*Werkinstructies*

Software ontwikkelstraat

*Dit zijn de werkinstructies voor de omgang met de door ons opgezette software ontwikkelstraat. In dit document wordt stap voor stap beschreven wat er allemaal moet gebeuren als je de ontwikkelstraat optimaal wilt gebruiken.*

**Datum:** 31-03-2014

**Groep:** S63-E

**Groepsleden:** Bas Rheiter, Ferdie Jonkers, Jelle v.d. Heijden, Remi Arts, Tim Lodder

2014

Inhoud

[Het toevoegen van een nieuw project: 2](#_Toc384031035)

[Mantis: 4](#_Toc384031036)

[Benodigheden 4](#_Toc384031037)

[Initialisatie 4](#_Toc384031038)

[Issue aanmaken 5](#_Toc384031039)

[Issue zoeken 5](#_Toc384031040)

[GIT: 6](#_Toc384031041)

[Benodigdheden 6](#_Toc384031042)

[Initialisatie 6](#_Toc384031043)

[Nieuw project 6](#_Toc384031044)

[Bestaand project 6](#_Toc384031045)

[Aanpassingen doorvoeren 6](#_Toc384031046)

[Aanpassingen ophalen 6](#_Toc384031047)

[Branches 7](#_Toc384031048)

[Merge conflict 7](#_Toc384031049)

[Rebasen 7](#_Toc384031050)

[Versie terughalen 8](#_Toc384031051)

[Eerdere versies 8](#_Toc384031052)

[Code analyse met Sonar: 8](#_Toc384031053)

# Het toevoegen van een nieuw project:

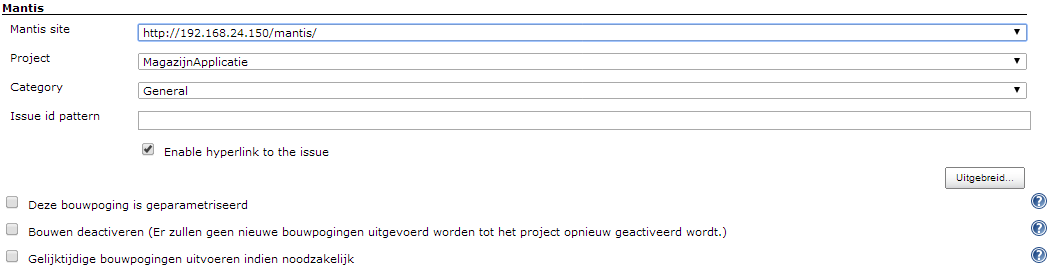
Voordat je gebruik kunt maken van onze ontwikkelstraat moet je verbonden zijn met de VPN van vpnseclab.fhict.nl. Je kunt verbinding maken met deze VPN door gebruik te maken Cisco AnyConnect Secure Mobility Client en in te loggen met je Fontys gegevens. Om een nieuw project aan de software ontwikkelstraat toe te voegen start je met het aanmaken van een Maven project in Netbeans. Vervolgens maak je een nieuw item aan in Jenkins met dezelfde naam als het Netbeans project. Dit doe je door naar de URL <http://192.168.24.150:8080> te gaan. Via deze URL kom je terecht in onze Jenkins omgeving. Nadat je bent ingelogd kun je in het linker menu kiezen voor een nieuw item. Je kiest als type een ‘Bouw een vrije stijl type software project’. Dit type biedt alle basisfunctionaliteit van Jenkins. Wanneer het nieuwe item is toegevoegd, dient het ook nog geconfigureerd te worden. Echter moeten we van tevoren nog enkele acties ondernemen om alle configuratie mogelijk te maken.

Alle projecten in onze ontwikkelstraat worden gemanaged via GitLab. Hier kun je alle verandering zien die in projecten zijn doorgevoerd met toegevoegd commentaar. Ook kun je eerdere versies van projecten terughalen en zo wijzigingen ongedaan maken. Voor ons nieuwe project moet er in GitLab ook een project worden aangemaakt. Dit doe je door naar GitLab te gaan (<http://192.168.24.150>) en vervolgens op ‘+ New project’ te klikken. Als naam vul je weer dezelfde naam in als die van het Netbeans project en Jenkins item. Vervolgens voegen we aan ons project meteen een webhook toe, die ervoor zorgt dat je project direct in Jenkins wordt gebuild nadat het gepushed is via git. Hiervoor gaan we via gitlab naar ‘Projects’ en kiezen we het nieuw toegevoegde project. Daarna kies je rechtsboven voor ‘Settings’ en ga je naar Web Hooks. Als URL vul je ‘http://Tim:be11ce7e4e736b16bbe0462d092f6efc@192.168.24.150:8080/job/MagazijnApplicatie/build?delay=0sec’ in en als trigger kies je ‘Push events’. Ook voeg je direct je Netbeans source code toe aan Git met behulp van Git Bash.

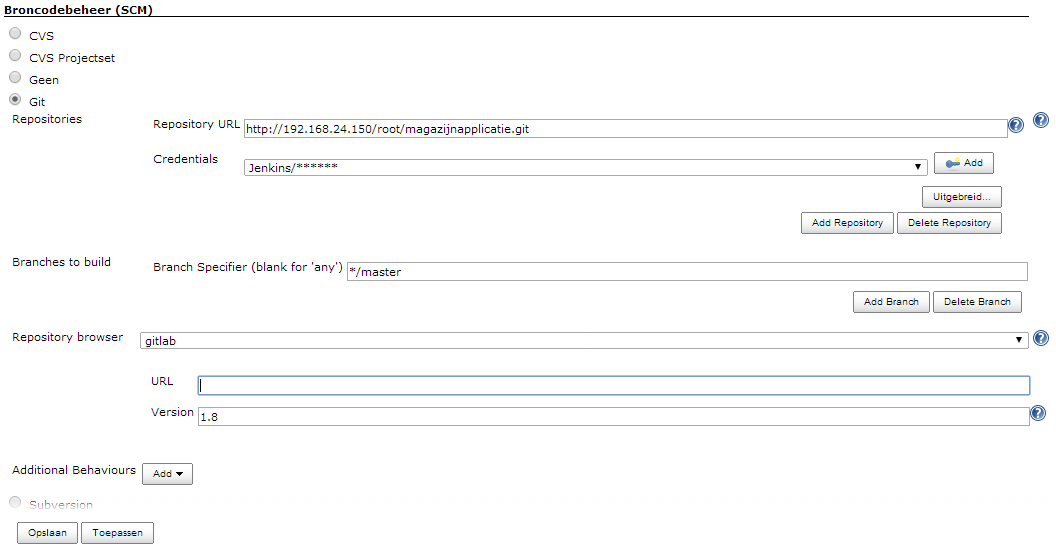
Ook in ons Issue Tracking moet het project worden toegevoegd. Om het project aan Mantis toe te voegen ga je naar de URL <http://192.168.24.150/mantis>. Wanneer je bent ingelogd kies je in het bovenste menu voor ‘Manage’ en vervolgens voor ‘Manage Projects’. Hier kun je een nieuw project toevoegen. Als projectnaam kies je weer dezelfde naam als het in Netbeans aangemaakte project.

Nu is het tijd om Jenkins te configureren. Je gaat naar Jenkins en klikt op het project dat je wilt configureren, daarna kies je in het linker menu voor ‘Configureren’. Aan Jenkins zijn al verschillende plug-ins toegevoegd, zoals beschreven in de installatie handleiding. Deze gaan we nu configureren.

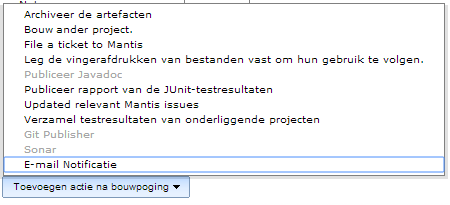
We beginnen met Mantis, dit heeft de volgende gegevens nodig:



Vervolgens gaan we verder met het broncodebeheer. Dit is in ons geval Git en we dienen de volgende gegevens in te vullen. De repository url kun je bovenaan de pagina vinden via GitLab wanneer je je project hebt geselecteerd. De credentials zijn door ons zelf toegevoegd en betreffen de inloggegevens voor git.



Ten slotte geef je nog aan dat je project Sonar gaat gebruiken. Dit doe je door onderaan te kiezen voor ‘Toevoegen actie na bouwpoging’ en sonar te selecteren. Via dit drop down menu kun je ook nog verschillende andere onderdelen toevoegen, maar dat is in onze straat niet nodig.



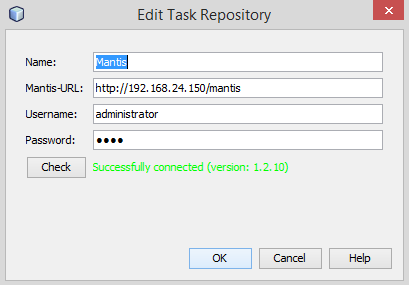
# Mantis:

## Benodigheden

Mantis plug-in voor Netbeans (<http://plugins.netbeans.org/plugin/45919/mantisintegration>).

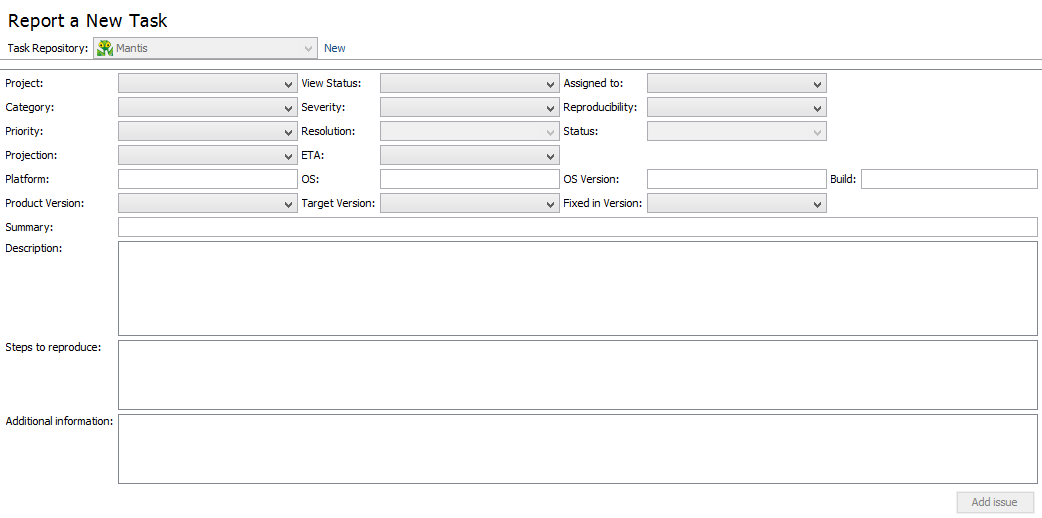
## Initialisatie

In Netbeans klikt je op tabblad ‘Services’, daarna klik je met de rechtermuisknop op ‘Task Repositories’ en kies create task repository. Hiervoor vul je de volgende gegevens in:



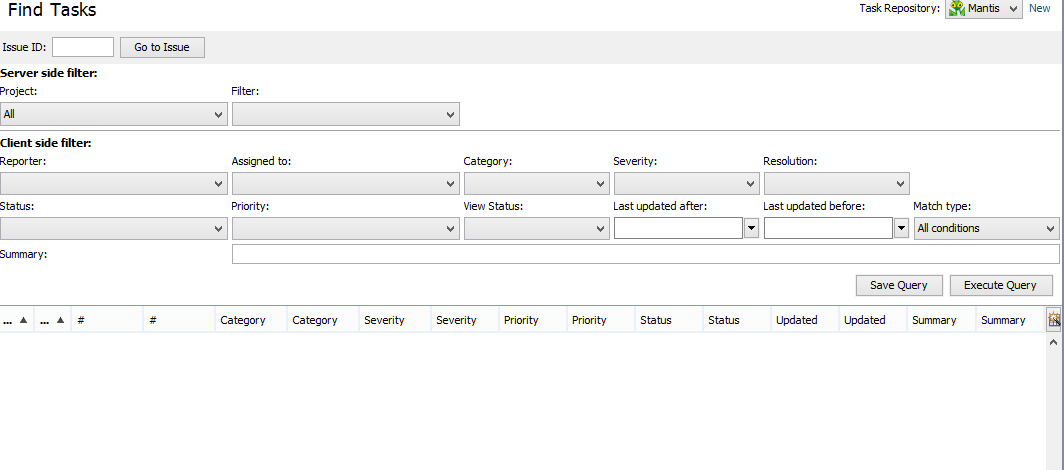
## Issue aanmaken

Binnen het tabblad ‘Services’ zie je onder de task repositories de door jouw toegevoegde mantis repository staan. Klik met de rechtermuisknop op die repository en kies ‘Report Task’. Als je alle velden volledig hebt ingevuld kun je de task aan mantis toevoegen.



## Issue zoeken

Navigeer naar mantis zoals hierboven beschreven. Klik daarna met de rechtermuisknop op mantis en kies daarna voor ‘Find Tasks’. Je kunt hierna het juiste project selecteren en eventueel enkele filters toevoegen aan je zoekopdracht. Klik op ‘Execute Query’, wanneer er issues worden gevonden zijn deze terug te vinden in de lijst onderaan.



# GIT:

## Benodigdheden

Git Bash (<http://git-scm.com/downloads>).

## Initialisatie

Alvorens het project te initialiseren, dient het gebruikersnaam en e-mail adres geïnitialiseerd te worden. Met behulp van de volgende regels code, kan dit worden ingesteld:

Git config –global user.name “Uw gebruikersnaam”

Git config –global user.email “Uw e-mail adres”

### Nieuw project

Vervolgens dient er en nieuwe map aangemaakt te worden, waar alles in moet worden gezet. Hier moeten de bestanden in worden gezet, die samen de standaardversie vormen.

Cd “map waar de bestanden in zitten”

Git init

Git commit –m “first commit”

Git remote add origin [git@192.168.24.150:Uw](mailto:git@192.168.24.150:Uw) gebruikersnaam/Uw project.git

Git push origin master

### Bestaand project

Indien het project al bestaat, is het de bedoeling dat het project lokaal wordt gekloond.

Git clone [git@192.168.24.150:Uw](mailto:git@192.168.24.150:Uw) gebruikersnaam/Uw project.git doelmapnaam

## Aanpassingen doorvoeren

Wanneer er een aanpassing klaar is en moet worden doorgevoerd, moeten de volgende regels worden uitgevoerd;

Git add .

Git commit –m “Toelicht van de huidige commit”

Git push origin “branchnaam/master”

## Aanpassingen ophalen

Het ophalen van aanpassingen gaat met behulp van een simple regel:

Git pull

## Branches

Een branch is een tak onder een project. Zo kun je grotere issues onderverdelen onder een aparte tak, waardoor het project niet direct wordt aangepast. Op deze manier kun je veilig grotere aanpassingen maken, zonder dat andere mensen hier last van hebben. Eerst is het de bedoeling om een branch aan te maken.

Git checkout –b “branchnaam”

Op deze manier maakt hij een nieuwe branch aan en gaat hij daar direct naartoe (dit is ook te zien in de commandline, naast de mapnaam). Wanneer je nu elementen pusht, wordt dit apart weergegeven en zal dit niet in conflict komen met het volledige project (master). Om het project weer samen te voegen, dient u, na de laatste commit, een merge in te zetten.

Git checkout master

Git merge –no-ff “branchnaam”

## Merge conflict

Het kan wel eens voorkomen dat er een merge conflict optreedt. Git heeft een handige tool ingebouwd om dit soort problemen gemakkelijk op te lossen. Met de regel;

Git mergetool

Verschijnt er een programma in beeld, waarmee je gemakkelijk de twee bestanden handmatig kunt mergen, waar het merge conflict optreedt. Vervolgens kun je het conflict weer opnieuw committen en pushen, zodat het project weer merge conflict-vrij is.

### Rebasen

Wanneer dit soort problemen vaak optreden, komen er veel overbodige commits door. Dit zorgt ervoor dat het commit overzicht erg onoverzichtelijk wordt. Indien de wijziging nog niet is gepusht, kun je door middel van een rebase de commits aanpassen en eventueel samenvoegen/wijzigen.

Git rebase –i

Hierdoor komt er een overzicht in beeld met de laatste commits. Deze commits kun je definiëren als “pick”, “squash”, of “edit”. Pick betekend simpelweg dat de commit wordt gebruikt, squash betekend dat de commit wordt samengevoegd met de vorige commit en edit betekend dat hij wordt gebruikt, maar wel de gewijzigde variant.

Git commit –amend

Git commit –continue

## Versie terughalen

Git is een versiebeheer systeem, waardoor het gemakkelijk is om terug te vallen op vorige versies.

Om de lokale wijzigingen te ontwaren, dient de volgende regel te worden gebruikt;

Git reset –hard

### Eerdere versies

Om een eerdere versie te gebruiken, dient de hashcode van de desbetreffende revisie te worden gebruikt;

Git checkout “hashcode”

Hierdoor wordt de revisie teruggezet van de gewenste revisie.

# Code analyse met Sonar:

## SonarCube in het kort

SonarCube (SC) scant de source code van projecten om te kijken hoeveel regels code, documentatie, issues, en inconsistenties er zijn. Daarbij wordt er gekeken naar code complexiteit, unit test coverage en duplicaatcode. Bij iedere build die door Jenkins CI wordt uitgevoerd, wordt er een code analyse uitgevoerd door SC.

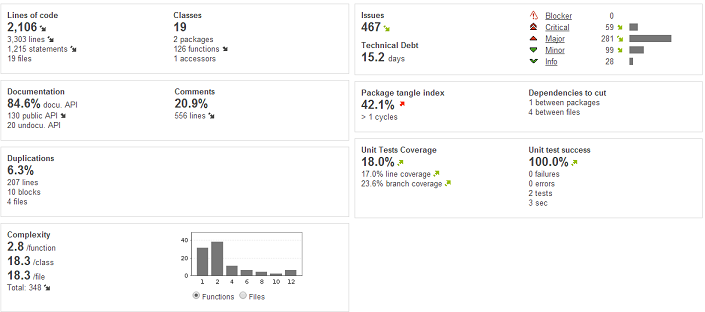
## Aannames

Zorg ervoor dat er een werkende SC-server is opgezet en is geïntegreerd met Jenkins CI. De server is correct geïnstalleerd op het moment dat het SC dashboard te bereiken is via de gespecificeerde poort. In deze context is SC bereikbaar via poort 9000.

## Projecten benaderen

Op het dashboard van SC vind je de geïntegreerde projecten (zie afbeelding). In dit overzicht is ook de LOC (Lines of Code), de laatste analyse en de Technical Debt per project zichtbaar.

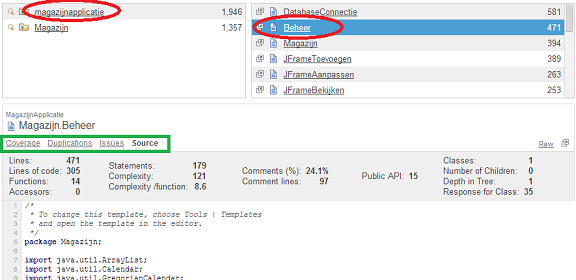
Door een project aan te klikken kom je op het dashboard van dat project. Hier krijg je een uitgebreid overzicht van de verschillende componenten waar SC de analyse op uitvoert.



## Analyse LOC en documentatie

SC analyseert de volledige source code en geeft een overzicht van het aantal regels code, LOC (non-commenting Lines of Code), statements, documentatie, duplicaatcode en daadwerkelijke files (zie afbeelding).

1. Klik op één van de geanalyseerde componenten
2. Selecteer het gewenste project binnen de solution
3. Selecteer een klasse. Dit toont de analyse die is uitgevoerd over deze klasse
4. Klik op ‘issues’ om een gedetailleerd overzicht van probleemkwesties te krijgen (zie afbeelding)
5. Klik op ‘Duplications’ om duplicaatcode binnen die klasse te bekijken
6. Klik op ‘Coverage’ om zien welke delen van de code nog niet in Unit tests worden meegenomen



Dezelfde workflow geldt voor de analyses voor de volgende componenten binnen het project:

* Lines of Code
* Classes
* Documentation
* Comments
* Duplicates
* Complexity

## Geavanceerde issue analyse

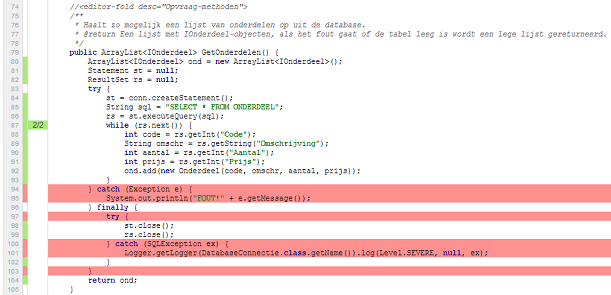
Per analyse wordt er een overzicht van issues gemaakt, welke gecategoriseerd worden op basis van prioriteit en invloed op de rest van het project.

1. Ga naar het dashboard van het gewenste project
2. Klik op ‘Issues’ of op één van de categorieën om de issues per prioriteit te bekijken
3. Bovenaan zie je een differentiatie van de verschillende issues (Rule’s genaamd). Klik op een rule om een overzicht te krijgen van waar een foute implementatie van de rule in de code is gevonden
4. Klik op één van de issues om details van de issue te krijgen, zoals status, laatst geüpdatet, de aangewezen persoon om deze issue op te lossen en de technical debt (tijd in uren dat gerekend wordt om deze issue te verwerken)

## Unit Tests Coverage

SC controleert ook of alle code wel door unit tests wordt behandelt en of alle unit tests wel zijn geslaagd of niet. Om hier meer inzicht in te krijgen dien je de volgende stappen te doorlopen:

1. Ga naar het dashboard van het gewenste project
2. Selecteer de gewenste klasse. Naast de klasse kun je het percentage behandelde code zien
3. Klik op Unit Tests Coverage om een overzicht te krijgen van welke code wel en welke code niet door unit tests wordt behandeld (zie afbeelding)



## Unit Tests resultaten

Om een overzicht te krijgen van de resultaten van de unit tests, dien je de volgende stappen uitgevoerd te worden (zie afbeelding):

1. Ga naar het dashboard van het gewenste project
2. Klik op de gewenste categorie (gefaalde tests, errors, geslaagde tests en alle tests)
3. In de tabbladen kun je nog kiezen voor ‘Issues’, ‘Tests’ en ‘Source’.