Basis apps

Testen en Verbeteren

hoofdstuk

1

Reflectie en verbetervoorstellen

Algemene informatie

Onderwerp	Verbetervoorstellen
Leerdoel(en)	1. De student kan de uitkomsten van technische testen verwerken in een verbetervoorstel. 2. De student kan de uitkomsten van een acceptatietest verwerken in een verbetervoorstel. 3. De student kan reflectie en feedback verwerken in een verbetervoorstel.
Vereiste voorkennis	Alle opgedane kennis van Thema's 1-4, 7 en 8.
Kwalificatiedossier	☐ B1-K1-W1: Plant werkzaamheden en bewaakt de voortgang ☐ B1-K1-W2: Ontwerpt software ☐ B1-K1-W3: Realiseert (onderdelen van) software ☒ B1-K1-W4: Test software ☒ B1-K1-W5: Doet verbetervoorstellen voor de software ☒ B1-K2-W1: Voert overleg ☐ B1-K2-W2: Presenteert het opgeleverde werk ☒ B1-K2-W3: Reflecteert op het werk

Inhoudsopgave

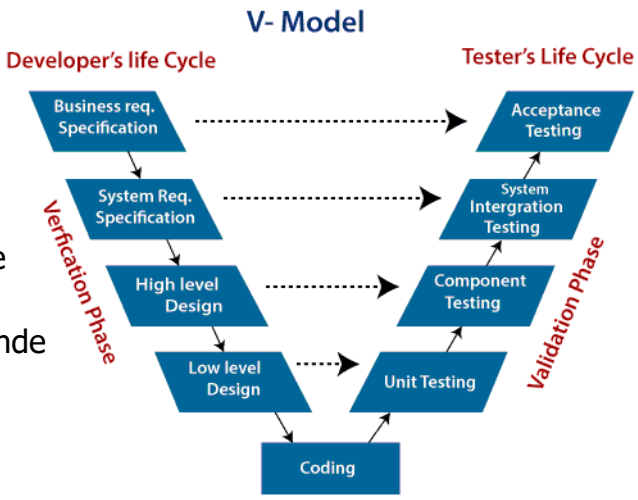
Algemene informatie 2
Inhoudsopgave 3
Introductie 4
Inhoud 4
  Terug naar het V-model 4
  Drie verschillende verbetervoorstellen 4
  Van specifiek naar algemeen 5
    Component en unit 5
    Acceptatietest 6
    Retrospective 9

Introductie

De afgelopen jaren hebben we gemerkt dat er diverse redenen zijn waarom een applicatie zou moeten worden aangepast om anders of beter te functioneren. Drie belangrijke verschillende manieren die in verschillende fasen worden toegepast worden in dit hoofdstuk behandeld.

Inhoud

Terug naar het V-model



Het V-model hebben we al eerder gezien.

Wat heel mooi duidelijk wordt is dat bepaalde fasen in het leven van een developer ook bepaalde soorten testen nodig hebben.
We gaan drie belangrijke testfasen, met bijbehorende verbetervoorstellen, hieronder behandelen.

Je kunt nu **T9_TV_Oefening_TV_1** maken.

Je kunt nu **T9_TV_Oefening_T_2** maken.

Je kunt nu **T9_TV_Oefening_V_2** maken.

Drie verschillende verbetervoorstellen

Wanneer we van specifiek naar algemeen gaan testen, zal elke fase zijn eigen verbetervoorstel hebben. De verbetervoorstellen kunnen namelijk komen vanuit:

1. het (technische of functionele) testen -- tijdens een sprint (iteratie).
2. de oplevering -- na de sprint.
3. de reflectie -- na het afsluiten van een sprint.

De verbetervoorstellen die voortkomen uit de bovenstaande drie fasen zijn:

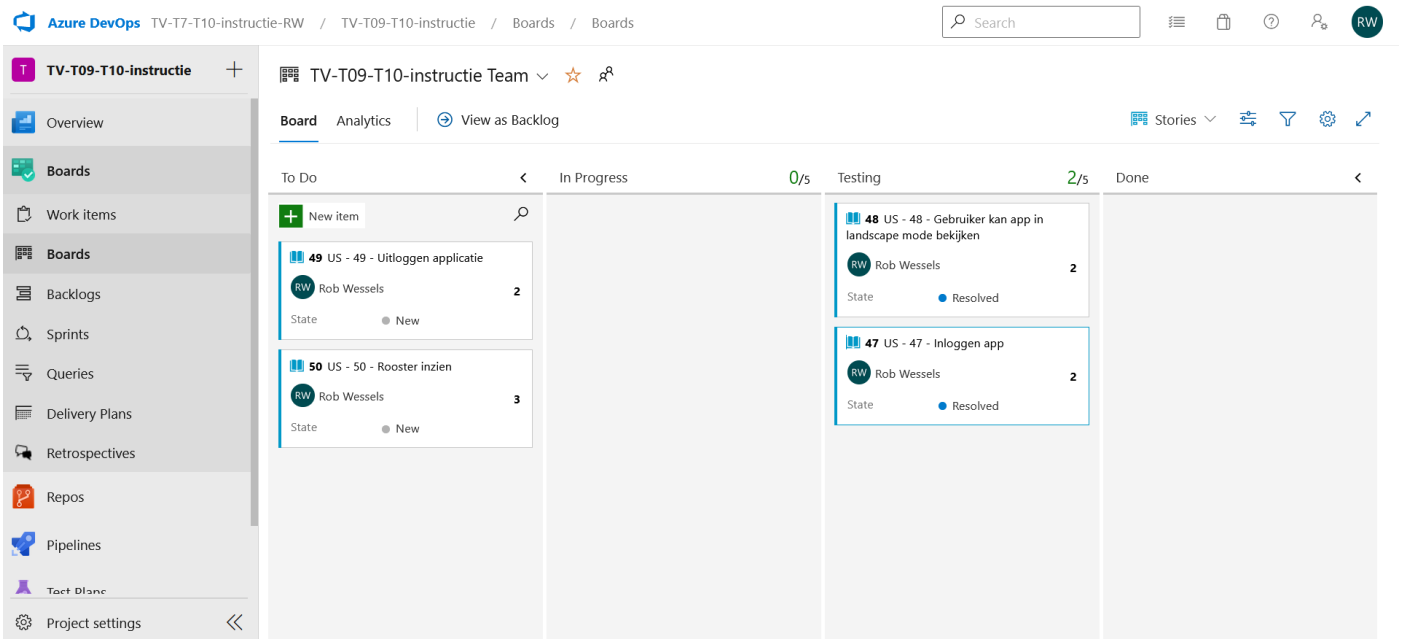
1. Een issue.
2. Een aangepaste userstory en testcases, geplaatst in de volgende sprint.
3. Een retrospective.

Let op: een issue kan ook gemeld worden door de opdrachtgever of een andere gebruiker van de app, nádat de app is gereleased. Dit komt vaker voor maar dit is uiteraard niet wat je wil. Dit kan de reputatie van je app of zelfs je bedrijf schaden. Zorg daarom er voor dat je zélf intern op alle mogelijke manieren je app hebt getest.

Van specifiek naar algemeen

Component en unit

Tijdens en vlak na het programmeren van diverse userstories wordt er doorgaans veel getest of de code goed werkt. Immers programmeert er bijna niemand zonder te kijken of het ook echt werkt.



Wanneer een userstory, zoals hierboven, afgerond is qua programmeren wordt er deze in de Testing-kolom van DevOps geplaatst. Er wordt door de developer een testcase aangemaakt als child van de userstory en toegewezen aan een collega. In de testcase staan testscenario's (normale scenario's en alternatieve scenario's – zoals het afhandelen van verkeerd ingevulde formulieren bijv.) die de tester gaat doornemen om zo te beoordelen of de userstory voldoet aan de acceptance criteria.
Testscenario's van een formulier bevatten ook een dataset. Een testcase bevat correcte, complete data, een ander testcase van dezelfde userstory bevat incorrecte, incomplete data (zie reader T9_TV_2_Testscenarios).

Dit kan dan gaan om functionele testen van bepaalde specifieke onderdelen (component testing) of het testen van heel specifieke functies (unit testing).
De developer kan enkele zeer kleine wijzigingen met de developer direct proberen op te lossen. Wanneer verder de test slaagt krijgt de testcase een PASS en wordt de userstory naar Done versleept.
Grotere problemen zorgen ervoor dat een testcase een FAIL met een beschrijving van hetgeen er mis gaat.

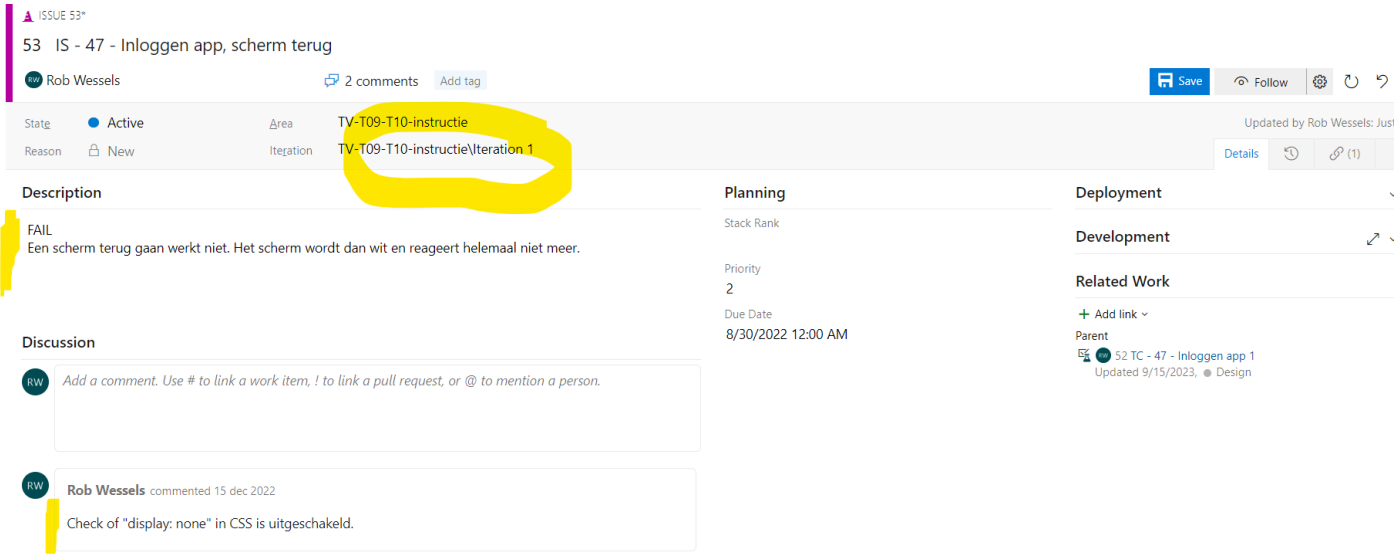
- Wanneer een test niet werkt wordt er in de titel van de testcase "FAIL" gezet:

  TEST CASE 52*

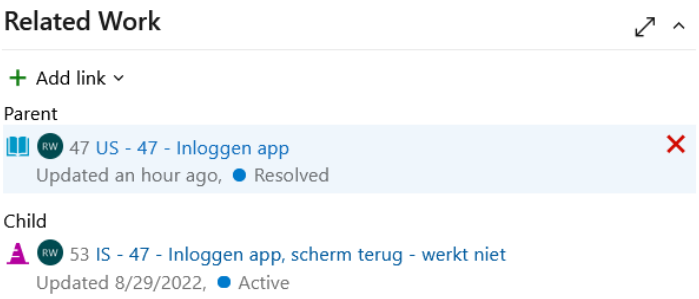
  52 TC - 47 - Inloggen app, scherm terug - FAIL

- Je vult comments in in de discussion: FAIL met daaronder wat er wel of niet gebeurt. Naam en datum worden door DevOps automatisch ingevuld:

Vervolgens wordt er een **issue** als child van de testcase aangemaakt waarin, in de discussion, diverse oplossingen voor het probleem worden aangedragen:



De userstory wordt weer terug gesleept van Testing naar To Do of naar In Progress van dezelfde sprint. Aan de Testcase met FAIL en de issue kan de developer zien dat het hier dan gaat om het corrigeren van reeds bestaande code n.a.v. een FAIL-test.



Tijdens bovenstaand proces is het belangrijk dat gegevens duidelijk en correct worden ingevuld zodat het proces overzichtelijk blijft. Dit geldt ook voor de naamgevingen.

Je verbetervoorstel is dus je aangemaakte issue

Acceptatietest

De acceptatietest volgt. Hierbij is de opdrachtgever betrokken. Samen met de developer(s) zal deze vanuit de Done testen of alle functionaliteiten werken zoals afgesproken én zoals de opdrachtgever voor ogen had. De acceptance criteria zijn hier belangrijk, maar toch zijn deze niet zaligmakend. Een opdrachtgever kan op dat moment tóch iets anders voor ogen hebben dan eerder bekend was bij de developer. Sterker nog: er zou zelfs een wens veranderd of bijgekomen kunnen zijn in de tussentijd van het developen.

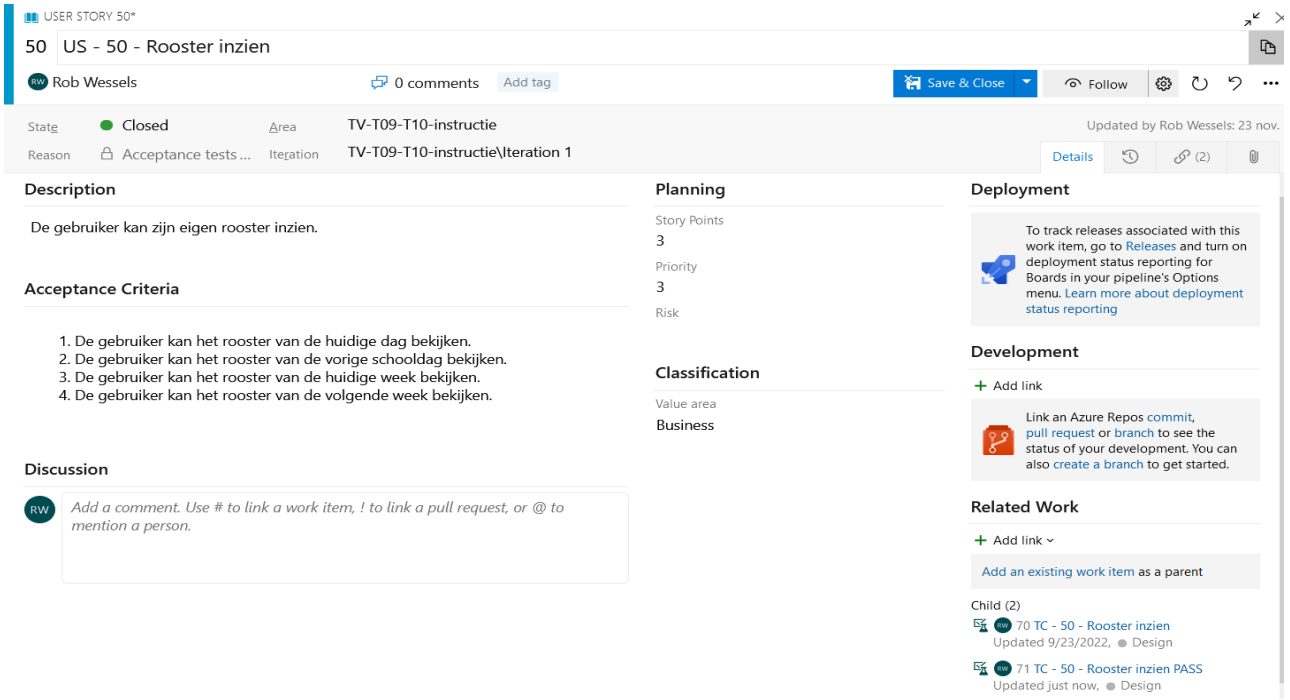
Dat is ook één van de redenen dat de opdrachtgever nauw wordt betrokken bij het ontwikkelproces. Niet de gehele applicatie wordt aan het eind van het totale ontwikkelproces getest door de opdrachtgever, maar diverse onderdelen en functionaliteiten worden op diverse momenten, het liefst continu, door de opdrachtgever getest. Hierdoor kunnen wijzigingen snel worden opgepakt en zal hopelijk het eindproduct ook meer up-to-date zijn.

Nieuwe userstory – of verplaatsen?

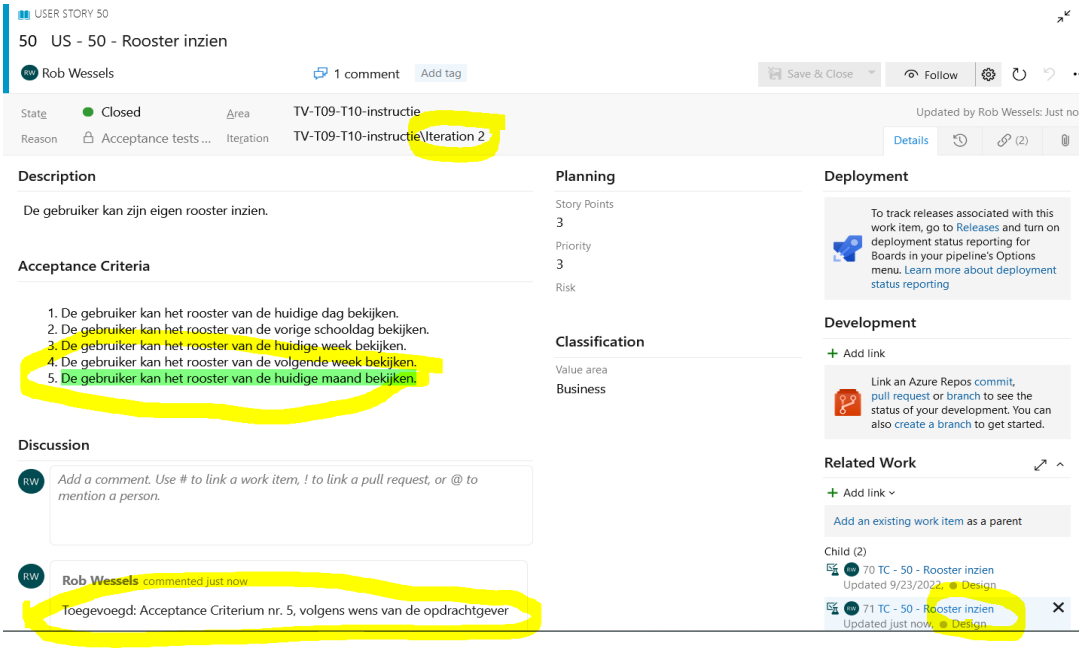
Wanneer de opdrachtgever tóch iets anders of iets extra's wil, zal de testcase van de betreffende userstory zijn PASS in de titel verliezen. Alhoewel de app technisch wel goed werkte is de testcase nu aangepast en moet in de nieuwe sprint de test gewoon nog een keer worden uitgevoerd alsof deze nog nooit is uitgevoerd.
De FAIL die al in de comments stond laten we staan. Dit omdat we dan goed kunnen zien wat er de vorige sprint met de testcase is gebeurd. (normale procedure van de testcase).
Hierna zal de bestaande parent-userstory worden aangepast:

- De extra acceptance criteria worden toegevoegd.
    - De acceptance criteria zijn genummerd.
    - De nieuwe acceptance criteria wordt gemarkeerd (in het voorbeeld: groen)
- In de discussion wordt vermeld welk punt erbij is gekomen + de reden.
- De iteration van de userstory wordt gewijzigd naar de volgende sprint.
- De testcase wordt ook naar de volgende sprint verplaatst. De PASS verdwijnt uit de titel, ***niet*** uit de comments.
- Het **testscenario** van de testcase wordt aangevuld zodat ook deze nieuwe functionaliteit wordt getest in de toekomst → Dus aangepaste **Action** en **Expected Result** (hieronder niet afgebeeld).
- De userstory wordt, in de volgende sprint, in kolom **To Do** geplaatst.

… en vanaf daar begint de hele cyclus weer opnieuw.



...wordt:



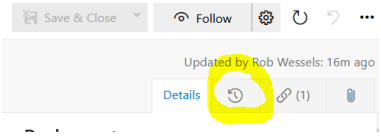
Je verbetervoorstel is dus:

de oorspronkelijke userstory in de volgende sprint met aangepaste acceptance criteria

- **de aangepaste testcases, in de volgende sprint geplaatst**

History

Mocht je tóch willen zien wat de vorige staat van deze userstory (of testcase of ieder ander item) was, dan kun je de history inzien door op het klokje te klikken. Zit er iemand te rommelen in al deze items? Dan staat er in de history ook de naam van de betreffende persoon.



Belangrijk is dus:

Bij het aanmaken van een nieuwe userstory geef je ook direct aan in welke iteratie deze zit.

Je kunt nu **T9_TV_Oefening_V_3** maken.

Je kunt nu **T9_TV_Oefening_T_3** maken.

Retrospective

→ Bekijk de pagina van de Visual Studio Marketplace waarvan de Retrospectives te installeren zijn.

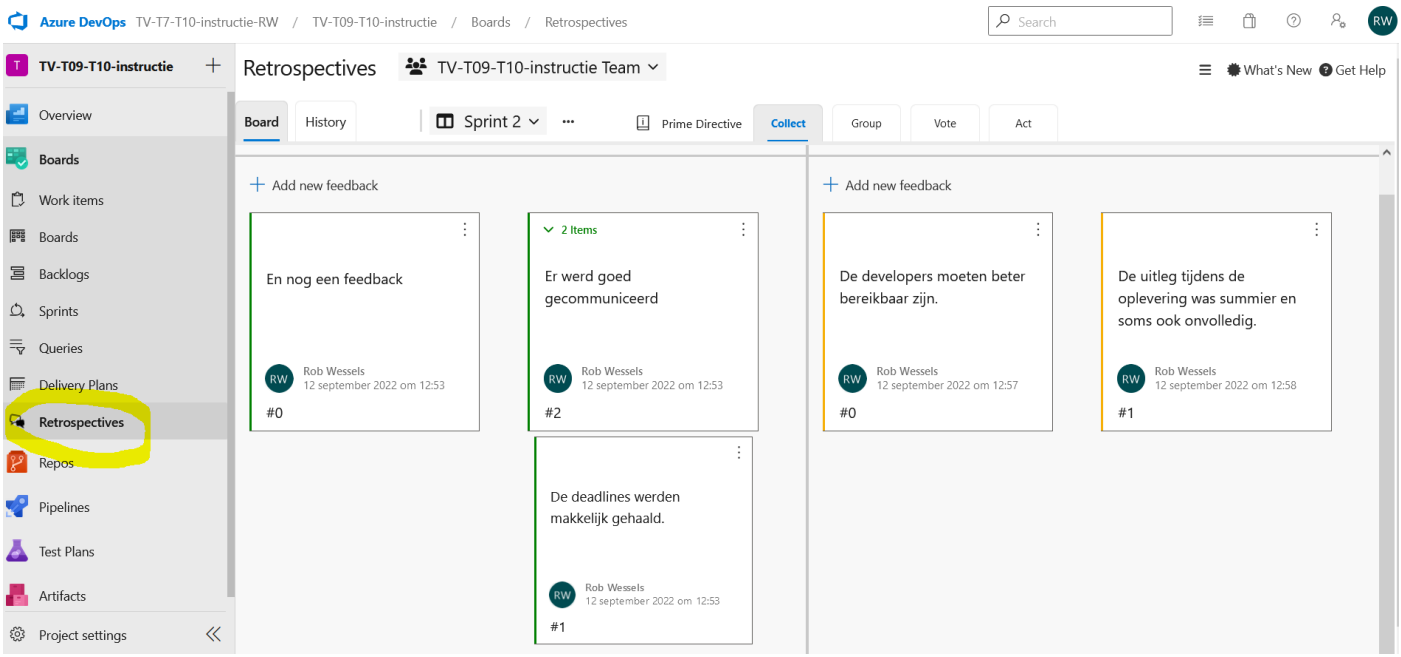
Wanneer een project helemaal is afgerond en de klant vrij en blij (denken we, hopen we) met de software aan de gang kan, lijkt het totale proces afgesloten te kunnen worden.

Dit is uiteraard niet waar. We willen leren van het doorlopen proces. We kunnen dus het beste achteraf gaan reflecteren op het proces. Dit gaat dan dus niet meer over het product, maar puur over welke onderdelen van het gehele proces goed gingen of beter konden. Want niet onbelangrijk: zaken die goed verliepen zijn ook belangrijk om te weten. Never change a winning team, dus mogelijk wil je procesonderdelen blijven hanteren wanneer deze goed lopen.

Wanneer een project is afgerond wordt er dus aan alle betrokkenen (opdrachtgever(s), mededevelopers, leidinggevenden, etc) gevraagd om constructieve feedback. Jijzelf hoort ook bij de betrokkenen, dus jouw feedback telt ook zeker mee.

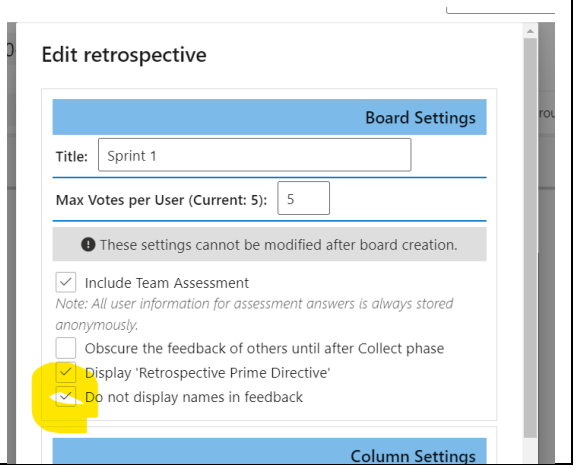
Alle feedback wordt verzameld en in de Retrospective geschreven. Betrokkenen kunnen dat zelf doen of via een ander medium (bijv. email) verschaffen ze feedback en reflectie, waardoor een developer deze feedback ook in de Retrospective kan zetten. Betrokkenen kunnen zelfs stemmen op diverse feedback-items, feedback-items kunnen worden gegroepeerd, een email-summary kan worden verkregen om in een email of word-bestand te plaatsen, etc.

>> LET OP: Een organisatie (ook KW1C) kan er ook voor kiezen om tevens iedere sprint met een retrospective af te sluiten.



Waar let je op met het maken van een Retrospective?

Naam	Sprint of Iteriation + nr. van de sprint, bijv: "Sprint 1"

Kleur en icoon	Kwaliteitspunten: groen, ☺ Verbeterpunten: oranje, **!**
Aantal punten	Per developer: minimaal 2 kwaliteitspunten en 2 verbeterpunten: • Je persoonlijke kwaliteitspunt. • Het kwaliteitspunt van jou m.b.t. het team. • Je persoonlijke verbeterpunt. • Het verbeterpunt van jou m.b.t. het team. **>> Let op** Wanneer de volgende checkbox is aangevinkt komt er geen naam bij je kaartje. In dát geval start je je tekst van elk kaartje met je voor- en achternaam. Bijvoorbeeld: *Rob Wessels – Er werd goed gecommuniceerd* 

→ Bekijk hier een voorbeeld van het gebruik van Retrospectives in DevOps.

Je verbetervoorstel is dus het ingevulde retrospective

Je kunt nu **T9_TV_Oefening_V_4** maken.