicatie zelf. Er was de wens van Vorsen om dit ook zichtbaar te maken, maar het was nog niet bekend hoe. Ik kwam tot de conclusie dat er twee mogelijkheden waren om notificaties in de applicatie te tonen: Door middel van een widget (venster) op het dashboard, of als uitklapbaar menu boven in de applicatie. Omdat het dashboard al erg vol stond met andere widgets, heb ik met de opdrachtgever besloten om notificaties in een notificatie-menu bovenin de applicatie te tonen. Bovendien zie je dit soort notificatie-menu’s tegenwoordig steeds vaker op andere websites, waardoor het voor gebruikers herkenbaar is.
Al met al was de wens vanuit Vorsen om een nieuw notificatiesysteem te maken waarbij de teksten lokaal stonden opgeslagen en waarbij de notificaties via e-mail werden verstuurd. Dit omdat de huidige aanpak te specifiek voor de VU was. De koppeling met systemen van de VU vormde een probleem om de applicatie naar andere onderwijsinstellingen uit te rollen.

Het was echter nog niet bekend wat de VU hiervan vond. Daarom moest eerst worden nagegaan bij de VU wat zij van het versturen van notificaties via e-mail vonden. Hiervoor (en voor informatie over de workflows) ben ik op 9 februari 2017 met Jeroen van Schagen naar een meeting bij de VU geweest.

De functioneel beheerder van de VU gaf tijdens de meeting aan het goed te vinden dat de notificaties via e-mail werden verstuurd. Dat was verrassend, maar de functioneel beheerder had er ook goede redenen voor: Het invoeren van de teksten voor de notificaties in het eigen notificatiesysteem was lastig, en er werden niet-relevante banners en logo’s aan verstuurde meldingen toegevoegd.

Al met al was de VU zeer positief over het nieuw te ontwikkelen notificatie-gedeelte. Ze vonden het een goede oplossing voor het probleem dat er regelmatig betrokken zijn die hun taken te laat of niet uitvoerden. Vanwege de sterke wens om het nieuwe notificatie-gedeelte te maken (zowel vanuit Vorsen als de VU), kreeg dit de hoogste prioriteit en werd het op de backlog voor de eerste sprint geplaatst.

Op basis van het verkennen en de bespreking met de VU konden nu de requirements voor de notificaties worden opgesteld. De volledige requirements staan in het Requirements document, maar hieronder staat een samenvatting van de requirements:

- Versturen van e-mailnotificaties (N-001 t/m N-006)
- Notificaties in de applicatie (N-007 t/m N-010)
- Sjablonen voor de teksten voor notificaties (N-011 en N-012)
- Handmatige notificaties (N-013)

Tot slot zijn alle requirements van de notificaties met Vorsen B.V. gevalideerd, en geprioriteerd volgens MoSCoW.

7.3 Eerste ontwerpen van de gebruikersinterface (GUI)

Nadat de requirements in de eerste week waren vastgesteld, heb ik tijdens de tweede week van de oriëntatiefase een serie ontwerpen van de gebruikersinterface (GUI-ontwerpen) van het te bouwen notificatie-gedeelte en de dynamische workflow-module gemaakt. Deze GUI-ontwerpen bestaan uit wireframes voor de workflow-module en mock-ups voor het notificatie-gedeelte. Al deze GUI-ontwerpen zijn te vinden in [bijlage 5].

Voor het notificatie-gedeelte heb ik geen wireframes, maar mock-ups gemaakt van de verschillende benodigde onderdelen. Mock-ups waren een beter middel om dit onderdeel vast te leggen, omdat de requirements op dit moment al duidelijk waren geworden. Bovendien moest de ontwikkeling hiervan spoedig beginnen, waardoor een ontwerp dat voor de gebruiker duidelijk is meer voor de hand lag.

Deze mock-ups omvatten een gewijzigd instellingenscherm voor gebruikers, het notificatie-menu wat bovenin de applicatie wordt getoond, schermen voor het beheren en bewerken van notificatie-sjablonen, en een scherm voor het instellen van notificaties bij een workflow.
Van de nieuw te bouwen workflow-module heb ik in de eerste sprint enkel wireframes gemaakt. Dit kwam omdat ik hiervoor meer tijd nodig had om de eisen en wensen in kaart te brengen en dit een lastiger te bouwen onderdeel was. Deze wireframes betreffen alleen de zichtbare kant van dit onderdeel, want van de interne componenten is pas tijdens een latere sprint een technisch ontwerp gemaakt. Een voorbeeld van zo'n wireframe staat hieronder.

![Wireframe voor bewerken van sjabloon met gegevens voor een workflow. Dit werden later de "Aangepaste workflow-pagina's."](image)

Figuur 6: Wireframe voor bewerken van sjabloon met gegevens voor een workflow. Dit werden later de "Aangepaste workflow-pagina's."

Op de wireframes van de workflow-module is te zien hoe een workflow-scherm kan worden samengesteld, en welke opties er voor het maken van een workflow beschikbaar zijn. Hierbij moet worden gedacht aan het instellen van teksten, het toekennen van gebruikersrollen en het kiezen waar de workflow betrekking op heeft (studies, vakken, faculteiten etc.). Per scherm kan een sjabloon met gegevens worden gebruikt, zodat pagina's herbruikbaar zijn binnen verschillende workflows. Een sjabloon samenstellen kan door te kiezen van welk object uit de database er gegevens moeten worden getoond, en deze eventueel te filteren. Vervolgens kan worden aangegeven welke modi (be kijken, wijzigen en geschiedenis) worden ondersteund. Ten slotte kan er een inleidende en een afsluitende pagina aan de workflow worden toegevoegd, waarmee de workflow volledig is samen te stellen.

De gemaakte mock-ups en wireframes heb ik tijdens de afsluitende meeting van de oriëntatiefase, op 22 februari 2017, bij een presentatie op de VU getoond. De VU was positief over mijn ideeën over de notificaties, en met een paar kleine wijzigingen aan de GUI-ontwerpen kon met de ontwikkeling hiervan worden gestart. Over de wireframes van de workflow-module kon de VU tijdens de meeting geen duidelijke mening geven. Dit kwam omdat deze schermen alleen voor de functioneel beheerder relevant zijn, terwijl zij niet bij deze meeting aanwezig kon zijn. De vervangend functioneel beheerder die wel aanwezig was, had hierover ook geen duidelijke mening. Later zou blijken dat de workflow-module ook meer voor het product in het algemeen is bedoeld, dan voor de implementatie bij de VU.
7.4 Requirements vanuit de VU

Tijdens de eerste meeting bij de VU (9 februari) kwam naar voren dat een aantal wijzigingen in de huidige versie van de workflow voor het uitvragen van studiegidsteksten noodzakelijk waren. (Requirements WFSG-001 t/m WFSG-005) De VU wenste echter dat deze wijzigingen direct werden geïmplementeerd, omdat zij dit, vanwege het naderende starten van de uitvraag van de studiegidsteksten, zeer snel in gebruik wilden nemen. De wens om dit door mij te laten implementeren ging echter tegen het doel van de oriëntatiefase van de afstudeeropdracht in, omdat ik had afgesproken dat er in de oriëntatiefase geen softwareontwikkeling zou plaatsvinden. Ik heb daarom samen met Jeroen besloten dat we deze wijzigingen samen doornemen, zodat ik op deze manier beter bekend kan raken met de werking van de applicatie. Dit hebben we vervolgens gedaan, en daarmee waren deze requirements snel voltooid. Jeroen heeft er vervolgens voor gezorgd dat deze code is opgenomen in een nieuwe release, en die is vervolgens door Operations op de acceptatie- en productieomgevingen geplaatst.
8. Sprint 1 - Vernieuwde notificatie-module

De eerste “echte” sprint van de afstudeerperiode richtte zich op het ontwikkelen van een nieuwe notificatie-module voor de applicatie. Deze bestond uit verschillende schermen en componenten en is in de eerste sprint deels gebouwd.

8.1 Plannen van de sprint

Omdat de notificatie-module de hoogste prioriteit had, heb ik de requirements hiervan bekeken en voorzien van story points. Deze Story Points heb ik zelf ingeschat, en zijn daarmee slechts een ruwe benadering van de benodigde tijd. Op basis van de story points in de backlog heb ik de planning voor de eerste sprint opgesteld. Deze sprint bestond uit de alle requirements voor het sturen van notificaties, behalve het versturen van notificaties voor workflows, omdat hiervoor de vernieuwde workflow-module vereist is. Hieronder staat een overzicht van de in deze sprint opgenomen backlog-items, inclusief het aantal Story Points.

<table>
<thead>
<tr>
<th>Req.</th>
<th>Naam item</th>
<th>Schatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-001</td>
<td>Ondersteuning voor e-mail notificaties</td>
<td>13 SP</td>
</tr>
<tr>
<td>N-002</td>
<td>Instellen frequentie e-mailnotificaties per gebruiker</td>
<td>5 SP</td>
</tr>
<tr>
<td>N-005</td>
<td>Taal e-mailnotificaties per gebruiker instelbaar</td>
<td>2 SP</td>
</tr>
<tr>
<td>N-006</td>
<td>Ondersteuning voor klikbare links in e-mailnotificaties</td>
<td>8 SP</td>
</tr>
<tr>
<td>N-007</td>
<td>Meldingenoverzicht in de applicatie</td>
<td>5 SP</td>
</tr>
<tr>
<td>N-008</td>
<td>Aantal ongelezen meldingen als rode badge in de applicatie</td>
<td>1 SP</td>
</tr>
<tr>
<td>N-009</td>
<td>Bij klikken op melding in meldingenoverzicht naar juiste pagina gaan</td>
<td>3 SP</td>
</tr>
<tr>
<td>N-010</td>
<td>Wissen meldingen in meldingenoverzicht</td>
<td>1 SP</td>
</tr>
<tr>
<td>N-011</td>
<td>Sjablonen voor teksten e-mailmeldingen beheren</td>
<td>8 SP</td>
</tr>
<tr>
<td>N-012</td>
<td>Meldingen direct of op schema versturen</td>
<td>13 SP</td>
</tr>
<tr>
<td>N-013</td>
<td>Handmatige notificatie versturen vanaf pagina faculteit of opleiding</td>
<td>8 SP</td>
</tr>
<tr>
<td>N-016</td>
<td>Frequentie e-mailnotificaties per faculteit instelbaar</td>
<td>8 SP</td>
</tr>
<tr>
<td>N-017</td>
<td>Loggen van verstuurde e-mailnotificaties</td>
<td>8 SP</td>
</tr>
</tbody>
</table>
8.2 Klassendiagram notificatie-module

Voor de notificatie-module heb ik van tevoren een UML-klassendiagram gemaakt welke de verwachte werking van het onderdeel toont. Dit UML-diagram toont enkel het notificatie-gedeelte van de applicatie, omdat het anders veel te groot wordt.

Het UML-ontwerp is opgesplitst in verschillende packages, omdat de componenten betrekking hebben op verschillende vormen notificaties (e-mail en in de applicatie). Ook staan de sjablonen apart, omdat deze zowel voor de e-mail als de in-app notificaties worden gebruikt. Deze sjablonen zijn later ook gebruikt bij het versturen van workflow notificaties.
8.3 Ontwerpen gebruikersinterface (GUI)

De ontwerpen van de grafische user interface (GUI) waren reeds tijdens de oriëntatiefase van het project gemaakt. Tijdens deze sprint heb ik deze ontwerpen op enkele punten verbeterd en aangepast. In deze paragraaf worden de aangebrachte wijzigingen en verbeteringen verder toegelicht.

Uit de meeting bij de VU van 22 februari bleek dat de VU niet wenst dat gebruikers e-mailnotificaties kunnen uitschakelen, en dat gebruikers ook geen ander e-mailadres mogen opgeven voor het ontvangen van notificaties. Na overleg bleek dat Vorsen het met deze beslissingen eens was, en dat de bijbehorende requirements (N-003 en N-004) kwamen te vervallen. Het GUI-ontwerp van het scherm met gebruikersinstellingen voor notificaties is hierop aangepast.

Daarnaast is tijdens het ontwikkelen van het notificatie-gedeelte gebleken dat notificaties momenteel handmatig en door middel van een aantal vooraf gedefinieerde gebeurtenissen (triggers of events) kunnen worden verstuurd. Hierop is het scherm voor het bewerken van notificatie-templates aangepast, en toont het scherm voor het bewerken van notificatie-sjablonen voortaan een lijst van triggers in plaats van een doelgroep en een rol.

![Image of updated user interface for notifications]

_Figuur 8: Vernieuwd ontwerp gebruikersinstellingen voor notificaties._

![Image of updated notification template]

_Figuur 9: Vernieuwd ontwerp bewerken van een sjabloon voor notificaties._
8.4 Realisatie notificatie-module

Tijdens de eerste sprint lag veruit de meeste (misschien zelfs wel te veel) nadruk op het bouwen aan de applicatie. Tijdens deze sprint zijn dan ook de meeste onderdelen van de notificatie-module gebouwd. Het doel van Sprint 1 was om aan het eind van de sprint een zo groot mogelijk deel van de notificatie-module te hebben gebouwd, omdat de nadruk van deze opdracht lag op het ontwikkelen van een workflow-module - de notificatie-module was erbij gekomen.

Er waren tijdens het ontwikkelen van het notificatie-gedeelte verschillende uitdagingen, waarbij er afwegingen en keuzes moesten worden gemaakt.

8.4.1 Versturen van automatische e-mailnotificaties

Als eerste onderdeel heb ik het versturen van e-mailnotificaties geïmplementeerd. Het versturen van e-mails is een taak die door de back-end van de applicatie uitgevoerd. Voor het versturen van e-mails heb ik gebruik gemaakt van de JavaMail API.

Het werken met de JavaMail API leverde nauwelijks problemen op, omdat het Spring framework faciliteiten biedt om op een uiterst eenvoudige manier e-mails te kunnen versturen. Het inbouwen van de functie was bovendien nog eenvoudiger omdat Vorsen B.V. al ondersteuning voor het versturen van e-mails in een ander product heeft ingebouwd. Hierdoor kon de code voor het versturen van e-mail grotendeels worden overgenomen.

Het laatste vereiste onderdeel voor het versturen van e-mailnotificaties was het bijhouden van een digitaal logboek. Hiervoor heb ik een tabel notification_log in de database opgenomen. Daarin staan allereerst de kolommen afzender, titel en inhoud. Deze bevatten alle gegevens van het bericht. Bovendien geeft de kolom success aan of het bericht succesvol is verzonden en bevat de kolom error_message een eventuele foutmelding of stacktrace. Voor de user-interface heb ik een pagina gebouwd waarin de items uit het logboek worden getoond. Bij elk item staat of het bijbehorende e-mailbericht succesvol is verzonden. Indien een foutmelding is opgetreden wordt deze ook in het logboek-item getoond.

De wachtrij met notificaties die in bovenstaande figuur wordt getoond, is momenteel nog niet echt nodig, maar kan wel worden gebruikt om het aantal verstuurde e-mails te beperken. Dit wordt in de andere producten van Vorsen en 42 ook gebruikt, omdat sommige mailservers het niet toestaan dat er non-stop mails worden verstuurd. In de laatste sprint heb ik deze wachtrij verwijderd, waarmee het versturen van een e-mail een directe methode werd, hoewel deze wel op een aparte thread moest worden uitgevoerd. Hiermee was dit onderdeel beter te testen.
Beheren en gebruiken van sjablonen voor het sturen van e-mailnotificaties

Het volgende onderdeel dat op de backlog van Sprint 1 stond was de mogelijkheid tot het beheren en gebruiken van sjablonen voor e-mailnotificaties. Met deze sjablonen kan de (functioneel) beheerder instellen welke tekst er in een e-mailbericht moet komen te staan, wanneer een e-mailbericht moet worden verzonden en of het om een dringend bericht gaat.

Voor het beheren van notificatie-sjablonen had ik gedurende de oriëntatie-sprint diverse schermen ontworpen, welke in deze sprint geïmplementeerd zijn.

Bij het implementeren van notificatie-sjablonen zat de uitdaging in het generiek krijgen van het notificatiesysteem. Immers, er kunnen op drie totaal verschillende manieren notificaties worden verstuurd:
- Handmatig, vanaf het scherm van een faculteit of een opleiding
- Automatisch, bij vooraf gedefinieerde gebeurtenissen die zich tijdens het gebruik van de applicatie voordoen (Zoals in de oude situatie, maar eventueel ook voor nieuwe gebeurtenissen)
- Automatisch, op basis van deadlines (afhankelijk van later aangebrachte aanpassingen in de workflows).

De notificaties die in de eerste sprint zijn geïmplementeerd, zijn de automatische notificaties bij vooraf gedefinieerde gebeurtenissen. Het doel was om ook de handmatige notificaties te implementeren, maar dit is vanwege tijdgebrek niet gelukt. Deze heb ik in sprint 3 alsnog geïmplementeerd. De notificaties bij deadlines waren niet in de backlog van deze sprint opgenomen, omdat hiervoor de workflow-module nodig was. Uiteindelijk heb ik deze vorm van notificaties pas in de laatste sprint volledig kunnen implementeren.

In elk van deze gevallen dient de juiste tekst en de juiste groep ontvangers gekozen te worden. Bovendien zijn ervoor al deze gevallen wisselende tags beschikbaar om variabele velden in de tekst van de notificaties te vullen.

Om uit te leggen hoe ik het notificatiesysteem generiek heb gemaakt, is eerst een korte uitleg nodig:


Helaas werd de Event Bus in het geval van het versturen van notificaties niet optimaal gebruikt. Dat kwam omdat de drie gebeurtenissen die een notificatie-triggerden in een externe bibliotheek (client-vu-api) waren vastgelegd. Hieraan konden niet zomaar nieuwe Event Listeners voor nieuwe notificatie-gebeurtenissen worden toegevoegd.
Om dit probleem op te lossen, heb ik geanalyseerd hoe de notificaties op de meest generieke manier via de applicatie konden worden verstuurd. Ik heb besloten om hiervoor wel van de Event Bus gebruik te maken, maar om de verstuurde notificatie-gebeurtenissen generiek te maken. Dit heb ik gedaan door een extra interface te introduceren - en hier slechts één EventListener voor aan te maken. Deze EventListener is dus verantwoordelijk voor het versturen van alle notificaties binnen de applicatie. Hierbij kan er tijdens het afhandelen van een notificatie-gebeurtenis eenvoudig het bijpassende notificatie-sjabloon worden opgehaald en wordt vervolgens een e-mail naar de juiste betrokkene(n) verstuurd.

De notificatie-interface wordt vervolgens door alle gebeurtenissen waarvoor een notificatie moet worden verstuurd geïmplementeerd. Om op deze manier notificaties te kunnen sturen, zijn wel een aantal gegevens en functies nodig. Daarom vereist de nieuwgebouwde Notificatie-interface dat een aantal methoden in elke notificatie-gebeurtenis dienen te worden geïmplementeerd. Deze zijn ervoor om sjablonen aan gebeurtenissen te kunnen koppelen, om precondities voor het versturen van de notificatie te controleren, om de lijst van ontvangers samen te stellen en om de ondersteunde variabele velden (tags) voor het sjabloon te tonen en te vullen.
Met de nieuwe implementatie kunnen er heel eenvoudig nieuwe gebeurtenissen aan de applicatie worden toegevoegd, en kunnen er door de beheerder vanuit de applicatie sjablonen voor deze gebeurtenissen worden aangemaakt. Er kunnen zelfs meerdere sjablonen voor een bepaalde gebeurtenis zijn - alhoewel dit in de praktijk niet of nauwelijks voor zal komen.

8.4.2 Editor voor html-emailberichten

Bij het versturen van e-mail werd als eis gesteld dat het e-mailbericht met HTML moest kunnen worden opgemaakt. Op deze manier kan tekst- en stijlopmaak aan het bericht worden toegevoegd. Om sjablonen voor met HTML opgemaakte berichten te maken, was een geschikte teksteditor nodig. Het is voor gewone gebruikers immers niet gewenst om zelf html-codes in te moeten typen. Deze teksteditor dient in de front-end van de applicatie te worden ingebouwd. Na enig zoekwerk op internet kwam ik tot de conclusie dat er hiervoor verschillende oplossingen beschikbaar zijn.

Ik heb door middel van een vergelijkende test onderzocht welk pakket het best geschikt is voor gebruik in de applicatie.

Hiervoor heb ik eerst opgezocht welke grote pakketten er zijn. Pakketten kwamen enkel in aanmerking als deze over goede documentatie beschikken en geschikt zijn voor JavaScript, omdat dit de technologie is die in de front-end wordt gebruikt.

De gevonden pakketten heb ik vervolgens op een aantal relevante functies beoordeeld. Deze relevante functies heb ik samengesteld op basis van de wensen aan de applicatie (het invoegen van tags, afbeeldingen en html-opmaak in een tekst), en deels op de mogelijkheden die bij verschillende pakketten op de voorpagina’s stonden aangeprezen (html plakken, toegankelijkheid). Op basis van requirement NF-004 heb ik de minimaal benodigde Internet Explorer-versie meegenomen in de vergelijking. Daarnaast maakt Vorsen B.V. graag gebruik van open-source software, dus dit punt heb ik ook in de vergelijking meegenomen. Tot slot heb ik, omdat de code in een commercieel product wordt gebruikt, de prijs voor een commerciële licentie meegenomen.
Hierbij zijn de volgende pakketten met elkaar vergeleken: TextAngular, Froala, TinyMce, CKEditor.

Uit de vergelijking kwam naar voren dat de pakketten Froala, TinyMce en CKEditor nagenoeg dezelfde mogelijkheden bieden. Een van de pakketten (Froala) heeft echter minimaal Internet Explorer 10 nodig en kan bij voorbaat al niet worden gebruikt. Bovendien bleek Vorsen niet bereid om betaalde licenties af te willen sluiten om software-functionaliteiten in te bouwen, dus daarmee bleef er nog maar één editor over: TextAngular. Hoewel deze minder functionaliteit biedt, bleek het geen probleem te zijn om hiermee de e-mailteksten te kunnen bewerken. Dit is dan ook de editor die ik in de applicatie heb ingebouwd en waarmee de teksten voor de notificatie-sjablonen worden bewerkt.

De tabel van de vergelijking is als [Bijlage 6] aan het verslag toegevoegd.

Het toevoegen van TextAngular aan de front-end was eenvoudig: Het pakket is via bower, een dependency manager, te installeren. Vervolgens voeg je het als module toe aan de configuratie van AngularJS en daarmee is de installatie afgerond.

De editor is te instantiëren door een html-element met de tag <text-angular> aan de front-end toe te voegen. Hierin kun je aangeven welke opties er zijn ingeschakeld, door een element met het attribuut tekst-angular-toolbar te specificeren. Deze toolbar bevat een dropdown menu met alle toegestane tags, en plaatst deze door middel van Javascript in de tekst. Dit gebeurt door de methode window.getSelection() aan te roepen, en op de huidige plaats van de cursor een text node met de geselecteerde tag in te voegen. Hiermee is het mogelijk om de dynamische tags te plaatsen en werkt het bewerken van html-teksten zoals benodigd is.

8.4.3 Inloggen met een link vanuit een e-mailbericht

Nu de e-mailnotificaties kunnen worden verstuurd, moet het ook mogelijk zijn om - indien een gebruiker op een link in een e-mail klikt - de gebruiker automatisch in te loggen en vervolgens naar de juiste pagina te gaan.

Voor het inloggen in de applicatie heb ik gebruik gemaakt van token-authenticatie. Token-authenticatie bestaat uit een stuk willekeurig gegenereerde tekst (token), waar vervolgens een gebruiker en een actie aan zijn gekoppeld. Bij het versturen van een e-mailnotificatie wordt een token gegenereerd. De link met dit token wordt eerst verwerkt in het notificatie-sjabloon: Hij wordt op de plaats waar een tag naar een link stond geplaatst. Vervolgens wordt de notificatie naar de gebruiker gemaild. Indien hij of zij op de link klikt, wordt de back-end van de applicatie rechtstreeks aangeroepen. De back-end zoekt het token op in de database, en kijkt of dit bestaat. Als dat zo is, wordt de opgeslagen gebruiker ingelogd en stuurt de back-end een HTTP redirect request (301) naar de juiste pagina in de front-end. Hierdoor komt de gebruiker op de relevante pagina terecht.
Het implementeren van token-authenticatie was eenvoudig en verliep zonder verdere problemen. Het Spring Framework biedt uitgebreide mogelijkheden om beveiliging in te stellen, en hierdoor was het eenvoudig om een gebruiker in te laten loggen indien een token-link was aangeklikt.

Wat bij dit onderdeel lastig was, was het genereren van links naar de pagina’s. Er zijn immers verschillende objecten waarna een link moet kunnen worden gegenereerd. Zo kan een e-mail betrekking hebben op een workflow, maar ook op een opleiding of een faculteit. Of zelfs nog specifieker: op een bepaalde pagina binnen het scherm van een workflow of een faculteit.

Omdat het systeem voor notificaties generiek moet zijn heb ik hiervoor opnieuw gekozen om een Java-interface te definiëren. De Linkable interface biedt welgeteld één methode: haal de link naar het betreffende object op. Meestal staat hierin de naam van de pagina plus het id van het betreffende object. Soms is er meer logica nodig, omdat een workflow bijvoorbeeld altijd bij een ander onderdeel (opleiding, faculteit etc.) hoort. Dit kan echter allemaal in deze methode worden uitgevoerd. Het nadeel is dat er veel pagina’s zijn, en dat dus veel objecten over deze methode moeten beschikken. Er is echter geen andere manier om links te genereren, want de front-end gebruikt soms afwijkende structuur of namen voor pagina’s van objecten.

8.4.4 Notificaties en notificatie-instellingen in de applicatie

Het laatste onderdeel wat ik in deze sprint heb ontwikkeld is het tonen van notificaties in de applicatie zelf. Dit was een relatief eenvoudig onderdeel, en bestond uit het volgende:

- Ontwikkelen van een notificatie-menu met klikbare notificaties en een indicator voor ongelezen meldingen
- Ontwikkelen van een instellingenscherm voor gebruikers waarop zij de taal en frequentie van e-mailberichten kunnen instellen.

Het notificatie-menu wat in de applicatie zelf wordt getoond is een pop-up menu, zoals dat tegenwoordig op vele websites in gebruik is. Het haalt notificaties uit de database op, en zet ze in een lijstje neer. Er wordt bijgehouden of een notificatie is gelezen of niet, en er wordt geteld hoeveel ongelezen notificaties een gebruiker heeft.

Het instellingenscherm voor gebruikers is eenvoudig: Er zijn twee opties beschikbaar en deze kunnen vervolgens worden opgeslagen. De eerste optie is het instellen van de taal voor e-mailberichten Deze
wordt per gebruiker opgeslagen en wordt tijdens het verwerken van een notificatie-gebeurtenis gebruikt voor het plaatsen van de tekst. De tweede optie is het instellen van de frequentie voor niet-urgente e-mailnotificaties. Ook dit wordt bij de gebruiker opgeslagen, maar in deze sprint wordt hier verder nog niets mee gedaan.

Het ontwikkelen van deze twee componenten verliep redelijk vlot. Voor het werkend krijgen van het pop-up-menu heb ik wat kennis opgedaan over JQuery, want dit is nodig om het menu te sluiten indien er naast wordt geklikt. Tijdens de demo vonden de aanwezigen het notificatie-menu er erg mooi uitzien.

8.5 Testen van de notificatie-module

Het testen van de applicatie bestaat zowel uit het doorlopen van alle schermen en het schrijven van unittesten voor de back-end.

Het doorlopen van de schermen was voor de meeste onderdelen relatief eenvoudig: Openen, alle acties uitvoeren en kijken of alles werkt. Het testen van het versturen van e-mailnotificaties was iets lastiger. Het is immers geen goed idee om testmails naar echte e-mailadressen te sturen, dus dat moest voorkomen worden. Het was echter wel nodig om tijdens het testen een echt e-mailbericht te versturen. Dit omdat het verbinding met de mailserver en de vormgeving van het bericht gecontroleerd moesten worden. Om op een veilige manier test e-mails te kunnen versturen heb ik aan het component dat de e-mailberichten verstuurt een check toegevoegd. Door middel van het invullen van een e-mailadres bij de systeeminstelling mail.to worden de mailberichten altijd naar dat e-mailadres gestuurd - ongeacht de werkelijke ontvanger. Om te controleren of het e-mailbericht wel naar de goede ontvanger wordt gestuurd heb ik onderaan het bericht een footer met het e-mailadres waarvoor het bericht was bedoeld geplaatst.

Helaas ben ik in de eerste sprint niet toegekomen aan het schrijven van de overige unittesten. De reden hiervoor is dat het ontwikkelen van sommige onderdelen langer duurde dan ik had gedacht. Met name het verbeteren van de Event Bus en het genereren van links naar de pagina’s van objecten kostten veel tijd, waardoor het werk aan deze onderdelen langer duurde dan gepland.

8.6 Sprint Demo

De eerste sprint eindigde op woensdag 8 maart 2017. In een presentatie waarbij Jeroen van Schagen en Steven Losekoot aanwezig waren, heb ik al het voltooide werk getoond. Jeroen en Steven waren beiden zeer te spreken over de gemaakte componenten.

Na de demo was er ook een bespreking voor de volgende sprint, waarbij een aantal nieuwe requirements naar voren kwamen. Dit waren het instelbaar maken van workflows per faculteit. Vorsen wilde dat de beheerder van een faculteit zelf kan bepalen welke workflows er ingeschakeld moeten worden, omdat er zo meer vrijheid is om bepaalde workflows wel of niet in gebruik te nemen.

Hieruit kwamen de requirements WFE-F001 t/m WFE-F004 en WFE-F006/7 voort.
8.7 Sprint Retrospective

Tijdens de eerste sprint heb ik veel voortgang kunnen tonen. Uit de retrospective kwam naar voren dat een aantal onderdelen goed gingen, maar dat er ook de nodige verbeterpunten waren.

De onderdelen die goed gegaan waren zijn:
- Het ontwikkelen van de front-end voor de notificaties, zoals het notificatie-menu en de schermen om sjablonen voor notificaties te beheren
- Het implementeren van het versturen van e-mails
- Het implementeren van een logbestand voor notificaties

De verbeterpunten waren:
- De unitittesten voor de back-end waren nog niet geschreven, waardoor geen enkel item aan de Definition of Done voldeed. Bij een volgende sprint moet ik erop letten dat er niet te veel backlog items worden ingepland. Vanwege de grootte van de applicatie kosten veel items toch relatief veel tijd.
- Het generiek opstellen van de gebeurtenissen voor notificaties en het maken van links voor e-mailberichten was lastig en kostte veel tijd
- Sommige componenten waren al aanwezig in deze of in andere applicaties, waardoor ik ze beter had kunnen overnemen.

Het ontbreken van de unitittesten vormde een probleem wat opgelost diende te worden. Er is tijdens de retrospective besloten dat de unitittests aan het begin van sprint 2 worden geschreven, en dat vervolgens de gebruikelijke cyclus van ontwerpen, ontwikkelen en testen wordt hervat. Hierdoor kon ik minder items op de backlog van de tweede sprint plaatsen, maar was de code van de eerste sprint wel volledig getest.
9 Sprint 2: Afronden notificatie-module / begin workflow-module

Tijdens de tweede sprint lag de nadruk op het testen en het afronden van het notificatie-systeem dat tijdens de eerste sprint is ontwikkeld. Ook heb ik een begin gemaakt met het ontwikkelen van de flexibele workflow-module.

9.1 Plannen van de sprint

Zoals uit paragraaf 8.7 bleek, waren de items van sprint 1 nog niet getest en moesten deze worden teruggesteld op de backlog. Deze items zijn in deze sprint opnieuw in sprint genomen. Hierdoor lijkt het alsof de sprint te vol is gepland. Omdat ik echter wist dat de tests van sprint 1 niet de gehele sprint in beslag zouden nemen heb ik ook een aantal nieuwe items in sprint genomen. Dit zijn:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Omschrijving</th>
<th>Schatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-F001</td>
<td>In- en uit schakelen workflows per faculteit</td>
<td>13 SP</td>
</tr>
<tr>
<td>WFE-F002</td>
<td>Startdatum workflow per faculteit instelbaar</td>
<td>5 SP</td>
</tr>
<tr>
<td>WFE-F003</td>
<td>Einddatum workflow per faculteit instelbaar</td>
<td>5 SP</td>
</tr>
<tr>
<td>WFE-F004</td>
<td>Bij afronden van een workflow een nieuwe workflow opstarten</td>
<td>13 SP</td>
</tr>
</tbody>
</table>

De items uit sprint 1 staan hier niet vermeld. Zie hiervoor paragraaf 8.1

9.2 Testen van de back-end code van Sprint 1

Het testen van AscMe beslaat aan de kant van Vorsen B.V. vrijwel volledig uit unittests. Daarnaast doet de VU aan gebruikersacceptatietesten, die niet door Vorsen worden vastgesteld. Bovendien worden er enkel unittests voor de back-end geschreven. Vorsen geeft aan dat er geen unittests voor de front-end worden geschreven, omdat deze door veranderingen te vaak moeten worden herschreven. Hierdoor kosten deze tests veel tijd en leveren ze relatief weinig op. In de evaluatie van de beroepstaken licht ik toe waarom ik dit achteraf gezien geen goede keuze vindt.

9.2.1 Verschillende soorten unittests

De aanwezige unittests zijn op te splitsen in twee verschillende soorten tests. Als eerste zijn er tests die zich enkel op één component of klasse richten. Deze tests zijn meestal kort en eenvoudig om te schrijven. Er zijn echter ook tests die een hele serie van componenten doorlopen. Dit zijn eigenlijk integratie-tests, maar omdat de front-end hierbij niet actief is kan er niet van volledige integratie-tests worden gesproken, maar van unit-integratietests. Deze tests beslaan een hele cyclus van de hiërarchie die Vorsen in de back-end hanteert: Controller -> Service -> Repository. Hierbij verwerken controllers verzoeken van browsers, handelen services de business logica af en halen repositories gegevens uit de database op. Uiteraard wordt alle opgehaalde data via de service en de controller weer teruggezonden naar de browser.
Doordat deze tests meerdere lagen code testen, zijn deze lastiger om te ontwikkelen. Er moet bij deze tests eerst testdata in de database worden opgeslagen, vervolgens wordt er een bezoek door een webbrowser gesimuleerd, waarna de uitvoer alsmede de data in de database worden gecontroleerd.


Voor het simuleren van het webrequest wordt mockMVC gebruikt. MockMvc is onderdeel van Spring Test, een onderdeel van het Spring Framework wat is bedoeld voor het testen van Spring-applicaties. MockMvc kan onder andere GET, POST, PUT en DELETE requests versturen. Hierbij is de volledige URL op te geven en – in het geval van POST en PUT requests – ook de body van het request.

Vervolgens wordt er gebruik gemaakt van JsonPath, een domeinspecifieke taal (DSL) voor het uitlezen van de JSON-response. Op basis van deze JSON kan worden getest of de back-end de juiste data naar de front-end verstuurde. Onder andere de waardes en het bestaan van objecten kunnen hiermee worden getest. Daarnaast heb ik bij create, update en delete requests ook getest of de juiste waardes in de database zijn opgeslagen. Dit doe ik door met een repository het object opnieuw op te halen en met een assertion de waarde te vergelijken met wat ik als waarde verwacht.

Het schrijven van de unittests voor de notificatie-module kostte uiteindelijk meer tijd dan verwacht. Dit kwam ten eerste doordat het testen van sommige gemaakte delen lastig was, en ten tweede doordat er tijdens het maken van de unittests verschillende bugs in de notificatie-module naar voren kwamen. Ik licht beide problemen hieronder verder toe.

Het lastigste om te testen waren de gemaakte wijzigingen aan de EventBus. De EventBus (zie sprint 1) was een bestaand onderdeel van de applicatie. Er waren echter nog geen unittests voor dit onderdeel geschreven. Om de nieuwe functionaliteit van de notificatie-module te testen was het noodzakelijk om unittests voor de volledige functionaliteit van de EventBus te schrijven.
Bij deze tests moeten er events naar de eventBus worden gestuurd en moet vervolgens worden gekeken of deze correct door de EventListeners worden uitgevoerd.

Vanwege de manier waarop de EventBus werkt (Spring doet *package scanning* naar klassen die *event-listener zijn*) moesten er speciale “Test” events worden aangemaakt, die anders waren dan de gewone events. Anders ging de EventBus ook reageren op events die andere onderdelen uit de bestaande applicatie betroffen, zoals het importeren van gegevens uit het (test-)studentensysteem van de VU. Dit was niet toegestaan, omdat het tot oncontroleerbare situaties tijdens het uitvoeren van de tests leidt.

Bovendien was het nodig om JMockit, een *mocking* framework, te gebruiken. Zo’n framework is bedoeld om nep-instanties van klassen aan te maken, maar biedt ook functies voor het stellen van verwachtingen tijdens het uitvoeren van een test. Met zo’n verwachting kun je eisen dat een bepaalde methode een *x* aantal keer wordt aangeroepen. Gebeurt dit niet of juist te vaak, dan slaagt de test niet.

Dit lijkt niet lastig, maar wat wel lastig is, is wanneer de test aangeeft dat de methode niet wordt aangeroepen terwijl dit wel gebeurt. Dit gebeurde bij het testen van de EventBus, en bleek te komen omdat de methode niet met dezelfde referentie naar een object wordt aangeroepen. Daardoor denkt de test dat de methode niet op de juiste manier wordt aangeroepen en faalt deze. Om dit probleem op te lossen was het nodig om de *equals* en *hashCode* methodes van een object te implementeren. Hiermee vertel je Java wanneer twee instanties van objecten aan elkaar gelijk zijn. Zodra dit was gebeurd, slaagde de unitest en werd de EventBus succesvol getest.

### 9.2.2 Bugs in notificatie-module

Tijdens het testen van de notificatie-module kwamen er diverse bugs naar voren. Deze varieerden van kleinigheden tot serieuze bugs. In deze paragraaf worden enkel de serieuze bugs toegelicht.

Als eerste probleem speelde dat de e-mail die bij het afkeuren van een individuele wijziging werd verstuurd, niet over de goede informatie beschikte. Het verkrijgen van informatie over de gemaakte wijziging, nodig om in de e-mail te plaatsen, bleek lastig. Dit kwam omdat wijzigingen betrekking kunnen hebben op verschillende soorten objecten (faculteiten, additionele velden, opleidingen etc). Elk van deze klassen heeft verschillende methoden waar de data in staat. Door te testen of het betreffende object van een bepaalde klasse is (*instanceof*), kon de vereiste data alsnog op een veilige en betrouwbare manier worden opgehaald.

Ook bleek tijdens het testen dat het klikken op links in een e-mail niet goed werkt. Bij het klikken op een e-mail die vanaf de testserver werd verstuurd, bleef de gebruiker hangen op een pagina in de back-end, waarop geen informatie zichtbaar was. De oplossing hiervoor klinkt vrij simpel: Stuur de gebruiker door naar een relatieve link in plaats van naar een volledige link. Een voorbeeld: Doorsturen naar *http://ascme.nl/workflow/1* werkt niet, maar doorsturen naar *workflow/1* werkt wel. Toch was het hiermee nog niet klaar, want hoewel het doorsturen nu wel werkte, kwam je als gebruiker altijd op het startscherm van de applicatie en niet op het relevante onderdeel (zoals een workflow of een faculteit) terecht. Omdat het maken van de tests en het oplossen van de bugs al langer dan gepland hadden geduurd, heb ik in overleg met Jeroen besloten om dit in een latere sprint aan te pakken.
9.3 Verwerken code review notificatie-module

Nadat ik de unit tests had voltooid, is er door Jeroen een code review over alle door mij gemaakte code uit Sprint 1 uitgevoerd. Uit de code review bleek dat er relatief weinig verbeterpunten aan de code nodig waren. De meeste verbeterpunten waren kleine inconsistenties in de code en waren snel verholpen. Er was eigenlijk maar één groter verbeterpunt, en dat was dat de gemaakte event-interface voor notificaties niet geheel overzichtelijk was. Het voorstel van de reviewer was om met Java-annotaties aan te geven hoe notificaties bij events moeten worden verstuurd. Dit is als item op de backlog voor sprint 3 geplaatst, omdat er vanwege de te maken unit tests weinig tijd was om nog meer verbeteringen uit sprint 1 door te voeren.
9.4 Klassendiagram instellen workflows per faculteit

Naast het testen van de notificaties, heb ik ook nieuwe functionaliteit toegevoegd. Dit omvatte het instelbaar maken van enkele workflow-instellingen per faculteit. Dit resulteerde in een aantal nieuwe klassen. Daarom heb ik een klassendiagram van dit onderdeel gemaakt. Dit diagram toont hoe de faculteit en de workflow met elkaar worden verbonden door middel van de koppelingsklasse WorkflowSettings. Bij het uitvoeren heb ik de klasse uiteindelijk FacultyWorkflow genoemd, omdat dit binnen de code conventies van het bedrijf vereist was.

Figuur 16: Klassendiagram instellen workflows per faculteit

9.5 Ontwerpen grafische userinterface

Voor het instellen van de workflows per faculteit heb ik een scherm ontworpen waarop alle instellingen zijn in te stellen. Dit scherm bestaat voor de helft uit basisinstellingen en voor de andere helft uit notificatie-instellingen. Hoewel de notificaties voor workflows nog niet in deze sprint stonden, heb ik ze wel bij dit ontwerp betrokken, omdat ik al wist dat ze op dit scherm moesten komen. Hierdoor hoefde ik maar één keer een GUI-ontwerp te maken.

Figuur 17: GUI-ontwerp instellen workflow per faculteit
9.6 Database-diagram workflow instellingen per faculteit

Voor het instellen van de workflows per faculteit waren enkele aanpassingen in de database nodig. Er moest een tabel komen waarin de workflow instellingen aan een faculteit en workflow konden worden gekoppeld. Deze koppeltabel heb ik als volgt ontworpen:

Duidelijk is dat de start- en einddatum plus de getriggerde workflow worden opgeslagen bij de instellingen per faculteit. De workflow bevat daarnaast nog een hoop basisinstellingen die voor alle faculteiten gelden.

9.7 Implementeren Workflow-instellingen per faculteit

De functionaliteiten van het instellen van workflows per faculteit zijn in Sprint 2 geïmplementeerd, maar waren aan het eind van de sprint nog niet afgerond. De reden dat dit nog niet klaar was, was omdat er in deze sprint weinig tijd beschikbaar was vanwege de nog te maken unittests uit sprint 1.

Het maken van beheerpagina’s en standaard-vensters verliep inmiddels behoorlijk snel, omdat ik gewend was aan de structuur van de applicatie. Het lastigste onderdeel was om in de database aangebrachte samengestelde sleutels werkend te krijgen in combinatie met Spring. Het blijkt dat bij het Spring Framework een aparte Java-klasse nodig is die de samengestelde sleutel definiert. Deze kan bij de daadwerkelijke entiteit worden opgegeven om als samengestelde sleutel te dienen. Dat ziet er ongeveer zo uit:

Het komt erop neer dat de klasse waarin de samengestelde sleutel staat precies de namen van de objecten waarnaar verwezen wordt moet hebben, maar dat deze als de waarde van het database-id (Long) moeten worden opgeslagen. Bovendien moet de klasse van de samengestelde sleutel een implementatie van de methoden equals en hashCode hebben, en moet deze klasse serializable zijn. De daadwerkelijke entiteit moet ook serializable zijn en moet bovendien een idClass annotatie bevatten, zodat Spring / Hibernate weten waar de samengestelde sleutel staat opgeslagen. Een hoop technische
termen, maar wel belangrijk om een datamodel goed te implementeren.

Later zou blijken dat dit niet binnen de code conventies van het bedrijf hoorde. Uiteindelijk was het dus niet nodig geweest om een samengestelde primaire sleutel te definiëren. Deze is toen vervangen voor een primaire sleutel op de kolom `id`, en een unieke sleutel op de kolommen `faculty_id` en `workflow_id`.

9.8 Sprint Demo

De tweede sprint eindigde op woensdag 22 maart 2017 en was vanwege bedrijfsevenementen twee werkdagen korter. In een presentatie waarbij Jeroen van Schagen en Steven Losekoot aanwezig waren, heb ik het uitgevoerde werk getoond. Ditmaal heb ik de ontwerpen voor de nieuwe schermen voor het instellen van de workflows gepresenteerd en in een korte demo aan de aanwezigen laten zien. Hierbij kwam de opmerking wat er gebeurde wanneer een ingeschakelde workflow werd uitgeschakeld / verwijderd. Er konden dan namelijk gegevens van degenen die de workflow nog niet hebben afgerond verloren gaan. Er is daarom besloten dat het workflow-scherm standaard alle workflows moet tonen, en dat het uitschakelen van een workflow niet betekent dat deze uit de `faculty_workflow` tabel uit de database wordt gehaald.

Bovendien ontstond de vraag of de huidige implementatie van de notificatie-links wel veilig was. Het is immers mogelijk om met een link vanuit een e-mail direct in te loggen op de applicatie, dus wanneer een link van een beheerder uitlekt kan een kwaadwillende direct beheerdersrechten in de applicatie krijgen. Er zal in de derde sprint door mij of door Jeroen met de VU moeten worden overlegd of zij dit risico erg vinden. Indien het aangepast moet worden, moet worden onderzocht of er een mogelijkheid is om gebruikers na het klikken op een link eerst te laten inloggen en daarna naar de juiste pagina door te sturen.

9.9 Sprint Retrospective

Na de Sprint demo volgde, zoals gebruikelijk, de Sprint Retrospective. Tijdens de retrospective kwam het volgende naar voren:

**Taken die goed gingen:**
- Verwerken code review van notificaties
- Ontwerpen nieuwe functionaliteiten voor instellen workflows per faculteit
- Deploren van code op eigen testserver

**Taken die beter konden:**
- Workflow-taken waren nog niet afgerond, deels vanwege testen en deels vanwege kortere sprint
10 Sprint 3: Workflows configureerbaar

De derde sprint bestond vrijwel volledig uit het werken aan de flexibele workflowmodule. In het begin zijn echter nog een aantal openstaande bugs in de notificatie-module opgelost. Het werk aan de workflow-module richt zich in eerste instantie op het instelbaar maken van workflows per faculteit, omdat dit als basis dient voor de rest van de te ontwikkelen workflow-module.

10.1 Wijzigingen aan de requirements

Tijdens deze sprint zijn er slechts enkele requirements gewijzigd. Alle overige requirements voor de uitgevoerde onderdelen waren al in de oriëntatie-sprint vastgelegd. De opgetreden wijzigingen in de requirements worden in deze paragraaf kort toegelicht.

De in de tweede sprint gemaakte functionaliteiten voor het inschakelen van workflows per faculteit waren aan het eind van de tweede sprint nog niet functioneel. Bij het functioneel maken bleek dat het requirement "Automatisch afsluiten op einddatum" helemaal niet nodig was, omdat bij het tonen van de workflows standaard wordt gekeken of deze op de huidige datum geldig zijn Hierdoor kwam requirement WFE-F007 te vervallen.

Daarnaast moest er bij het instellen van de workflows zowel een workflow bij het goedkeuren van de huidige workflow als bij het afkeuren van de huidige workflow kunnen worden gestart. Dit requirement komt voort uit een door de opdrachtgever gestelde vraag tijdens de sprint-demo van sprint 2 en is ondergebracht in requirement WFE-F005.

10.2 Plannen van de sprint

Tijdens deze sprint zijn de workflow-instellingen per faculteit, die aan het eind van sprint 2 nog niet waren afgerond, in sprint 3 opgenomen. Bovendien waren de handmatige notificaties nog niet in de eerste sprint opgenomen en konden deze nu op de backlog worden geplaatst.

De backlog van de derde sprint werd als volgt:

Opnieuw geplaatste items (in progress)

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Omschrijving</th>
<th>Schatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-005</td>
<td>Taal e-mailnotificaties instelbaar (had bug)</td>
<td>2 SP</td>
</tr>
<tr>
<td>WFE-F001</td>
<td>In- en uit schakelen workflows per faculteit</td>
<td>8 SP</td>
</tr>
<tr>
<td>WFE-F002</td>
<td>Startdatum workflow per faculteit instelbaar</td>
<td>5 SP</td>
</tr>
<tr>
<td>WFE-F003</td>
<td>Einddatum workflow per faculteit instelbaar</td>
<td>5 SP</td>
</tr>
<tr>
<td>WFE-F004</td>
<td>Bij afronden van een workflow een nieuwe workflow opstarten</td>
<td>8 SP</td>
</tr>
</tbody>
</table>

Nieuw toegevoegde items:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Omschrijving</th>
<th>Schatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Code review)</td>
<td>Verbeteren events voor notificaties</td>
<td>5 SP</td>
</tr>
<tr>
<td>N-013</td>
<td>Handmatige notificaties versturen</td>
<td>13 SP</td>
</tr>
<tr>
<td>WFE-F007</td>
<td>Bij afkeuren van een workflow een nieuwe workflow opstarten</td>
<td>8 SP</td>
</tr>
<tr>
<td>N-016</td>
<td>Versturen notificaties bij workflow-gebeurtenissen</td>
<td>13 SP</td>
</tr>
</tbody>
</table>
10.3 Klassendiagram en gewijzigd datamodel

Tijdens deze sprint zijn er met name in het datamodel wijzigingen opgetreden. Deze wijzigingen waren nodig om de workflows beter instelbaar te kunnen maken, en om notificaties over workflow-gebeurtenissen te kunnen versturen.

In de oude situatie waren workflows aan een status van een object gekoppeld. Dit was niet gewenst, want hierdoor kon er slechts één workflow tegelijk actief zijn. Daarom was het noodzakelijk om de koppeling tussen de status van een object en een workflow te verbreken. Workflows worden voortaan ingeschakeld op datum of na het afronden van een andere workflow. Hierdoor konden er kolommen uit de workflow-tabel uit de database worden verwijderd.

Voor het implementeren van meldingen bij de start- en einddatum van workflows heb ik besloten om twee nieuwe tabellen in de database te maken, zodat notificatie-instellingen per workflow opgeslagen en gepland kunnen worden. Deze tabellen waren nodig omdat het versturen van een geplande notificatie eenvoudig moet kunnen verlopen.

Notificaties worden gekoppeld aan de workflow-instellingen van de faculteit, zodat verschillende faculteiten verschillende notificaties kunnen instellen. Het koppelen gebeurt in de eerste tabel (faculty_workflow_notification). Vervolgens wordt bij het opslaan van de workflow-instellingen uitgerekend wat de daadwerkelijke datum voor het verzenden van de notificatie is, en deze wordt in de tweede tabel (scheduled_workflow_notification) opgeslagen. Indien de start- of einddatum van de workflow wordt aangepast, worden alle opgeslagen notificaties van een nieuwe verzenddatum voorzien.

Figuur 20: Klassendiagram en gewijzigd datamodel
Dit is ook zichtbaar in het hierboven getoonde klasdiagram. De notificatie-instellingen zijn gekoppeld aan de workflow-instellingen per faculteit (FacultyWorkflow) door middel van een verwijzing naar de id. Bij de notificatie-instellingen wordt niet de daadwerkelijke datum opgeslagen, maar het type gebeurtenis (start- en einddatum workflow, afronden workflow en afkeuren workflow) en het aantal dagen voor of na de gebeurtenis. De daadwerkelijke datum voor de notificatie wordt vervolgens berekend en in een aparte klasse (ScheduledNotification) opgeslagen. Op deze manier kunnen in de toekomst ook andere notificaties worden gepland, waarmee het notificatiesysteem flexibeler en veelzijdiger te gebruiken is.

10.4 Implementeren versturen handmatige notificaties

Het onderdeel "handmatig versturen van notificaties met behulp van een notificatie-sjabloon" stelt gebruikers in staat om een e-mailbericht naar een vooraf gedefinieerde groep gebruikers te sturen. De nieuwgebouwde functionaliteit vervangt de voorziening waarbij een gebruiker de e-mailadressen van een groep gebruikers op een klem bord (voor het kopieren en plakken van teksten) kreeg getoond. Het bouwen van dit onderdeel verliep zonder grote problemen. Bij de back-end kon ik gebruik maken van de bestaande code voor het ophalen van de doelgroepen, en het versturen van e-mails was met de in Sprint 1 gebouwde notificatie-module al mogelijk.
10.5 Verbeteringen notificatie-events (Code review sprint 1)

In de code review die over sprint 1 was gehouden (nadat in sprint 2 de unitests waren voltooid) was als verbeterpunt naar voren gekomen dat de notificatie-events voor automatische notificaties niet geheel overzichtelijk waren. Ik heb hier samen met Jeroen naar gekeken, en we zijn tot de conclusie gekomen dat dit onderdeel het best in tweeën gesplitst kan worden.

Als eerste heb je de events, welke door de EntityEventBus worden verwerkt. Hierin zit de data die vanuit de applicatielogica vereist is (zoals een reactie, een statusovergang of een afgekeurde wijziging)

Aan de andere kant heb je de code voor het versturen van de notificaties. Deze bevat functies voor het filteren van de betrokkenen, het vullen van de variabele velden van het notificatie-sjabloon en het controleren van preconditions. Daarnaast bevatte de code een aantal statische velden, zoals de naam van de gebeurtenis en de ondersteunde variabele velden.

Ik heb dit vervolgens opgesplitst. De statische velden heb ik via de Java-annotatie @NotificationEvent op de events geplaatst. Daardoor staat deze informatie op de klasse waar het betrekking op heeft. Dit ziet er zo uit:

```java
@NotificationEvent{
    name = "COMMENT_REPLIED",
    displayNames = [@DisplayName(language = EN, value = "Reply to a comment"),
                   @DisplayName(language = NL, value = "Gereageerd op een reactie")],
    supportedTags = [RECIPIENT_NAME, USER_NAME, OBJECT_NAME, DYNAMIC_LINK, REPLY])
public class CommentRepliedEvent implements Serializable, Notifyable ... 
```

Figuur 22: Gebruik van Java-annotaties voor de statische gegevens van notificatie-events

De dynamische methodes heb ik in een aparte klasse gezet, die overerft van de interface NotificationHandler. Deze interface biedt methodes voor het controleren van de preconditions, het ophalen van de ontvangers, het vullen van de variabele velden, en het ophalen van een entiteit voor een check of notificaties door de faculteit worden ondersteund.

Alle events implementeren nóg een interface, genaamd Notifyable. Deze interface heeft een methode om de NotificationHandler te specificeren. Hiermee kan de klasse die overerft van NotificationHandler worden opgehaald, en kan data vanuit het event (zoals een reactie) worden meegegeven aan de handler.

Vervolgens heb ik een EntityEventListener gemaakt welke alle events uit de EventBus van het type Notifyable afvangt. Deze listener kan dus alle notificaties verwerken. Het haalt hierbij de NotificationHandler op, en gebruikt de waarde “name” uit de @NotificationEvent annotatie om het template op te halen. Vervolgens kan de bestaande NotificationScheduler de notificatie versturen. Op deze manier is het eenvoudig om later nieuwe events toe te voegen, wat handig is voor het uitbreiden van de applicatie.
Om dit alles inzichtelijk te maken, heb ik tevens onderstaand klassendiagram gemaakt:

![Klassendiagram vernieuwde code voor notificatie-events](image)

**Figuur 23:** Klassendiagram vernieuwde code voor notificatie-events. Opmerking: Alle andere klassen zijn i.v.m. de leesbaarheid van het diagram als attributen opgenomen en worden verder niet getoond.

### 10.6 Hervatten van het instelbaar maken van workflows

Na het werk aan de notificatie-module ben ik verder gegaan met het instelbaar maken van workflows. In de tweede sprint was de basis hiervoor al gelegd, maar er werd met de ingevoerde gegevens nog niets gedaan. In de derde sprint heb ik een werkend component ontwikkeld waarbij workflows per faculteit kunnen worden ingesteld.

Het ontwikkelen van dit component bleek moeilijker dan gedacht. Hoewel het instelbaar maken van de workflows goed te doen was, lagen de problemen tijdens deze sprint in het tonen van de ingeschakelde workflows op het dashboard (het beginscherm van de applicatie). Dit kwam vanwege performance-problemen.

Bij het openen van de applicatie werd namelijk steeds gecontroleerd welke workflows voor de gebruiker van toepassing waren. Dit kostte in de oude situatie al veel rekenkracht, maar met de nieuwe instellingen per faculteit liet de benodigde tijd om het dashboard te laden verder op. Bij het hebben van 20 actieve workflows (dit is voor een vakcoördinator zeker mogelijk) vanuit slechts 1 faculteit bedroeg de berekentijd al 2 seconden, en daarbij werd er niet eens goed rekening gehouden met de groepering van workflows (wat betekent dat er nog meer data moet worden opgehaald) en uitzonderingsgevallen. Tel hier de tijd
van het laden van alle andere gegevens en HTML-bestanden bij op en de applicatie voelt erg traag aan. Gezien de werking van de bestaande code zou de benodigde laadtijd nog verder oplopen indien er meer faculteiten workflows gingen inschakelen. Aangezien er 15 faculteiten zijn is ligt dit voor de hand.


Als oplossing heb ik gekozen om het bepalen van de ingeschakelde workflows op de achtergrond te laten uitvoeren, en de resultaten van de bepaling in een cache-tabel in de database op te slaan. Op deze manier kan de applicatie zeer snel laden - er is enkel een enkele query op de database met ingeschakelde workflows nodig.


Om dit te doen heb ik een achtergrondtaak ontwikkeld die:
- Regelmatig (elke nacht) naloopt welke workflows er voor iedereen moeten worden ingeschakeld, en vervolgens workflows in- of uitschakelt waar dit nodig is.
- Bij het wijzigen van instellingen van een workflow, of andere wijzigingen die tot het gevolg kunnen hebben dat een workflow opent of sluit, elke workflow waar de wijzigingen betrekking op hebben controleert en deze waar nodig voor gebruikers in- of uitschakelt.
Door het ontwikkelteam van AscMe werd als eis gesteld dat voor het uitvoeren van achtergrondtaken van Spring Batch gebruik werd gemaakt. Spring Batch is onderdeel van het Spring-framework en biedt functies voor het uitvoeren van achtergrondprocessen (ook wel batchtaken genoemd). Hiervoor biedt het ondersteuning om processen (Jobs) in verschillende delen (Tasklets) op te breken, en om via transactiebeheer alle wijzigingen pas naar de database weg te schrijven indien de gehele taak succesvol is uitgevoerd. Indien een taak mislukt dan wordt er dus niets aangepast en is er geen beschadigde of onvolledige data in de database aanwezig. Dit had tot het gevolg dat ik eerst moest uitzoeken hoe dit framework werkte, waardoor het ontwikkelen van de batchtaak extra tijd kostte.

Bij het maken van de achtergrondtaak liep ik opnieuw tegen performance-problemen aan. Ik had bij het ontwikkelen in eerste instantie zoveel mogelijk code van het laden van de workflows op dashboard hergebruikt. Dit zorgde er echter voor, dat het uitvoeren van de achtergrondtaak bij enkele ingeschakelde workflows al meer dan een half uur duurde. Als deze taak ook nog eens bij een wijziging in een workflow werd gestart, dan kreeg de server het erg zwaar…

De grote boosdoener was de database. Deze gebruikte tijdens het uitvoeren van de batchtaak bijna alle processorcapaciteit van de server. Dit kwam omdat er gebruik werd gemaakt van de bestaande code voor het tonen van het dashboard. Hierbij werden er voor iedere gebruiker vijf hoofdquery’s naar de database verstuurd, waarna er voor elk object waarmee de gebruiker een relatie had (faculteit, opleiding, studie etc.) nog een query naar de database werd uitgevoerd om deze relatie op te halen. Met ongeveer 3500 actieve gebruikers (3499) en 89750 relaties (een gemiddelde van 25,65 relaties per persoon) liep dit aantal query’s naar schatting op tot 448875 individuele query’s (3500 gebruikers * 5 hoofdquery’s * 25,65 relatiequery’s). Dit is absurd veel en daarmee op zeker de oorzaak van de enorme vertraging.

Om dit probleem op te lossen heb ik besloten om aan het begin van de batchtaak eerst alle benodigde data (faculteiten, opleidingen, vakken, minoren en relaties) op te halen. Vervolgens werd voor elke gebruiker op basis van deze data bepaald welke workflows moesten worden ingeschakeld. Op deze manier werden er veel minder query’s naar de database gestuurd. Een database presteert dus kennelijk beter indien er één query naar wordt verstuurd die 89750 objecten teruggeeft, dan dat er 89750 query’s naartoe worden verstuurd die steeds 1 object teruggeven.

Een gedeelte van deze achtergrondtaak staat in de codefragmenten hieronder.
// Obtain all required data at once. This saves a lot of CPU power vs querying for each person.
Map<String, List<?>> baseEntities = new HashMap<>();
baseEntities.put(Faculty.class.toString(), entityService.findAll(Faculty.class));
baseEntities.put(Study.class.toString(), entityService.findAll(Study.class));
baseEntities.put(Module.class.toString(), entityService.findAll(Module.class));
baseEntities.put(ModuleGroup.class.toString(), entityService.findAll(ModuleGroup.class));
final List<ExecutedWorkflow> executedWorkflows = executedWorkflowRepository.findAll();
final List<Relation> relations = relationRepository.findAll();
final List<RelationType> relationTypes = relationTypeRepository.findAll();
final List<Workflow> workflows = workflowRepository.findAllByDisabledIsFalse();

Figur 26: Ophalen van alle benodigde database-objecten voordat het werk begint

// Obtain all relations for the current person.
Set<Relation> relationsForPerson = relations
    .stream()
    .filter(r -> r.getTarget() != null &&
        r.getTarget().getRawClass() != null &&
        r.getTarget().getRawClass() == Person.class
        && isAcademicYearAware(r)
        && isSupported(r)
        && academicYearMatches(academicYearId, baseEntities, r))
    .collect(Collectors.toList());

final List<RelationType> relationTypesForPerson = relationTypes
    .stream()
    .filter(relationType -> relationsForPerson.stream().anyMatch(relation ->
        relation.getRoleId().equals(relationType.getId())))
    .collect(Collectors.toSet());

// Obtain all applicable workflows for the person using the relation types and the workflow relations.
Set<Workflow> workflowsForPerson = workflows
    .stream()
    .filter(workflow -> relationTypesForPerson
        .stream()
        .anyMatch(relationType -> (workflow.getRelations()
            .stream()
            .anyMatch(workflowRelation -> workflowRelation.getRoleId().equals(relationType.getId())))))
    .collect(Collectors.toSet());

// Pair the workflows with their relations to allow creating new ExecutedWorkflow instances for
// the right entity type.
Map<Long, Set<Relation>> workflowsWithRelationsForPerson = new HashMap<>();
for (Workflow workflow : workflowsForPerson) {
    Set<Long> relationTypeIds = relationTypesForPerson
        .stream()
        .anyMatch(relationType -> (workflow.getRelations()
            .stream()
            .anyMatch(workflowRelation -> workflowRelation.getRoleId().equals(relationType.getId()))))
        .distinct()
        .map(RelationType::getId).collect(Collectors.toSet());
}

Figur 27: Filteren van de data. Op basis van de WorkflowsForPerson konden workflows worden
ingeschakeld, dit was eenvoudig door het opslaan van een ExecutedWorkflow object.
De nieuwgebouwde achtergrondbaan was veel sneller. Ik heb het verschil in performance tussen de oude en de nieuwe code gemeten. Hierbij zijn met zowel de oude als de nieuwe code drie runs op de testomgeving uitgevoerd. Er werden hierbij steeds zes workflows ingeschakeld. Bij alle runs werd dezelfde serverhardware gebruikt en werd de database van tevoren via een sql-dump teruggezet, zodat deze bij alle runs identiek was. Voor het starten van het achtergrondproces werd steeds gekeken of de CPU-belasting van de server tegen de 0% lag, om te voorkomen dat er andere processen voor extra vertraging konden zorgen.

Uit de performance-test kwam naar voren dat de oude code er gemiddeld 29 minuten en 30 seconden over deed om de workflows in te schakelen. De nieuwe code deed over ditzelfde proces gemiddeld precies één minuut. Dit was dus een performance verbetering van bijna 30x.

![Diagram](image)

**Figuur 28: Performance-verbetering na herschrijven code voor importeren workflows**

Bovendien viel het op dat de nieuwe code met een identieke database veel minder workflows inschakelde (365 i.p.v. 772). Dit komt omdat er inmiddels meer controles zijn toegevoegd die nagaan of een workflow wel van toepassing is. Deze controles zijn toegevoegd aan de hand van de uitkomsten van de tests en worden in de paragraaf “Testen” verder toegelicht.
10.7 Testen

10.7.1 Handmatige notificaties

Handmatige notificaties konden worden getest met eenvoudige unittests voor de controller en de service. Deze unittests gingen na of de juiste doelgroep werd geselecteerd en of een e-mail met de juiste tekst werd aangemaakt. Hiervoor heb ik voor elke doelgroep die beschikbaar was een unittest gemaakt. De testgevallen die hieruit voortkwamen waren als volgt:

- Versturen naar personen met een rol bij een specifieke studie van een faculteit
- Versturen naar personen met een rol bij een specifiek vak van een faculteit
- Versturen naar alle personen met een rol bij een vak van een specifieke faculteit
- Versturen naar de eigenaar van een bepaald vak van een opleiding
- Versturen naar personen met een rol bij een vak uit een bepaalde minor

In elke unittest wordt een testgebruiker aangemaakt, en worden een faculteit plus een vak / opleiding aangemaakt. Eerst wordt de send() methode aangeroepen zonder dat de relatie gekoppeld is, nu zou er geen e-mail verstuurd mogen worden. Vervolgens wordt de persoon op de juiste manier gekoppeld aan het vak of aan de opleiding. De mail moet nu wel worden verzonden en bevat de juiste naam van de persoon en een link naar het dashboard. Door hierop te testen worden alle mogelijke notificatie tags gelijk getest.

10.7.2 workflow-instellingen per faculteit

Om de nieuwe workflow-instellingen te testen was het allereerst vereist dat de bestaande unittests voor het tonen van de workflows op het dashboard slaagden. Deze verzekeren ervan dat de voor een gebruiker ingeschakelde workflows correct werden getoond, maar testten niet de specifieke onderdelen voor de faculteiten.

Voor het instellen van de start- en einddatum heb ik twee tests opgesteld die testen of een workflow met een startdatum van morgen, en een workflow met een einddatum van gisteren, niet wordt ingeschakeld. Uiteraard heb ik ook een derde test opgesteld die met geldige data test, de workflow moet dan worden ingeschakeld.

Verder heb ik ook tests opgesteld voor het ophalen en opslaan van de instellingen per faculteit. Een aantal scenario's worden hierbij getest:

- Ophalen ingeschakelde en niet-ingeschakelde workflows
- Opslaan zonder inschakelen (niks opgeslagen in database)
- Opslaan met einddatum voor startdatum (foutmelding)
- Opslaan met geldige gegevens, zowel met als zonder getriggerde workflows, start- en einddatum

Tijdens het testen bleken er nog bugs in de code voor het inschakelen van de workflows te zitten, die aan de hand van de testresultaten verholpen konden worden. Daarnaast heb ik nieuwe tests aangemaakt voor het importeren van de workflows, om te kijken of de juiste workflows werden geïmporteerd.
10.8 Sprint Demo

De derde sprint eindigde op woensdag 5 april 2017. Bij de demo-presentatie waren Jeroen van Schagen en Marcel Noordzij aanwezig. Steven Losekoot was verhinderd. Tijdens de presentatie heb ik het gemaakte werk getoond en zijn er een aantal (verbeter)punten naar voren gekomen. Het belangrijkste punt was dat de nadruk in de komende sprint moest komen te liggen op het maken van dynamische workflow-schermen, omdat het hoofddoel van de opdracht anders in gevaar kwam. Daarom is besloten om tijdens de vierde sprint geen verbeteringen of bug-fixes meer voor het inschakelen van de workflow-instellingen te doen. Ook de notificatie-instellingen voor workflows, die nog niet effectief waren, konden niet in de volgende sprint worden geïmplementeerd. Sprint 4 werd volledig volgepland met het maken van dynamische workflow-pagina’s.

10.9 Sprint Retrospective

Tijdens de gebruikelijke retrospective kwamen weer een aantal belangrijke (verbeter)punten naar voren. Deze worden kort toegelicht:

Punten die goed gingen:
- Het implementeren en testen van het versturen van handmatige e-mailnotificaties
- Het oplossen van openstaande bugs in de notificatie-module

Verbeterpunten:
- De performance-problemen bij het instelbaar maken van workflows kostten erg veel tijd, waardoor er wederom backlog-items niet afgerond konden worden. Het inschatten van backlog-items blijft erg lastig.
11 Sprint 4: Aanpasbare workflow-pagina’s

In de vierde sprint heb ik ondersteuning voor het zelf aanmaken van workflow-pagina’s toegevoegd. Deze pagina’s, genaamd "aanpasbare workflow-pagina’s", zijn workflow-pagina die de beheerder kan inrichten. Op deze manier kan de beheerder zelf bepalen wat er op een workflow-scherm moet worden getoond. Op deze manier kunnen delen van workflows worden ingericht, zonder dat er een nieuwe versie van de applicatie ontwikkelt hoeft te worden.

11.1 Vaststellen en verifiëren van de requirements

Voor het ondersteunen van aanpasbare workflow-pagina’s waren in de oriëntatie-sprint nog niet alle requirements vastgelegd, omdat de wensen voor dit onderdeel nog niet volledig in kaart waren gebracht. Daarom heb ik voor het opstarten van de vierde sprint de requirements verzameld en laten valideren.

De requirements zijn in kaart gebracht op basis van een bespreking met Jeroen en Steven, aansluitend aan de sprint-demo van sprint 3. De requirements zijn vervolgens door Marcel gevalideerd.

Uit de requirements voor de aanpasbare workflow-pagina kwam naar voren dat er per type workflow (facultair niveau, opleidingsniveau of vakniveau) verschillende beschikbare opties moeten komen. De beschikbare opties die op een workflow-pagina kunnen worden gezet, moeten worden ingebouwd in de applicatie - omdat deze specifiek op database-objecten zijn gericht. De beheerder moet echter vrij zijn om te kiezen welke items er wel en niet op de workflow-pagina worden gezet, en ook de volgorde van de items op de pagina moet instelbaar zijn.

Daarnaast zijn er ook pagina’s met vaste componenten, welke dusdanig specifiek zijn dat ze niet dynamisch kunnen worden gemaakt. Deze componenten moeten in zijn geheel in de workflow kunnen worden geplaatst en zijn slechts beperkt configureerbaar. Voorbeelden van zulke componenten zijn een opleidingsrooster of een planbord voor het inplannen van docenten en lokalen. Hiervandaan komen de termen “ingebouwde workflow-pagina” en “aangepaste workflow-pagina”.

11.2 Plannen van de sprint

Omdat het ondersteunen van aangepaste pagina’s de hoogste prioriteit had, is dit onderdeel in de vierde sprint geplaatst. Hierbij is eerst enkel ingegaan op het samenstellen en het invullen van de pagina, ondanks dat deze nog niet op een nette manier in de workflow kan worden geplaatst. Dit om te voorkomen dat ik wederom backlog-items niet zou kunnen afronden.

De samenstelling van de vierde sprint is hiermee als volgt:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Omschrijving</th>
<th>Schatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-P002</td>
<td>Titel van aangepaste workflow-pagina</td>
<td>5 SP</td>
</tr>
<tr>
<td>WFE-P003</td>
<td>Uitlegtekst op aangepaste workflow-pagina</td>
<td>5 SP</td>
</tr>
<tr>
<td>WFE-P006</td>
<td>Items op aangepaste pagina te selecteren</td>
<td>13 SP</td>
</tr>
<tr>
<td>WFE-P007</td>
<td>Volgorde items aangepaste pagina instelbaar</td>
<td>5 SP</td>
</tr>
<tr>
<td>WFE-P008</td>
<td>Invullen / verwerken van aangepaste pagina</td>
<td>20 SP</td>
</tr>
</tbody>
</table>
11.3 Klassendiagram aangepaste workflow-pagina's

Als eerste heb ik een klassendiagram gemaakt om de aangepaste workflow-pagina's te kunnen ondersteunen. Ik heb hierbij gebruik gemaakt van overerving: Een workflow-pagina bevat namelijk altijd een aantal vaste gegevens (naam, titel, uitlegtekst) en een aantal specifieke onderdelen (items op aangepaste pagina, AngularJS route op ingebouwde pagina). Verder is er een service die de pagina's tijdens een workflow vult met reeds opgeslagen gegevens (retrieve methode) en verwerkt (process methode). Bovendien kan de beheerder de pagina's beheren (create, edit en delete methoden).

![Klassendiagram aangepaste workflow-pagina's](image)

Figuur 29: Klassendiagram aangepaste workflow-pagina's

11.4 Ontwerpen van de gebruikersinterface

Nadat de requirements voor de workflow-pagina waren gevalideerd, heb ik de gebruikersinterface voor het bewerken van aangepaste workflow-pagina's ontworpen. Hiervoor heb ik enkele mock-ups gemaakt. Omdat volgens requirement WFE-B002 het inrichten van workflows in een wizard moest plaatsvinden, heb ik hier bij het maken van de ontwerpen rekening mee gehouden.

Wizards worden vaker in de applicatie gebruikt, waardoor ik dezelfde structuur voor het vormgeven van de wizard heb aangehouden. Op deze manier blijft de gebruikersinterface consistent.
Bovenstaande figuur toont dat het inrichten van de aangepaste workflow-pagina wordt gedaan door middel van het slepen van items naar de rechter tabel. In de linker tabel staan alle invoervelden die op de pagina kunnen worden geplaatst. De beschikbare invoervelden zijn afhankelijk van het type object waarop de workflow wordt uitgevoerd. Dit type was in deze sprint nog niet in te stellen, maar is in sprint 5 toegevoegd.

Aan de rechterkant staat de lijst met items die reeds op de pagina aanwezig zijn. Om aan requirement WFE-P007 te voldoen, kan de volgorde van de items worden aangepast door deze omhoog of omlaag te verslepen.

11.5 Ontwerpen van een nieuw gegevensmodel

Om de wijzigingen aan de workflow-module te implementeren heb ik een nieuw database-ontwerp voor het opslaan van workflow-pagina’s gemaakt.

Het oude gegevensmodel voldeed niet langer, omdat een workflow-pagina aan slechts één workflow kon worden gekoppeld. Dit betekende dat er voor iedere workflow nieuwe pagina’s moesten worden opgeslagen. Dit was niet handig omdat - met het oog op uitbreidbaarheid van de applicatie - workflow-pagina’s aan meerdere workflows moeten kunnen worden toegevoegd. Bovendien is het efficiënter om dezelfde pagina’s maar één keer op te slaan en vervolgens via een koppeltabel aan de betreffende workflows te koppelen.

Daarom heb ik bij het ontwerpen van het nieuwe datamodel besloten om een koppeltabel tussen workflows en workflow-pagina’s te maken. Elke rij in deze tabel geeft aan dat een bepaalde pagina voor
een bepaalde workflow is ingeschakeld, en op welke positie de betreffende pagina in de workflow moet worden getoond. De combinatie workflow, workflow pagina en positie is uniek, waardoor deze een samengestelde primaire sleutel vormt. Bovendien zijn de verwijzing naar de betreffende workflow en de pagina een vreemde sleutel.

De items van aangepaste workflow-pagina's moesten in ieder geval in een aparte tabel komen te staan, omdat een pagina uit meerdere items kan bestaan.

Uit het overleg na de demo van de derde sprint was gebleken dat er momenteel een database tabel genaamd “Aanpasbare velden” aanwezig is. Deze bevatt velden die voor willekeurige objecten in te stellen zijn, en werd in de oude situatie gebruikt voor het tonen van sommige gegevens op tabbladen van faculteiten en opleidingen. De data werd bij het importeren vanuit het studentensysteem van de onderwijsinstelling ingevoerd, deze kon daarna niet worden bewerkt.

Er werd door Jeroen sterk aangeraden om hier gebruik van te maken, en eventueel meer velden aan deze tabel toe te voegen als er nog velden ontbreken. Dit leek mij een goed idee, want deze tabel was generiek opgezet zodat de velden voor alle objecten gebruikt konden worden. Als ik het zelf had ontworpen was ik waarschijnlijk op een vergelijkbaar iets uitgekomen. Door deze keuze heb ik in het database-ontwerp besloten om de items op de workflow-pagina naar deze tabel te laten refereren.

Vervolgens moest er onderscheid worden gemaakt tussen ingebouwde en zelfgemaakte workflow-pagina’s. Er waren meerdere opties om dit in de database te modelleren:
- Alle gegevens over workflow-pagina’s in één tabel opslaan
- Twee aparte tabellen voor de verschillende soorten workflow-pagina’s
- Een hoofdtabel voor gegevens die bij elke workflow-pagina van toepassing zijn, plus nog twee tabellen waarin gegevens voor de specifieke typen pagina’s staan.

Omdat beide workflow-pagina’s over de meeste basisgegevens beschikten en de items van de aangepaste workflow-pagina al apart stonden, heb ik gekozen om alle gegevens over workflow-pagina’s in één tabel op te slaan.

11.6 Realisatie

Nadat de gebruikersinterface en het datamodel waren ontworpen kon het werk aan de implementatie beginnen. Bij het werk aan de dynamische pagina waren er verschillende uitdagingen die elk een aparte aanpak vereisten. De aanpak wordt hieronder per uitdaging toegelicht.

11.6.1 Modelleren van de database in Java

Database-objecten worden in principe altijd in Java gemodelleerd (door middel van ORM), maar het modelleren van de overerving op workflow-pagina’s, het modelleren van de koppeltabel voor workflow-pagina’s en het koppelen van items aan de pagina leverde veel fouten van het ORM-framework Hibernate op. Het bleek dat Hibernate heel gevoelig is bij het opslaan van gegevens in zulke tabellen, en dat de koppelingen precies goed moeten staan voordat dit werkt. Ook is het - vreemd genoeg - nodig om altijd dezelfde set (Java HashSet) mee te geven aan de database. Als je een gewijzigde samenstelling van een pagina wilt opslaan mag dit niet door een nieuwe lijst met items in de pagina te plaatsen, maar moet dit door de set met items leeg te gooi en opnieuw te vullen.
11.6.2 Samenstellen van de pagina

Omdat tijdens deze sprint enkel requirements voor het samenstellen van een enkele pagina en het tonen van de pagina in een workflow in sprint zijn genomen, heb ik een los scherm ontwikkeld voor het samenstellen van de pagina. Op deze manier kon de functionaliteit worden ingebouwd zonder al te veel bestaande onderdelen te hoeven vervangen.

Omdat de items door middel van slepen & neerzetten (drag&drop) moesten kunnen worden neergezet, moest ik eerst op zoek naar een pakket wat dit in de front-end mogelijk maakte. Er werd al drag & drop in de applicatie gebruikt, maar dit zag er niet heel fraai uit en werkte erg stroperig. Dit gebeurde door middel van *angular-drag-and-drop-lists*. Om te kijken of dit beter kon, heb ik een vergelijking gemaakt tussen dit pakket, en tussen Dragula drag & drop.

Angular-drag-and-drop-lists ("Angular directives for sorting nested lists using the HTML5 Drag & Drop API") is een opensource-oplossing voor het inbouwen van drag&drop in AngularJS applicaties. Hiervoor biedt het verschillende configuratie-opties aan. De website oogt niet heel erg fraai en is vooral op technische kenmerken gericht.

Dragula ("Drag and drop so simple it hurts") is een vergelijkbaar pakket, wat al langer bestaat. De bijbehorende website toonde betere voorbeelden en demo’s, waardoor ik snel kon zien welke mogelijkheden dit pakket bood. Bovendien toonden de demo’s dat het pakket vloeiend oogt, waardoor het slepen van items erg intuïtief verloopt.

Hierom heb ik besloten om Dragula drag & drop uit te proberen. Het was vervolgens eenvoudig om dit pakket te configureren voor het slepen en neerzetten van items op de workflow-pagina.

11.6.3 Tonen van gegevens op de pagina

Het tonen van gegevens op een aangepaste pagina was geen eenvoudige klus, omdat de pagina uit willekeurige velden bestaat, en deze velden uit de database afkomstig zijn. Omdat van tevoren niet bekend over welke objecten de pagina gaat en van welk type de ingevoerde waarde is, moest ik hiervoor andere programmeer technieken gebruiken. Dit waren Reflection en native queries. Beide onderdelen worden hieronder toegelicht.

Reflection is een onderdeel van de Java-programmeertaal wat zich richt op het dynamisch aanroepen van klassen, methodes en variabelen. Met Reflection kunnen acties worden uitgevoerd die normaliter niet mogelijk zouden zijn, omdat de *statistische typering* van de Java-programmeertaal dit niet toelaat.

![Figure 31: Schema van de Reflection API uit Java. Afbeelding afkomstig van [7].](image)

Een voorbeeld van een actie die niet zomaar kan, maar enkel met Reflection mogelijk is, is het uitlezen van een veld op een willekeurig object. In geïnterpreteerde programmeertalen (zoals JavaScript) kan dit, omdat er geen controle op aanwezigheid van benaderde velden is. Er wordt enkel *undefined* als resultaat
gegeven als het veld niet aanwezig is. Als het bovenliggende object ook niet aanwezig is, ontstaat een run-time error.

Reflection maakt dit mogelijk door het betreffende veld niet direct uit een klasse op te halen, maar dit via het framework (te benaderen met de op elk Java-object aanwezige methode getClass) te laten verlopen. Hierbij geef je het framework de veldnaam en het betreffende willekeurige object waaruit je de waarde wilt krijgen mee. Reflection probeert vervolgens de waarde uit dit object op te halen. Lukt dit, dan krijg je de uitgelezen waarde als resultaat. Lukt dit niet, dan ontstaat er een run-time error.

Het tweede onderdeel wat ik nodig had, waren "native queries". Hiermee worden directe query's op de database bedoeld, die geen gebruik maakten van het reguliere systeem voor het uitvoeren van query's (Spring Data JPA). Door variabel te maken uit welke databasetabel de data komt, en hier als variabele de databasetabel van het betreffende "aanpasbare veld" neer te zetten, kon een object uit een willekeurige database tabel worden geselecteerd. Hiervoor waren twee hoofdlijnen: Het ophalen van data op basis van entity-referenties (dit zijn het id en het type entity), en op basis van overerving (waarbij een klasse moduleCredits een module id heeft). Voor beide manieren heb ik een aparte query-methode opgesteld.

Vervolgens kon met Reflection de de waarde uit het veld van deze instantie worden uitgevraagd. De vereiste naam van het veld stond ook in de tabel "Aanpasbare velden" opgeslagen.

Helaas was het werk hiermee nog niet voltooid. Er waren namelijk twee manieren waarop gegevens uit de "aanpasbare velden" in objecten werden opgeslagen. Als eerste waren er de objecten waarbij de gegevens in de velden van het object zelf waren opgeslagen. Deze objecten kon ik met de reeds genoemde aanpak verwerken.

Er waren echter ook objecten die hun gegevens niet in de velden zelf opslagen, maar dit in een HashMap met waardes deden. Een HashMap is een verzameling waarin gegevens op basis van een unieke sleutel staan opgeslagen (ook wel bekend als Key-Value pair). Een voorbeeld van een key-value pair is een rij met daarin als sleutel "Naam" en als waarde "Arjan".

Waardesten stonden in objecten die over zo’n HashMap beschikken, moesten op een andere manier worden uitgelezen dan objecten die gegevens in hun eigen velden opslagen. Hierbij was het nodig om te weten wat de naam van de HashMap is, zodat de HashMap zelf via Reflection kan worden uitgelezen. Als de HashMap eenmaal is uitgelezen, kan vervolgens de waarde op de normale manier uit de HashMap worden opgehaald.

Om de naam van de HashMap op te slaan, heb ik gebruik gemaakt van annotaties. Annotaties zijn regels code die je boven een klasse of een methode kunt plaatsen, waarna je ze door middel van Reflection weer kunt uitvragen. Een annotatie kun je argumenten meegeven, welke vervolgens ook weer uit te lezen zijn.

Ik heb in dit geval een annotatie gemaakt die aangeeft wat de naam is van de HashMap met de gegevens. Door deze annotatie, genaamd "CustomFieldValues", met Reflection uit te lezen had ik beschikking over de naam van de HashMap, en na deze ook met Reflection uitgelezen te hebben konden hier de benodigde gegevens uit worden opgehaald. Dit gebeurt door middel van onderstaande code:

```java
CustomFieldValues values = clazz.getAnnotation(CustomFieldValues.class);
Field classField;
if(values != null) {
```
classField = clazz.getDeclaredField(values.storedIn());
classField.setAccessible(true);
return new Pair<>(MAP, classField);
}
else {
classField = clazz.getDeclaredField(field.getName());
classField.setAccessible(true);
return new Pair<>(FIELD, classField);
}

Figuur 32: Dynamisch ophalen van een veld van een klasse. De linker waarde in het Pair is het type waarde, wat in de volgende methode (onderstaande figuur) wordt gebruikt om het veld op te halen.

Object value = null;
if(typeAndField.getKey() == MAP) {
  Object valuesContainer = typeAndField.getValue().get(entity);
  if(valuesContainer instanceof Map) {
    Map valuesMap = (Map) valuesContainer;
    value = valuesMap.get(customField.getName());
    if(value != null && !value.equals(StringUtils.EMPTY) &&
      customField.getValueType() != null) {
      switch (customField.getValueType()) {
        case DATE:
          value = LocalDateTime.ofInstant(Instant.ofEpochMilli(Long.parseLong((String) value)), ZoneId.systemDefault());
          break;
        case BOOLEAN:
          value = Boolean.parseBoolean((String) value);
          break;
        case NUMBER:
          value = Double.parseDouble((String) value);
          break;
        case ELEMENT:
          // Elements and modules are always stored as IDs (which are of type Long).
          value = Long.parseLong((String) value);
          break;
      }
    }
  } else if(typeAndField.getKey() == FIELD) {
    // Fields are always stored in the right format. No date / number parsing is necessary here.
    value = typeAndField.getValue().get(entity);
  }
}
return value;

Figuur 33: Ophalen van een waarde uit een veld. Ondersteunt zowel HashMaps als directe velden.

11.7 Sprint demo

Op 19 april 2017 was de demo-presentatie van de vierde sprint. Tijdens deze presentatie heb ik het gemaakte werk getoond. De aanwezigen waren tevreden, maar hadden enkele opmerkingen over het samenstellen van een aangepaste workflow-pagina.

- Items die reeds op de pagina waren geplaatst mochten niet langer in de lijst met beschikbare items komen te staan
- De pagina heeft betrekking op een bepaald type objecten (faculteit, opleiding etc.) maar dit werd nog niet opgeslagen. Dit moest ook bij de gegevens over de pagina worden opgeslagen

11.8 Sprint retrospective

Aansluitend aan de sprint demo volgde de sprint retrospective. De punten die tijdens deze sprint goed gingen, waren:

- Het ontwerpen en ontwikkelen van de gebruikersinterface voor het samenstellen van de workflow-pagina
- Het ontwikkelen van de functies om het formulier te verwerken. Ondanks dat dit een uitdaging was, ging dit goed: het gehele formulier kon aan het eind van de sprint worden samengesteld en verwerkt.

Er waren echter ook een aantal verbeterpunten:

- Het modelleren van het datamodel in de back-end had te lang geduurd. Bovendien was het van het bedrijf niet nodig geweest om de samengestelde database-sleutels te implementeren. Dit had ik beter kunnen overleggen en had mij een hoop tijd gescheeld.
- Er werd aangegeven dat er nog geen automatische tests voor de workflow-pagina waren geschreven, maar dit had ik bewust niet gedaan. Dit omdat de omliggende functionaliteit voor het samenstellen van de workflow pas in de volgende sprint werd uitgevoerd. Achteraf had ik beter wel de tests kunnen schrijven, omdat fouten in het verwerken van de pagina dan eerder konden worden gevonden.
12 Sprint 5: Wizard voor samenstellen workflow

In de vijfde sprint heb ik een wizard ontwikkeld waarmee een (functioneel) beheerder eenvoudig een workflow kan samenstellen. Deze workflow bestaat zowel uit gewone als uit aangepaste pagina’s (sprint 4). Bovendien beschikt de workflow over alle aanwezige functies uit de oude situaties, zodat de bestaande klant ermee overweg kan.

12.1 Vaststellen requirements

Na afloop van de vierde sprint retrospective heb ik in een bespreking met Steven en Jeroen bepaald welke taken er in de vijfde sprint dienden te worden uitgevoerd. Zij gaven aan dat de nadruk moest liggen op het samenstellen van de workflow, en dat dit op een gebruiksvriendelijke manier moest plaatsvinden.

Omdat de opdrachtsomschrijving aandroeg om zoveel mogelijk processen binnen de applicatie in wizards onder te brengen, en het inrichten van workflows altijd uit een bepaalde set stappen bestaat, heb ik aangeraden om het inrichten van de workflows ook in een wizard te laten plaatsvinden. Dit werd als een goed idee bevonden, en hierdoor konden de requirements voor het samenstellen van de workflow worden uitgevraagd.

Allereerst dienden alle huidige opties van de workflow instelbaar te blijven. Dit resulteerde in de requirements WFE-I001 t/m WFE-I014. Deze requirements omvatten onder andere het instellen van de naam, het type en de betrokkenen van de workflow.

Daarnaast dienden de pagina’s van de workflow op een gebruiksvriendelijke manier in de workflow te worden geplaatst. Dit resulteerde in de requirements WFE-S003 t/m WFE-S006. Hieronder vallen het toevoegen, verwijderen en verplaatsen van pagina’s binnen de workflow.

Opmerking: Bij deze sprint is er geen UML-diagram gemaakt, omdat het systeemontwerp (figuur 26) wat is gemaakt bij het maken van de aangepaste workflow-pagina’s alle klassen bood die nodig waren om workflows in het systeem op te kunnen slaan.
12.2 Plannen van de sprint

Nu de requirements bekend waren kon de backlog voor de vijfde sprint worden gevuld. Deze is als eerste gevuld met de requirements voor het instellen van de basisinstellingen van de workflow. Omdat de basisinstellingen al bestonden, betrof het werk hieraan enkel het ontwerpen en ontwikkelen van een nieuwe gebruikersinterface. Hierdoor kregen deze requirements weinig story points en konden ze allemaal worden ingepland.

Verder heb ik de requirements WFE-S001 t/m WFE-S006 in de backlog geplaatst. Hiermee kunnen de workflow-pagina’s aan de workflow worden toegevoegd en vervolgens worden geconfigureerd.

Ten slotte heb ik de requirements voor het beheren en verbeteren van aangepaste workflow-pagina’s toegevoegd, omdat deze functionaliteit benodigd was om aangepaste workflow-pagina’s te kunnen gebruiken (het was na sprint 4 enkel mogelijk om deze aan te maken, niet te verwijderen of in te zien). Dit betroffen de requirements WFE-P001, WFE-P004 en WFE-P011.

Hoewel na de derde sprint aangegeven was dat na het instellen van de workflow een voorbeeldmodus van de workflow moest volgen (requirements WFE-V001 en WFE-V002), kon dit onderdeel vanwege tijdgebrek niet in deze sprint worden genomen.

Hiermee was de sprint backlog als volgt:

| Requirement | Omschrijving | Schattin
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-P001</td>
<td>Beheerpagina voor aangepaste workflow-pagina’s</td>
<td>8SP</td>
</tr>
<tr>
<td>WFE-P004</td>
<td>Instellen objecttype aangepaste workflow-pagina</td>
<td>5SP</td>
</tr>
<tr>
<td>WFE-P011</td>
<td>Alleen niet-gekozen velden in lijst beschikbare velden</td>
<td>5SP</td>
</tr>
<tr>
<td>WFE-I001</td>
<td>Titel workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I002</td>
<td>Objecttype workflow instelbaar</td>
<td>3SP</td>
</tr>
<tr>
<td>WFE-I003</td>
<td>Betrokkenen workflow instelbaar</td>
<td>5SP</td>
</tr>
<tr>
<td>WFE-I004</td>
<td>Introductietekst workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I005</td>
<td>Afsluitende tekst workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I006</td>
<td>Omschrijving workflow op dashboard instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I007</td>
<td>Instellen of workflow kan worden afgekeurd</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I010</td>
<td>Link naar een externe handleiding over de workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I011</td>
<td>Link naar een externe uitlegvideo over de workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I012</td>
<td>Systeemnaam workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I013</td>
<td>Naam van icoontje workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-I016</td>
<td>Gegroepeerd weergeven van workflow instelbaar</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-S001</td>
<td>Pagina’s toevoegen en verwijderen aan / uit de workflow</td>
<td>8SP</td>
</tr>
<tr>
<td>WFE-S002</td>
<td>Wijzigen volgorde van pagina’s op een workflow</td>
<td>2SP</td>
</tr>
<tr>
<td>WFE-S003</td>
<td>Instellen toegestane weergave-modi ingebouwde workflow-pagina</td>
<td>5SP</td>
</tr>
<tr>
<td>WFE-S004</td>
<td>Instellen standaard weergave-modi ingebouwde workflow-pagina</td>
<td>5SP</td>
</tr>
<tr>
<td>WFE-S005</td>
<td>Aangepaste titel voor pagina in de huidige workflow</td>
<td>5SP</td>
</tr>
<tr>
<td>WFE-S006</td>
<td>Aangepaste uitlegtekst voor pagina in de huidige workflow</td>
<td>5SP</td>
</tr>
</tbody>
</table>
12.3 Ontwerpen van de gebruikersinterface

Voor het instellen van de workflows was een nieuwe gebruikersinterface nodig. Hiervoor heb ik mock-ups gemaakt, zodat het ontwerp vervolgens alleen nog maar gebouwd hoefde te worden.

Aangezien het instellen van de workflows in een wizard moest plaatsvinden, is het ontwerp van de gebruikersinterface daar ook op gebaseerd. De wizard bestaat uit verschillende stappen, welke ik eerst heb vastgesteld.

Hierbij heb ik besloten om twee basisstappen voor het instellen van de workflow aan te maken: De eerste stap bestaat uit het instellen van de basisgegevens en de betrokkenen van de Workflow. Hiervoor heb ik een formulier ontworpen waarop al deze gegevens kunnen ingevoerd.

Hierbij viel het mij op dat de gegevens van de workflow onder te verdelen waren in vier categorieën: Basisinstellingen, Opties, Help en Uitleg en Betrokkenen.

Bij het ontwerpen van deze pagina heb ik de invoervelden in deze vier categorieën verdeeld. Dit maakt de pagina overzichtelijker.

Figuur 34: GUI-ontwerp voor het instellen van de basisgegevens van een workflow
De tweede stap bestaat uit het selecteren van de pagina’s. Hiervoor had ik twee GUI-ontwerpen gemaakt. Het eerste ontwerp toonde een scherm waarbij workflow-pagina’s uit een lijst met beschikbare pagina’s omhoog werden gesleept. Hierna kwamen de pagina’s in een tijdlijn te staan, die de workflow vormde. Het tweede ontwerp bestond uit twee tabellen met pagina’s (beschikbare en gekozen) en leek veel op het ontwerp voor het plaatsen van invoervelden op een aangepaste workflow-pagina (sprint 4).

Beide ontwerpen zijn voorgelegd. Vervolgens is voor het tweede ontwerp gekozen, omdat het volgens Vorsen consistent is met de eerder gebouwde functionaliteit voor het toevoegen van items aan een aangepaste workflow-pagina.
De derde fase voor het inrichten van de workflows bestaat uit het instellen van de pagina’s. Hierbij is er voor elke gekozen pagina een extra stap benodigd. Deze stap biedt de gebruiker de mogelijkheid om de gekozen pagina naar wens in te stellen.

Van elke pagina zijn de titel en de uitleg instelbaar, en van ingebouwde pagina’s is ook de weergavemodus instelbaar (WFE-S001/002). Deze opties zijn terug te vinden in de figuur die hieronder wordt getoond.

![GUI-ontwerp voor het instellen van een gekozen workflow-pagina]

**Figuur 37: GUI-ontwerp voor het instellen van een gekozen workflow-pagina**

Tenslotte diende er een overzichtspagina voor aangepaste workflow-pagina’s te worden ontworpen. Deze pagina was vrijwel identiek aan de overzichtspagina voor het instellen van workflows per faculteit (sprint 3), dus hiervan heb ik het ontwerp grotendeels hergebruikt.

### Aangepaste workflow-pagina’s

<table>
<thead>
<tr>
<th>Titel pagina</th>
<th>Heeft betrekking op</th>
<th>#Items</th>
<th>Acties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interne gegevens</td>
<td>Faculteit</td>
<td>1</td>
<td>Bewerken</td>
</tr>
<tr>
<td>Studiegidssteken</td>
<td>Opleiding</td>
<td>10</td>
<td>Bewerken</td>
</tr>
<tr>
<td>Gegevens over aanmelding</td>
<td>Module</td>
<td>5</td>
<td>Bewerken</td>
</tr>
</tbody>
</table>

**Figuur 38: GUI-ontwerp voor het beheren van aangepaste workflow-pagina’s.**

Met deze serie GUI-ontwerpen is het mogelijk om alle items voor het inrichten van de workflow te kunnen bouwen. Het overzichtsscherm met de lijst met workflows kreeg geen nieuw ontwerp, omdat het huidige ontwerp (lijkend op de lijst met aangepaste workflow-pagina’s) voldeed.
12.4 Realisatie workflow-wizard

Het bouwen van de workflow-wizard had vrijwel volledig betrekking op de front-end van de applicatie. De back-end code voor het opslaan van workflow-gegevens was immers al aanwezig en hoefde enkel te worden aangeroepen. In deze paragraaf staat per stap die benodigd is voor het opstellen van een workflow het proces toegelicht.

12.4.1 Wizard stap 1: Basisgegevens voor de workflow

Het eerste scherm met daarop de basisgegevens van de workflow was relatief het eenvoudigst om te ontwikkelen. Ik had een goed schermontwerp gemaakt wat vrij snel naar bruikbare code kon worden omgezet. De front-end beschikte over uitgebreide directives en componenten om invoervelden op een grafisch mooie manier te plaatsen, waardoor dit snel was gebeurd. Wel heb ik een aantal teksten opnieuw geformuleerd, omdat deze in de oude situatie te technisch of onduidelijk waren.

12.4.2 Wizard stap 2: Pagina’s aan de workflow toevoegen

Het tweede scherm waarop de pagina’s aan de workflow moeten worden toegevoegd leverde iets meer problemen op. Deze problemen lagen met name bij het gebruik van drag & drop. Hoewel ik bij de vorige sprint het Dragula drag & drop pakket had gebruikt, bleek het, bij het toevoegen van de eisen voor het slepen van pagina’s, toch minder flexibel te zijn dan gedacht. Er was met name een grote vertraging tussen het slepen van items en het weghalen uit de lijst met beschikbare pagina’s. Pagina’s mochten namelijk slechts een keer aan de workflow worden toegevoegd en dienden dus uit de lijst met beschikbare pagina’s te verdwijnen. Hiervoor had ik een functie geschreven die dit in JavaScript deed. Hoewel de functie de items direct uit de lijst verwijderde, was deze lijst gekoppeld aan het Dragula pakket voor weergave in de HTML (en niet via reguliere 2-way binding, waarbij wijzigingen als deze direct in de browser verwerkt zouden worden). Het tijdsverschil was telkens 1 a 2 seconden, waarbij de items soms ook versprongen. Dit was niet acceptabel voor de gebruiker, en daarom was ik genoodzaakt om een ander framework voor drag&drop te gebruiken. Dit werd het eerder besproken angular-drag-and-drop-lists. Een nadeel hiervan was dat het slepen er minder vloeiend uit zag. Met het aanpassen van de CSS zag het slepen er echter wél vloeiend uit. De nieuwe CSS heb ik uit een demo op internet overgenomen en in de applicatie gezet.

12.4.3 Wizard stap 3: Pagina’s configureren

De derde stap (of fase, want bij meerdere pagina’s zijn er meerdere “stappen”) was het lastigst te implementeren. Dit lag aan het feit dat er dynamisch stappen aan de wizard moesten worden toegevoegd, verwijderd of verplaatst voor alle in stap 2 gekozen pagina’s. Dit kwam met name door de structuur van wizards in de applicatie AscMe. Deze bestaan uit een buitenste schil (de indicatoren voor de stappen en de “volgende” en “vorige” knoppen, met daarbinnen een view voor de inhoud van de huidige actieve stap. De actieve stap is een apart Angular component en in feite dus een aparte klasse.

Bij het toevoegen van stappen dient dit naar de bovenste laag te worden gecommuniceerd. Hiervoor vond ik al vrij snel een handige manier: Het gebruik van Angular events. Angular events zijn een mechanisme waarbij een component een bepaald sleutelwoord uitzendt naar bovenliggende componenten (parents), of naar onderliggende componenten (children). Het component wat de code ontvangt kan een functie specificeren die wordt uitgevoerd zodra het event binnenkomt.
In mijn geval werd in de buitenste laag van de wizard een functie geplaatst die de stappen toevoegt, wijzigt of verwijderd. In de component van het toevoegen van de pagina’s werd bij iedere actie op het slepen van de pagina’s naar de rechter kolom (gekozen pagina’s) het event verstuurd.

Wat ook lastig was, was het afhandelen van het herladen van een pagina. Normaliter kom je dan terug op hetzelfde scherm waar je was, maar deed je dit in een dynamisch toegevoegde stap, dan ging dat fout. De wizard moet namelijk weten bij welke stap je op dat moment bent, en moet hiervoor ook de gekozen pagina’s uit de vorige stap hebben. Hierbij heb ik geprogrammeerd dat indien de wizard de huidige stap niet kan detecteren, dat deze verder niets doet. Vervolgens roept het component voor het instellen van de pagina’s (wat wel de lijst met pagina’s heeft) een ander event aan wat ervoor zorgt dat de wizard de huidige stap weer kan vinden. Door het op deze manier vertraagd plaatsen van de stappen wordt de pagina wel correct getoond.

12.5 Realisatie beheren van aangepaste workflow-pagina’s

Het beheren van de aangepaste workflow-pagina’s was relatief eenvoudig, omdat ik het scherm voor het bewerken ervan al in de vorige sprint had gemaakt. Het was hierbij vooral een kwestie van het koppelen van dit scherm aan een nieuw te bouwen beheerpagina. Het bouwen van de beheerpagina bracht geen uitdagingen mee, en was dus snel gebeurd. Bij het bewerken van de pagina heb ik enkel nog een veld voor het bewerken van het type pagina toegevoegd (requirement WFE-P004), verder was het compleet.

12.6 Aanpassingen onderdelen vorige sprint a.d.h.v. code review

Aan het eind van de vorige sprint was er een code review gehouden en hieruit waren enkele interessante punten naar voren gekomen. Het eerste grote punt was dat het gebruik van de Java Reflection API voor het verwerken van de aangepaste workflow-pagina’s niet nodig was. Jeroen had namelijk al eerder, voor het bijhouden van digitale audits, een stuk code ontwikkeld wat door middel van het Spring framework waardes in Java-klassen kan plaatsen en ook weer kan ophalen. Dit component was zelfs in staat om te detecteren of de klasse over een (Hash)Map met waardes of veldwaardes beschikte (helaas door middel van een nietszeggende annotatie @SourceChange). Het gebruik hiervan was eenvoudig en zorgde ervoor dat de code voor het verwerken van het workflow-formulier significant korter werd.

Hoewel het gebruik van Reflection niet nodig bleek te zijn, was het wel leerzaam om hiermee te kunnen werken. Met deze tools is het namelijk mogelijk om geavanceerde operaties in Java uit te voeren.

Het tweede punt wat werd opgemerkt, was dat de overerving in de database op onnodig complexe wijze was gemodelleerd. Omdat er namelijk heel weinig velden specifiek voor een ingebouwde of aangepaste workflow-pagina zijn, was het volgens Jeroen beter om één object, genaamd Page, aan te maken en hier elke pagina in op te slaan. Het enige verschil is dat een ingebouwde pagina een waarde voor het veld route heeft, terwijl een aangepaste pagina items (voorheen CustomWorkflowPageItem) kan bevatten. Het hernemen van alles naar Page (in plaats van WorkflowPage) had nog een tweede voordeel: Hiermee wordt het in de toekomst ook mogelijk om pagina’s voor andere doeleinden dan workflows aan te maken. Met het oog op uitbreidbaarheid van de applicatie een goed idee.
Na deze wijzigingen heb ik een schema gemaakt waarin de werking van het workflow-systeem staat afgebeeld. Dit staat hieronder.

**Figuur 39: Globale werking van het workflow-systeem in AscMe, na de refactoring van pagina’s.** Een ingebouwde pagina wordt direct door de browser geladen en bestaat uit andere, niet getoonde klassen. Deze zijn dan weer verbonden met de entiteit die links onderin het diagram staat.

### 12.7 Testen aangepaste (workflow-)pagina’s

De aangepaste workflow-pagina’s konden pas deze sprint worden getest, omdat hiervoor ook de tooling eromheen nodig was om deze op te slaan. Achteraf was dit geen goede keuze, want door de verbeteringen van de code review was er een hoop logica veranderd. Indien de tests al eerder waren geschreven, kon hiermee worden gekeken of de code nog steeds werkte na het doorvoeren van de verbeteringen.

Bij het testen van de pagina’s heb ik unittests voor verschillende scenario’s opgesteld. Deze testen omvatten onder andere het testen van elke pagina met items waarvoor geen gegevens bekend zijn (allemaal lege invoervelden), en pagina’s waarvoor verschillende soorten gegevens bekend zijn. Bij de pagina’s met beschikbare gegevens zijn alle mogelijke combinaties voor het ophalen van gegevens (entity-referentie of overerving, zie paragraaf 11.6.3) gebruikt. Onderstaande figuur toont de verschillende testgevallen. (1 = items waarvan geen gegevens bekend zijn, 2 = items waarvan gegevens wel bekend zijn, 3 = testen pagina, 4 = helper-methode voor testen pagina-items)
Uiteraard heb ik ook het opslaan van de pagina uitvoerig getest. In de daarbij behorende test worden verschillende formulieren nagebootst die over alle mogelijke types invoervelden beschikken, en waarvan invoervelden zijn ingevuld of bewust zijn leeggelaten. Bij het maken van deze test kwam naar voren dat er inderdaad teveel digitale audits worden opgeslagen. Deze test kon initieel niet worden voltooid, omdat de code nog niet aan de specificaties voldeed. In sprint 6 is deze test wel voltooid, omdat ik toen het opslaan van de digitale wijzigingen op de goede manier heb geïmplementeerd.
12.8 Testen inrichten van workflows

Het tweede onderdeel wat ik heb getest is het in deze sprint gebouwde component voor het inrichten van de workflows. Bij het testen van het inrichten van workflows is onder andere rekening gehouden met verplichte en niet verplichte invoervelden. Ook pagina’s dienden in de juiste volgorde te worden opgeslagen en te kunnen worden ingesteld.

Voor het testen van het instellen van de pagina’s heb ik in de unittests verschillende pagina’s (zowel ingebouwde als aangepaste pagina’s) opgeslagen. Sommige pagina’s bevatten enkel gegevens in de pagina zelf (vooraf ingevulde gegevens), terwijl andere pagina’s gegevens specifiek voor de workflow hadden opgeslagen. In beide gevallen moesten de juiste gegevens in de workflow worden getoond. Hierbij werd de volgorde van de pagina’s gelijk mee getest.

Het testen van het instellen van de basisinstellingen van de workflows koste relatief weinig tijd, omdat dit bestaande mogelijkheden van het workflow-systeem waren. Hiervoor waren er al unittests aanwezig, en deze waren in een eerdere sprint al aangepast aan het nieuwe datamodel voor workflows (met instellingen per faculteit). Deze tests behoorden uiteraard te slagen, waarmee bevestigd werd dat de code voor het opslaan van de workflows nog correct werkte.
13 Sprint 6: Voortgangsoverzicht, beheerpagina en afronden

13.1 Requirements

Bij aanvang van de zesde sprint heb ik een bespreking met Jeroen gehad over wat er tijdens deze sprint nog moet gebeuren. Gedurende de eerste week van de sprint waren Marcel en Steven afwezig, waardoor zij niet bij de bespreking aanwezig konden zijn. Bovendien konden zij de requirements niet prioriteren. Als dit pas na een week zou gebeuren, was er te weinig tijd over om nog tot het uitvoeren van de nieuwe requirements te komen. Daarom heeft Jeroen enkel een schatting van de prioriteit van de requirements van deze sprint kunnen geven.

Voorafgaand aan de bespreking heb ik doorgenomen welke backlog items er allemaal over waren gebleven (doordat ze niet aan bod waren gekomen of omdat ze niet aan de Definition of Done voldeden). Hierbij kwamen de items van de volgende requirements naar voren:

- Het instellen van workflows was nog niet volledig geïmplementeerd, waardoor de requirements WFE-F004, WFE-F005, WFE-F006 terug op de backlog waren geplaatst. Dit betrof het automatisch opstarten van een workflow bij het goed/ afkeuren van een workflow en het versturen van meldingen bij workflows. Het automatisch opstarten van workflows (WFE-F004, WFE-F005) was volgens Jeroen belangrijk voor de VU en kreeg daarom de prioriteit Must Have. De notificaties voor de workflows (WFE-F006) waren minder belangrijk en kregen de prioriteit Could Have.
- Het importeren van de ingestelde workflows per faculteit (i.v.m. de performance-problemen) werkte niet altijd goed en duurde soms toch erg lang. Daardoor voldeed requirement WFE-F007 niet aan de Definition of Done en is het teruggeplaatst op de backlog. Dit item kreeg de prioriteit Should have.
- Het opslaan van een aangepaste pagina verliep wel goed, maar er werden teveel digitale wijzigingen (audits) aangemaakt. Dit was ook al bij het testen gebleken en hierdoor voldeed dit requirement (WFE-P011) niet aan de Definition of Done. Dit item kreeg de prioriteit Must Have, omdat het belangrijk is voor het verwerken van aangepaste formulieren.
- Er was behoefte aan een voortgangsoverzicht voor functioneel beheer, waarop zij de voortgang in workflows konden zien en eventueel workflows konden (her)starten. Hieruit ontstonden de nieuwe requirements VO-003 t/m VO-007.
- Er was behoefte om extra kolommen aan een bestaand voortgangsoverzicht met vakken toe te voegen. Deze kolommen waren alle van toepassing zijnde workflows, en hierin moest de laatst uitgevoerde workflow worden getoond. Uit deze wens kwam het nieuwe requirement VO-008 voort.
- Het requirement WFE-I007 (Instelbaar of workflow kan worden afgekeurd) was niet door de test van Jeroen heen gekomen, omdat het afkeuren van workflows niet goed werkt.
- De overige requirements die nog op de backlog stonden waren allen te veel werk en kregen daardoor de prioriteit “Won’t Have” met als reden Out of Scope. Deze konden helaas niet meer binnen de afstudeerperiode worden uitgevoerd.
13.2 Plannen van de sprint

Op basis van de in de vorige paragraaf omschreven requirements heb ik de planning voor de laatste sprint opgesteld. De items met de hoogste prioriteit kregen uiteraard voorrang, zowel bij de planning als bij de uitvoering. Vanwege de geschatte duur van het uitvoeren van deze taken zijn de requirements VO-007 (Einddatum instellen bij handmatig opstarten workflow) en WFE-F006 (Notificaties bij workflows) niet in deze sprint genomen.

De definitieve sprint planning is als volgt:

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Omschrijving</th>
<th>Schatting</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-I007</td>
<td>Instelbaar of workflow kan worden afgekeurd</td>
<td>8SP</td>
</tr>
<tr>
<td>WFE-F004</td>
<td>Opstarten nieuwe workflow bij afronden workflow</td>
<td>13SP</td>
</tr>
<tr>
<td>WFE-F005</td>
<td>Opstarten nieuwe workflow bij afkeuren workflow</td>
<td>8SP</td>
</tr>
<tr>
<td>WFE-P011</td>
<td>Correct bijhouden wijzigingen bij opslaan workflow-pag.</td>
<td>8SP</td>
</tr>
<tr>
<td>WFE-F007</td>
<td>Correct importeren van workflows naar het dashboard</td>
<td>8SP</td>
</tr>
<tr>
<td>VO-003</td>
<td>Beheerpagina voor workflows</td>
<td>18SP (tot.)</td>
</tr>
<tr>
<td>VO-004</td>
<td>Totaalaantal betrokkenen workflow op VO-003</td>
<td>5SP</td>
</tr>
<tr>
<td>VO-005</td>
<td>Aantal betrokkenen die workflow hebben afgerond op VO-003</td>
<td>5SP</td>
</tr>
<tr>
<td>VO-006</td>
<td>Handmatig (her)starten en afronden workflow vanuit VO-003</td>
<td>8SP</td>
</tr>
<tr>
<td>VO-008</td>
<td>Extra kolommen voor workflows in voortgangsoverzicht vakken</td>
<td>3SP</td>
</tr>
</tbody>
</table>
13.3 Systeemontwerpen

In deze sprint waren de meeste onderdelen al tijdens eerdere sprints ontworpen. De onderdelen die wel nieuw waren betroffen:
- De beheerpagina voor het handmatig starten, stoppen en herstarten van workflows
- De workflowkolommen in de rapportage met vakken

Van beide onderdelen heb ik een UML-ontwerp gemaakt. Het eerste ontwerp (Figuur 38) bevat enkele nieuwe methoden in de WorkflowService en de ImportWorkflowService. Daarnaast is het ManageWorkflowResult nieuw toegevoegd: Dit bevat alle data die nodig is om het beheerscherm te kunnen tonen. Gezien de performance-problemen die bij het beheren van workflows kunnen ontstaan, heb ik dit keer meteen besloten om de beheeracties op de workflows op een achtergrond thread te laten verlopen. Op deze manier kan de gebruiker direct verder met het gebruik van het systeem, terwijl de actie op de achtergrond wordt uitgevoerd.

Figuur 41: UML-ontwerp voor de beheerpagina met workflows
Bij de rapportage van de vakken heb ik voor de duidelijkheid een UML-ontwerp van de bestaande situatie gemaakt, omdat de benodigde workflow-kolommen met de huidige code eenvoudig kunnen worden toegevoegd. Het enige onderdeel wat achteraf nieuw moest worden toegevoegd, was een methode in de WorkflowService die de voortgang van de workflows bepaalt en omzet in een gebruikersvriendelijke tekst.

Figuur 42: UML-ontwerp voor de rapportage met vakken. Dit is een bestaand rapportagesysteem.
13.4 Ontwerp gebruikersinterface

Voor de beheerpagina met workflows heb ik een ontwerp gemaakt van de gebruikersinterface. Dit is een eenvoudig scherm met daarop de naam van de workflow, de status en de knoppen voor de acties. Een belangrijk verschil t.o.v. de opgestelde requirements is, dat het bij een onderdeel belangrijk is dat de workflow is afgerond, maar niet hoe vaak. Daarom heb ik het aantal betrokkenen wat de workflow heeft uitgevoerd t.o.v. het aantal betrokkenen wat de workflow te zien krijgt (VO-004, VO-005) niet opgenomen in het ontwerp van deze pagina.

In sommige gevallen zijn acties niet toegestaan. In deze gevallen worden de niet-toegestane knoppen grijs gemaakt. De gebruiker kan er uiteraard niet op klikken.

![VAK](image)

**Workflow-acties**

<table>
<thead>
<tr>
<th>Workflow</th>
<th>Status</th>
<th>Acties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uitvraag studiegidteksten</td>
<td>Voltooid op 02-06-2017 door Arjan Vlek</td>
<td>Starten</td>
</tr>
<tr>
<td>Goedkeuren studiegidteksten</td>
<td>Niet ingeschakeld door de faculteit</td>
<td>Stoppen</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Herstarten</td>
</tr>
</tbody>
</table>

_Figuur 43: Beheerpagina voor workflows. Is aanwezig op elk onderdeel en toont de relevante workflows._
13.5 Realisatie nieuwe onderdelen

13.5.1 Voortgang workflows in rapportage met vakken

Als eerste nieuwe onderdeel heb ik de kolommen aan de rapportage met de vakken toegevoegd. Dit was vrij eenvoudig, omdat het bestaande rapportage goed was ingericht. Eigenlijk was hiervoor enkel het volgende nodig:

- Toevoegen workflows aan beschikbare kolommen rapportage
- Ophalen voortgang per workflow
- Sorteerbaar maken voortgang per workflow

Het toevoegen van de workflows aan de kolommen van de rapportage verliep volledig via de bestaande code. Het ophalen van de voortgang van de workflow was nieuw. Hiervoor heb ik een stukje code geschreven wat de rapportage voorziet van de juiste gegevens voor een betreffend veld. In het rapportagesysteem wordt dit een decorator genoemd. Een Decorator krijgt de volgende data mee van het rapportagesysteem:

- Het object (vak) waar de rij betrekking op heeft mee (entityReference)
- Zelf in te stellen headercode (In dit geval: in de reportservice ingesteld op het ID van de workflow)
- Taal waarin de rapportage moet worden getoond.

Hiermee kan ik de meest recente voortgang van de workflow bepalen. Om dit te doen, kijk ik eerst of de workflow is ingeschakeld door de faculteit. Vervolgens haal ik alle uitgevoerde workflows op en sorteer ik deze. Hiervoor heb ik een Java Comparator geschreven. Dit is een middel waarmee Java een lijst met gegevens kan sorteren. Een Comparator werkt als volgt: Er komen telkens 2 items binnen. Als deze gelijk zijn aan elkaar, levert de functie de waarde 0 op. Moet het eerste object boven (in dit geval: Meer voortgang in workflow) het andere object komen, dan levert de functie de waarde 1 op. Moet het tweede object boven het eerste object komen, dan levert de functie de waarde -1 op.

De voortgang van workflows wordt als volgt bepaald:

- Als de workflow niet is ingeschakeld door de faculteit, of nu niet open is vanwege een start- of einddatum, is er geen voortgang. Er kan in dit geval niets worden getoond.
- Verwijder alle nog niet-voltooide workflows, waarbij de gebruiker op de eerste pagina is, uit het resultaat. Ik heb aangenomen dat deze gebruikers nog nooit aan de workflow zijn begonnen, dus het is niet eerlijk om de voortgang hiervan in de vergelijking op te nemen.
- Pas de hierboven omschreven Comparator toe. Deze krijgt steeds twee workflows mee en werkt als volgt:
  - Allebei de workflows zijn nog niet voltooid: Vergelijk het paginanumber waar de huidige gebruiker zich bevindt. De workflow met de meeste voortgang (eenzelfde workflow heeft altijd evenveel pagina’s) is het meest recent.
  - Workflow 1 nog niet voltooid maar workflow 2 wel: Workflow 1 is recente. Dit omdat de gebruiker al verder is dan de introductiepagina en er dus actief mee bezig is
  - Workflow 1 voltooid maar workflow 2 nog niet: Workflow 2 is recente. Idem als hierboven
  - Workflow 1 en 2 allebei voltooid: Vergelijk de datum waarop de workflow is voltooid. De meest recente workflow wordt hierbij gekozen.

Onderstaande code toont dit proces.
Optional<FacultyWorkflow> facultyWorkflow = 
facultyWorkflowService.findForWorkflowAndEntity(workflowId, entityReference);

if(!facultyWorkflow.isPresent() || !facultyWorkflow.get().isEnabled() || 
!Utils.isOptionalDateRangeValidOn(LocalDate.now(), facultyWorkflow.get().getStartDate(), facultyWorkflow.get().getEndDate())) {
// If the faculty didn’t enable the workflow, return this case in the result.
return new MostRecentExecutedWorkflowResult(null, null);
} else {
List<ExecutedWorkflow> executedWorkflows = 
executedRepository.findAllByEntityAndFacultyWorkflowAndInvalidatedIsFalse(entityReference, facultyWorkflow.get());

// If the workflow has been enabled, find the most recent executed workflow. If nobody ever 
opened the workflow, return null.
return new MostRecentExecutedWorkflowResult(facultyWorkflow.get(), executedWorkflows.stream()
// Remove all workflows that have not been completed and are still at page 1. We can 
assumed those were never opened by the user.
.filter(executedWorkflow -> executedWorkflow.getCompleted() != null || 
executedWorkflow.getIndex() > 0)
.max((o1, o2) -> {
  if(o1.getCompleted() == null && o2.getCompleted() == null) {
    return Integer.compare(o1.getIndex(), o2.getIndex()); // If both have not been 
completed, compare the page number (index);
  } else if(o1.getCompleted() == null && o2.getCompleted() != null) {
    return 1; // If this workflow has not yet been completed but the other already 
has been, this workflow is newer.
  } else if(o1.getCompleted() != null && o2.getCompleted() == null) {
    return -1; // If this workflow has been completed but the other one has not, 
the other workflow is newer.
  } else {
    return o1.getCompleted().compareTo(o2.getCompleted()); // If both have been 
completed, compare the completion date.
  }
}).orElse(null));
}

**Figuur 44:** Code voor het bepalen van de voortgang van een workflow.

Met bovenstaande functie kan het sorteeralgoritme van lijsten de lijst met uitgevoerde workflows sorteren. De *max* functie uit Java 8 neemt het eerste item uit de gesorteerde lijst, wat neer komt op de meest recente workflow.

Verder was er enkel nog code nodig om de voortgang in tekst te tonen. Deze toont in het geval van een voltooide workflow de naam van de betrokkene en datum waarop deze is voltooid. In het geval van een workflow die wordt uitgevoerd, wordt ook de naam van de betrokkene getoond, maar dit keer staat het huidige paginanummer t.o.v. het totaal aantal pagina’s in de tekst. Om de rapportage sorteerbaar te maken, was het nodig om de datum in het Engelse formaat te tonen. Dit omdat de rapportagemodule geen mogelijkheden biedt voor het specificeren hoe een kolom moet worden gesorteerd. Er wordt dus enkel naar de tekst die er in staat gekeken.
13.5.2 Beheerpagina voor workflows

De beheerpagina voor workflows was iets meer werk om te maken dan gepland. Dit kwam met name door het beperken van de toegestane acties bij het beheren van de workflows. Deze beperkingen waren niet uit de eerste requirements gekomen, maar heb ik gedurende de sprint deels zelf en deels met Marcel vastgesteld.

De toegestane acties voor het beheren van een workflow zijn als volgt:
- Workflows die niet zijn ingeschakeld door de faculteit, kunnen niet worden beheerd. Dit betekent dat starten, stoppen en herstarten niet zijn toegestaan.
- Workflows die geen betrokkenen hebben kunnen niet worden gestart. Dit omdat de start-operatie in dit geval niets kan doen. Effectief betekent dit dat er ook geen operaties mogelijk zijn.
- Workflows die gestart zijn kunnen niet nogmaals worden gestart. Idem voor workflows die zijn gestopt: Een keer starten of stoppen is wel genoeg.
- Workflows kunnen alleen worden herstart als er enige voortgang is. Als helemaal niemand de workflows heeft aangeklikt heeft herstarten geen effect, dus daarom moet het ook niet mogelijk zijn. Zodra er wel voortgang is, is herstarten uiteraard mogelijk.

Onderstaande tabel toont de toegestane acties bij het beheren van workflows:

<table>
<thead>
<tr>
<th>Status / toestand workflow</th>
<th>Toegestane acties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Niet ingeschakeld door de faculteit</td>
<td>-</td>
</tr>
<tr>
<td>Ingeschakeld, maar geen betrokkenen</td>
<td>-</td>
</tr>
<tr>
<td>Ingeschakeld, betrokkenen aanwezig, nog niet gestart</td>
<td>X</td>
</tr>
<tr>
<td>Gestart, maar nog geen voortgang</td>
<td>-</td>
</tr>
<tr>
<td>Gestart en voortgang gemaakt</td>
<td>-</td>
</tr>
<tr>
<td>Gestart en voortgang voltooid</td>
<td>-</td>
</tr>
<tr>
<td>Gestopt (= ingeschakeld, nog niet gestart)</td>
<td>X</td>
</tr>
<tr>
<td>Systeem is bezig (start, stop, of herstart in uitvoering)</td>
<td>-</td>
</tr>
</tbody>
</table>

*Figuur 45: Tabel met toegestane acties voor beheren van workflows*

Een voordeel wat ik hierbij had was dat sommige statussen al te bepalen waren door de code die ik voor de rapportage met vakken had geschreven. Of een workflow voortgang heeft of dat een workflow is ingeschakeld door de faculteit, was al te bepalen.

Voor het bepalen van de overige statussen had ik nog geen bestaande code. Voor sommige statussen was de code makkelijk te schrijven, voor anderen was dit iets moeilijker. Ik licht het per status toe.

Om te bepalen of een workflow, die is ingeschakeld, betrokkenen heeft, simuleer ik de start-operatie van de workflow. Hierbij wordt exact dezelfde code voor het starten van de workflow uitgevoerd, enkel worden er geen gegevens naar de database geschreven. Indien de uitkomst van deze operatie "0" is, kunnen er geen workflows worden gestart en zijn er dus geen betrokkenen. Onderstaande code toont hoe dit werkt.
@Async
public ListenableFuture<Integer> startWorkflowAsync(FacultyWorkflow facultyWorkflow,
EntityReference reference, boolean simulate) {
    importWorkflowsTasklet.clearCache();
    List<ExecutedWorkflow> executedWorkflows = executedWorkflowRepository.findAll();
    List<ExecutedWorkflow> started = new ArrayList<>();
    Map<WorkflowRelation, Set<Relation>> relationData =
importWorkflowsTasklet.findRelationsForWorkflow(facultyWorkflow.getWorkflow(), reference);
    relationData.forEach((workflowRelation, relations) -> {
        Set<Person> personsWithRelation =
importWorkflowsTasklet.findPersonsForRelations(relations);
        personsWithRelation.forEach(person -> {
            Optional<ExecutedWorkflow> existingWorkflowProgress =
findCurrentWorkflowProgressForPerson(person, reference, facultyWorkflow.getWorkflow(),
executedWorkflows, false);
            // If this user is already performing this workflow, don’t start it again. Only if
not, start it up.
            if(!existingWorkflowProgress.isPresent()) {
                ExecutedWorkflow executedWorkflow = new ExecutedWorkflow(facultyWorkflow,
reference, person.getExternalId(), true);
                if(!simulate) {
                    LOGGER.info("Manually started workflow <" +
executedWorkflow.getFacultyWorkflow().getWorkflow().getName() + "> for <" +
person.getExternalId() + ">...");
                }
                started.add(executedWorkflow);
            }
        });
    });
    if(!simulate) {
        executedWorkflowRepository.save(started);
    }
    return new AsyncResult<>(started.size());
}


De volgende status, of een workflow al is gestart, is te bepalen aan het aantal ExecutedWorkflows
waarbij de status niet “Invalidated” (Ongeldig verklaard of gestopt) is. Als dit aantal voor de betreffende
workflow “0” is, is er geen enkele gebruiker die de workflow heeft uitgevoerd. Bij het stoppen van een
workflow worden alle ExecutedWorkflows op “Invalidated” gezet, waardoor het starten opnieuw kan
worden uitgevoerd.

De laatste status, of het systeem bezig is, heb ik toegevoegd zodat gebruikers niet twee keer op de
knoppen kunnen klikken. Dit zou tot het gevolg hebben dat een workflow dubbel wordt geprobeerd te
starten, en leidt waarschijnlijk tot een constraint violation in de database. Om dit te bepalen plaats ik de
huidige workflow plus het object waarop de operatie wordt uitgevoerd in een lijst.
Bij het bepalen van de voortgang van de workflow kijk ik of het huidige object zich in deze lijst bevindt. Is
dit zo, dan zal er “Systeem is bezig” worden getoond en kan er niet op de knoppen worden geklikt.
Testen
Om het beheerscherm voor workflows te testen, heb ik een aantal testgevallen opgesteld. De volgende scenario's moeten correct worden afgehandeld:

Laden scherm:
- De verschillende statussen, zoals in de tabel aan het begin van deze paragraaf staan vermeld, moeten correct worden verwerkt. Het resultaat van de pagina (ManageWorkflowsResult) dient de correcte waardes voor de verschillende variabelen te hebben. Dit betekent dat alle gevallen moeten worden getest.

Starten workflow:
- Bij starten van een workflow mag enkel die workflow worden gestart, voor het betreffende object / entityReference
- Als een workflow wordt gestart terwijl er al mensen bezig zijn (is niet mogelijk via de userinterface) moet de workflow enkel voor de mensen die er niet mee bezig zijn worden gestart

Stoppen workflow:
- Bij stoppen van een workflow mag enkel de betreffende workflow worden gestopt, voor het betreffende object / entityReference
- Reeds afgeronde workflows mogen niet als gestopt worden gemarkeerd
- Een reeds gestopte workflow nogmaals stoppen levert geen resultaat op en verandert geen data.

Herstarten workflow:
- Het herstarten van een workflow is in feite het stoppen en vervolgens starten van een workflow. Na het herstarten moeten er nieuwe ExecutedWorkflows zijn die allemaal op pagina 1 staan. Uiteraard behoren deze tot hetzelfde object en workflow.
- Bij het herstarten moeten alle oude ExecutedWorkflows als gestopt zijn gemarkeerd. Dit zijn dus ook de workflows die al waren voltooid. Anders zouden mensen die de workflow ooit hebben afgerond hem nooit meer opnieuw kunnen starten!

Op basis van deze data heb ik unittests opgesteld die deze gevallen testen. Bij het starten van de workflow test ik met verschillende workflows en controleer ik of voor elke workflow een ExecutedWorkflow wordt aangemaakt.

// Ensure there are no workflows enabled yet.
Assert.assertEquals(0, executedWorkflowRepository.findAll().size());

// Start the workflow. The workflow should be enabled for the first 6 persons.
int startedWorkflows = importWorkflowService.startWorkflowAsync(facultyWorkflow, faculty.toReference(), false).get();
Assert.assertEquals(6, startedWorkflows);

List<ExecutedWorkflow> executedWorkflowList = executedWorkflowRepository.findAll().stream().sorted(Comparator.comparing(ExecutedWorkflow::getUserName)).collect(Collectors.toList());
Assert.assertEquals(6, executedWorkflowList.size());

Figuur 47: Gedeelte van unit-tests voor het starten van workflows. Het aanmaken van data is hier niet getoond.
De unittests voor het starten, stoppen en herstarten van workflows testen zowel met niet-gerelateerde personen als met gerelateerde personen. Bovendien worden er in sommige testgevallen afgeronde workflows en ongeldig verklaarde workflows toegevoegd.

Voor het laden van de beheerpagina heb ik per status een testgeval toegevoegd. In elk geval wordt de juiste set met gegevens toegevoegd.

13.5.3 Nieuwe code voor workflow-importer

Naast de nieuwe onderdelen heb ik ook een aantal onderdelen afgerond die in eerdere sprints niet volledig waren afgerond.

Als eerste heb ik gekeken naar de nog steeds aanwezige problemen met het importeren van workflows (requirement WFE-F007). Het was namelijk het geval, dat indien een gebruiker een nieuwe rol kreeg binnen een vak of opleiding, en hiervoor al een workflow aanwezig was, dat deze gebruiker de nieuwe workflow niet te zien kreeg. Dit kwam doordat de code voor het importeren van de workflows niet goed naar de gedefinieerde relaties tussen betrokkene en workflow keek. Bovendien was de code voor het importeren van workflows onnodig complex geworden. Dit kwam doordat ik bij het importeren begon vanuit een lijst met personen. Een beter idee was om vanuit (een lijst met) workflows te beginnen, om hiervoor de gerelateerde personen op te halen, en voor deze personen de workflows in te schakelen. Ook moesten workflows zonder start- en einddatum niet worden geïmporteerd, omdat deze handmatig zijn ingeschakeld. Als dit wel zou gebeuren, zouden handmatig gestopte workflows opnieuw worden geïmporteerd.

Het resultaat is een nieuwe versie van de importer voor workflows. Voor de code, zie [bijlage 7].

Om te zorgen dat de performance goed blijft heb ik ervoor gezorgd dat het ophalen van entiteiten uit de database in één keer gebeurt. Het ophalen van 10000 objecten in een keer is namelijk vele malen sneller dan 10000 keer 1 object uit een database ophalen. Zie hiervoor ook paragraaf 10.6.

Alle opgehaalde objecten worden in een cache gestopt, zodat ze binnen de verschillende methodes van de klasse (ImportWorkflowsTasklet), beschikbaar zijn. Hiermee is de nieuwe code voor het importeren van workflows veel overzichtelijker en compacter geworden. Bovendien kunnen een aantal methodes nu ook voor andere onderdelen (zie de paragrafen hieronder) worden gebruikt, wat dubbele of vrijwel gelijke code voorkomt.

13.5.4 Afkeuren van workflows

Vervolgens heb ik gekeken naar het instellen of een workflow kan worden afgekeurd (WFE-I007). Hierbij liep ik al snel vast. De logica voor het afkeuren van workflows was namelijk nog veel te afhankelijk van de oude situatie, waarbij workflows aan één status waren gekoppeld. Ik kon direct beginnen om het afkeuren van workflows helemaal opnieuw te implementeren, maar omdat ik twijfelde over eisen die de aanwezige code stelde over wanneer een workflow kan worden afgekeurd, heb ik eerst bij Jeroen en Marcel nagevraagd wat hier de wensen en eisen voor zijn. Hieruit bleek dat er eigenlijk meerdere uitkomsten voor workflows moesten kunnen, en dat deze allemaal per workflow, per faculteit instelbaar zijn. Omdat dit onderdeel dus duidelijk andere requirements had en naar mijn mening veel ontwikkeltime ging vergen, heb ik besloten om het niet in deze sprint uit te voeren. Ik heb dit aan het eind van de sprint in het overdrachtsdocument vermeld, zodat het bedrijf weet dat dit probleem nog moet worden opgelost.
13.5.5 Automatisch opstarten (triggeren) van workflows

Wat ik wel kon implementeren, was het automatisch opstarten (triggeren) van een nieuwe workflow bij het afronden en afkeuren van een workflow. De code voor het opslaan van de gegevens had ik al in sprint 2 gemaakt, maar er werd met deze opgeslagen gegevens nog niets gedaan. Daarom heb ik aan de methode voor het afronden workflows (complete) in de WorkflowService een stukje code toegevoegd wat dit proces opstart.

Hiervoor wordt weer gebruik gemaakt van de ImportWorkflowService, zodat het opstarten van de workflow op een achtergrond thread plaatsvindt. Anders zou het afronden van een workflow erg lang kunnen duren, wat natuurlijk niet prettig is voor de gebruiker.

Bij het triggeren van workflows wordt een deel van de code voor het importeren van workflows (ImportWorkflowsTasklet) gebruikt om de betrokkenen van de workflow op te halen. Wat er hierna gebeurt is dat er voor elke betrokkene van de nieuwe workflow een ExecutedWorkflow wordt aangemaakt. Op deze manier verschijnt de nieuw opgestarte workflow op het dashboard van de betreffende persoon. Uiteraard wordt hierbij gecontroleerd of de workflow wel is ingeschakeld door de faculteit. Onderstaande code wordt gebruikt voor het automatisch opstarten van workflows.

```java
// First, check if the faculty has enabled the workflow.
Optional<FacultyWorkflow> facultyWorkflowOptional = facultyWorkflowService.findForWorkflowAndEntity(triggeredWorkflow, reference);
 Preconditions.checkState(facultyWorkflowOptional.isPresent(), "Can't trigger workflow "+ triggeredWorkflow.getName() + " for object " + reference.toString() + ": The faculty didn't enable the workflow!");

FacultyWorkflow facultyWorkflowForTriggered = facultyWorkflowOptional.get();
 Preconditions.checkState(facultyWorkflowForTriggered.isEnabled(), "Can't start triggered workflow "+ triggeredWorkflow.getName() + " for object " + reference.toString() + ": The faculty didn't enable the workflow!");

Map<WorkflowRelation, Set<Relation>> relationData = importWorkflowsTasklet.findRelationsForWorkflow(triggeredWorkflow, reference);

// Enable the triggered workflow for all related persons.
List<ExecutedWorkflow> triggered = executedWorkflowRepository.save(relationData.entrySet().stream()
    .map(entry -> importWorkflowsTasklet.findPersonsForRelations(entry.getValue())
        .stream()
        .map(person -> new ExecutedWorkflow(facultyWorkflowForTriggered, reference, person.getExternalId()))
        .collect(Collectors.toList()))
    .flatMap(Collection::stream)
    .collect(Collectors.toList()));

triggered.forEach(executedWorkflow -> LOGGER.info("Auto-started workflow " + triggeredWorkflow.getName() + " for user " + executedWorkflow.getUserName() + " and object " + reference.toString() + "...");

return new AsyncResult<>(triggered.size());
```

Figuur 48: Code voor automatisch opstarten van workflows
13.5.6 Bijhouden audits bij aangepaste workflow-pagina’s

Bij de demo van sprint 4 was naar voren gekomen dat er bij het invullen van een aangepaste workflow-pagina alleen digitale audits per gewijzigd object mogen worden aangemaakt. Dit gebeurde eerst per veld van de pagina, wat uiteraard niet goed was. Om dit te verbeteren heb ik bij het invullen van de aangepaste workflow-pagina alle gewijzigde entiteiten in een lijst gestopt en vervolgens per entiteit een audit aangemaakt. Er bleek echter nog meer verbetering mogelijk, want Jeroen tipe mij dat er voor het aanmaken van de audits al bestaande functionaliteit in het systeem aanwezig is. Deze functionaliteit, beschikbaar in de basisklasse HistoricalCrudService, en door alle objecten die digitale wijzigingen ondersteunen overgeërfd in een eigen klasse, biedt de business logica voor het aanmaken van wijzigingen, beveiliging en het eventueel versturen van notificaties. De bedoeling is dat het workflow-formulier de juiste service gebruikt om het gewijzigde object op te slaan. Als er geen service beschikbaar is, wordt er geen digitale wijziging aangemaakt.

Het lastige om dit voor elkaar te krijgen was, dat deze service gebruik maakt Java Generics. Generics zijn een manier om klassen voor verschillende types objecten bruikbaar te maken, waarbij een bepaalde methode (bijvoorbeeld save) automatisch een object van het juiste type teruggeeft. Een voorbeeld is HistoricalCrudService<Faculty, FacultyView>, waarbij deze service geschikt is voor het opslaan van objecten van het type Faculty, en er een database view aanwezig is voor het bekijken van objecten van het type facultyView.

Het is echter niet bekend welk object er in het formulier geplaatst is, dus daarom was het nodig om alle in het systeem aanwezige HistoricalCrudServices in een lijst te stoppen en op basis van het op te slaan object (met Java Reflection in te zien) de juiste service te kiezen.

Voor het vullen van de lijst met services bood het Spring Framework twee functies aan: ApplicationContext.getBeansOfType(<klasse>), en GenericTypeResolver.resolveTypeArguments(<instantie, basisklasse>).

Met de getBeansOfType methode kunnen alle Spring Beans (Service, Controller, Repository etc.) van een bepaalde type (in dit geval HistoricalCrudService) worden opgehaald. Deze komen in een set met naam en klasse terug.

De GenericTypeResolver stelt je in staat om het dynamisch toegekende type van een generieke klasse (in het voorbeeld hierboven: Faculty), te achterhalen. Op deze manier kan met opvallend weinig code een lijst met alle aanwezige services worden opgeslagen. Per klasse (Faculty, Study etc.) wordt de betreffende service opgeslagen.
private final Map<Class<?>, HistoricalCrudService> data = new HashMap<>();

@Autowired
public HistoricalCrudServiceRegistry(ApplicationContext applicationContext) {
    applicationContext.getBeansOfType(HistoricalCrudService.class, false, true).forEach((string, service) -> {
        data.put(GenericTypeResolver.resolveTypeArguments(service.getClass(), HistoricalCrudService.class)[0], service);
    });
}

public HistoricalCrudService getService(ChangeableEntity entity) {
    return data.get(entity.getClass());
}

Figuur 49: Ophalen van alle aanwezige HistoricalCrudServices.

Bij het opslaan van de aangepaste workflow-pagina kijk mijn code enkel of de lijst met services over een service beschikt, en gebruik ik de service om het gewijzigde object te slaan. Hierbij wordt van alle bestaande code voor het aanmaken van digitale wijzigingen gebruik gemaakt. Als er geen service beschikbaar is (niet aangemaakt of fout bij ophalen) wordt een waarschuwing gelogd en wordt enkel de entiteit opgeslagen. Indien dit niet gewenst is, kan er altijd nog een foutmelding / exception worden ingebouwd. In dat geval gaan alle ingevoerde gegevens verloren.

for (ChangeableEntity entity : changedEntities) {
    HistoricalCrudService service = historicalCrudServiceRegistry.getService(entity);
    if (service != null) {
        service.save(entity, comment);
    } else {
        LOGGER.warn("No HistoricalCrudService found for entity "+ entity.toReference().toString() + ": Defaulting to regular save without making a change.");
        entityService.save(entity);
    }
}

Figuur 50: Code voor opslaan van objecten met dynamisch opgehaalde HistoricalCrudService

13.5.7 Workflow-notificaties

Na het afronden van de onderdelen uit de vorige paragrafen had ik nog wat tijd over. Dat kwam omdat de laatste sprint-demo pas op de vrijdag was, terwijl de sprint al op de dinsdag (= 2.5 week) was begonnen. Ik heb toen getwijfeld om alsnog het afkeuren van de workflows te implementeren, of om nog iets voor de workflow-notificaties te doen. Omdat ik de meeste code voor de workflow-notificaties al in sprint 3 had geschreven, kon ik dit onderdeel nog binnen de resterende tijd afronden.

Even ter herinnering: De workflow-notificaties, waaraan ik in sprint 3 ben begonnen, bevatten het volgende requirements:

- Instelbaar per faculteit, op het scherm met workflow-instellingen
- Gepland versturen van een notificatie bij de start- en einddatum van een workflow
- Instelbaar hoeveel dagen voor of na de start- of einddatum de notificatie wordt verzonden.
- Versturen bij het goed- en afkeuren van een workflow
Voor de workflow-notificaties heb ik aan de klasse die notificaties verstuurt ({NotificationScheduler}) code toegevoegd. Deze bevat nu een methode die via CRON (vergelijkbaar met Geplande Taken onder Windows) elke nacht om 02:00 (configureerbaar voor systeembeheer) wordt aangeroepen om te controleren op notificaties. Is er een notificatie die voor de betreffende datum staat gepland (zoals 7 dagen voor het einde van een workflow), dan wordt deze vervolgens verzonden. Verzonden notificaties worden als verzonden gemaild, zodat ze niet nogmaals verzonden kunnen worden. Bovendien heb ik een methode toegevoegd voor het versturen van workflow-notificaties die niet van een datum afhankelijk zijn, zoals bij het voltooien of afkeuren van de workflow. Deze gebruiken dezelfde code voor het versturen van de notificatie, behalve dat de parameter {ScheduledWorkflowNotification} met een “geplande” notificatie met daarin de datum van vandaag wordt gevuld. Ook is deze methode weer "Async", omdat het afronden van de workflow anders te lang kan duren.

Voor dit onderdeel heb ik drie unittests gemaakt. De eerste unittest test de functie voor het versturen van alle notificaties voor vandaag. Hiervoor worden er zowel een start- als einddatum notificatie met de datum van vandaag aangemaakt. De notificatie bij de startdatum wordt naar alle betrokkenen verstuurd. De notificatie bij de einddatum van de workflow wordt enkel naar personen die de workflow nog niet hebben voltooid gestuurd. In het eerste testgeval wordt deze notificatie niet verzonden, omdat de testgebruiker zijn workflow al heeft afgerond. In een tweede testgeval wordt deze notificatie wel verstuurd, omdat de testgebruiker nu bezig is met zijn workflow. Om te kijken of de notificatie is verstuurd, kijk ik of er een in-app notificatie is opgeslagen in de database. Een in-app notificatie wordt als laatste onderdeel van het versturen van notificaties opgeslagen. Deze kan ik direct ophalen, omdat tijdens de unittests ondersteuning voor async functies uit staat. Alle functies die door unittests worden aangeroepen worden dus op de main thread uitgevoerd.

De tweede en derde unittests testen de functie voor het versturen van notificaties bij het afronden en afkeuren van de workflow. Hiervoor heb ik een testworkflow opgeslagen met bijbehorende notificatie-instellingen. Vervolgens kijk ik of deze naar alle betrokkenen van de workflow is verstuurd, ook weer door de opgeslagen in-app notificaties te bekijken. Uiteraard test ik of het juiste sjabloon is gebruikt en of de gebruikersnaam overeenkomt met de ontvanger van de notificaties.

De notificaties bij het afronden en afkeuren van de workflow hadden, naar mijn mening, naar een of meer specifieke personen moeten worden verstuurd. Dit is echter nooit in de requirements en de ontwerpen opgenomen. Ik had ook niet meer genoeg tijd om die functionaliteit alsnog te implementeren. Daardoor wordt deze notificatie nu naar alle betrokkenen gestuurd - een aandachtspunt voor het bedrijf bij het uitrollen van de vernieuwde applicatie.

13.6 Sprint demo

Op vrijdag 19 mei was de demo van deze sprint. Dit was tevens mijn laatste sprint-demo presentatie. Bij deze presentatie waren veel aanwezig, waaronder ook een aantal collega’s die aan AscMe werken. Er was één oppervlakkige afwezige: Steven Losekoot. Steven kon enkel op een aantal momenten in de week daarna komen, terwijl Marcel en Jeroen op die momenten niet konen. Bovendien vond ik het niet prettig als de presentatie pas zo dicht tegen het inlevermoment van het afstudeerverslag lag. Ik vond het wel jammer dat Steven niet aanwezig kon zijn.

Tijdens de presentatie heb ik uiteraard het gemaakte werk getoond. Hierover waren de meesten te spreken, er waren enkel wat twijfels bij Jeroen over het simuleren van het opstarten van de workflows. Hij was bang dat dit te lang zou duren, maar op de testserver laadde de pagina snel genoeg (< 3 seconden).
13.7 Sprint retrospective

Na de demo was de gebruikelijke sprint retrospective. Er waren tijdens deze sprint zeer veel dingen goed gegaan: Ik heb heel veel werk kunnen verrichten en heel veel onderdelen die eerst niet af waren gekomen zijn nu alsnog voltooid. Bovendien verliep het werken aan de back-end dit keer goed, en heb ik goed rekening gehouden waar eventuele performance problemen konden ontstaan door betreffende methoden Async te maken, zodat ze op de achtergrond worden uitgevoerd.

Ik had zelf eigenlijk maar één punt wat beter had gekund: De verbeteringen aan de workflow-importer (paragraaf 12.4.3) zorgden ervoor dat een aantal unittests niet meer slaagden. Dit betroffen veelal tests waarin workflows zonder start- en einddatum werden opgeslagen, maar ook tests waarin workflows werden gestopt die eigenlijk al waren afgerond. Het verbeteren hiervan kostte nog relatief veel tijd, dit moest vlak voor de demo nog gebeuren. Gelukkig heb ik het wel kunnen afronden.

13.8 Afronding en retrospective over het gehele project

Na afloop van de demo en de retrospective heb ik ook een terugblik gegeven op het hele project. Hierbij heb ik kort alle gemaakte onderdelen toegelicht, en heb ik er ook bij stilgestaan welke onderdelen nog niet waren afgerond. Hiervan heb ik een document opgesteld, zodat het bedrijf weet waar rekening mee moet worden gehouden bij het uitrollen van de vernieuwde versie van AscMe. Zie hiervoor [Bijlage 8]. In evalutaties (Hoofdstukken 14, 15 en 16) ga ik hier nog verder op in.

De niet-afgeronde onderdelen licht ik hieronder toe:

- Een aantal onderdelen hadden nog enkele bugs of problemen. Met name het inschakelen van gegroepeerde workflows, het afronden van workflows en het starten van workflows via een e-maillink moeten nog worden verbeterd voordat de vernieuwde applicatie in gebruik kan worden genomen.
- Het groeperen en vertraagd versturen van notificaties (N-002 en N-012) is in de eerste sprint niet gelukt. Het is in latere sprints nooit meer op de backlog geplaatst omdat andere items hogere prioriteit hadden. Er waren enkele GUI-elementen voor aanwezig. Indien dit niet meer gemaakt wordt, dienen deze uit de front-end verwijderd te worden.
- Het tonen van een voorbeeld van een workflow na het instellen van de workflow (WFE-V001 en WFE-V002) is nooit op de backlog geplaatst. Dit kwam omdat het Could Have requirements waren. In principe is het systeem ook zonder deze functionaliteit bruikbaar.
- Het instellen van de weergavemodi van de ingebouwde workflow-pagina’s (WFE-S003 en WFE-S004) is tijdens sprint 5 niet gelukt. Ook dit waren Could Have requirements, dus het was niet meer noodzakelijk om het alsnog af te ronden.
- Het ontwikkelen van uitgebreide voortgangsoverzichten voor eindverantwoordelijken (VO-001, VO-002, verder niet vastgesteld) was niet mogelijk, omdat er geen tijd meer voor was. Dit staat in de productevaluatie verder toegelicht.
14 Procesevaluatie

In dit hoofdstuk kijk ik terug op het proces en de manier van werken gedurende de afstudeerperiode. Ik sta hierbij stil bij de positieve en de negatieve punten.

14.1 Ontwikkeling volgens Scrum

Het werken volgens Scrum beviel mij goed en heb ik, naar mijn idee, goed kunnen uitvoeren. Door steeds een sprint-demo en een sprint-retrospective te houden was het voor de betrokkenen duidelijk hoe het met de voortgang van het project was gesteld. Door telkens aan het begin de requirements bij te werken en de sprint te plannen had ik een goed beeld wat er in de komende sprint te wachten stond. Het enige wat ik misschien beter kon doen, was in de eerste sprints niet te veel taken inplannen. Hierdoor bleef er minder tijd over voor testen, waardoor dit in de volgende sprint alsnog moest gebeuren. Dit was echter lastig in te schatten, omdat de applicatie nog betrekkelijk nieuw voor mij was.

14.2 Technologieën en (programmeer) vaardigheden

Tijdens dit project heb ik beduidend beter leren programmeren. Ik ben met name in het hergebruiken van bestaande functionaliteiten en het gebruiken van functioneel programmeren gegroeid. Ook het werken aan grote projecten was voor mij nieuw, hiervan heb ik een hoop geleerd.

Wat betreft (nieuwe) technologieën heb ik beduidend minder geleerd dan gehoopt. Omdat ik al bekend was met een aantal technologieën die binnen Vorsen worden gebruikt (uit eerdere ervaring bij 42) heb ik hier relatief weinig nieuws van geleerd. Ook mochten er aan de bestaande applicatie weinig frameworks en libraries worden toegevoegd. In een van de gevallen dat ik een nieuwe library had toegevoegd (Dragula drag & drop) moest ik deze toch weer verwijderen, omdat deze niet voldeed. Het leerproces zat met name in de complexiteit van deze applicatie, en het gebruik van geavanceerde technologieën als Spring Batch en Reflection.

14.3 Meetings bij de VU

In het begin van het project ben ik twee keer bij de VU geweest. De eerste keer heb ik een presentatie bijgewoond en de tweede keer heb ik zelf een presentatie gegeven. Tijdens deze besprekingen maakte ik kennis met de klant en kreeg ik inzicht in hoe zij de applicatie gebruiken. Dit vond ik erg interessant, en ik had gedacht dat ik hier vaker naar toe kon gaan. Doordat de bij de VU de procedures voor het versturen van notificaties niet rond waren (functioneel beheer wilde alles via e-mail, maar van hogerop mocht dit niet), was het niet mogelijk om verdere meetings over het door mij ontwikkelde notificatie-gedeelte te houden. Het workflow-gedeelte was veelal te technisch en bedoeld voor het bedrijf zelf, dus daar kon ik bij de VU ook niet uitgebreid over spreken. Dit vond ik best jammer, vooral omdat ik het pas later te horen kreeg, nadat ik een aantal keer had gevraagd of ik nog naar een meeting bij de VU zou gaan.
14.4 Verbeterpunten voor een volgend project

Tot slot licht ik toe wat ik bij een volgend project beter zou doen. Ik zal in de eerste weken van een nieuw project proberen nog meer contact met de opdrachtgever te krijgen, om de requirements en de omschrijving van de opdracht duidelijker in beeld te krijgen. Daarnaast zal ik in de eerste sprint niet te veel items plannen, want de kans dat deze niet allemaal af komen is erg groot. Tot slot zou ik tijdens het ontwikkelen iets meer (verbeter)voorstellen kunnen doen, en eventueel ook kunnen kijken of het mogelijk is om meer nieuwe technologieën binnen het project te gebruiken. Dit moet uiteraard niet ten koste gaan van de kwaliteit en de performance van de applicatie, en het bedrijf moet het hiermee eens zijn.
15 Productevaluatie

In dit hoofdstuk kijk ik terug op de producten die ik tijdens mijn afstudeerstage heb opgeleverd. Omdat deze opdracht betrekking heeft gehad op een aantal specifieke onderdelen binnen één applicatie, evalueer ik het project per onderdeel.

15.1 Vernieuwde notificatie-module

De notificatie-module is naar mijn mening erg goed geworden. Deze biedt verschillende mogelijkheden voor het versturen van notificaties, zodat er op eenvoudige manier handmatige en workflow-notificaties kunnen worden verstuurd. Ook het invullen van teksten via sjablonen verloopt goed en biedt beheerders mogelijkheden om nette e-mailberichten op te stellen. Daarnaast worden alle bij aanvang aanwezige automatische notificaties via dit systeem verstuurd. Tot slot is het voor programmeurs eenvoudig om nieuwe systeemgebeurtenissen voor notificaties toe te voegen, zodat er in de toekomst nog meer automatische notificaties kunnen worden verstuurd.

Het bedrijf is hier ook tevreden over, en hoopt dat het betrokkenen beter op de hoogte gaat stellen van nog uit te voeren taken. Het is nu aan de onderwijsinstellingen om hier ook effectief gebruik van te gaan maken.

15.2 Vernieuwde workflow-module

Ik heb verschillende onderdelen aan de applicatie toegevoegd die het inrichten en gebruiken van workflows hebben verbeterd. Over de meeste van deze onderdelen ben ik zeer tevreden, eigenlijk alleen over de mogelijkheden van de aangepaste pagina’s iets minder.

De nieuwe wizard voor het inrichten van workflows vind ik zelf zeer goed. Het configureren van workflows verloopt nu zo veel beter en duidelijker dan in de oude situatie, dat ik hoop dat hier ook echt meer mee gedaan gaat worden. Dat licht uiteraard ook aan de klanten die de applicatie gaan gebruiken.

Daarnaast vind ik de nieuwe beheerpagina voor het starten en stoppen van workflows heel goed. Deze is eenvoudig te gebruiken en maakt het voor de beheerder mogelijk om zelf een workflow op te starten of te stoppen. Het bedrijf is hier uitermate tevreden over en denkt dat klanten dit veelvuldig zullen gaan gebruiken.

Over de aangepaste workflow-pagina’s ben ik ook behoorlijk tevreden, maar iets minder dan de hierboven genoemde onderdelen. De pagina’s zijn een mooie manier om zelf de inhoud van de workflow samen te stellen, en het opslaan en tonen van gegevens werkt goed, met nette code. Wat ik iets minder vind, zijn de mogelijkheden voor het inrichten van de pagina’s. Ik had eigenlijk meer opties voor het aanpassen van de stijl van de pagina willen toevoegen, zoals de mogelijkheid om groepen items, gescheiden door zelf in te voeren teksten, aan een pagina toe te voegen. Ook het instellen of een veld bewerkbaar is of enkel voor weergave is, was nog een mogelijkheid geweest. Op deze manier zouden de pagina’s nog geavanceerder, maar vooral veel mooier kunnen worden.
Over de workflow-instellingen per faculteit ben ik het minst tevreden, maar dit betekent niet dat ik ontevreden ben. Hoewel de workflow-instellingen per faculteit uiteindelijk goed zijn gelukt, heeft het ontwikkelen ervan naar mijn mening veel te veel tijd gekost. Vooral vanwege de performance-problemen die dit onderdeel met zich meebracht. Gelukkig is alle functionaliteit toch nog af gekomen, zodat het onderdeel wel succesvol kan worden gebruikt. Wat ik mij echter af vraag, is of het systeem door dit onderdeel niet foutgevoeliger wordt. Er waren in de oude situatie namelijk meer controles voordat een workflow werd opengesteld. De kans lijkt mij reëel, dat een beheerder van een faculteit een verkeerde workflow inschakelt, of een verkeerde datum invult. Op deze manier kunnen betrokkenen onverwachts een workflow te zien krijgen die zij eigenlijk niet mogen invullen. Bovendien moeten de automatische (getriggerde) workflows heel goed worden ingesteld, want anders moet er elke keer handmatig een nieuwe workflow worden opgestart.

15.3 Voortgangsoverzichten

Dit was duidelijk het onderdeel waar ik het minste aan gewerkt heb. Al vanaf het begin was het niet duidelijk wat de requirements waren en hoe belangrijk het voor het bedrijf was. Uiteindelijk heb ik toch nog een klein stukje voortgang kunnen tonen in de rapportage met vakken. Hoewel ik voor dit onderdeel dus niet veel heb kunnen doen, zijn de andere gemaakte onderdelen wel van goede kwaliteit geworden. Ik vind zelf dat kwaliteit belangrijker is dan kwantiteit, dus daarom vind ik het zelf niet heel erg dat dit onderdeel niet is afgerond. Het bedrijf gaf bij de laatste presentatie eigenlijk hetzelfde aan. Zij zeiden toen ook dat de grotere voortgangsoverzichten eigenlijk bij een andere, nog op de plank liggende, afstudeeropdracht horen. Het was volgens hun mooi geweest als ik een deel daarvan in deze opdracht mee had kunnen nemen, maar het was absoluut geen vereiste voor het bedrijf.
16 Evaluatie van de beroepstaken

In dit hoofdstuk evalueer ik de voor deze afstudeeropdracht gekozen beroepstaken. Ik sta bij elke gekozen beroepstaak stil bij hoe het toepassen hiervan is verlopen. De beroepstaken zijn uiteraard ook te vinden in het afstudeerplan [bijlage 9].

16.1 Voorbereiden en opstarten software-ontwikkeltraject

Ik heb gedurende de eerste twee weken van mijn afstudeerstage uitgebreid aandacht besteed aan het analyseren van de bestaande situatie, het kiezen van een ontwikkelmethode en het vaststellen van de eerste set van requirements. Door middel van diverse interviews en twee meetings bij de VU heb ik van tevoren een goed beeld gekregen van de wensen aan de applicatie. Met deze verkregen informatie kon ik een goed begin maken met het uitvoeren van het project – ondanks dat in dezelfde periode het afstudeerplan was gewijzigd.

16.2 Uitvoeren analyse door definitie van requirements

Requirements speelden een belangrijke rol bij het uitvoeren van dit project. Aan het begin van het project waren de requirements echter nog niet bekend. Door middel van het voorbereiden van het project (zie bovenstaande paragraaf) en verschillende interviews met de betrokkenen heb ik steeds de requirements verkregen. Ik heb er zoveel mogelijk voor gezorgd dat de requirements door Marcel Noordzij, de requirements analyst van Vorsen, werden gevalidateerd, en dat de requirements door verschillende stakeholders werden geprioriteerd. Met deze prioritering kon ik steeds de juiste requirements in de sprint plaatsen.

16.3 Ontwerpen systeemdeel

Naast het verkrijgen en uitwerken van de requirements vormde het ontwerpen van de nieuw te bouwen of te wijzigen onderdelen een belangrijk onderdeel van het project. Van elk te bouwen onderdeel, wat invloed had op de code van de applicatie, heb ik een UML-diagram gemaakt. Van een aantal onderdelen die invloed hadden op de structuur van de bestaande database, heb ik een Entity Diagram (EER) gemaakt.

Van alle onderdelen die invloed hadden op de gebruikersinterface, heb ik wireframes en/of mock-ups gemaakt. Op deze manier was het goed te zien hoe de onderdelen eruit kwamen te zien, en het hielp zeker mee bij het ontwikkelen aan de applicatie. Opvallend was dat het bedrijf het bijna altijd met het ontwerp wat ik had gemaakt eens was.

16.4 Bouwen applicatie

Na het ontwerpen van een nieuw/te wijzigen onderdeel volgde uiteraard het bouwen ervan. Bij het bouwen van de verschillende onderdelen van de applicatie heb ik zoveel mogelijk gebruik gemaakt van de conventies en de afspraken die binnen het bedrijf golden. Omdat dit een bestaande applicatie was, was ik echter beperkt in het gebruiken van nieuwe frameworks en libraries. Dit was echter geen probleem: De applicatie beschikte over een uitstekende set van frameworks en libraries.
Voor het ontwikkelen van de aangepaste workflow-pagina’s heb ik een aantal nieuwe technieken geleerd, waaronder het gebruik van Reflection, Generics en het toepassen van ORM mapping voor complexere datastructuren, zoals overerving. Achteraf bleek dit niet altijd nodig te zijn, maar ik heb er toch veel van geleerd.

Bij het ontwikkelen van de back-end van de applicatie heb ik bovendien veelvuldig gebruik gemaakt van een aantal functies die bestaan sinds Java 8. Dit zijn Lambda-expressies, Functional interfaces en Streams. Met deze onderdelen is het eenvoudig om operaties op lijsten en verzamelingen uit te voeren. Dit verloopt op een functionele wijze, waarbij het resultaat van de ene functie (bijvoorbeeld Filter) gebruikt kan worden voor een volgende functie (Map). Een hoop van deze functies / technieken heb ik tijdens de cursus Functioneel Programmeren (bij mijn minor bij de Universiteit Utrecht) geleerd.

16.5 Uitvoeren van en rapporteren over het testproces

Nadat een onderdeel was gebouwd, diende het uiteraard getest te worden om er zeker van te zijn dat de gemaakte code werkte zoals verwacht. Het testen van de applicatie vormde veelal het laatste onderdeel van een sprint, waarna een onderdeel als “done” bevonden kon worden.

16.5.1 Unittests

Voor alle onderdelen die ik tijdens dit project heb gemaakt heb ik unittests geschreven. Het streven hierbij was om in de back-end 100% code coverage te behalen. Op de meeste plekken is dit gelukt, maar op een aantal andere plekken lag de coverage net onder de 100%. Dit kwam omdat een aantal excepties die je voor het gebruik van sommige frameworks moest afvangen (JavaMail en Reflection) eigenlijk nooit in de praktijk voorkomen. Het enige onderdeel waar de coverage echt lager lag, was bij de workflow-notificaties. Dit omdat deze in de derde sprint niet waren afgerond, en omdat ik het in de laatste sprint extra heb gemaakt. Ik had niet meer voldoende tijd om alle unittests hiervoor te maken, maar heb dit aan het bedrijf laten weten.

16.5.2 Aanbevelingen op het gebied van testen

Ten slotte heb ik een aantal opmerkingen en aanbevelingen over het testen die belangrijk zijn om te weten, en waardoor ik misschien minder aandacht aan testen heb besteed dan verwacht. Dit komt met name door de manier van testen binnen de organisatie.

Als eerste maakt het bedrijf voor dit project geen gebruik van front-end (unit)tests. Dit kwam omdat, zoals het bedrijf aangaf, de front-end zeer actief wordt ontwikkeld en dat er hierbij vaker nieuwe frameworks en technologieën worden toegepast. De front-end tests moesten, indien deze wel waren gemaakt, veelvuldig worden aangepast om met deze technieken om te kunnen gaan. Hierdoor kosten deze tests relatief veel tijd (en dus geld) en leverden ze weinig op. Zelf ben ik het er mee eens dat het maken van tests veel tijd kost, maar juist bij het frequent gebruiken van nieuwe frameworks en technologieën kunnen (unit)tests handig zijn om te kijken of de code nog functioneert zoals wordt verwacht.

De tweede opmerking die ik heb, gaat over het (gebrek van het) toepassen van andere testtechnieken. Hoewel het bedrijf veelvuldig gebruik maakt van unittests en unit-integratietests, worden (in ieder geval bij dit project) nauwelijks andere testsoorten uit het V-model voor testen (TMap Next) toegepast. Dit betekent dat er dus geen systeemtests worden gehouden, en dat er ook geen volledige integratietests (end-to-end tests tussen front- en backend) worden gehouden. De gebruikersacceptatietests die de VU houdt, worden door de VU zelf geregeld, dus niet door Vorsen. Ik heb zelf dan ook geen idee hoe deze
tests worden gespecificeerd en wat hiermee wordt getest. Ik neem aan dat dit het doorlopen van de verschillende onderdelen van de applicatie is.

Door het gebrek aan verschillende testsoorten bestaat het risico dat de applicatie problemen kan veroorzaken indien deze wordt uitgerold naar een daadwerkelijke server. Zeker gezien het verschil qua database (lokale ontwikkeling met HSQLDB en Postgresql op de server) kunnen er eenvoudig fouten ontstaan.

Het bedrijf is echter bezig met een oplossing, zodat ontwikkelaars eenvoudig (via het virtualisatieplatform Docker) met Postgresql als DBMS kunnen ontwikkelen. Op deze manier verdwijnt in ieder geval het verschil in databases, zodat een grote bron van fouten en ergernis is verdwenen. Hiermee zijn echter niet alle tekortkomingen op het gebied van testen, zoals het gebrek aan testsoorten, opgelost.

De overige tekortkomingen kunnen grotendeels met de beschikbare OTAP-omgeving worden opgelost. Hierbij kan met name op de testserver veel meer worden getest. Door hier integratietests en systeemtests op te zetten en deze (indien mogelijk) automatisch uit te voeren, kan de applicatie voortdurend in de gaten worden gehouden. Hierdoor kunnen een aantal problemen, zoals fouten die ontstaan na refactoring of upgrades van frameworks, worden gedetecteerd. Het zou mooi zijn als er door deze tests minder bugs aanwezig zijn, want die leken toch vrij regelmatig voor te komen.
III. BIJLAGEN
Overzicht bijlagen:

Bijlage 1: Overzicht figuren
Bijlage 2: Begrippenlijst
Bijlage 3: Klassendiagram oude situatie workflows
Bijlage 4: Klassendiagram oude situatie notificaties
Bijlage 5: GUI-ontwerpen gemaakt tijdens de opstart van het project
Bijlage 6: Vergelijking mogelijkheden HTML-editors voor de front-end
Bijlage 7: Code voor importeren van workflows
Bijlage 8: Aandachtspunten voor ingebruikname
Bijlage 9: Afstudeerplan
Bijlage 10: Plan van Aanpak
Bijlage 11: Requirements document
Bijlage 12: Scrum-backlog, aan het einde van project
### Bijlage 1: Overzicht figuren

<table>
<thead>
<tr>
<th>Naam</th>
<th>Omschrijving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figuur 1</td>
<td>Organogram Vorsen B.V. en 42 B.V.</td>
</tr>
<tr>
<td>Figuur 2a en 2b</td>
<td>Tabel en figuur van de globale werking van een AngularJS-applicatie</td>
</tr>
<tr>
<td>Figuur 3</td>
<td>Diagram van de mogelijkheden van het Spring-framework</td>
</tr>
<tr>
<td>Figuur 4</td>
<td>Onduidelijke rapportages in het systeem</td>
</tr>
<tr>
<td>Figuur 5</td>
<td>Gedeelde klasseendiagrammen oude situatie workflows en notificaties</td>
</tr>
<tr>
<td>Figuur 6</td>
<td>Voorbeeld wireframe workflow-module</td>
</tr>
<tr>
<td>Figuur 7</td>
<td>UML-ontwerp voor het versturen van notificaties</td>
</tr>
<tr>
<td>Figuur 8</td>
<td>Vernieuwd GUI-ontwerp voor het versturen van notificaties</td>
</tr>
<tr>
<td>Figuur 9</td>
<td>Vernieuwd GUI-ontwerp bewerken notificatie-sjabloon</td>
</tr>
<tr>
<td>Figuur 10</td>
<td>Diagram van de werking van het versturen van e-mailnotificaties</td>
</tr>
<tr>
<td>Figuur 11</td>
<td>Globale werking oude notificatie-module AscMe</td>
</tr>
<tr>
<td>Figuur 12</td>
<td>Globale werking nieuwe notificatie-module AscMe</td>
</tr>
<tr>
<td>Figuur 13</td>
<td>Flowchart inloggen met een token</td>
</tr>
<tr>
<td>Figuur 14</td>
<td>Linkable interface voor genereren links naar objecten</td>
</tr>
<tr>
<td>Figuur 15</td>
<td>Schematisch overzicht integratie-unittests en gewone unittests</td>
</tr>
<tr>
<td>Figuur 16</td>
<td>Klasseendiagram instellen workflows per faculteit</td>
</tr>
<tr>
<td>Figuur 17</td>
<td>GUI-ontwerp instellen workflows per faculteit</td>
</tr>
<tr>
<td>Figuur 18</td>
<td>Database-ontwerp instellen workflows per faculteit</td>
</tr>
<tr>
<td>Figuur 19</td>
<td>Modelleren van een samengestelde primaire sleutel in Hibernate</td>
</tr>
<tr>
<td>Figuur 20</td>
<td>Dammodel voor plannen notificaties per workflow (per faculteit)</td>
</tr>
<tr>
<td>Figuur 21</td>
<td>Klasseendiagram plannen notificaties per workflow (per faculteit)</td>
</tr>
<tr>
<td>Figuur 22</td>
<td>Gebruik van Java-annotaties voor statiche gegevens vernieuwde notificatie-events</td>
</tr>
<tr>
<td>Figuur 23</td>
<td>Klasseendiagram vernieuwde notificatie-events</td>
</tr>
<tr>
<td>Figuur 24</td>
<td>Oude situatie ophalen workflows voor het dashboard</td>
</tr>
<tr>
<td>Figuur 25</td>
<td>Nieuwe situatie ophalen workflows voor het dashboard</td>
</tr>
<tr>
<td>Figuur 26</td>
<td>Ophalen benodigde database-objecten voor caching of workflows</td>
</tr>
<tr>
<td>Figuur 27</td>
<td>Filteren aanwezige database-objecten om workflows te cachen</td>
</tr>
<tr>
<td>Figuur 28</td>
<td>Grafiek met performance-verbetering na herschrijven code cachen workflows</td>
</tr>
<tr>
<td>Figuur 29</td>
<td>Klasseendiagram aangepaste workflow-pagina’s</td>
</tr>
<tr>
<td>Figuur 30</td>
<td>GUI-ontwerp bewerken van een aangepaste workflow-pagina</td>
</tr>
<tr>
<td>Figuur 31</td>
<td>Schematische werking Reflection API van Java</td>
</tr>
<tr>
<td>Figuur 32</td>
<td>Ophalen van een veld uit een entiteit met Reflection</td>
</tr>
<tr>
<td>Figuur 33</td>
<td>Ophalen van de waarde uit een veld met Reflection</td>
</tr>
<tr>
<td>Figuur 34</td>
<td>GUI-ontwerp instellen basisgegevens workflow</td>
</tr>
<tr>
<td>Figuur 35</td>
<td>GUI-ontwerp (optie 1) voor selecteren van workflow-pagina’s</td>
</tr>
<tr>
<td>Figuur 36</td>
<td>GUI-ontwerp (optie 2) voor selecteren van workflow-pagina’s</td>
</tr>
<tr>
<td>Figuur 37</td>
<td>GUI-ontwerp voor het instellen van een gekozen workflow-pagina</td>
</tr>
<tr>
<td>Figuur 38</td>
<td>GUI-ontwerp beheerpagina aangepaste workflow-pagina’s</td>
</tr>
<tr>
<td>Figuur 39</td>
<td>Schematische werking nieuw workflow-paginasysteem (na refactoring)</td>
</tr>
<tr>
<td>Figuur 40</td>
<td>Gedeelten van unittests voor aangepaste workflow-pagina’s</td>
</tr>
<tr>
<td>Figuur 41</td>
<td>UML-ontwerp voor beheerpagina starten / stoppen / herstarten van workflows</td>
</tr>
<tr>
<td>Figuur 42</td>
<td>UML-ontwerp voor rapportagesysteem met vakken</td>
</tr>
<tr>
<td>Figuur 43</td>
<td>GUI-ontwerp beheerpagina voor starten / stoppen / herstarten van workflows</td>
</tr>
<tr>
<td>Figuur 44</td>
<td>Code voor het bepalen van de voortgang van een workflow</td>
</tr>
<tr>
<td>Figuur 45</td>
<td>Tabel met statussen voor opstarten / stoppen / herstarten van workflows</td>
</tr>
<tr>
<td>Figuur 46</td>
<td>Code voor het starten van een workflow, met simulatie-modus</td>
</tr>
<tr>
<td>Figuur 47</td>
<td>Gedeelten van unittests voor het starten van workflows</td>
</tr>
<tr>
<td>Figuur 48</td>
<td>Code voor automatisch opstarten / triggeren van workflows</td>
</tr>
<tr>
<td>Figuur 49</td>
<td>Code voor ophalen alle aanwezige HistoricalCrudServices</td>
</tr>
<tr>
<td>Figuur 50</td>
<td>Code voor opslaan wilkeurige objecten met HistoricalCrudService.</td>
</tr>
</tbody>
</table>
Bijlage 2: Begrippenlijst

Technische begrippen kunnen lastig zijn. Daarom heb ik deze toegelicht. De begrippen staan ook schuin gedrukt wanneer ze in het verslag voor het eerst voorkomen.

<table>
<thead>
<tr>
<th>Begrip</th>
<th>Omschrijving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stakeholder</td>
<td>Belanghebbende binnen een (software-)project.</td>
</tr>
<tr>
<td>Scrum</td>
<td>Iteratieve software-ontwikkelmethode, waarbij in periodes (Sprints) wordt gewerkt aan het opleveren van de te ontwikkelen onderdelen van een product.</td>
</tr>
<tr>
<td>Klasse</td>
<td>Onderdeel van een software-applicatie. Bevat een stukje code voor het uitvoeren van bepaalde activiteiten.</td>
</tr>
<tr>
<td>Library</td>
<td>Softwarepakket waarmee ontwikkelaars extra functionaliteit beschikbaar krijgen. Librariers bieden extra functies aan waar ontwikkelaars gebruik van kunnen maken. In tegenstelling tot frameworks voeren libraries geen taken automatisch uit. Het zijn puur uitbreidingen voor het ontwikkelen van een applicatie (zoals een csv-export, of mooiere invoer-elementen)</td>
</tr>
<tr>
<td>Dependency</td>
<td>Automatisch inladen van klassen wanneer deze nodig zijn. Hierdoor hoeft een ontwikkelaar geen nieuwe instantie van het object aan te maken.</td>
</tr>
<tr>
<td>injection</td>
<td></td>
</tr>
<tr>
<td>REST</td>
<td>Representational state transfer. Eenvoudige manier om gegevens tussen een webserver en een browser uit te wisselen.</td>
</tr>
<tr>
<td>JSON</td>
<td>Formaat voor het encoderen van gegevens in tekst. Kan op deze manier objecten en lijsten met gegevens opslaan.</td>
</tr>
<tr>
<td>Front-end</td>
<td>Voorkant van een applicatie. In de meeste gevallen een website of een GUI-applicatie</td>
</tr>
<tr>
<td>Back-end</td>
<td>Achterkant van een applicatie. Meestal code die op een server of een computer wordt uitgevoerd, maar die voor de gebruikers niet zichtbaar is.</td>
</tr>
<tr>
<td>Merge</td>
<td>Samenvoegen van twee bronnen met code. Meestal het werk van twee collega’s. Conflicten die hierbij ontstaan (merge conflicts) dienen eerst te worden opgelost</td>
</tr>
<tr>
<td>Branch</td>
<td>Afspilting van de basisversie van de applicatie die op versiebeheer staat. Op deze manier kunnen verschillende nieuwe onderdelen gelijktijdig worden ontwikkeld.</td>
</tr>
<tr>
<td>Pull request</td>
<td>Verzoek om code terug te brengen (mergen) naar een andere branch. De reviewer die is toegevoegd voert een code-review uit en geeft het aan als er verbetering nodig is. Pas bij het goedkeuren door de reviewer wordt de code gemerged.</td>
</tr>
<tr>
<td>Continuous</td>
<td>Pakket voor het automatisch laten bouwen en uitrollen van een applicatie naar een server.</td>
</tr>
<tr>
<td>integration</td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td>Het uitbrengen van een nieuwe versie van de applicatie, en het op een server plaatsen ervan.</td>
</tr>
<tr>
<td>Wireframes</td>
<td>Een wireframe is een ruwe schets van een te bouwen onderdeel van de applicatie. Hierop is te zijn waar bijvoorbeeld knoppen, teksten en tekstvakken komen te staan. Het doel is om een stakeholder een globaal beeld te geven van wat hij gaat maken, zodat hij kan zien of dit enigszins aan zijn of haar verwachtingen voldoet.</td>
</tr>
<tr>
<td>Mock-up</td>
<td>Een mock-up is een gedetailleerder ontwerp dan een wireframe. Door zoveel mogelijk bestaande elementen te gebruiken ontstaat een ontwerp wat bijna op de echte applicatie lijkt. Hierdoor kunnen de stakeholders precies zijn wat er gemaakt gaat worden.</td>
</tr>
<tr>
<td>Story Point</td>
<td>Eenheid voor het schatten hoeveel tijd het kost om iets te ontwikkelen</td>
</tr>
<tr>
<td>Backlog</td>
<td>Takenlijst met items die voor een project moeten worden uitgevoerd</td>
</tr>
<tr>
<td>Sprint</td>
<td>Periode van enkele weken waarbinnen een gedeelte van software wordt ontworpen, ontwikkeld en getest.</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modeling Language. “Taal” voor het ontwerpen van systeemcomponenten, zodat de business de architectuur van een systeem kan begrijpen</td>
</tr>
<tr>
<td>Klassendiagram</td>
<td>UML-diagram met daarin de klassen van een applicatie. Bevat ook de verhoudingen en afhankelijkheden tussen klassen.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programmers Interface. Set met beschikbare functies die door een leverancier van een library, framework of applicatie wordt aangeboden. Hiermee kunnen ontwikkelaars de functionaliteit van het betreffende pakket gebruiken.</td>
</tr>
<tr>
<td>Term</td>
<td>Beschrijving</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Event Bus</td>
<td>Principe om gebeurtenissen binnen een applicatie generiek af te handelen. Vanuit allerlei kanten worden events erin gestopt, terwijl deze op een centrale plek worden afgehandeld.</td>
</tr>
<tr>
<td>Interface</td>
<td>In context van programmeren: Set met methodes / functies die door een klasse die deze aanbiedt / implementeert, moeten worden geïmplementeerd.</td>
</tr>
<tr>
<td>Unittest</td>
<td>Code voor het testen van een specifiek onderdeel van de applicatie</td>
</tr>
<tr>
<td>Integratietest</td>
<td>Code voor het testen van de volledige cyclus voor het gebruiken van een onderdeel van de applicatie</td>
</tr>
<tr>
<td>Annotatie (Java)</td>
<td>Beschrijvend element wat op een methode, variabele of klasse kan worden geplaatst. Wordt meestal door een framework uitgelezen om zo extra functionaliteit te bieden.</td>
</tr>
<tr>
<td>Statisch getypeerde taal</td>
<td>Programmeertaal waarbij functies of variabelen alleen kunnen worden aangesproken als het 100% zeker is dat deze bestaan. Hier wordt bij het compileren (compile-time) gecontroleerd. In niet-statisch getypeerde programmeertalen, zoals JavaScript, wordt hier niet op gecontroleerd en ontstaat er pas een foutmelding bij het aanspreken van een niet bestaande functie of variabele (Runtime error)</td>
</tr>
<tr>
<td>EntityReference</td>
<td>Principe wat in de applicatie AscMe wordt gebruikt om verschillende objecten generiek te benaderen. Bevat een combinatie database-id en type object, waarmee elk object uniek identificeerbaar is.</td>
</tr>
<tr>
<td>Reflection</td>
<td>API van de programmeertaal Java om klassen en methoden dynamisch aan te spreken. Kan worden gebruikt om de statische typering van de programmeertaal te omzeilen.</td>
</tr>
<tr>
<td>Constraint violation</td>
<td>Foutmelding van de database die optreedt wanneer gegevens worden opgeslagen op een manier die niet is toegestaan, zoals het leeg laten van een als niet-leeg gemarkeerd veld, of een uniek veld waarin toch een bestaande waarde wordt ingevoerd.</td>
</tr>
<tr>
<td>Decorator</td>
<td>Functie voor het opmaken van een rapportage in de applicatie AscMe.</td>
</tr>
<tr>
<td>Bean</td>
<td>Component wat door een framework wordt geïnstantieerd. Kan ook zelf worden gedeclareerd en vervolgens worden opgeroepen. Is bedoeld om het creëren van nieuwe object-instanties te beperken.</td>
</tr>
</tbody>
</table>
Afstudeerscriptie - Ontwikkelen van een flexibele workflow-module

Bijlage 3: Klassendiagram oude situatie workflows
Bijlage 4: Klassendiagram oude situatie notificaties
Bijlage 5: GUI-ontwerpen gemaakt bij opstart project

In de eerste twee weken heb ik veel GUI-ontwerpen gemaakt. Daarom heb ik deze niet allemaal in het hoofdstuk over deze sprint gezet, maar als bijlage opgenomen. Onder elk ontwerp staat een korte toelichting.

B5.1 Notificaties

*In-app notificaties optie 1: Bovenin de applicatie*

*In-app notificaties optie 2: Als widget op het dashboard (Optie 1 is gekozen)*
Optie voor het versturen van notificaties voor workflows. Later ondergebracht bij de instellingen per faculteit.

Beheerscherm voor notificatie-sjablonen

Bewerken van een notificatie-sjabloon. Doelgroep en rol zijn vervangen voor events.
Handmatige notificaties versturen vanuit e-mailscherm

Versturen van een handmatige notificatie, na klikken op “Verstuur met een sjabloon”
Logboek van verzonden notificaties. Is vrijwel volledig op deze manier geïmplementeerd.

Gebruikersinstellingen voor notificaties. Inschakelen en e-mailadres zijn later verwijderd.
B5.2 Workflows

**Bewerken algemene instellingen workflow**

**Bewerken introductie workflow-pagina**
Bewerken van een workflow-pagina met gegevens

Bewerken van een afrondende workflow-pagina
Beheerscherm voor aangepaste workflow-pagina’s (toen nog: sjablonen gegevens)

Bewerken aangepaste workflow-pagina (toen nog: sjabloon voor gegevens)

Beheren van workflows
Bijlage 6: Vergelijking HTML-editors voor de front-end

Voor het bewerken van notificatie-sjablonen heb ik vergelijkend onderzoek gedaan naar de mogelijkheden van vier bekende tekst-editors die overweg kunnen met HTML (WYSIWYG-editors). Deze editors heb ik op de voor notificatie-sjablonen relevante onderdelen beoordeeld. De uitkomst is in onderstaande tabel zichtbaar.

<table>
<thead>
<tr>
<th>Functie / Editor</th>
<th>TextAngular</th>
<th>Froala</th>
<th>TinyMce</th>
<th>CKEditor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tekst opmaken</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Afbeeldingen invoegen</td>
<td>✔* (1)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Spelling controleren</td>
<td>X</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>HTML-tekst plakken</td>
<td>✔* (2)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Toegankelijkheidsfuncties</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Teksteditor configureerbaar</td>
<td>✔</td>
<td>✔</td>
<td>✔* (3)</td>
<td>✔</td>
</tr>
<tr>
<td>Minimale IE-versie</td>
<td>IE9</td>
<td>IE10</td>
<td>IE 8</td>
<td>IE 10 (4)</td>
</tr>
<tr>
<td>Open source</td>
<td>✔</td>
<td>X</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>AngularJS compatible</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔* (5)</td>
</tr>
<tr>
<td>Prijs</td>
<td>Gratis</td>
<td>$299/jaar</td>
<td>$1/user</td>
<td>$499</td>
</tr>
</tbody>
</table>

Uiteindelijk heb ik, in overleg met het bedrijf, gekozen voor TextAngular. Deze editor was gratis en bood, gezien het beperkte gebruik voor sjablonen, voldoende mogelijkheden.

Opmerkingen:
1 Alleen invoegen via externe link
2 Mogelijk via apart filter voor AngularJS
3 Beperkt mogelijk
4 Internet Explorer 9 wordt beperkt ondersteund
5 Enkel via plug-in van onafhankelijke ontwikkelaar / hobbyist

Toelichting op de velden:

Spelling controleren: Dit omvat een eigen spellingscontrole en suggesties voor het verbeteren van woorden.

HTML-tekst plakken: Hiermee wordt het plakken van html-content, zoals uit een opgemaakt Word-document of website, bedoeld. Wordt dit ondersteund, dan wordt de tekst inclusief opmaak overgenomen, anders wordt het als platte tekst geplakt.

Toegankelijkheidsfuncties: Dit zijn speciale functies voor gebruikers van schermlezers, braille-apparaten etc. Deze kunnen worden geïmplementeerd op basis van de standaarden WAI, WCAG en RFC-508.

Editor configureerbaar: Hiermee wordt bedoeld dat het mogelijk is om een eigen thema en de knoppen van de werkbalk in te stellen. Dit kan door middel van CSS (styling) of Javascript (knoppen inschakelen)
AngularJS: Hiermee wordt bedoeld of er een plug-in of directive voor het framework AngularJS beschikbaar is, zodat de editor volgens de voor AngularJS geldende programmeerconventies gebruikt kan worden. Is dit niet het geval, dan verloopt het gebruik van de editor via plain javascript of JQuery.

Bijlage 7: Code voor importeren van workflows

```java
/*
 * (C) 2017 Vorsen bv (www.vorsen.nl). All rights reserved.
 */
public class ImportWorkflowsTasklet implements Tasklet {

    @Autowired
    private AcademicYearService academicYearService;

    @Autowired
    private StudyService studyService;

    @Autowired
    private ExecutedWorkflowRepository executedWorkflowRepository;

    @Autowired
    private FacultyWorkflowRepository facultyWorkflowRepository;

    @Autowired
    private RelationService relationService;

    @Autowired
    private RelationRepository relationRepository;

    @Autowired
    private EntityService entityService;

    private Map<String, BaseEntity> entityReferencesCache = new HashMap<>();
    private final Logger LOGGER = LoggerFactory.getLogger(ImportWorkflowsTasklet.class);

    @Override
    public RepeatStatus execute(StepContribution contribution, ChunkContext chunkContext) throws Exception {
        cacheAllCommonEntities();
        AcademicYear academicYear = academicYearService.findOne((Long) chunkContext.getStepContext().getJobParameters().get("academicYearId"));

        facultyWorkflowRepository.findAll().forEach(facultyWorkflow ->
            doImport(academicYear, facultyWorkflow, null, chunkContext.getStepContext().getStepExecution());

        return RepeatStatus.FINISHED;
    }

    public ImportWorkflowsResult doImport(AcademicYear academicYear, FacultyWorkflow facultyWorkflow, @Nullable EntityReference reference, @Nullable StepExecution stepExecution) {
        return doImport(academicYear, facultyWorkflow, reference, stepExecution, false);
    }

    /**
     * Import workflows for a faculty.
     * @param facultyWorkflow Faculty settings for the workflow
     * @param reference If supplied, only import the workflow for a single object. Otherwise, import the workflow for all applicable objects.
     * @param stepExecution Spring Batch context. If supplied, batch logging to JobExecutionView is performed. Otherwise, no logging is performed.
     * @return Amount of modified workflow entries. Both enabled and disabled workflows count.
     */
    public ImportWorkflowsResult doImport(AcademicYear academicYear, FacultyWorkflow facultyWorkflow, @Nullable EntityReference reference, @Nullable StepExecution stepExecution,
```
```java
boolean deleteDisabled) {

    // If the workflow is not enabled or has no start and end date set (= enabled for manual run only), skip importing it.
    if (facultyWorkflow == null || (!facultyWorkflow.isEnabled() && !deleteDisabled) ||
        (facultyWorkflow.getStartDate() == null && facultyWorkflow.getEndDate() == null)) {
        return new ImportWorkflowsResult();
    }

    // If the import is called manually, clear the cache in this method. Otherwise, do it outside to prevent unnecessary cache clearing during the import process.
    if (stepExecution == null) {
        cacheAllCommonEntities();
    }

    ImportWorkflowsResult result = new ImportWorkflowsResult();
    List<ExecutedWorkflow> modified = new ArrayList<>();

    LOGGER.debug("Started importing workflow " + facultyWorkflow.getWorkflow().getName() + " for faculty " + facultyWorkflow.getFaculty().getEnglishName());
    LOGGER.debug("Workflow has " + facultyWorkflow.getWorkflow().getRelations().size() + " relations.");

    List<ExecutedWorkflow> executedWorkflows = executedWorkflowRepository.findByFacultyWorkflow(facultyWorkflow);

    facultyWorkflow.getWorkflow().getRelations().forEach(workflowRelation -> {
        LOGGER.debug("Processing workflow relation with ID " + workflowRelation.getId());
        // Find all relations for the current workflow. Relations which are invalid (no target, because deleted) are ignored.
        Set<Relation> relationsForWorkflow = findRelationsForWorkflowRelation(workflowRelation, reference);

        LOGGER.debug("Number of regular relations for workflow relations: " + relationsForWorkflow.size());

        // Obtain the persons from the relations.
        Set<Person> personsWithRelationForWorkflow = findPersonsForRelations(relationsForWorkflow);

        LOGGER.debug("Number of persons with this relation: " + personsWithRelationForWorkflow.size());

        // Loop through the related person.
        personsWithRelationForWorkflow.forEach(person -> {
            LOGGER.debug("Processing person " + person.getExternalId());
            // Find all entities that should be enabled for the current person.
            Set<BaseEntity> entitiesForPerson = findRelatedWorkflowEntitiesForPerson(person, relationsForWorkflow, facultyWorkflow, academicYear);

            LOGGER.debug("Person " + person.getExternalId() + " has " + entitiesForPerson.size() + " related entities");

            List<ExecutedWorkflow> executedWorkflowsForPerson = executedWorkflows.stream()
                .filter(executedWorkflow ->
                    executedWorkflow.getUserName().equals(person.getExternalId()))
                .collect(Collectors.toList());

            // Loop through all the entities for the current person.
            entitiesForPerson.forEach(entity -> {
                // Check if workflow data already exists. Otherwise create new workflow data.
                ExecutedWorkflow executed = executedWorkflowsForPerson.stream()
                    .filter(executedWorkflow ->
                        executedWorkflow.getEntity().equals(entity.toReference()))
                    .findAny()
                    .orElse(new ExecutedWorkflow(facultyWorkflow, entity.toReference(),
                        person.getExternalId(), false));

                // If the workflow is not complete and not invalidated, and the enabled status of the workflow differs from the faculty status, modify the entry.
                if (executed.getCompleted() == null || executed.isInvalidated() ||
                    executed.getEntity().equals(entity.toReference()) &&
                    entity.getEnabled() != facultyWorkflow.getWorkflow().getEnabled())) {
                    LOGGER.debug("Updating workflow entity for person " + person.getExternalId() + " with ID " + entity.getId());
                    executed.setCompleted(true);
                    executed.setInvalidated(true);
                    executed.setEntity(entity.toReference());
                    executed.setWorkflow(facultyWorkflow.getWorkflow());
                    executed.setFaculty(facultyWorkflow.getFaculty());
                    executed.setAcademicYear(facultyWorkflow.getAcademicYear());
                    executed.setExternalId(person.getExternalId());
                    executed.setUserName(person.getUserName());
                    LOGGER.debug("Updated workflow entity for person " + person.getExternalId() + " with ID " + entity.getId());
                    modified.add(executed);
                }
            });
        });
    }

    // If the import is not enabled or has no start and end date set (= enabled for manual run only), skip importing it.
    if (facultyWorkflow == null || (!facultyWorkflow.isEnabled() && !deleteDisabled) ||
        (facultyWorkflow.getStartDate() == null && facultyWorkflow.getEndDate() == null)) {
        return new ImportWorkflowsResult();
    }

    // If the import is called manually, clear the cache in this method. Otherwise, do it outside to prevent unnecessary cache clearing during the import process.
    if (stepExecution == null) {
        cacheAllCommonEntities();
    }

    ImportWorkflowsResult result = new ImportWorkflowsResult();
    List<ExecutedWorkflow> modified = new ArrayList<>();

    LOGGER.debug("Started importing workflow " + facultyWorkflow.getWorkflow().getName() + " for faculty " + facultyWorkflow.getFaculty().getEnglishName());
    LOGGER.debug("Workflow has " + facultyWorkflow.getWorkflow().getRelations().size() + " relations.");

    List<ExecutedWorkflow> executedWorkflows = executedWorkflowRepository.findByFacultyWorkflow(facultyWorkflow);

    facultyWorkflow.getWorkflow().getRelations().forEach(workflowRelation -> {
        LOGGER.debug("Processing workflow relation with ID " + workflowRelation.getId());
        // Find all relations for the current workflow. Relations which are invalid (no target, because deleted) are ignored.
        Set<Relation> relationsForWorkflow = findRelationsForWorkflowRelation(workflowRelation, reference);

        LOGGER.debug("Number of regular relations for workflow relations: " + relationsForWorkflow.size());

        // Obtain the persons from the relations.
        Set<Person> personsWithRelationForWorkflow = findPersonsForRelations(relationsForWorkflow);

        LOGGER.debug("Number of persons with this relation: " + personsWithRelationForWorkflow.size());

        // Loop through the related person.
        personsWithRelationForWorkflow.forEach(person -> {
            LOGGER.debug("Processing person " + person.getExternalId());
            // Find all entities that should be enabled for the current person.
            Set<BaseEntity> entitiesForPerson = findRelatedWorkflowEntitiesForPerson(person, relationsForWorkflow, facultyWorkflow, academicYear);

            LOGGER.debug("Person " + person.getExternalId() + " has " + entitiesForPerson.size() + " related entities");

            List<ExecutedWorkflow> executedWorkflowsForPerson = executedWorkflows.stream()
                .filter(executedWorkflow ->
                    executedWorkflow.getUserName().equals(person.getExternalId()))
                .collect(Collectors.toList());

            // Loop through all the entities for the current person.
            entitiesForPerson.forEach(entity -> {
                // Check if workflow data already exists. Otherwise create new workflow data.
                ExecutedWorkflow executed = executedWorkflowsForPerson.stream()
                    .filter(executedWorkflow ->
                        executedWorkflow.getEntity().equals(entity.toReference()))
                    .findAny()
                    .orElse(new ExecutedWorkflow(facultyWorkflow, entity.toReference(),
                        person.getExternalId(), false));

                // If the workflow is not complete and not invalidated, and the enabled status of the workflow differs from the faculty status, modify the entry.
                if (executed.getCompleted() == null || executed.isInvalidated() ||
                    executed.getEntity().equals(entity.toReference()) &&
                    entity.getEnabled() != facultyWorkflow.getWorkflow().getEnabled())) {
                    LOGGER.debug("Updating workflow entity for person " + person.getExternalId() + " with ID " + entity.getId());
                    executed.setCompleted(true);
                    executed.setInvalidated(true);
                    executed.setEntity(entity.toReference());
                    executed.setWorkflow(facultyWorkflow.getWorkflow());
                    executed.setFaculty(facultyWorkflow.getFaculty());
                    executed.setAcademicYear(facultyWorkflow.getAcademicYear());
                    executed.setExternalId(person.getExternalId());
                    executed.setUserName(person.getUserName());
                    LOGGER.debug("Updated workflow entity for person " + person.getExternalId() + " with ID " + entity.getId());
                    modified.add(executed);
                }
            });
        });
    }
```

executed.isEnabled() != facultyWorkflow.isEnabled())
    {
        boolean enabled = facultyWorkflow.isEnabled();
        executed.setEnabled(enabled);
        String logText = enabled ? "Enabled" : "Disabled";
        BatchLogger.info(person.toReference(), logText + " workflow " + getWorkflowName(executed) + "\n" + "+ " + person.getExternalId());
        if (stepExecution != null) {
            stepExecution.setWriteCount(stepExecution.getWriteCount() + 1);
        }
        if (enabled) {
            result.setNumberEnabled(result.getNumberEnabled() + 1);
        } else {
            result.setNumberDisabled(result.getNumberDisabled() + 1);
        }
        modified.add(executed);
    }
    executedWorkflowRepository.save(modified);
    executedWorkflowRepository.flush();
    return result;
}
/**
 * Find all relations for a workflow on a given object
 * @param workflow Workflow to find the relations for
 * @param reference Object to which the relations must be related
 * @return List of all relations for the workflow.
 */
public Map<WorkflowRelation, Set<Relation>> findRelationsForWorkflow(Workflow workflow,
@Nullable EntityReference reference) {
    Map<WorkflowRelation, Set<Relation>> result = new HashMap<>();
    for (WorkflowRelation workflowRelation : workflow.getRelations()) {
        result.put(workflowRelation, findRelationsForWorkflowRelation(workflowRelation,
            reference));
    }
    if (workflow.isGrouped()) {
        result.putAll(findRelationsForGroupedWorkflow(workflow, reference));
    }
    return result;
}
private Map<WorkflowRelation, Set<Relation>> findRelationsForGroupedWorkflow(Workflow
workflow, EntityReference reference) {
    Preconditions.checkState(workflow.isGrouped(), "This function can only be used with
groupable workflows");
    Map<WorkflowRelation, Set<Relation>> result = new HashMap<>();
    BaseEntity entity = findEntityReference(reference);
    for (WorkflowRelation workflowRelation : workflow.getRelations()) {
        // Add the workflow relations for a study in case of a groupable workflow that
        // belongs to a module or module group.
        if (entity instanceof Module || entity instanceof ModuleGroup) {
            List<Relation> allRelationsForStudy = relationService.findAllActiveByEntity(new
                EntityReference(Study.class, ((StudyAware) entity).getStudyId()));
            result.put(workflowRelation, allRelationsForStudy.stream().filter(relation ->
                relation.getRoleId().equals(workflowRelation.getTypeId())).collect(Collectors.toSet()));
        }
    }
}
// If the groupable workflow belongs to a study, add all applicable relations for the faculty it belongs to.

  else if (entity instanceof Study) {
    List<Relation> allRelationsForFaculty = relationService.findAllActiveByEntity(new EntityReference(Faculty.class, (Study) entity).getFacultyId()));
    result.put(workflowRelation, allRelationsForFaculty.stream().filter(relation -> relation.getRoleId().equals(workflowRelation.getTypeId())).collect(Collectors.toSet()));
  }

return result;

/**
 * Find regular relatios for a workflow relation on a given object
 * @param workflowRelation Workflow relation to find relations for
 * @param reference Object to which the relations must be related
 * @return List of all relations for the workflow and the object.
 */
public Set<Relation> findRelationsForWorkflowRelation(WorkflowRelation workflowRelation, @Nullable EntityReference reference) {
  return relationRepository.findAllByRoleId(workflowRelation.getTypeId()).stream()
    // If a reference is supplied, only process the import for relations for this object (a.k.a. enable one workflow for one object)
    .filter(relation -> reference == null || relation.getEntity().equals(reference))
    .filter(relation -> relation.getTarget() != null && relation.getTarget().getId() != null && relation.getTarget().getType() != null)
    // Only relations to persons should be considered. As only humans fill in workflows, relations to other objects do not apply.
    .filter(relation -> relation.getTarget().getRawClass() == Person.class)
    .collect(Collectors.toSet());
}

/**
 * Find all persons for a set of relationships
 * @param relations Relations to find persons for
 * @return Persons
 */
public Set<Person> findPersonsForRelations(Set<Relation> relations) {
  return relations.stream()
    .map(r -> (Person)findEntityReference(r.getTarget()))
    .collect(Collectors.toSet());
}

/**
 * Find all related entities for a person and a workflow
 * @param person Person to find related entities for
 * @param relationsForPerson All relationships of the person
 * @param facultyWorkflow Workflow to find related entities for
 * @param academicYear Academic year to limit relationships to.
 * @return List of all related entities.
 */
public Set<BaseEntity> findRelatedWorkflowEntitiesForPerson(Person person, Set<Relation> relationsForPerson, FacultyWorkflow facultyWorkflow, AcademicYear academicYear) {
  return relationsForPerson.stream()
    .filter(r -> r.getTarget() != null && r.getTarget().equals(person.toReference()))
    .map(rel -> findEntityReference(rel.getEntity()))
    // The related entity must be related to the faculty whose workflow is enabled / disabled.
    .filter(be -> be instanceof FacultyAware && ((FacultyAware)be).getFacultyId() != null && ((FacultyAware)be).getFacultyId().equals(facultyWorkflow.getFaculty().getId()))
    // The related entity must be related to the current academic year. Older entities should never be considered.
    .filter(be -> be instanceof AcademicYearAware && ((AcademicYearAware)be).getAcademicYearId().equals(academicYear.getId()))
    .collect(Collectors.toSet());
/**
 * Helper method to speed up finding entity references by using a cache.
 * @param reference Entity reference to resolve to BaseEntity
 * @return BaseEntity for the given reference. If present in the cache, no database operations will be performed.
 */
private BaseEntity findEntityReference(EntityReference reference) {
    Preconditions.checkNotNull(reference.getId(), "Entity reference can not be null");
    Preconditions.checkNotNull(reference.getType(), "Entity reference can not be null");
    String cacheKey = getCacheKey(reference);
    if(!entityReferencesCache.containsKey(cacheKey)) {
        LOGGER.trace("Caching entity reference " + reference.toString());
        entityReferencesCache.put(cacheKey, entityService.findByReference(reference));
        LOGGER.trace("Retrieving entity reference " + reference.toString());
        return entityReferencesCache.get(cacheKey);
    }
}

/**
 * Get the English name of a workflow to print it in human readable format.
 * @param executedWorkflow Workflow data
 * @return English workflow name, or null if no titles have been set.
 */
private String getWorkflowName(ExecutedWorkflow executedWorkflow) {
    return Text.translate(executedWorkflow.getFacultyWorkflow().getWorkflow().getTitles(), Language.EN);
}

/**
 * Cache all common entities (Faculty, Study, Module, ModuleGroup and Person) to speed up the import process of workflows.
 */
private void cacheAllCommonEntities() {
    clearCache();
    for(RootType rootType : RootType.values()) {
        List<BaseEntity> entities = entityService.findAllAsBaseEntity(rootType.getEntityType());
        for(BaseEntity entity : entities) {
            EntityReference reference = entity.toReference();
            LOGGER.trace("Caching entity reference " + reference.toString());
            entityReferencesCache.put(getCacheKey(reference), entity);
        }
    }
    List<Person> people = entityService.findAll(Person.class);
    for(Person person : people) {
        EntityReference reference = person.toReference();
        LOGGER.trace("Caching entity reference " + reference.toString());
        entityReferencesCache.put(getCacheKey(reference), person);
    }
}

/**
 * Remove every object from the entity cache of this service.
 */
public void clearCache() {
    this.entityReferencesCache = new HashMap<>();
}

/**
 * Generate a cache key for the given object
 * @param reference Object to cache
 * @return Generated cache key (object_type-object_id)
 */
private String getCacheKey(EntityReference reference) {
    return reference.getType() + "=" + reference.getId();
}
Bijlage 8: Aandachtspunten voor ingebruikname

Een aantal onderdelen vereisen de aandacht bij het in gebruik nemen van de vernieuwde applicatie. Deze onderdelen staan in onderstaand overzicht toegelicht.

Notificaties
- Email-gegevens voor het versturen van notificaties moeten in application.properties worden ingesteld.
- Frontend url en CRON voor workflow-notificaties moeten in application.properties worden ingesteld
- Het groeperen (samenvatten tot "U heeft x meldingen") van e-mailberichten is niet geïmplementeerd
- Het vertraagd versturen (1x per week of vaker, afhankelijk van instellingen per gebruiker) van niet-dringende e-mailnotificaties is niet geïmplementeerd
- Bij de gebruikersinstellingen (klikken op avatar, instellingen) staat een optie "Frequentie e-mailmeldingen", deze doet verder niets
- Bij notificatie-sjablonen staat een optie "Dringend", deze doet verder niets.
- Het is nog niet mogelijk om op de e-maillink van een workflow-notificatie te klikken, omdat de front-end code voor het opstarten van een workflow via email (route workflow/<id>/begin) nog niet aanwezig is
- Bij handmatige notificaties is een notificatie-sjabloon nu altijd verplicht. Volgens mij moest je ook handmatig een tekst kunnen invullen…

Importeren van workflows
- CRON voor importeren van workflows moet in application.properties worden ingesteld.
- Het importeren van workflows wordt automatisch opgestart bij het bewerken van relaties en het inschakelen van workflows. Mogelijk moet dit ook op andere plekken gebeuren, of juist niet indien er te veel serverbelasting ontstaat.

Aangepaste workflow-pagina’s
- Route voor aangepaste workflow-pagina moet in application.properties worden gezet (momenteel: workflow.custom-page)
Instellen van workflows
- De workflows moeten nu per faculteit worden ingesteld. Dit moet op de productie-omgeving worden ingericht volgens de wensen van de VU / WU.
- Het database script voor de WU (009_workflow.sql) moet nog worden aangepast aan nieuwe datamodel, anders kan deze niet worden uitgevoerd bij het deployen. Gebruik als voorbeeld 012_workflow_migration.sql van de VU.
- Bij het toevoegen van ingebouwde pagina’s (Planboard etc.) aan een workflow is het niet mogelijk om de standaard weergavemodus en de toegestane weergavemodi (Lezen, bewerken of geschiedenis) van de pagina in te stellen.
- Het is niet mogelijk om na het instellen van een workflow een voorbeeld / preview van de workflow te bekijken.
- Het is niet direct mogelijk om bij een workflow in te stellen of deze afhankelijk is van een andere workflow. Hetzelfde effect kan worden bereikt door het juist instellen van twee workflows: Stel bij workflow 1 de workflow, die van workflow 1 afhankelijk is, in als getriggerde workflow bij het afronden van workflow 1. Vervolgens wordt workflow 2 opgestart als een gebruiker workflow 1 heeft afgerond.
- Bij het verlaten van de wizard voor het instellen van workflows verschijnt altijd de melding dat "aangebrachte wijzigingen verloren gaan", ook al zijn er helemaal geen wijzigingen aangebracht.

Voortgangsoverzicht
- De workflow-kolommen in het voortgangsoverzicht bij de vakken hebben vrij lang nodig om te laden.
- De lijst met beschikbare kolommen in de rapportage springt irritant van links naar rechts, doordat de balk bovenaan het scherm komt te staan als er omlaag wordt gescrolld.
Bijlage 9: Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2017-1.1 (start uiterlijk 6 februari 2017)
Startdatum uitvoering afstudeeropdracht: 6 februari 2017
Inleverdatum afstudeerdossier volgens jaarrooster: 2 juni 2017

Studentnummer: 13052888
Achternaam: Vlek
Voorletters: A.P.G.
Roepnaam: Arjan
Adres: Berglaan 52
Postcode: 2716 ED
Woonplaats: Zoetermeer
Telefoonnummer: 079 351 41 67
Mobiel nummer: 06 429 524 68
Privé emailadres: arjan1995@gmail.com

Opleiding:
Locatie: Zoetermeer
Variant: voltijd

Naam studieloopbaanbegeleider: A.A.A.M. Jacobs
Naam begeleidend examinator: A.A.A.M. Jacobs
Naam tweede examinator: A.A. Nederend

Naam bedrijf: Vorsen B.V.
Afdeling bedrijf: Ontwikkeling
Bezoekadres bedrijf: Koraalrood 33
Postcode bezoekadres: 2718 SB
Postbusnummer: -
Postcode postbusnummer: -
Plaats: Zoetermeer
Telefoon bedrijf: 088 4242 042
Internetsite bedrijf: www.vorsen.nl

Achternaam opdrachtgever: dhr Losekoot
Voorletters opdrachtgever: S
Titulatuur opdrachtgever: Drs.
Functie opdrachtgever: Directeur
Doorkiesnummer opdrachtgever: 06 415 144 30
Email opdrachtgever: steven@vorsen.nl

Achternaam bedrijfsmentor: dhr van Schagen
Voorletters bedrijfsmentor: J
Titulatuur bedrijfsmentor:
Functie bedrijfsmentor: Lead developer
Doorkiesnummer bedrijfsmentor: 088 4242 078
Email bedrijfsmentor: jeroen@vorsen.nl

Doorkiesnummer afstudeerder: -
Functie afstudeerder (deeltijd/duaal): -
Titel afstudeeropdracht:
Ontwikkelen van een flexibele workflowmodule binnen het bestaande systeem AscMe

Opdrachtomschrijving

1. Bedrijf

Vorsen is een IT-bedrijf wat in 2016 is opgericht vanuit 42 B.V. Vorsen richt zich specifiek op het optimaliseren van bedrijfsprocessen binnen het hoger onderwijs. Vorsen (en voorheen 42) hebben twee innovatieve producten ontwikkeld: DiCE en AscMe (voorheen UAS). AscMe is in samenwerking met de Vrije Universiteit Amsterdam ontwikkeld, en DiCE in samenwerking met de Universiteit Wageningen.

De komende jaren richten we ons op de doorontwikkeling van deze twee producten en blijven we op zoek naar nieuwe innovatieve toepassingen binnen de onderwijslogistiek, en op het snijvlak tussen het gebruikersproces en de administratieve processen. Door te starten vanuit de gebruiker creëren wij producten die in eerste instantie de gebruiker echt helpen in het uitvoeren van zijn werk, en als bijvangst maken we ook de onderwijslogistieke processen eenvoudiger te beheren / efficiënter.

Probleemstelling

In de afgelopen twee jaar hebben in eerste instantie 42, en hierna Vorsen een applicatie ontwikkeld voor het hoger onderwijs, voor het ondersteunen van het vaststellen van het onderwijsprogramma van een instelling.

AscMe biedt hiervoor een procesmatige ondersteuning aan de diverse betrokkenen in het proces. Dit omvat o.a. de ondersteuning van de opleidingsverantwoordelijke die de opbouw van een vak moet vaststellen, een examencommissie die advies geeft, de vakcoördinator die de opbouw van het vak voorstelt en de studiegidsteksten invoert. Elk van de betrokkenen wordt ondersteund bij het uitvoeren van hun deeltaak, waarbij AscMe het volledige proces monitort en inzichtelijk maakt. Alle wijzigingen doorlopen een proces van indienen tot af-/goedkeuring, voordat ze effectief worden. In het kader van de accreditatie wordt een volledig digitaal logboek (audit) bijgehouden van alle gemaakte wijzigingen.

AscMe is initieel ontwikkeld voor één onderwijsinstelling, waarbij vanaf het begin rekening gehouden is met het feit dat de applicatie inzetbaar moet zijn voor meerdere onderwijsinstellingen. Echter zijn er hierbij op een aantal plekken shortcuts genomen, en zijn er keuzes gemaakt die tijdens implementatie bij nieuwe onderwijsinstellingen minder gelukkig bleken te zijn.

Deze afstudeeropdracht richt zich op het probleem dat de momenteel aanwezige ondersteunde processen ‘hard geconfigureerd’ zijn binnen het systeem, en dat er meestal geen goede flow tussen de verschillende processen te vinden is. Op sommige van de schermen die bij het uitvoeren van een proces langskomen staat te veel en / of niet relevante informatie.

Wijzigingen in het proces zijn niet te configureren en vragen ontwikkelwerk en een uitrol van een nieuwe versie. In het verlengde hiervan geldt dat de deeltaken binnen een proces idealiter via een Wizard doorlopen moeten kunnen worden, wat nu slechts beperkt het geval is. Ook zitten er nog geen harde deadlines en voortgangsrapportages gekoppeld aan deeltaken, waardoor de eindverantwoordelijke van de instelling het hele proces lastig kan overzien.
2. **Doelstelling van de afstudeeropdracht**  
Deze opdracht heeft vier sub-doelstellingen, die samen tot een flexibeler en intuïtiever systeem moeten leiden.

1. Het moet mogelijk worden, om zonder ontwikkelwerk het systeem voor nieuwe onderwijsinstellingen in te kunnen richten. Dit zal gebeuren door het configureerbaar maken van processen en deeltaken.
2. Het systeem zal voor de bestaande klant (VU) verbeterd worden, zodat het voortaan een logische indeling in processen en flows bevat. De indeling van de processen in flows zal door middel van onderzoek en overleg tot stand komen.
3. Het systeem moet beter kunnen omgaan met (menselijke) fouten die tijdens het uitvoeren van een proces kunnen optreden. Het systeem moet de mogelijkheid bieden om in bepaalde gevallen, naar een vorige fase binnen een proces terug te keren.
4. Het systeem moet aan de eindverantwoordelijke van een proces een overzicht kunnen bieden met daarin de voortgang van alle betrokkenen binnen het proces, en indien de voortgang in gevaar komt moeten de eindverantwoordelijke en de betrokkene hiervan op de hoogte worden gesteld.

3. **Resultaat**

Een beheerpagina voor de functioneel beheerder waarin alle processen en deeltaken van een instelling kunnen worden beheerd. De user interface moet begrijpelijk zijn voor een niet ontwikkelaar (functioneel beheerder) en de verschillende processen van instellingen kunnen ondersteunen. Door het aanbieden van de user interface zijn de instellingen zelf in staat om wijzigingen in hun taken door te voeren en bespaart dat de klant tijd en geld.

Wizards waarin gebruikers de verschillende processen binnen het systeem op een eenvoudige, intuïtieve manier kunnen uitvoeren. Deze wizards tonen enkel voor de gebruiker relevante gegevens / invoervelden en zijn op een logische volgorde ingedeeld. In eerste instantie zullen er wizards voor de processen “Curriculum vaststellen” en “Studiegidsen samenstellen” worden gemaakt. Afhankelijk van de beschikbare tijd kunnen eventueel meer processen worden voorzien van wizards.

Voortgangsoverzichtsscherm voor de eindverantwoordelijke van een proces, bijv. de opleidingsdirecteur die verantwoordelijk is voor het wijzigings- en goedkeurproces van de studiegids teksten. Hierin ziet de directeur alle vakken binnen zijn opleiding en een voortgangsbalk die aangeeft hoeveel de vakcoördinator(en) is in de wijzigingsvoorstel Wizard. Hierdoor krijgt de directeur een duidelijk overzicht van de voortgang en kan hij bijsturen indien de deadline in gevaar dreigt te komen. Ook dit voortgangsoverzicht zal zich in eerste instantie op de processen “Curriculum vaststellen” en “Studiegidsen samenstellen” richten, omdat de opdracht anders te groot wordt.
4. **Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten**

Er is bij het bepalen van de activiteiten van de opdracht uitgegaan van 70 dagen voor de uitvoering (14 weken) en 15 dagen voor het maken van het afstudeerdossier. De ontwikkeling verloopt bij Vorsen via Scrum. Er is voor deze opdracht een oriëntatiefase (sprint-0) van vier weken. Verder zijn er vijf Scrum-sprints van ieder twee weken.

Tijdens de oriëntatie-sprint (sprint-0) zal de nadruk voornamelijk liggen op het opstarten van het traject en op het uitvoeren van onderzoek. Het onderzoek zal bestaan uit praktijkonderzoek om verbeterpunten / aanpassingen binnen de verschillende flows van de ondersteunende processen in kaart te vinden. N.B. de processen zelf liggen vast, enkel de inrichting hiervan kan worden verbeterd.

Er zullen tijdens deze sprint vragensessies en besprekingen met de klant worden gehouden, en hierbij worden er verschillende voorstellen / ontwerpen voor het inrichten van de flows gemaakt. Op deze manier kan de klant de voor hun beste ideeën kiezen, en worden de requirements van het nieuw te bouwen gedeelte vastgelegd. Aan de hand van MoSCoW-prioritering zal worden bepaald aan welke wensen van de klant wel en niet kan worden voldaan.

Tijdens de daadwerkelijke sprints is er steeds ruimte voor het ontwerpen, ontwikkelen en testen van de nieuw te bouwen functionaliteit. Het resultaat zal aan het einde van elke sprint in een demo worden getoond. Hierbij zijn minstens twee van deze demo’s bij de klant (Vrije Universiteit Amsterdam). De precieze werkzaamheden van deze sprints zijn nog niet bekend, want deze hangen af van welke verbeterpunten en aanpassingen er tijdens het onderzoek in de eerste weken worden gevonden.

Op basis van de uitkomst van de demo’s kunnen eventuele wijzigingen en verbeteringen in de volgende sprint worden doorgevoerd, alvorens de volgende fase van ontwerp, ontwikkeling en testen wordt gestart.

**Ontwikkeltools**

Voor het ontwerpen zal onder andere gebruik worden gemaakt van klasendiagrammen. Eventueel worden nieuw te ontwikkelen processen / workflows ook in flowcharts of sequentiagrammen vastgelegd.

Voor het ontwikkelen zal worden gebruikgemaakt van de ontwikkelstraat van 42 / Vorsen: Java met Spring en Hibernate voor de back-end, AngularJS voor de front-end en Postgres als dbms.

Voor het testen zal gebruik worden gemaakt van in ieder geval unit-testing, en zeer waarschijnlijk ook van integratie-testing en end-to-end testing. Unit-tests worden bij applicaties van Vorsen voor zowel de front-end als de back-end geschreven. Integratiets tests vinden meestal in de back-end plaats, en end-to-end tests worden vanuit de front-end gestart, waarbij het hele systeem wordt doorlopen en het resultaat wordt gecontroleerd.
5. **Op te leveren (tussen)producten**

Tijdens deze opdracht zullen er verschillende producten worden opgeleverd. Er is bij deze opdracht een duidelijke verdeling tussen initiële producten en eindproducten aanwezig, omdat bij het gebruik van Scrum er in elke sprint verschillende activiteiten worden uitgevoerd.

De op te leveren producten zijn:

- Na afloop van de oriëntatie-sprint (sprint-0): Plan van Aanpak, Onderzoeksrapport, Requirements-document en gemaakte ontwerpen / mock-ups
- Van alle sprints: Van elke sprint een Sprint-demo presentatie, aantekeningen van de demo met daarin eventuele verbeterpunten en wijzigingen, en de backlog van de betreffende sprint.
- Na afloop van de laatste sprint: Het eindproduct (de nieuw gebouwde module(s) voor het AscMe-systeem). Indien het stabiel genoeg is verklaard om op de productieomgeving te mogen worden gebruikt, kan de klant hier direct over beschikken.

6. **Te demonstreren competenties en wijze waarop**

- **1.2 Voorbereiden en opstarten softwareontwikkeltraject:** Bij deze applicatie zal door middel van vooronderzoek moeten worden ontdekt hoe de klant de nieuw te bouwen module wilt hebben ingericht

- **1.4 Uitvoeren analyse door definitie van requirements:** Om de scope van deze opdracht te beperken, is het noodzakelijk dat er aan de hand van requirements en prioritering volgens MoSCoW wordt bepaald welke wensen van de klant wel of niet in de nieuw te bouwen module worden opgenomen.

- **3.2 Ontwerpen systeemdeel:** Er zal een volledig en gebruikersgericht ontwerp moeten worden gemaakt van de nieuw te bouwen module. De klant gebruikt het systeem AscMe voor alle verantwoordelijke medewerkers, waardoor het van uiterst belang is dat de nieuwe module(s) logisch en gebruiksvriendelijk is / zijn ingericht.

- **3.3 Bouwen applicatie:** De nieuwe functionaliteit zal volledig en op productie-niveau moeten worden ontwikkeld. De applicatie moet stabiel, onderhoudbaar en veilig zijn zodat deze in een echte bedrijfsomgeving kan worden gebruikt.

- **3.5 Uitvoeren van en rapporteren over het testproces:** Omdat dit systeem in een productieomgeving wordt gebruikt is het van groot belang dat het systeem goed wordt getest en dat het zo min mogelijk fouten bevat. Alle nieuw te ontwikkelen back-end en front-end code moet door middel van automatische tests worden gecontroleerd op juistheid, en moet worden bewaakt tegen ongewenste wijzigingen. Door middel van reeds bestaande tools wordt de code coverage van het systeem voortdurend in de gaten gehouden.
Bijlage 10: Plan van Aanpak

B10.1 Inleiding

Hogescholen en universiteiten beschikken over grote hoeveelheden gegevens. Zo beschikken ze over alle studentgegevens, over gegevens van hun medewerkers, en ook over de gegevens van al hun opleidingen. Al deze gegevens worden bewaard in hun Studenten Informatie Systeem (SIS). Dit is een centrale applicatie die door de onderwijsinstelling zelf wordt beheerd. Slechts enkele personen hebben hier toegang tot, zodat zij de gegevens kunnen beheren. Dit systeem is het best te vergelijken met een enorme kaartenbak gevuld met gegevens.

Om een voorbeeld te geven van de omvang van de gegevens: Een universiteit biedt vele opleidingen aan, van Economie tot Rechten en van Wiskunde tot Informatica. Voor elke opleiding die wordt aangeboden, wordt bijgehouden welke vakken er in de opleiding zitten, hoeveel studiepunten elk vak waard is, en uit welke werkvormen (hoorcolleges, werkcolleges, practica etc.) elk vak bestaat. Daarnaast moet er algemene informatie over elk vak beschikbaar zijn, zodat studenten kunnen zien wat ze ervan kunnen verwachten, welke voorkennis vereist is en op welke manieren ze zullen worden getoetst.

Het zal duidelijk zijn dat het beheren van al deze gegevens een hele klus is. In het verleden moesten bijna alle gegevens handmatig worden ingevoerd, wat een tijdrovend en foutgevoelig proces was. Enige jaren geleden is hiervoor een betere oplossing beschikbaar gekomen, in de vorm van de webapplicatie AscMe, die door verschillende personen binnen de onderwijsinstelling kan worden gebruikt. Omdat ook deze webapplicatie nog niet gebruiksvriendelijk genoeg is, zal deze opdracht zich richten op het verder verbeteren van deze applicatie, die de gegevens over de opleidingen zo eenvoudig mogelijk te beheren maakt.
B10.2 Aanleiding en context

Deze opdracht wordt uitgevoerd voor het bedrijf Vorsen B.V. Vorsen B.V. is een in 2015 opgericht IT-bedrijf, wat zich zich op het aanbieden van IT-producten voor de onderwijsmarkt richt.


Hoewel Vorsen B.V. nog een jong bedrijf is, werkt het momenteel hard aan expansie en een eigen omgeving. Zo is het bedrijf momenteel bezig om beide producten bij nieuwe onderwijsinstellingen te installeren, en wordt er steeds meer een eigen huisstijl voor de producten ontwikkeld.

Het product AscMe, waar deze opdracht over gaat, wordt momenteel afgenomen door de Vrije Universiteit Amsterdam (VU). De VU is een grote universiteit: In 2015 studeerden hier 22.889 studenten. De VU heeft vele faculteiten en beschikt ook over een eigen ziekenhuis (VUmc). Daarnaast heeft de VU een eigen IT-afdeling, welke ook een indirecte opdrachtgever voor het product AscMe is. Er zijn bij deze opdracht zowel wensen vanuit Vorsen B.V. als vanuit de VU.
B10.3 Opdrachtgever en contactpersonen

De opdrachtgever van deze opdracht is Steven Losekoot (tel. 06 415 144 30). Hij is directeur van Vorsen B.V. en houdt zich vooral bezig met administratieve zaken, en het winnen van nieuwe klanten voor het bedrijf.

Daarnaast is Jeroen van Schagen (tel. 06 2415 1642) mijn begeleider vanuit het bedrijf. Jeroen is software-ontwikkelaar binnen het bedrijf, en heeft regelmatig contact met de VU voor wensen en verbeteringen binnen de applicatie. Hij zal mij waar nodig de principes en ontwikkelmethoden binnen het bedrijf aanleren, en is beschikbaar voor vragen of onduidelijkheden tijdens het uitvoeren van deze opdracht.

Tot slot is Nanny Jacobs mijn begeleidend examinator vanuit de Haagse Hogeschool. Zij geeft waar nodig begeleiding en advies over het uit te voeren afstudeertraject.
B10.4 Probleemanalyse en probleemstelling

Achtergrond

De applicatie AscMe wordt momenteel door de Vrije Universiteit Amsterdam (VU) gebruikt en biedt ondersteuning voor het vaststellen van het onderwijsprogramma, het roosteren en het beheren van opleidingsgegevens.

AscMe wordt door verschillende personen gebruikt. Voorbeelden van personen die AscMe gebruiken zijn: een opleidingsdirecteur die de opbouw van zijn opleiding(en) moet kunnen vaststellen, de examencommissie die hierbij advies geeft, en de vakcoördinatoren die de opbouw van hun vakken vaststellen en de basisinformatie van hun vakken (ook wel studiegidsteksten genoemd) invoeren.

Elk van deze personen wordt door de applicatie ondersteund bij het uitvoeren van zijn of haar deeltaak(en). Alle gegevens die door een persoon zijn ingevoerd of gewijzigd moeten worden goedgekeurd voordat ze definitief worden opgeslagen. Omdat het belangrijk is om te zien welke persoon welke wijzigingen heeft gemaakt, wordt van alle wijzigingen een digitaal logboek (audit) bijgehouden.

Probleemstelling

Hoewel AscMe actief wordt gebruikt en een hoop functies biedt, is de applicatie nog niet perfect - zeker niet met het oog op de toekomst gericht. Er zijn een aantal problemen in de applicatie aanwezig die in deze opdracht dienen te worden aangepakt. Deze problemen worden hieronder toegelicht.

Probleem 1: Gebrek aan flexibiliteit

De momenteel aanwezige processen zijn ‘hard geconfigureerd’ binnen de applicatie. Wijzigingen binnen (de structuur van) een proces zijn in de huidige versie van de applicatie niet in te stellen. Hiervoor moet de code worden aangepast en moet er een nieuwe versie worden getest en uitgerold. Dit maakt dat de applicatie relatief veel onderhoud vergt, en dat de afnemer regelmatig beroep op de ontwikkelaars moet doen om wijzigingen binnen hun processen doorgevoerd te krijgen. Dit probleem wordt alleen maar groter indien de applicatie naar verschillende onderwijsinstellingen wordt uitgerold, omdat deze andere bedrijfsprocessen hanteren. Het is onmogelijk om alle processen van verschillende onderwijsinstellingen in dezelfde codebase op te nemen, omdat deze met elkaar conflicteren en de applicatie slecht onderhoudbaar wordt.

Daarnaast is het momenteel onmogelijk om verschillende workflows per faculteit / opleiding tegelijkertijd uit te voeren, omdat elke opleiding / faculteit in de applicatie is gekoppeld aan één actieve status. De momenteel aanwezige processen zijn bovendien gekoppeld aan één begin- en eindstatus. Er zijn bij de huidige afnemer reeds problemen door deze implementatie van workflows, want de VU wil op sommige momenten eigenlijk twee workflows tegelijk uit kunnen voeren terwijl dit niet kan.

Probleem 2: Gebrek aan voortgangsoverzichten

Het is voor gebruikers met een leidinggevende rol onvoldoende mogelijk om de voortgang van hun processen te monitoren. Zo wil de portefeuillehouder onderwijs kunnen zien hoe het met het vaststellen van het curriculum van zijn faculteit gaat, wil een opleidingsdirecteur kunnen zien hoe het met het
uitvragen van de studiegidsteksten is gesteld, en wil de beheerder academische structuur een globaal overzicht van het gehele proces van het vaststellen van het onderwijsprogramma kunnen zien.

Er is hiervoor momenteel een voortgangsrapportage in de applicatie aanwezig, maar deze zit verstopt per item (faculteit, opleiding) en is niet intuïtief ingericht. Het gevolg is dat deze niet goed te vinden is en dat deze niet wordt gebruikt. Het moet voor alle rollen bij het inloggen in de applicatie direct duidelijk zijn hoe het met de voortgang van de lopende processen is gesteld.

Probleem 3: De applicatie is niet pro-actief
Er is vanuit de VU aangegeven dat gebruikers van de applicatie vaak vergeten dat zij nog een of meer taken moeten uitvoeren. Dit komt omdat het niet mogelijk is om gebruikers op de hoogte te stellen van nog uit te voeren taken. Ook is het niet mogelijk om een deadline aan een proces te koppelen. Hierdoor moeten de Beheerders Academische Structuur gebruikers aansporen om hun openstaande taken op tijd te voltooien, en moeten zij vervolgens de processen handmatig afsluiten. Dit kost de beheerders onnodig veel tijd. Het is veel beter als de applicatie deze taken van de beheerders kan overnemen.
B10.5 Doelstelling en eindresultaat

Doelstelling

Het uiteindelijke streven van deze opdracht is om een versie van de applicatie op te leveren die onafhankelijk is van de processtructuur van een specifieke onderwijsinstelling. Specifiek gaat het om de volgende verbeteringen:

- Er wordt een dynamische workflow-engine aan de applicatie toegevoegd, waarin elk proces naar wens kan worden ingesteld. Dit is een uiterst ingrijpende wijziging, want de applicatie is groot (momenteel 33,311 regels code), is reeds in gebruik bij een onderwijsinstelling, en zal over niet al te lange termijn door meerdere onderwijsinstellingen worden gebruikt. Deze onderwijsinstellingen hebben verschillende business logica en processen. Bovendien is het lastig om het nieuw te bouwen gedeelte te testen, want vanwege het feit dat het zo dynamisch moet zijn kan niet van te voren worden bepaald welke combinatie van proces-onderdelen binnen een workflow wordt gekozen. Er zal dus een speciale test-suite moeten worden ingericht die de workflow-engine zo goed mogelijk kan testen.

- Er zal een voortgangsrapportage-module worden ontwikkeld, waarbinnen elke eindverantwoordelijke die de applicatie gebruikt een relevant overzicht krijgt van de voortgang van de betrokkenen binnen zijn of haar proces(sen). Voor een opleidingsdirecteur is dit per opleiding, voor een portefeuillehouder of beheerder academische structuur zal een rapportage van elke faculteit worden getoond. De module geeft tevens een goede indicatie van naderende deadlines. Indien de voortgang van één of meer betrokkenen gevaar dreigt te lopen krijgt zowel de eindverantwoordelijke als de betrokkene(n) hierover op tijd een melding via het notificatiesysteem (zie hieronder). De rapportage-module dient om te kunnen gaan met de dynamische workflow-engine, en moet per ingestelde workflow een op maat gemaakte rapportage kunnen tonen.

- Er zal een nieuw notificatiesysteem in de applicatie worden ingebouwd. Hiermee wordt het mogelijk om via e-mail meldingen naar gebruikers van de applicatie te sturen. Ook komt er een notificatie-menu waarin de gebruiker openstaande meldingen kan zien. Bovendien zullen er aan de instellingen van elke workflow opties worden toegevoegd waarmee de beheerder kan instellen of er meldingen verspreid moeten worden, en er kunnen hierbij deadlines worden ingesteld. Er kan worden ingesteld wat er moet gebeuren indien de deadline is verstreken. Denk hierbij aan het automatisch afsluiten van een workflow.
B10.6 Eindresultaten

Gedurende het uitvoeren van het afstudeerproject zullen er verschillende producten worden gemaakt. De op te leveren producten worden hieronder besproken.

Allereerst wordt dit Plan van Aanpak zelf opgeleverd. Dit document dient als basis voor het uitvoeren van het project.

Bij het ontwikkelen van de verschillende doelstellingen van het project zal er van requirements-analyse gebruik worden gemaakt om de wensen in onderdelen op te delen en te prioriteren. Deze requirements zullen in een daarvoor bedoelde specificatie worden vastgelegd. De requirements-documenten zullen tijdens de besprekingen met de afnemer worden voorgelegd, gecontroleerd en aangevuld waar nodig. De uiteindelijke requirements-specificatie wordt op de einddatum van het project aan Vorsen B.V. opgeleverd en is als bijlage in de afstudeerscriptie voor De Haagse Hogeschool beschikbaar.

Daarnaast zullen er door de gekozen Scrum-methodiek van elke sprint de afsluitende presentatie en de daaruit voorkomende backlog worden opgeleverd. Deze onderdelen worden aan het eind van elke sprint (iedere twee weken) aan Vorsen B.V. opgeleverd. De backlog wordt aan het eind van het project als bijlage van de afstudeerscriptie aan De Haagse Hogeschool opgeleverd.

Uiteraard zal de verbeterde applicatie worden opgeleverd. Hiervan wordt de broncode in de laatste week van het project aan Vorsen B.V. opgeleverd.

Omdat dit project een afstudeerproject voor de Haagse Hogeschool is, zal er tijdens het project een afstudeerscriptie worden gemaakt. De uiteindelijke afstudeerscriptie zal op de einddatum van het project worden aan zowel Vorsen B.V. als De Haagse Hogeschool worden opgeleverd.
B10.7 Randvoorwaarden en risicoanalyse

Scope

Voor het uitvoeren van het project zijn in totaal 17 werkweken van elk 5 werkdagen beschikbaar. Het project is gestart op 6 februari 2017, en eindigt op 2 juni 2017. Dit betekent dat alle uit te voeren activiteiten binnen deze periode plaatsvinden.

Omdat de hoofdactiviteit, het maken van de dynamische workflow-engine, grote gevolgen heeft voor de werking van de applicatie en de datastructuur, zal de engine in eerste instantie worden beperkt tot het generiek maken van het vastleggen en rapporteren over het invoeren van studiegidsteksten. Het lijkt hierdoor misschien dat de te ontwikkelen engine niet dynamisch is, maar ook binnen één proces zijn er tal van configuraties en instellingen mogelijk - wat de engine zeker wel dynamisch maakt.

Afhankelijk van hoeveel tijd dit kost, zal hierna worden afgewogen of er nog tijd beschikbaar is om de dynamische workflow-engine ook voor (één van) de overige processen of zelfs binnen de gehele applicatie te implementeren.

Tot slot zullen, om te voorkomen dat het project te breed wordt, alle overige onderdelen van de applicatie (die niets met workflows te maken hebben) niet worden gewijzigd. Een voorwaarde is wel dat deze onderdelen ondanks de gemaakte wijzigingen blijven werken zoals bedoeld.

Kwaliteit

De kwaliteit van deze afstudeeropdracht zal op meerdere manieren worden gewaarborgd. Er valt hierbij onderscheid te maken in algemene kwaliteitswaarborging en inhoudelijke kwaliteitswaarborging.

Algemene kwaliteitswaarborging

Alle tekstuele producten die zullen worden opgeleverd (scriptie, verslagen, requirements-documenten) zullen door middel van spellingcontrole op taalfouten worden gecontroleerd. Verder zal er bij deze documenten zoveel mogelijk van eenzelfde lay-out gebruik worden gemaakt. Dit betekent dat er op deze documenten een logo van Vorsen B.V. en van de afnemer (VU) aanwezig is, dat er een duidelijke indeling in titels en koppen is, en dat er een inhoudsopgave aanwezig is. PowerPoint-presentaties zullen steeds met hetzelfde thema worden gemaakt, zodat ze een duidelijke, herkenbare structuur hebben. Aantekeningen van de sessies / presentaties zullen aan de betrokkenen worden verspreid, zodat zij de voortgang en de kwaliteit van het project kunnen controleren.
Inhoudelijke kwaliteitswaarborging

De inhoud van de afstudeeropdracht zal op verschillende manieren op kwaliteit worden gecontroleerd. Ik zal dit per product toelichten.

Het requirements-document wordt volgens de algemeen geldende specificaties opgebouwd. Dit betekent dat alle requirements worden voorzien van een uniek nummer, dat het duidelijk is voor welke actor(en) een requirement geldt, dat het duidelijk is waar de requirements uit voortkomen en dat bij elk requirement een prioritering volgens MoSCoW is opgenomen.

De verschillende Scrum-producten (backlog, sprint demo presentaties en verslagen) zullen volgens de geldende regels voor Scrum worden gemaakt. Dit houdt in dat bij alle backlog items duidelijk staat omschreven wat deze inhouden, dat er voor elk item een inschatting in uren (story points) is gemaakt en dat per sprint wordt bepaald wat er wel en niet in de sprint kan worden opgenomen. Scrum-taken zullen vervolgens op het Scrum bord worden geplaatst en lopen tijdens een sprint de algemeen geldende fases (To do, In progress, To Test en Done) door.

In de demo-presentaties zal altijd worden aangegeven welke taken er in de afgelopen sprint zijn uitgevoerd, waar er problemen waren en wat er goed ging. Ook geven deze presentaties een kort vooruitzicht in wat er de komende sprint gepland staat. Op deze manier blijft het duidelijk waar de activiteiten zich op richten en is dit voor het bedrijf en de afnemer te controleren.

Ten slotte de te ontwikkelen verbeteringen aan de applicatie (implementatie en programmacode). De op te leveren verbeteringen aan de applicatie zullen uitvoerig moeten worden getest en op fouten moeten worden gecontroleerd. Allereerst zal er hierbij aan de afspraken die Vorsen B.V. voor het schrijven van programmacode heeft gemaakt, moeten worden gehouden. Onderdeel van deze afspraken is dat alle code volledig wordt getest door middel van unittests, en dat grotere modules (waar de workflow-engine, de rapportage-module en notificaties onder vallen) ook door integratietests of test-suites zullen moeten worden gecontroleerd. Daarnaast zal de geschreven programmacode door middel van peer-reviewing (Pull requests) door collega's worden gecontroleerd. Waar nodig zal er door deze personen feedback of commentaar op de gemaakte code worden gegeven. Deze verbeterpunten zullen dan direct worden verwerkt, zodat de code kwalitatief goedgekeurd is voordat deze in de bestaande applicatie wordt opgenomen.
Risico's

Er zijn tijdens dit project een aantal risico's. Deze verschillen in waarschijnlijkheid en hebben een verschillende mate van impact. De risico's van het project zijn in hieronder opgenomen.

Onderstaande tabel toont de legenda van de klassen in de gemaakte risico-overzichten.

<table>
<thead>
<tr>
<th>Effect-klasse</th>
<th>Bijbehorend effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tijd</td>
<td>Kwaliteit</td>
</tr>
<tr>
<td>Klein</td>
<td>Minder dan anderhalve week vertraging</td>
</tr>
<tr>
<td>Middel</td>
<td>anderhalf tot vier weken vertraging</td>
</tr>
<tr>
<td>Groot</td>
<td>Meer dan vier weken vertraging</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Risico</th>
<th>Onvoldoende terugkoppeling vanuit de stakeholders van de VU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Omschrijving</td>
<td>Er bestaat de kans dat stakeholders van de VU onvoldoende beschikbaar zijn voor het leveren van terugkoppeling over de aangebrachte verbeteringen in de applicatie</td>
</tr>
<tr>
<td>Gevolgen</td>
<td>Indien dit voorkomt, is het voor mij niet duidelijk of een aangebrachte verbetering aan de wensen van de verschillende stakeholders voldoet. Hierdoor dient een aangebrachte verbetering later alsnog verbeterd te worden om aan de wensen te voldoen. Dit kost extra tijd.</td>
</tr>
<tr>
<td>Kans</td>
<td>Middel. De VU is een grote organisatie, en helaas hebben veel mensen het tegenwoordig druk.</td>
</tr>
<tr>
<td>Impact</td>
<td>Middel. Afhankelijk van de mate waarin dit voorkomt, kan een substantiële vertraging optreden.</td>
</tr>
<tr>
<td>Risiconiveau</td>
<td>Middel</td>
</tr>
<tr>
<td>Oplossing / preventie</td>
<td>Stakeholders zullen tijdig moeten worden uitgenodigd voor presentaties en meetings. Indien een stakeholder niet kan, moet een andere stakeholder met dezelfde rol binnen de applicatie worden uitgenodigd. Idealiter zijn bij elke presentatie ten minste één Beheerder Academische Structuur, opleidingsdirecteur en vakcoördinator aanwezig.</td>
</tr>
<tr>
<td>Risico</td>
<td>Bugs in de applicatie</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Omschrijving</td>
<td>Er blijkt een bug in de bestaande applicatie te zitten, waardoor een workflow niet of niet goed kan worden doorgelopen.</td>
</tr>
<tr>
<td>Gevolgen</td>
<td>Tijdens het ontwikkelen van de workflow-engine dient de verbeterde applicatie te worden getoond, waarbij workflows worden doorgelopen. Door de bug in de workflow kan dit niet.</td>
</tr>
<tr>
<td>Kans</td>
<td>Groot: AscMe is een grote applicatie, en er werden vlak voor de afstudeeropdracht begon regelmatig bugs gevonden.</td>
</tr>
<tr>
<td>Impact</td>
<td>Klein: Het kost in de meeste gevallen slechts een kleine hoeveelheid tijd om de bug op te lossen. Vervolgens kan de het werk aan de workflow-engine worden hervat.</td>
</tr>
<tr>
<td>Risiconiveau</td>
<td>Klein, vanwege kleine impact.</td>
</tr>
<tr>
<td>Oplossing / preventie</td>
<td>Als er een bug in een workflow wordt gevonden, zal deze moeten worden opgelost voordat de workflow kan worden getoond. Bugs die door mij in de applicatie worden gevonden, maar geen betrekking op de workflow hebben, worden door een collega opgelost omdat deze buiten de scope van de opdracht vallen.</td>
</tr>
</tbody>
</table>
B10.8 Aanpak en planning

Aanpak

Vorsen B.V. werkt met de software-ontwikkelmethode Scrum. Hierbij wordt er op iteratieve wijze (in Sprints van twee weken) aan de opdracht wordt gewerkt. Er is voor deze opdracht een oriëntatiefase (sprint-0) van vier weken toegewezen. Vervolgens zijn er vijf Scrum-sprints van ieder twee weken. In dit hoofdstuk staat de planning van de uit te voeren activiteiten.

Planning

Tijdens de oriëntatie-sprint (sprint-0) zal de nadruk liggen op het voorbereiden en het opstarten van het software-ontwikkeltraject en het uitvoeren van een analyse door het verkrijgen van de requirements. Tijdens deze sprint zullen er de volgende werkzaamheden / activiteiten plaatsvinden:
- Verkennen van de bestaande applicatie door middel van het navigeren door de user-interface, inzicht in relevante delen van de code, doorlezen van de relevante documentatie en het maken van klassesdiagrammen van relevante delen van de huidige situatie.
- Het maken van een voorstel en ontwerpen voor de dynamische workflow-engine. Dit omvat het maken van wireframes en een presentatie.
- Het houden van één of meer besprekingen bij de VU, waarbij de gemaakte voorstellingen en de hieronder vermelde roadmap worden aangedragen.
- Het uitwerken van de uitkomsten van deze besprekingen, waarbij steeds de voortgekomen requirements in kaart worden gebracht en op basis hiervan de analyse kan worden uitgebreid en verbeterde voorstellen kunnen worden gedaan.

Nadat de oriëntatie-sprint is afgerond, is het de bedoeling dat de wensen in kaart zijn gebracht en dat er op basis van de gemaakte prioritering van de requirements een backlog kan worden opgesteld. Vanwege het iteratieve karakter van Scrum kan er geen volledige planning worden gemaakt, maar er is op basis van de opdrachtomschrijving wel mogelijk om een globale roadmap van uit te voeren taken te maken. Deze staat hieronder.

Sprint 1: **Vereiste onderdelen voor de workflow-engine.** Uit de eerste bespreking met de VU is al gebleken dat ondersteuning voor notificaties eerst moeten worden ontwikkeld. Dit onderdeel wordt dus in de eerste sprint ontwopen, ontwikkeld en getest.

Sprint 2: **Workflow-engine.** Ontwerpen van klassesdiagrammen van te wijzigen onderdelen, bouwen initiele versie workflow-engine en testen van gebouwde workflow-engine

Sprint 3: **Uitbreiden workflow-engine.** Verwerken feedback vorige sprint, aanvullen ontwerpen workflow-engine, bouwen uitgebreidere versie workflow-engine en testen hiervan

Sprint 4: **Afronden workflow-engine en starten rapportage-module.** Verwerken feedback vorige sprint, starten met ontwerpen van de rapportagemodule, starten met bouwen van rapportage-module en testen initiële versie rapportagemodule
Sprint 5: **Afronden Rapportage-module en overige kleine verbeteringen.** Verwerken feedback vorige sprint, verbeteren en testen van de rapportagemodule, ontwerpen wizards voor ontbrekende workflows, bouwen wizards die nog ontbreken, testen wizards die nog ontbreken.

Sprint 6: **Afronden project.** Er zal worden gecontroleerd of de opgeleverde applicatie klaar is om in de productieomgeving van de VU te worden gebruikt. Afhankelijk van de kwaliteit en de bruikbaarheid zal er al dan niet direct een release worden uitgevoerd. Indien het systeem nog niet geschikt is voor gebruik in een productieomgeving, wordt een lijst van nog uit te voeren taken opgesteld, zodat Vorsen B.V. hiermee aan de slag kan om het nieuw gebouwde systeem alsnog geschikt voor productie te krijgen.
Bijlage 11: Requirements document

Opmerkingen:
Beheren is aanmaken, bekijken, wijzigen en verwijderen (CRUD).
Gebruiker betekent in principe iedereen (dus BAS, opleidingsdirecteur, vakcoördinator etc.)

B11.1 Business rules voor de VU

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Oorsprong</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR-001</td>
<td>Meldingen mogen slechts op één manier verspreid worden, namelijk per e-mail aan de betrokkene.</td>
<td>Meeting VU 9 februari 2017</td>
</tr>
<tr>
<td>BR-002</td>
<td>Het roosteren van een opleiding mag enkel beginnen indien het curriculum van de betreffende opleiding is goedgekeurd</td>
<td>Meeting VU 9 februari 2017</td>
</tr>
<tr>
<td>BR-003</td>
<td>Het uitvragen van de studiegidsteksten van een opleiding mag enkel beginnen indien het curriculum van de betreffende opleiding is goedgekeurd</td>
<td>Meeting VU 9 februari 2017</td>
</tr>
</tbody>
</table>
# B11.2 Functionele requirements

## Workflows

### Workflows per faculteit instelbaar

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-F001</td>
<td>Als beheerder van een faculteit wil ik kunnen instellen welke workflows er voor mijn faculteit zijn ingeschakeld. Dit doe ik door een workflow in- of uit te schakelen.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 1</td>
<td></td>
</tr>
<tr>
<td>WFE-F002</td>
<td>Als beheerder van een faculteit wil ik de startdatum van een workflow voor mijn faculteit kunnen instellen</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 1</td>
<td></td>
</tr>
<tr>
<td>WFE-F003</td>
<td>Als beheerder van een faculteit wil ik de einddatum van een workflow voor mijn faculteit kunnen instellen</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 1</td>
<td></td>
</tr>
<tr>
<td>WFE-F004</td>
<td>Als beheerder van een faculteit wil ik kunnen instellen, of er bij het succesvol afronden van een workflow een andere workflow moet worden gestart</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 1</td>
<td></td>
</tr>
<tr>
<td>WFE-F005</td>
<td>Als beheerder van een faculteit wil ik kunnen instellen, of er bij het afkeuren van een workflow een andere workflow moet worden gestart</td>
<td>Must Have</td>
<td>Door opdrachtgever voorgesteld tijdens demo van sprint 2</td>
<td></td>
</tr>
<tr>
<td>WFE-F006</td>
<td>Als beheerder van een faculteit wil ik kunnen instellen, of op een in te stellen aantal dagen voor of na de startdatum, einddatum, bij goedkeuren of bij afkeuren van een workflow een melding moet worden verstuurd.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 1</td>
<td>Heeft lagere prio gekregen nadat het in de derde sprint niet is afgerond.</td>
</tr>
<tr>
<td>WFE-F008</td>
<td>Als systeembeheerder wil ik dat de ingeschakelde workflows elke nacht in een cache-tabel worden opgeslagen, zodat het dashboard sneller kan laden. Bovendien moet dit cache-proces handmatig (via het importscherm) kunnen worden opgestart</td>
<td>Must Have</td>
<td>Bespreking tijdens sprint 3</td>
<td>Komt voort uit performance problemen bij het laden van het dashboard, nadat de workflows per faculteit instelbaar waren gemaakt.</td>
</tr>
<tr>
<td>WFE-F007</td>
<td>Als beheerder van een faculteit wil ik kunnen instellen, of een nog niet-afgeronde workflow op de einddatum moet worden afgerond.</td>
<td>Could Have</td>
<td>Bespreking na de demo van sprint 1</td>
<td>Afgewezen tijdens sprint 3: Dit gebeurt vanzelf en is dus geen requirement.</td>
</tr>
</tbody>
</table>
### Beheren workflows in de applicatie

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-B001</td>
<td>Als beheerder wil ik een overzichtspagina waarop ik de workflows kan beheren en uitschakelen</td>
<td>Must Have</td>
<td>Huidige mogelijkheden workflowsysteem (beheren), meeting VU 9 februari 2017 (uitschakelen)</td>
<td></td>
</tr>
<tr>
<td>WFE-B002</td>
<td>Als beheerder wil ik elke workflow door middel van een wizard kunnen instellen.</td>
<td>Must Have</td>
<td>Opdrachtomschrijving</td>
<td></td>
</tr>
</tbody>
</table>

### Requirements voor systeembeheer

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-SB001</td>
<td>Als systeembeheerder wil ik dat er in een logbestand wordt bijgehouden wie er met een workflow is begonnen, wanneer een workflow is afgerond en of er foutcondities tijdens het uitvoeren van de workflow zijn opgetreden</td>
<td>Must Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
</tbody>
</table>
Gebruikersvriendelijke manier voor het instellen van algemene workflow-gegevens

- Er bestond al een scherm om workflow-gegevens in te stellen. Dit scherm wordt echter opnieuw gebouwd, omdat het gebruiksvriendelijker moet zijn en met de aangepaste workflow-pagina’s overweg te kunnen gaan. Zie ook requirement WFE-B002 (Inrichten workflow via wizard)

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-I001</td>
<td>Als beheerder wil ik de titel van een workflow kunnen instellen</td>
<td>Must Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I002</td>
<td>Als beheerder wil ik het objecttype waarop de workflow betrekking heeft kunnen instellen</td>
<td>Must Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td>Objecttypes zijn: Faculteit, Opleiding, Vak en Vakgroep</td>
</tr>
<tr>
<td>WFE-I003</td>
<td>Als beheerder wil ik de betrokkenen van een workflow kunnen beheren</td>
<td>Must Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I004</td>
<td>Als beheerder wil ik de introductietekst van een workflow kunnen instellen</td>
<td>Should Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I005</td>
<td>Als beheerder wil ik de afsluitende tekst van een workflow kunnen instellen</td>
<td>Should Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I006</td>
<td>Als beheerder wil ik bij de workflow een tekst kunnen instellen die wordt getoond bij het plaatsen van de muis op het informatie-icoontje wat op het dashboard wordt getoond.</td>
<td>Should Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I007</td>
<td>Als beheerder wil ik kunnen instellen, of een workflow moet kunnen worden afgekeurd</td>
<td>Should Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I010</td>
<td>Als beheerder wil ik een link naar een handleiding over de workflow kunnen toevoegen, wijzigen of verwijderen.</td>
<td>Could Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I011</td>
<td>Als beheerder wil ik een link naar een video over het gebruik van de workflow kunnen toevoegen, wijzigen of verwijderen</td>
<td>Could Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I012</td>
<td>Als beheerder wil ik de systeemnaam van de workflow kunnen instellen</td>
<td>Could Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td>Afgekeurd tijdens bespreking na sprint demo 4: Moet intern worden geregeld. Niet relevant voor de gebruiker.</td>
</tr>
<tr>
<td>WFE-I013</td>
<td>Als beheerder wil ik bij de workflow de naam van het icoon voor op het dashboard kunnen instellen.</td>
<td>Must Have</td>
<td>Huidige mogelijkheden workflowsysteem</td>
<td></td>
</tr>
<tr>
<td>WFE-I014</td>
<td>Als beheerder wil ik kunnen instellen, of</td>
<td>Could Have</td>
<td>BR-002, BR-003</td>
<td>Out of scope,</td>
</tr>
</tbody>
</table>
het openstellen van een workflow afhankelijk is van een andere workflow

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-P001</td>
<td>Als beheerder wil ik aangepaste workflow-pagina’s vanuit een overzichtspagina kunnen beheren. (Was: Als beheerder wil ik een sjabloon voor een scherm met workflow-gegevens kunnen beheren)</td>
<td>Must Have</td>
<td>Opdrachtomschrijving , Bespreking na de demo van sprint 3</td>
<td>Inzien, aanmaken en verwijderen</td>
</tr>
<tr>
<td>WFE-P002</td>
<td>Als beheerder wil ik een standaard titel aan een aangepaste workflow-pagina kunnen geven.</td>
<td>Must Have</td>
<td>Initieel in een bespreking na de demo van sprint 3. Dat dit een standaard tekst is tijdens een bespreking na de demo van sprint 4</td>
<td>Tekst kan tijdens plaatsen in een workflow voor die workflow worden overschreven</td>
</tr>
<tr>
<td>WFE-P003</td>
<td>Als beheerder wil ik een standaard uitlegtekst aan een aangepaste workflow-pagina kunnen toevoegen</td>
<td>Must Have</td>
<td>Initieel in een bespreking na de demo van sprint 3. Dat dit een standaard tekst is tijdens een bespreking na de demo van sprint 4</td>
<td>Tekst kan tijdens plaatsen in een workflow voor die workflow worden overschreven</td>
</tr>
<tr>
<td>WFE-P004</td>
<td>Als beheerder wil ik bij het bewerken van een aangepaste workflow-pagina kunnen instellen op welk type objecten deze betrekking heeft. Het gekozen objecttype bepaalt welke gegevens er op de pagina kunnen worden geplaatst, en in welke workflows de pagina kan worden geplaatst. Het objecttype mag alleen tijdens het nieuw aanmaken van de pagina worden ingesteld.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 3, Instelbaar op zelfde pagina tijdens bespreking na demo van sprint 4 (Jeroen)</td>
<td>Beschikbare objecten zijn: Faculteit, Opleiding, Module en Modulegroep.</td>
</tr>
<tr>
<td>WFE-P006</td>
<td>Als beheerder wil ik een aangepaste workflow-pagina door middel van slepen en neerzetten (drag&amp;drop) kunnen vullen met relevante invoervelden. (Was oorspronkelijk:</td>
<td>Must Have</td>
<td>Opdrachtomschrijving, Opnieuw geformuleerd a.d.h.v. bespreking na de demo van sprint 3</td>
<td>Relevantie komt voort uit het type van de pagina. De lijst met in te stellen dynamische invoervelden is in de applicatie ingebouwd.</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>WFE-P007</td>
<td>Als beheerder wil ik de volgorde van de op een aangepaste workflow-pagina gekozen items door middel van slepen &amp; neerzetten (drag&amp;drop) kunnen wijzigen.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 3</td>
<td></td>
</tr>
<tr>
<td>WFE-P008</td>
<td>Als gebruiker wil ik een workflowscherm dat uit een aangepaste workflow-pagina bestaat als onderdeel van mijn workflow kunnen invullen. Wijzigingen die ik aanbreng worden vervolgens in het wijzigingenoverzicht van het betreffende onderdeel weergegeven.</td>
<td>Must Have</td>
<td>Opdrachtomschrijving, bespreking na de demo van sprint 3</td>
<td></td>
</tr>
<tr>
<td>WFE-P009</td>
<td>Als beheerder wil ik dat er bij het opslaan van een aangepaste workflow-pagina één en slechts één digitale wijziging (audit) in het systeem wordt bijgehouden.</td>
<td>Should Have</td>
<td>Bespreking na de demo van sprint 3, bespreking na de demo van sprint 4 (slechts één wijziging aanmaken)</td>
<td>Digitale wijziging is nodig voor exporteren en eventueel terugdraaien als een fout is gemaakt.</td>
</tr>
<tr>
<td>WFE-P010</td>
<td>Als beheerder wil ik per gekozen invoerveld kunnen instellen of het bewerkbaar is</td>
<td>Could Have</td>
<td>Bespreking na de demo van sprint 4</td>
<td></td>
</tr>
<tr>
<td>WFE-P011</td>
<td>Als beheerder wil ik bij het bewerken van een aangepaste workflow-pagina dat reeds gekozen velden niet langer in de lijst met beschikbare velden worden getoond.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 4</td>
<td></td>
</tr>
</tbody>
</table>
Samenstellen van een workflow door het kiezen en configureren van pagina’s

Gebruikte termen:
- Ingebouwde workflow-pagina: Dit is een pagina die op maat is gebouwd en in de code van de applicatie is verwerkt. Deze pagina is slechts beperkt te configureren. Deze pagina’s bestonden reeds bij aanvang van de afstudeerperiode.
- Aangepaste workflow-pagina: Dit is een pagina die door de beheerder is samengesteld. De inhoud van deze pagina is door de beheerder te configureren. Bij het in de workflow plaatsen zijn een beperkt aantal opties configureerbaar.

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-S001</td>
<td>Als beheerder wil ik pagina’s aan een workflow kunnen toevoegen of verwijderen. De beschikbare en de gekozen pagina’s moeten in een overzicht worden getoond. (Was: Als beheerder wil ik de op een workflowscherm te tonen gegevens, afkomstig uit een sjabloon of als kant en klaar component, kunnen kiezen)</td>
<td>Must Have</td>
<td>Opdrachtdoel, Bespreking na de demo van sprint 4</td>
<td></td>
</tr>
<tr>
<td>WFE-S002</td>
<td>Als beheerder wil ik de volgorde van gekozen pagina’s in een workflow kunnen wijzigen</td>
<td>Must Have</td>
<td>Sprint 0, Bespreking na de demo van sprint 4</td>
<td></td>
</tr>
<tr>
<td>WFE-S003</td>
<td>Als beheerder wil ik bij het in een workflow plaatsen van een ingebouwde workflow-pagina kunnen instellen welke weergave-modi (alleen-lezen, bewerken of wijzigingen bekijken) zijn toegestaan</td>
<td>Could Have</td>
<td>Sprint 0, Bespreking na de demo van sprint 3</td>
<td></td>
</tr>
<tr>
<td>WFE-S004</td>
<td>Als beheerder wil ik bij het in een workflow plaatsen van een ingebouwde workflow-pagina kunnen instellen in welke modus de gegevens bij het openen van het de pagina worden getoond (alleen-lezen, bewerken of wijzigingen bekijken)</td>
<td>Could Have</td>
<td>Sprint 0, Bespreking na de demo van sprint 3</td>
<td>Geldt alleen voor de ingebouwde pagina’s die dit ondersteunen. Alleen toegestane modi mogen worden gekozen.</td>
</tr>
<tr>
<td>WFE-S005</td>
<td>Als beheerder wil ik bij het in een workflow plaatsen van een pagina een andere titel kunnen opgeven. Deze aangepaste titel is alleen van toepassing op de betreffende workflow.</td>
<td>Must Have</td>
<td>Sprint 0, Bespreking na de demo van sprint 4</td>
<td>De standaard titel wordt eerst ingevuld.</td>
</tr>
<tr>
<td>WFE-S006</td>
<td>Als beheerder wil ik bij het in een workflow plaatsen van een pagina een andere uitlegtekst kunnen opgeven. Deze aangepaste uitlegtekst is alleen van toepassing op de betreffende workflow.</td>
<td>Must Have</td>
<td>Sprint 0, Bespreking na de demo van sprint 4</td>
<td>De standaard uitlegtekst wordt eerst ingevuld.</td>
</tr>
</tbody>
</table>
### Voorbeeld / Preview van een ingestelde workflow tonen

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFE-V001</td>
<td>Als beheerder wil ik voor het afronden van het instellen van de workflow een voorbeeld van de workflow kunnen zien, zodat ik kan controleren of de workflow correct is geconfigureerd.</td>
<td>Could Have</td>
<td>Bespreking na de demo van sprint 3</td>
<td>Relevantie wordt bepaald door het objecttype van de workflow.</td>
</tr>
<tr>
<td>WFE-V002</td>
<td>Als beheerder wil ik bij het tonen van het voorbeeld een relevant voorbeeld-item (vak, studie etc.) kunnen kiezen waarmee het voorbeeld van de workflow wordt getoond.</td>
<td>Could Have</td>
<td>Bespreking na de demo van sprint 3</td>
<td></td>
</tr>
</tbody>
</table>
Wijzigingen huidige workflow Uitvragen Studiegids (vanuit de VU aangedragen)

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFSG-001</td>
<td>Als Beheerder Academische Structuur wil ik dat de workflows “Start uitvraag studiegids” en “Afronden uitvraag studiegids” voor alle opleidingen worden uitgeschakeld, omdat deze niet relevant zijn</td>
<td>Must Have</td>
<td>Meeting VU 9 februari</td>
</tr>
<tr>
<td>WFSG-002</td>
<td>Als opleidingscoördinator wil ik op de studiegidspagina de taal voor de studiegidsteksten kunnen instellen</td>
<td>Could Have</td>
<td>Meeting VU 9 februari</td>
</tr>
<tr>
<td>WFSG-003</td>
<td>Als opleidingscoördinator wil ik de studiegidsteksten per vak kunnen controleren, in plaats van per wijziging</td>
<td>Must Have</td>
<td>Meeting VU 9 februari</td>
</tr>
<tr>
<td>WFSG-004</td>
<td>Als opleidingscoördinator wil ik dat de links naar de te controleren studiegidsteksten gegroepeerd per opleiding worden weergegeven op het dashboard</td>
<td>Must Have</td>
<td>Meeting VU 9 februari</td>
</tr>
<tr>
<td>WFSG-005</td>
<td>Als vakcoördinator wil ik dat de links naar de uit te vragen studiegidsteksten gegroepeerd per opleiding worden weergegeven op het dashboard</td>
<td>Must Have</td>
<td>Meeting VU 9 februari</td>
</tr>
</tbody>
</table>

Opmerking: Deze requirements ontstonden tijdens een meeting bij de VU (in de oriëntatiefase) en zijn daarom samen met Jeroen uitgevoerd.
## B11.3 Notificaties

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-001</td>
<td>Als Beheerder Academische Structuur, portefeuillehouder onderwijs en opleidingscoordinator wil ik dat de betrokkenen binnen mijn afdeling door middel van e-mailberichten op de hoogte kunnen worden gesteld van nog uit te voeren taken</td>
<td>Must Have</td>
<td>Vernieuwde opdrachtomschrijving, BR-001 (e-mail)</td>
<td></td>
</tr>
<tr>
<td>N-002</td>
<td>Als gebruiker wil ik kunnen instellen hoe vaak ik volgens schema meldingen per e-mail ontvang</td>
<td>Must Have</td>
<td>Bespreking VU 9 februari</td>
<td></td>
</tr>
<tr>
<td>N-003</td>
<td>Als gebruiker wil ik kunnen instellen naar welk e-mailadres meldingen worden verzonden</td>
<td>Should Have</td>
<td>Opdrachtgever</td>
<td>Afgekeurd tijdens meeting VU op 23 februari: bleek niet gewenst</td>
</tr>
<tr>
<td>N-004</td>
<td>Als gebruiker wil ik kunnen instellen of ik e-mailmeldingen wil ontvangen</td>
<td>Should Have</td>
<td>Opdrachtgever</td>
<td>Afgekeurd tijdens meeting VU op 23 februari: bleek niet gewenst</td>
</tr>
<tr>
<td>N-005</td>
<td>Als gebruiker wil ik kunnen instellen in welke taal ik e-mailmeldingen ontvang.</td>
<td>Should Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
<tr>
<td>N-006</td>
<td>Als gebruiker wil ik indien een e-mailbericht een link bevat, en ik op de link in een e-mailmelding klik, op de bijbehorende pagina terecht komen</td>
<td>Should Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
<tr>
<td>N-007</td>
<td>Als gebruiker wil ik een overzicht van alle meldingen in de applicatie zien</td>
<td>Could Have</td>
<td>Overleg tijdens sprint 0</td>
<td></td>
</tr>
<tr>
<td>N-008</td>
<td>Als gebruiker wil ik in een rode badge kunnen zien hoeveel ongelezen meldingen er nog zijn</td>
<td>Could Have</td>
<td>Overleg tijdens sprint 0</td>
<td></td>
</tr>
<tr>
<td>N-009</td>
<td>Als gebruiker wil ik, indien er een bijbehorende pagina is, dat ik naar de bijbehorende pagina ga als ik op een melding in het meldingenoverzicht klik</td>
<td>Could Have</td>
<td>Overleg tijdens sprint 0</td>
<td></td>
</tr>
<tr>
<td>N-010</td>
<td>Als gebruiker wil ik alle meldingen in het meldingenoverzicht in één keer kunnen wissen</td>
<td>Could Have</td>
<td>Overleg tijdens sprint 0</td>
<td></td>
</tr>
<tr>
<td>N-011</td>
<td>Als beheerder wil ik sjablonen met daarin de titels en de teksten van e-mailmeldingen kunnen beheren, zodat deze op verschillende plekken hergebruikt kunnen worden.</td>
<td>Must Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
<tr>
<td>N-012</td>
<td>Als beheerder wil ik per sjabloon voor meldingen kunnen instellen of een melding direct of volgens schema moet worden</td>
<td>Must Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
<tr>
<td></td>
<td>verzonden</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>--------------------------------------------------------------------------</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>N-013</td>
<td>Als beheerder van een faculteit of een opleiding wil ik vanaf de pagina van de opleiding een handmatige e-mail met behulp van een sjabloon voor meldingen kunnen versturen. De teksten van de e-mail moeten voor het versturen van de e-mail nog aanpasbaar zijn.</td>
<td>Must Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
<tr>
<td>N-014</td>
<td>Als beheerder wil ik bij een workflow kunnen instellen of er automatisch meldingen moeten worden verstuurd</td>
<td>Should Have</td>
<td>Vernieuwde opdrachtomschrijving</td>
<td></td>
</tr>
<tr>
<td>N-015</td>
<td>Als beheer wil ik bij een workflow kunnen instellen hoeveel dagen voor de deadline er een melding moet worden verstuurd</td>
<td>Should Have</td>
<td>Vernieuwde opdrachtomschrijving</td>
<td></td>
</tr>
<tr>
<td>N-016</td>
<td>Als beheerder wil ik per faculteit in kunnen stellen hoe vaak er standaard e-mails worden verzonden</td>
<td>Should Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
<tr>
<td>N-017</td>
<td>Als beheer wil ik dat er een log wordt bijgehouden over verstuurd notificaties, zodat ik kan zien wanneer er notificaties zijn verstuurd, naar wie en of dit is gelukt.</td>
<td>Must Have</td>
<td>Opdrachtgever</td>
<td></td>
</tr>
<tr>
<td>N-018</td>
<td>Als productontwikkelaar wil ik, uit veiligheidsoverwegingen, dat gebruikers slechts zeven dagen de tijd hebben om op een gegenereerde link in een e-mailmelding te klikken.</td>
<td>Must Have*</td>
<td>Jeroen van Schagen, bij de demo van sprint 1</td>
<td></td>
</tr>
</tbody>
</table>

* Was geïmplementeerd, maar is later verwijderd: Links bleken onveilig te zijn. Voortaan eerst inloggen bij klikken op een link, en daarna doorsturen naar de relevante pagina.

| N-019 | Als productontwikkelaar wil ik dat er in de database wordt bijgehouden wanneer er voor het laatst op een gegenereerde link in een e-mailmelding is geklikt. | Won't have | Jeroen van Schagen, bij de demo van sprint 1 |

* Niet nodig: Inloggen is uit veiligheidsoverweging verplicht geworden, hierdoor zijn de verstuurde links niet specifiek voor de gebruiker en kan dit niet worden bijgehouden.
## B11.4 Voortgangsoverzichten

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Prioriteit</th>
<th>Oorsprong</th>
<th>Opmerkingen</th>
</tr>
</thead>
<tbody>
<tr>
<td>VO-001</td>
<td>Als opleidingsdirecteur wil ik een voortgangsoverzicht van de betrokkenen binnen mijn opleiding op het dashboard zien, zodat ik kan zien of zij hun taken tijdig uitvoeren.</td>
<td>Won’t Have</td>
<td>Opdrachtomschrijving</td>
<td>Out of scope</td>
</tr>
<tr>
<td>VO-002</td>
<td>Als portefeuillehouder wil ik een voortgangsoverzicht per opleiding van mijn faculteit op het dashboard zien, waarop ik kan zien of de voortgang van het vaststellen van het onderwijsprogramma in orde is</td>
<td>Won’t Have</td>
<td>Opdrachtomschrijving</td>
<td>Out of scope</td>
</tr>
<tr>
<td>VO-003</td>
<td>Als functioneel beheerder wil ik op de beheerpagina van een opleiding, modulegroep- of vak alle relevante workflows kunnen zien, zodat ik hiervan de voortgang kan zien en acties kan uitvoeren.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 5</td>
<td>Of een workflow mogelijk is wordt bepaald door het objecttype van de workflow.</td>
</tr>
<tr>
<td>VO-004</td>
<td>Als functioneel beheerder wil ik op het scherm VO-003 bij iedere workflow het aantal personen die de workflow moeten uitvoeren kunnen zien.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 5, VO-003</td>
<td></td>
</tr>
<tr>
<td>VO-005</td>
<td>Als functioneel beheerder wil ik op het scherm VO-003 het aantal personen wat de workflow heeft voltooid kunnen zien</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 5, VO-003</td>
<td></td>
</tr>
<tr>
<td>VO-006</td>
<td>Als functioneel beheerder wil ik vanuit het scherm VO-003 een workflow handmatig kunnen opstarten, herstarten of afbreken. Deze actie heeft invloed op alle personen die de workflow moeten uitvoeren</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 5, VO-003</td>
<td></td>
</tr>
<tr>
<td>VO-007</td>
<td>Als functioneel beheerder wil ik bij het handmatig opstarten van een workflow (VO-006) vanuit het scherm VO-003 een einddatum kunnen selecteren. De handmatig opgestarte workflow moet op deze einddatum vanzelf worden afgesloten</td>
<td>Could Have</td>
<td>Bespreking na de demo van sprint 5, VO-003, VO-006</td>
<td></td>
</tr>
<tr>
<td>VO-008</td>
<td>Als eindverantwoordelijke van een faculteit of opleiding wil ik bij het overzicht op de pagina “Vakken” per workflow een kolom kunnen toevoegen. Zo’n kolom bevat de naam van de betrokkene en het tijdstip waarop hij / zij de workflow heeft uitgevoerd. Dit gaat alleen om de laatste betrokkene die de workflow heeft uitgevoerd.</td>
<td>Must Have</td>
<td>Bespreking na de demo van sprint 5</td>
<td></td>
</tr>
</tbody>
</table>
### B11.5 Niet-Functionele requirements

<table>
<thead>
<tr>
<th>#</th>
<th>Omschrijving</th>
<th>Oorsprong</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF-001</td>
<td>Het systeem moet minstens 50 simultane workflow-gebruikers aankunnen</td>
<td>Opdrachtgever</td>
</tr>
<tr>
<td>NF-002</td>
<td>Het systeem moet in één nacht minstens 10.000 e-mails kunnen versturen</td>
<td>Opdrachtgever</td>
</tr>
<tr>
<td>NF-003</td>
<td>Het importeren van workflows naar het dashboard dient de applicatie niet te vertragen</td>
<td>Jeroen van Schagen</td>
</tr>
<tr>
<td>NF-004</td>
<td>Alle functionaliteiten in het systeem dienen compatibel zijn met Internet Explorer 9</td>
<td>Opdrachtgever (vanwege de VU)</td>
</tr>
</tbody>
</table>
Bijlage 12: Scrum-backlog, aan het einde van het project

Onderstaande tabel toont de volledige Scrum-backlog. Deze is afkomstig uit JIRA. De issue-nummers zijn interne JIRA-nummers, maar de titels komen grotendeels overeen met de requirements uit het Requirements Document.

Toelichting statussen:
- Closed, fixed: Afgerond
- Closed, duplicate: Was dubbel en heb ik daarom afgesloten
- Closed, won't do / won't fix: Won't have requirements
- Open of Reopened, unresolved: Nog op de backlog. Reopened betekent afgekeurd bij testen.

<table>
<thead>
<tr>
<th>Issue</th>
<th>Omschrijving</th>
<th>Status</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>UAS-799</td>
<td>Afbeelding / icon introductie workflow instelbaar</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-800</td>
<td>Afbeelding / icon afronden workflow instelbaar</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-795</td>
<td>Tone ten beschikbare workflow-componenten</td>
<td>Closed</td>
<td>Duplicate</td>
</tr>
<tr>
<td>UAS-893</td>
<td>Dynamische workflow-pagina</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-794</td>
<td>Instelbaar of workflow kan worden afgekeurd</td>
<td>Reopened</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-792</td>
<td>Instelbaar welke workflow bij het afronden van een workflow wordt opgestart</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-801</td>
<td>Instelbaar welke workflow bij het afkeuren van een workflow wordt opgestart</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-909</td>
<td>Voortgangsoverzicht met workflow-voortgang voor functioneel beheer</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-810</td>
<td>Automatisch versturen meldingen instelbaar per workflow</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-910</td>
<td>Aantal personen die de workflow moeten uitvoeren zichtbaar op voortgangsoverzicht uitvoeren workflow (UAS-909)</td>
<td>Closed</td>
<td>Won't Fix</td>
</tr>
<tr>
<td>UAS-911</td>
<td>Aantal personen die de workflow hebben afgerond zichtbaar op voortgangsoverzicht workflow (UAS-909)</td>
<td>Closed</td>
<td>Won't Fix</td>
</tr>
<tr>
<td>UAS-914</td>
<td>Bugfix voor importeren van workflows</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-912</td>
<td>Workflow starten, herstarten en afbreken vanuit voortgangsoverzicht workflow (UAS-909)</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-913</td>
<td>Kolommen voor workflows in voortgangsoverzicht met vakken</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-894</td>
<td>Volgorde items dynamische workflow-pagina instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-784</td>
<td>Titel van workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-901</td>
<td>Toelichting van workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-786</td>
<td>Introductietekst workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-787</td>
<td>Afsluitende tekst workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-797</td>
<td>Basisinstellingen workflows instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-906</td>
<td>Statustransitie workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-785</td>
<td>Betrokkenen workflow te beheren</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-891</td>
<td>Overzicht met aan de workflow toe te voegen pagina’s</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS</td>
<td>Beschrijving</td>
<td>Status</td>
<td>Status Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------------------------------------------------------------</td>
<td>------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>UAS-889</td>
<td>Pagina aan workflow toevoegen / verwijderen</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-890</td>
<td>Volgorde workflow-pagina's te wijzigen</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-892</td>
<td>Instellen toegestane weergave-modi van een ingebouwde workflow-pagina</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-798</td>
<td>Standaardmodus voor het tonen van gegevens op een ingebouwde workflow-pagina</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-900</td>
<td>Naam van workflow-pagina instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-788</td>
<td>Toelichting op workflowpagina instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-902</td>
<td>Objecttype aangepaste workflow-pagina instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-904</td>
<td>Overzicht met aangepaste workflow-pagina's</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-905</td>
<td>Toevoegen / verwijderen van aangepaste workflow-pagina</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-908</td>
<td>Naam van icoon workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-895</td>
<td>Preview van workflow</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-896</td>
<td>Object voor preview workflow instelbaar</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-802</td>
<td>Logging voor acties op workflows</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-820</td>
<td>Groeperen van notificaties</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-813</td>
<td>Frequentie versturen notificaties per faculteit instelbaar</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-804</td>
<td>E-mailadres voor ontvangen notificaties instelbaar</td>
<td>Closed</td>
<td>Won't Do</td>
</tr>
<tr>
<td>UAS-807</td>
<td>Aantal ongelezen notificaties zichtbaar in applicatie</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-812</td>
<td>Notificatie templates beheren</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-808</td>
<td>Actie notificatie in de applicatie instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-814</td>
<td>Instelbaar of melding regulier of urgent is</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-818</td>
<td>Filter voor versturen notificatie instelbaar</td>
<td>Reopened</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-817</td>
<td>Genereren unieke link met automatische login</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-805</td>
<td>Taal e-mailnotificaties instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-789</td>
<td>Startdatum van workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-790</td>
<td>Einddatum van workflow instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-793</td>
<td>Automatisch afronden op einddatum</td>
<td>Closed</td>
<td>Won't Do</td>
</tr>
<tr>
<td>UAS-811</td>
<td>Automatisch versturen meldingen instelbaar op aantal dagen voor einddatum workflow</td>
<td>Closed</td>
<td>Duplicate</td>
</tr>
<tr>
<td>UAS-816</td>
<td>Notificatie handmatig versturen</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-819</td>
<td>Versturen notificatie bij afkeuren workflow</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-783</td>
<td>Beheerpagina voor workflows per faculteit</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-838</td>
<td>Workflow niet afhankelijk van status maar van start / einddatum</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-770</td>
<td>Notificatie-overzicht in het menu</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-815</td>
<td>Loggen van acties op notificaties</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-806</td>
<td>Actie e-mailnotificatie instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-809</td>
<td>Wissen van alle notificaties in notificatie-overzicht</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-803</td>
<td>Frequentie ontvangen reguliere notificaties instelbaar</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-821</td>
<td>Versturen e-mails vanuit de back-end</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
<tr>
<td>UAS-791</td>
<td>Afhankelijkheden van een workflow instelbaar</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------</td>
<td>-------</td>
<td>-----------</td>
</tr>
<tr>
<td>UAS-766</td>
<td>Voortgangsoverzicht per opleiding</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-767</td>
<td>Voortgangsoverzicht per faculteit</td>
<td>Open</td>
<td>Unresolved</td>
</tr>
<tr>
<td>UAS-775</td>
<td>Begeleiden / goedkeuren workflow per vak</td>
<td>Closed</td>
<td>Fixed</td>
</tr>
</tbody>
</table>
Bijlage 13: Gebruikte referenties


[4] BeanMapper: Library for converting from one Java class to a dissimilar Java class with similar attribute names based on the Bean convention. Opgehaald in mei 201 (http://beanmapper.io/)

