IT-Project 2020-2021

DevOps

Afstudeerrichting	AON
Taaknummer - taaknaam	3 - DevOps
Weging (%) van de taak	20%

Algemene taakomschrijving

Als applicatie ontwikkelaar kom je in aanraking met de volgende onderdelen in de DevOps opstelling van je team:

- Issue tracking
- Versiebeheer
- Kwaliteitscontrole

We verwachten dat je tijdens het IT-project voor deze onderdelen een significante bijdrage levert in je team. Deze taak bevat voor elk van bovenstaand DevOps onderdeel een deelopdracht die onderaan verder wordt toegelicht.

Verantwoordelijke senior collega's

Tim Dupont, Wesley Hendrikx

Praktische afspraken

- Tweewekelijks zit je (+- 20min.) samen met een senior collega die verantwoordelijk is voor deze taak. Tijdens dit feedbackmoment licht je de status van de taak toe en kan de senior collega gerichte feedback geven.
- In week 4 van het project en tijdens week 8 van het project wordt dit feedbackmoment een evaluatiemoment. De eerste evaluatie telt voor 40% mee. De tweede evaluatie telt voor 60% mee.

Deelopdracht 1: Issue tracking

Er wordt verwacht dat je team werkt met Jira van Atlassian. Een onderdeel van dit systeem is ‘Issue tracking’. In een issue (of ticket) wordt het werk omschreven dat gedaan moet worden (een vereiste, bug, ...). Elk teamlid zal bij de start van het IT-Project toegang hebben tot Jira. Indien de opdrachtgever graag wil dat jullie met het issue tracking systeem werken van het bedrijf, dan is dat bespreekbaar. Een voorwaarde is wel dat de technische coaches (en teamcoach) toegang krijgen tot dit systeem.

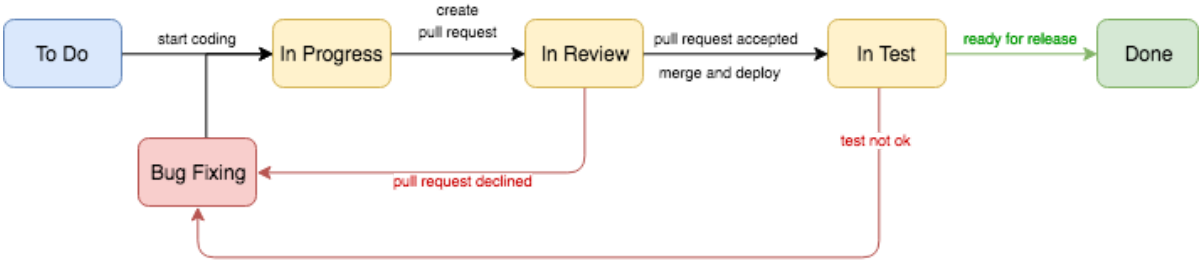
Als applicatieontwikkelaar hou je jezelf aan de volgende afspraken:

- Je registreert consequent met welk issue je in de sprint bezig bent (status = ‘In Progress’). Als je stopt met werken aan een issue dan sleep je de issue ofwel terug naar ‘To Do’ ofwel verder naar ‘In Review’. In principe kan er op het bord maar één issue op jouw naam de status ‘In Progress’ hebben.
- Issues die klaar zijn om na te kijken (code review) geef je de status = ‘In Review’. Eventueel wijs je de persoon die gaat nakijken toe aan de issue.
- Issues waarvan de code is nagekeken gaan van ‘In Review’ naar ‘In Test’ als er geen gebreken gevonden werden. In het andere geval gaat de issue naar ‘Bugfixing’. Eventueel wijs je de volgende persoon die de issue gaat behandelen toe aan de issue.
- De tester zal issues die getest en goed bevonden zijn van ‘In Test’ naar ‘Done’ verhuizen. Is er iets mis dan gaat de issue naar ‘Bugfixing’. Eventueel wijs je de volgende persoon die de issue gaat behandelen toe aan de issue.
- De status ‘Bugfixing’ kan je beschouwen als de ‘ToDo’ status voor issues die al eens gereviewed en/of getest geweest zijn. Vanuit ‘Bugfixing’ gaan de issues dus naar ‘In Progress’ vanaf het moment dat er iemand er terug aan begint te werken.
- Als je een issue een andere status geeft pas je ook de resterende tijd aan die je nog nodig hebt voor die issue.

Wanneer mag je een issue als ‘Done’ (DOD of Definition of Done) beschouwen:

- De code is geschreven.
- Er is een pull request gemaakt in het versiebeheersysteem (zie ook deelopdracht 2 en 3).
- De code in de pull request is door een junior collega nagekeken.
- De pull request is gemerged.
- De code is gedeployed in de testomgeving.
- De tester heeft de issue getest en goedgekeurd.

Workflow schema:



Indien deze workflow niet past voor jouw project, dan kan je hierover altijd één van de verantwoordelijke senior collega's aanspreken.

Tip: afhankelijk van het git systeem dat je gebruikt is het mogelijk om bepaalde acties in git (bijvoorbeeld het goedkeuren van een pull request) te koppelen aan een statusovergang in Jira (bijvoorbeeld issue naar 'In Test' verhuizen).

Voorbeeld

John start met de implementatie van *Issue-1*. Hij sleept de issue van 'To Do' naar 'In Progress'. Twee uren later is hij klaar met de implementatie. Hij maakt een pull request. Vervolgens sleept hij *Issue-1* naar 'In Review' en zet de issue op naam van Jane. Jane is immers de persoon die die pull request gaat nakijken. Ook past John de resterende tijd aan naar 1 uur, want hij denkt dat dit de tijd die nodig is om de issue in de testomgeving te krijgen en te testen.
Een half uur later heeft Jane tijd. Ze kijkt de code van John na en vindt geen problemen. Ze merged de pull request met de 'Develop' branch (zie deelopdracht 2). Dankzij CI (Continuous Integration) wordt de code automatisch in de testomgeving gedeployed en dus kan Jane *Issue-1* slepen naar 'In Test'. Jane past de resterende tijd aan naar 30 minuten, want zij denkt dat dit de tijd nodig is om de issue te testen.
Bill, de tester voert enkele testen en vindt nog een probleem. Hij voegt zijn opmerkingen toe aan de issue en sleept de issue naar 'Bugfixing'. Vervolgens zet hij de issue op naam van 'John'.

Er wordt verwacht dat minstens de applicatie features en bugs in het issue tracking systeem geregistreerd worden. Optioneel kan je ook ander werk registreren (o.a. research, documentatie, refactorings, proof of concepts, projecttaken, ...)

Deelopdracht 2: Versiebeheer

Er wordt verwacht dat je team de code bewaart m.b.v. een versiebeheersysteem. Er zijn vele systemen op de markt, maar voor het project wordt het ‘Git’ versiebeheer systeem verplicht.

Binnen ‘Git’ zal je team ook een bepaalde ‘branching’ strategie of workflow moeten kiezen. Zie https://www.atlassian.com/git/tutorials/comparing-workflows.

Voor dit project moet je verplicht werken met één van deze workflows:

- Feature Branch Workflow (aanbevolen)
- Gitflow Workflow. Zie https://github.com/nvie/gitflow, https://danielkummer.github.io/git-flow-cheatsheet
- Forking Workflow
- Als de opdrachtgever graag met een andere flow wil werken, dan is dat steeds bespreekbaar

Maak regelmatig een commit. Als algemene richtlijn kan je stellen dat je een commit maakt zodra je iets hebt dat werkt en dat niets stuk maakt voor anderen. Doorgaans maak je meerdere keren per dag een commit.

Schrijf bij elke commit een goede commit message. Een goede commit message geeft ontwikkelaars (ook jezelf) op een later ogenblik in de tijd een goed idee van de wijzigingen die gebeurd zijn.

Tip: meer informatie over het schrijven van goede commit messages vind je in:
https://wiki.openstack.org/wiki/GitCommitMessages#Information_in_commit_messages

Tip: Jira maakt het mogelijk om issues te koppelen aan code wijzigingen als je een ‘commit’ doet.

Tip: Jira maakt het ook mogelijk om rechtstreeks een (feature) branch aan te maken vanuit een issue afhankelijk van het gebruikte git systeem.

Deelopdracht 3: Kwaliteitscontrole

We verwachten dat jullie de kwaliteit van je applicatie continu in het oog houden. Zo moet er aandacht zijn voor de kwaliteit van de code en het (automatisch) testen van de code.

Code

Zorg voor leesbare code:

- Zinvolle naamgeving voor variabelen, functies, classes
- Indien nodig, commentaar in de code
- Volg code conventies (bv. Naam van een variabele begint met een kleine letter)
- Als je met een IDE werkt gebruik dan de plugins zoals PMD, checkstyle en findBugs.

Vermijd 'code smells' (https://en.wikipedia.org/wiki/Code_smell) zoals daar zijn:

- Ongebruikte variabelen
- Classes die te veel doen / te groot zijn
- Gekopieerde code (Don't Repeat Yourself)
- Lange lijst van parameters
- Code die niet bereikt kan worden (Dead Code)
- Hard coded configuratie (bv. uri van de log file staat rechtstreeks in de code)
- ...

Er bestaan tools om de kwaliteit van je code te monitoren (o.a. SonarQube, Code Analysis in Visual Studio, …). Maak gedurende het project gebruik van zo een tool. Wij raden SonarQube aan.

Toon aan hoe je de kwaliteit van je code zou kunnen verbeteren met behulp van de gekozen tool. De gekozen tool zal in de loop van het project deel gaan uitmaken van de CI pipeline (dit maakt deel uit van de DevOps taak voor SNB). Wacht echter niet tot deze integratie gebeurd is om gebruik te maken van de tool. Tools, zoals SonarQube, kan je ook gemakkelijk lokaal op je machine installeren en gebruiken.

Naast het meten van de kwaliteit van de code met een tool dien je ook code reviews te doen. De code reviews moeten traceerbaar zijn in een bijhorende pull request. Zie https://www.atlassian.com/git/tutorials/making-a-pull-request. Bij elke feature of bug hoort een pull request (zie ook deelopdracht 1).

Testing

Toon aan dat je systematisch gebruik hebt gemaakt van unit tests. De unit tests dienen deel uit te maken van het automatisch build process (CI). Gebruik een tool om je 'code coverage' aan te tonen. Motiveer waarom sommige code niet 'gecovered' is (als dit het geval is).

Beoordelingscriteria

Criterium	**Gewicht**
Correct gebruik van issue tracking systeem	20%
Correct gebruik van een versiebeheer systeem (git)	20%
Aanpak voor het schrijven van kwaliteitsvolle code	40%
Systematisch gebruik van unit testen	20%