Git branching strategie & richtlijnen

# Branching strategie

Om de Git repositories van dit project overzichtelijk te houden zal er gebruik gemaakt worden van de aangepaste versie van de GitHub flow branching strategie. In het diagram hieronder is een voorbeeld van deze strategie te zien.

Afbeelding met duisternis, waas, nacht, licht

Automatisch gegenereerde beschrijving

Bij de GitHub flow branching strategie wordt er gebruik gemaakt van een master branch en dienen er losse branches per feature of bugfix gemaakt te worden. Op het moment dat de feature of bugfix klaar is, dan kan deze via een pull request getest, gereviewed en vervolgens gemerged worden met de master branch. Het is vooral van belang dat hetgeen wat op de master branch staat werkt naar behoren en dus van tevoren getest is.

Bij dit project hebben wij als team de branch s3nj24\_dev aangeleverd gekregen. Deze branch zal in ons geval als de master branch behandeld worden. Na iedere oplevering kunnen we indien gewenst door de docenten een pull request aanmaken om s3nj24\_dev te mergen met de main.

<https://docs.github.com/en/get-started/using-github/github-flow>

# Richtlijnen

## Commits

Het is in het algemeen beter om meerdere kleine commits te maken in plaats van één grote commit. Door kleinere commits te maken kun je je commit messages duidelijker maken en biedt het meer mogelijkheden om terug te rollen naar een eerdere commit bij problemen, zodat je minder werk verliest op het moment dat een rollback nodig is.

## Commit messages

Om goed bij te kunnen houden waar en wanneer bepaalde wijzigingen aangebracht zijn is het belangrijk om duidelijke commit messages te schrijven. Om deze duidelijk en consistent te houden is het handig om de volgende richtlijnen te volgen:

Schrijf commit messages altijd in het Engels en laat ze met een werkwoord beginnen. Denk hierbij aan: Added, Refactored, Removed, Fixed, etc.. Beschrijf daarna verder de veranderingen die je gedaan hebt. Probeer daarbij ook specifiek te zijn. Maak commit messages ook niet al te lang, zodat je het hele bericht in een oogopslag kan lezen. GitKraken heeft rechts van de commit message een karakterlimiet van 72 karakters staan. Probeer die aan te houden.

**Enkele voorbeelden:**

* “Added unit tests for UpdateRole and Update methods in UserService”
* “Removed unused RoleUtils class”
* “Fixed current password not being validated in MyAccount/Edit”
* “Refactored release related DAL and service to use new (Data)Result classes”

## Branches

Bij branches zijn er minder strenge richtlijnen van toepassing aangezien deze regelmatig opgeschoond zullen worden. Toch is het fijn om hier wat consistentie in te hebben, waardoor het handig is om de volgende richtlijnen aan te houden:

Zorg dat je branches altijd in het Engels schrijft en vernoemt naar hetgeen waar je op dat moment mee bezig bent.

## Mergen

Wanneer je feature klaar en getest is is hij in principe klaar om gemerged te worden met de ‘s3nj24\_dev’ branch. Voordat je dit gaat doen is het handig om eerst de ‘s3nj24\_dev’ in jouw branch te mergen, zodat de kans op merge conflicts klein blijft en eventuele conflicten op jouw branch opgelost kunnen worden.

Het mergen zelf zal niet handmatig gaan, maar juist verlopen via pull requests. Een pull request kun je op GitHub aanmaken. Zet in de titel van de pull request welke feature of user story je in de dev branch gaat mergen, liefst in het Engels.