



Opleiding Applicatie Ontwikkelaar

DG3 - Documentatie,
Versiebeheer en Scrum
Samen projecten doen II

Auteur: Aminah Balfaqih & Erik Mast

Datum: 17-9-2019

Versie: 1.0

Inhoudsopgave

Overzicht	4
Voorkennis.....	4
Materialen	4
Bronnen	4
Instructies	4
Beschrijving	5
Doelen	5
Beoordeling	5
Documentatie.....	6
Definitiestudie	6
Doel van het document.....	7
Inhoud van het document.....	7
Opdracht 1.....	8
Waterval als projectmethode.....	9
Conclusie over de watervalmethode	9
Nadelen	9
Waterval of Scrum.....	9
Scrum in een groep	10
De opdracht.....	10
Versiebeheer	11
Uitleg	11
Introductie.....	11
Wat is een branch?.....	11
Voorbeeld situatie	12
Verschil tussen branch en commit.	12
Een branch maken.....	13
Een branch kiezen	14
Een branch samenvoegen met een andere branch	15
Oefening 2	16
Eindopdracht	18
Eigen project als opdracht.....	18



DG3: SAMEN PROJECTEN DOEN II

Alternatieve opdracht	18
Eisen	18
Bronnen	19

Overzicht

Level: Domein DG Level 3

Duur: 3-4 weken

Methode: Weekplanning

Voorkennis

DG2 – Samen project doen

én

A3 PHP òf A3 Java (ook te gebruiken als combinatie)

Ook kan B3 toegevoegd worden aan het eindproject

Materialen

Je laptop met:

- Programmeeromgeving
- Webbrowser
- Geïnstalleerde git software

Bronnen

- Zie bijlage Bronnen

Instructies

- Wat zijn Git-branches en forks
- Hoe werk je met Scrum in een groep (backlog/userstories)
- Definitiestudie
- Scrum <-> Watervalmethode



Beschrijving

In deze module leer je hoe je project met meer mensen kunt aanpakken. Deze module kun je dus het beste doen met minimaal 2 personen, maar kun je ook met een team van 4 mensen doen.

Je gaat een project beschrijven, en opdelen in taken, en die taken weer onderverdelen in sprints. Die sprints zul je in de code terug vinden als branches.

Doelen

Na deze module zal je in staat zijn om een project met een aantal mensen te doen en er behalve een planning en een beschrijving van te maken, ook in code de sprint en branch structuur te laten zien.

Beoordeling

Deze opdracht wordt beoordeeld aan de hand van de eindopdracht en een gesprek over het eindproduct.

Documentatie

We gaan verder in onze zoektocht naar meer controle over en begrip van de projecten die we doen. We hebben in de vorige module een Functioneel Ontwerp en Technisch Ontwerp gemaakt, maar eigenlijk is dat normaal gesproken verkeerd om.

Je wilt toch eerst in kaart brengen wat er nu eigenlijk gemaakt moet worden, voor dat je begint.

Je hebt al eens een (onderdeel van een) FO gemaakt. Daarvoor maakte je gebruik van de eisen die de opdrachtgever je hebben verteld. Dat ging redelijk gemakkelijk, want je krijgt dat een lijstje met wat je moet doen. Maar echte klanten zijn niet zo gemakkelijk. Ze willen een programma of een website. En dat is alles wat ze je kunnen uitleggen. Dus je moet zelf aan het werk met de

Definitiestudie

Stel je voor: jouw opdrachtgever zegt je dit:

“Wij hebben een bedrijfje dat handelt in metaalwaren(schroeven, moeren en andere metalen voorwerpen) voor computers. We leveren aan bedrijven die computers op maat maken, en kopen onze onderdelen in grote hoeveelheden van onze leveranciers.

We hebben ongelooflijk veel gedoe om ons klantensysteem op orde te houden. We hebben een administrateur die alles op kaartjes schrijft en doorstreept als er dingen veranderd zijn. Elke bestelling die gedaan wordt, komt op een nieuw kaartje en als die verzonden is, komen die kaartjes in een te facturen bak. Hij maakt dan een factuur, en als die betaald is, komt dat kaartje in de bak achter de klant te staan.

Elk kwartaal blijft dat die kaartjes niet in de bak passen en dan kopen we een nieuwe bak en maken ruimte in de oude bakken voor nieuwe bestellingen.

Hier willen we graag een programma voor die in één klap alle kaartjes overbodig maakt.”

Op zich is dit een simpel en helder verhaal. Je begint met programmeren en op een bepaald moment is het af. Je laat het aan de opdrachtgever zien. En die zegt:

“Dit is niet wat ik wil, hier ga ik niet voor betalen”

Dat is een groot probleem, je hebt er veel tijd in gestopt, maar krijgt er geen geld voor (probleem 1) en bovendien het is niet wat de opdrachtgever wil (probleem 2).

Het tweede probleem kun je oplossen door voor dat je begint met programmeren, in kaart te brengen wat de opdrachtgever wil. En dan los je ook probleem 1 op, immers als de klant akkoord gaat met jouw voorstel, dan kun je daarna praten over hoeveel het gaat kosten. En als je ook daar met de opdrachtgever eens over wordt, dan heb je alle opstartproblemen opgelost.

Doel van het document

De opdrachtgever moet op basis van de definitiestudie kunnen beslissen om wel of niet met het project te beginnen. In het document moet duidelijk worden wat de opbrengsten van het project zijn; eindproduct, oplossing van de probleem(stelling) en doelstelling. Daarnaast moeten de kosten duidelijk worden; materieel en mensen die nodig zijn, materialen die aangeschaft moeten worden en de tijd die nodig is voor de uitvoering (planning).

Inhoud van het document

Algemeen

Voor een definitiestudie document gelden dezelfde regels als voor alle andere projectdocumenten die je maakt:

- Het heeft een voorblad en bevat
 - De naam van het project
 - Titel van het document (bv Definitiestudie)
 - Namen van de schrijvers
 - Versienummer van het document
- Een revisietabel (wie heeft op welke datum wat aangepast?)
- Een inhoudsopgave (dit kun je zelfs automatisch laten genereren in Word met hoofdstukken en paragrafen. Tip: maak een document waarin dit werkt, en begin daarna pas te typen)
- Heeft paginanummers

Contactgegevens

Noteer hier de contactgegevens van de opdrachtgever en van jezelf of van jouw bedrijf.

Werkwijze

Omschrijf de structuur van je werkwijze. Wanneer je volgens een projectfasering werkt en de opdrachtgever is daar niet van op de hoogte, is het belangrijk om dat uit te leggen. De opdrachtgever weet dan wat hij mag en kan verwachten van jou als bedrijf. Dit kun je doen in de inleiding. De werkwijze kun je ook omschrijven bij de planning, maar als je bijvoorbeeld het functioneel ontwerp benoemd in je tekst, dan is dat belangrijk om van tevoren aan te geven wat je daarmee bedoelt.

Bedrijfsinformatie

Beschrijf het bedrijf van de opdrachtgever. Denk hierbij aan de grootte; hoeveel mensen werken er, wat doen ze; wat zijn de belangrijkste werkzaamheden van het bedrijf, wat is de structuur; bijvoorbeeld afdelingen of een organigram.

Wat is het werk van de opdrachtgever (huidige situatie)

Voor wie ga je het project uitvoeren, de eindgebruikers; afdelingen, werknemers en/of opdrachtgevers en beschrijf de werkzaamheden/handelingen die met het project te maken hebben. Geef hierbij een duidelijk beeld van de beginsituatie, hoe wordt er op het moment gewerkt.

Wat is het probleem/doelstelling (gewenste situatie)

Welke problemen zijn er op dit moment die moeten worden opgelost. Welke wensen zijn bij de opdrachtgever, bijvoorbeeld; uitbreiding of nieuwe functionaliteit. Het oplossen van de genoemde problemen en/of het mogelijk maken van de wensen is de doelstelling voor het project. Kijk uit dat je hier niet de oplossing noemt maar het probleem of doelstelling. Bijvoorbeeld '*het maken van een database*' is niet een wens, maar '*het verbeteren van de informatiestroom binnen de organisatie*' wel. Daarbij kan het maken van een database-toepassing wel de oplossing zijn.

Probleemanalyse

Analyseer de eerder genoemde problemen, wensen en doelstellingen. Verzamel informatie die met het probleem of de wens te maken hebben. Bekijk of het probleem of de wensen is op te delen in deelproblemen/wensen. Onderzoek wat de oorzaak is van het probleem. Waar heeft het probleem of de wens mee te maken; wat zijn de verbanden.

Oplossing(-srichting)

Welke oplossing heb je om de problemen te verhelpen en de doelstellingen te halen. Geef aan welke eigenschappen je oplossing moet hebben.

Middelen

Welke middelen heb je nodig van de opdrachtgever, bijvoorbeeld een werkplek, specifieke hard- en software, externe deskundigheid, gegevens, inlogmogelijkheden. Hier moet de opdrachtgever een compleet overzicht hebben van de zaken die hij moet aanleveren/regelen voor de uitvoering van het project. Dit kan dus ook zijn; informatie, bijvoorbeeld bestaande rapportage, gegevens, bijvoorbeeld ingevulde formulieren die op dit moment worden gebruikt of toegang tot een of meerdere locaties, als je daar zaken moet inspecteren of controleren om het project te kunnen uitvoeren.

Opdracht 1

Schrijf aan de hand van het korte interview op bladzijde 5 een definitiestudie.

- Gebruik als hoofdstuknamen alle dingen die hierboven genoemd zijn (Contactinformatie, Werkwijze, Bedrijfsinformatie enzovoort tot en met Middelen).
- Het hoeft geen vloeiend verhaal te zijn

Waterval als projectmethode

Lees deze wikipedia pagina eerst door:

<https://nl.wikipedia.org/wiki/Watervalmethode>

Conclusie over de watervalmethode

Als je dit hebt gelezen zul je in eerste instantie denken dat dit een goed idee is. Sommige kleine projecten kunnen prima zo (een website met 3 pagina's bijvoorbeeld, of een opdracht uit een module in level 1) al zal je waarschijnlijk veel tijd besteden aan het schrijven van documentatie.

Je doorloopt elke keer alle fases, en er kan dus steeds ingegrepen worden als een project qua uitvoering de verkeerde kant op gaat. De kracht van de watervalmethode zit in de regelmatige beoordeling door de klant. Dit aspect zie je ook terug in Scrum.

Nadelen

De praktijk leert dat deze methode vaak misbruikt wordt om aan het begin het project te definiëren, om daarna heel weinig terugkoppeling te hebben met de opdrachtgever. Dat levert meestal een uitwerking van het project op, waar de opdrachtgever niet optimaal tevreden over is, omdat delen niet zo uitgewerkt zijn dat uitwerking aansluit bij hoe de opdrachtgever het wil gebruiken.

Dus de opdrachtgever krijgt niet wat hij wil en nodig heeft, en uiteindelijk kan dat opleveren dat de opdrachtgever er niet voor wil betalen.

Als je deze methode goed uitvoert, zal het prima werken. Echter als je dat niet doet, levert het veel extra documentatie op, die vervolgens niet goed gebruikt worden.

Waterval of Scrum

In de volgende link vind je een uitgebreide beschrijving van voor- en nadelen van beide methodes. Lees dit door zodat je goed begrijpt wat deze zijn:

<https://www.insyde.nl/blog/watervalmethode-vs-scrum/item325>

Scrum in een groep

Om level 3 te kunnen halen ga je in een groep van minimaal 4 personen werken. Het doel van Level 3 is dat je gaat oefenen met het samenwerken met een opdrachtgever. Een docent zal de rol van opdrachtgever op zich nemen. Op het moment dat je wilt starten met deze module heb je een groep nodig van minimaal 4 personen waarmee je jouw groep aanmeld bij een docent. De docenten beslissen dan wie jou opdrachtgever wordt.

Bij het afronden van dit level heb je ook automatisch level 2 van samenwerking behaald.

Afhankelijk van jouw taak kan je ook eventueel level 3 van samenwerking laten aftekenen.

De opdracht

De opdracht wordt bepaald door de opdrachtgever. De scrum groep moet alles gaan inventariseren en op de gebruikelijke scrum werkwijze het project aanpakken. Nu is het ook van belang om een scrum master aan te wijzen.

Uitleg: <http://scrumguide.nl/scrum-master/>

Omdat de groep nu groter is zal er ook een daily scrum meeting moeten plaatsvinden.



Uitleg: <https://www.scrum.nl/Begrippenlijst/daily-scrum/>

Versiebeheer

In deze module leer je hoe je je samenwerking kunt verbeteren door gebruik te maken van branching en forking.

Uitleg

In deze module wordt zoveel mogelijk gebruik gemaakt van icoontjes om iets duidelijk te maken:

	Dit tekstgedeelte is van toepassing als je in de command prompt met Git werkt
	Als je met TortoiseGit werkt moet je deze opdrachten uitvoeren.

Introductie

Normaal gesproken als je lekker in je eentje aan het programmeren bent, en je hebt alle tijd en niemand werkt mee in jouw project, dan kun je rustig je gang gaan. Je kunt daar programmeren waar je wilt, en als het niet werkt: jammer.

Dat wordt anders als je gaat samenwerken. Want als jij iets kapot maakt dan kunnen jouw teamgenoten misschien ook niet verder.

Een van de manieren om dat te voorkomen is het gebruiken van branches.

Wat is een branch?

De letterlijke uitleg is een tak of aftakking. Denk aan een boom. Bij het programmeren betekent het:

“nu, hier in de code, ga ik verder. Het stukje code dat nu komt geef ik zijn eigen naam, en dat programmeren gebeurt niet meer in de hoofdlijn”

Dat is nog niet duidelijk, dus even een voorbeeld:

Voorbeeld situatie

Je werkt aan een website met een paar pagina's. In alle pagina's zit een stukje waarin het menu wordt beschreven. Dat bestand heet navbar.php en wordt in elke pagina gebruikt.

Probleem bij het toevoegen van functionaliteit

Je wilt een nieuwe pagina toevoegen, de contact pagina. Als je dat in de hoofdlijn zou doen (master-branch) dan zou alles wat jij verandert aan navbar.php voor alle teamleden zichtbaar worden zodra zij de code ophalen van de repository. Dus ook als je de pagina toevoegt aan navbar.php voordat je de echte pagina hebt gemaakt. De andere teamleden gaan zeuren dat code nog niet klaar is. En dat wil je niet.

Oplossing

Maakt je een nieuwe branch aan. Je noemt die bijvoorbeeld contactpagina. Daarna ga je aanpassingen maken aan navbar.php, je maakt ook nog een nieuwe pagina aan (contact.php) en misschien nog wel meer.

Als je klaar bent, stuur je in één keer alles terug naar de master-branch, door een merge (samenvoegen van code) te doen.

Verschil tussen branch en commit.

Een branch is een traject dat onafhankelijk is van andere trajecten en waarbinnen je verschillende commits kunt hebben. Dus binnen een ontwikkeltraject van een gedeelte van de software (bijvoorbeeld een functioneel deel, of juist een deel dat de hele userinterface aanpast) kun je weer aan versiebeheer doen. Later kun je, door middel van een merge alles toevoegen aan de masterbranch.

In dit voorbeeld zou je het prima af kunnen met een commit. Maar waar het om gaat: een branch is een manier om twee ontwikkeltrajecten te scheiden. Een branch is groter.

Voorbeelden van branches

Een commit is bedoeld voor een lopend spoor van ontwikkeling, een branch is bedoeld om onderscheid te maken tussen verschillende sporen. Dus bijvoorbeeld:

Master-branch: Is de code die op de website staat



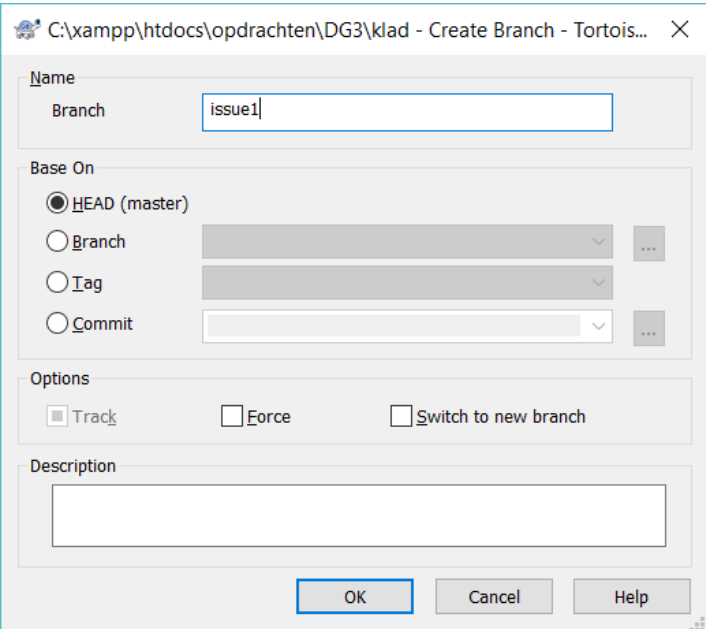
Development-branch: Is de code waaraan gewerkt wordt in het team.

Pas als de opdrachtgever tevreden is over dat stuk, dan wordt dit samengevoegd in de master-branch.

Bij sommige bedrijven/projecten wordt elke scrum-sprint in een branch gezet. Ook worden branches gebruikt om nieuwe technieken of inzichten te testen in bestaande code, vooral als men niet zeker weet of het gaat werken in een project.



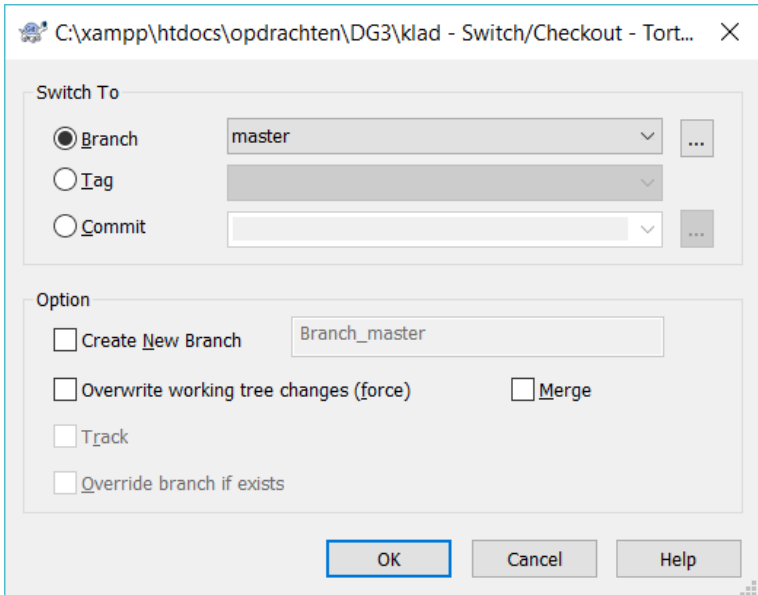
Een branch maken

Het aanmaken van een branch is eenvoudig, hier onder vind je twee mogelijkheden, maar als je een andere GIT client gebruikt zal het vergelijkbaar zijn. Dit kan echter pas als je al een keer een commit gedaan hebt in de werkmapp.

	<p>Ga in de command prompt naar de map waar je project staat, en voer deze opdracht uit:</p> <pre>git branch issue1</pre> <p>Deze opdracht zal een branch aanmaken met de naam "issue1"</p>
	<p>Klik rechts op de map waar je project staat en kies TortoiseGit-> Create Branch</p> <p>Je krijgt dan dit scherm te zien, als je bij Branch "issue1" invult en op OK klikt, zal je een branch aanmaken met de naam "issue1"</p> 



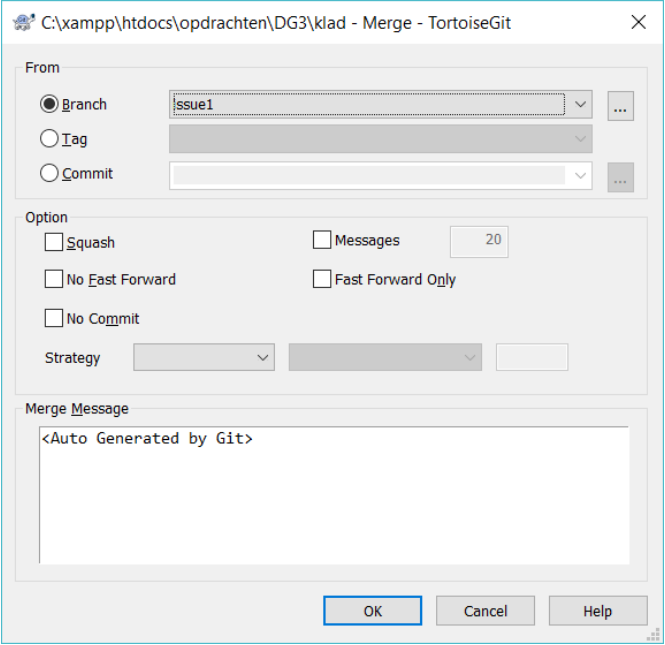
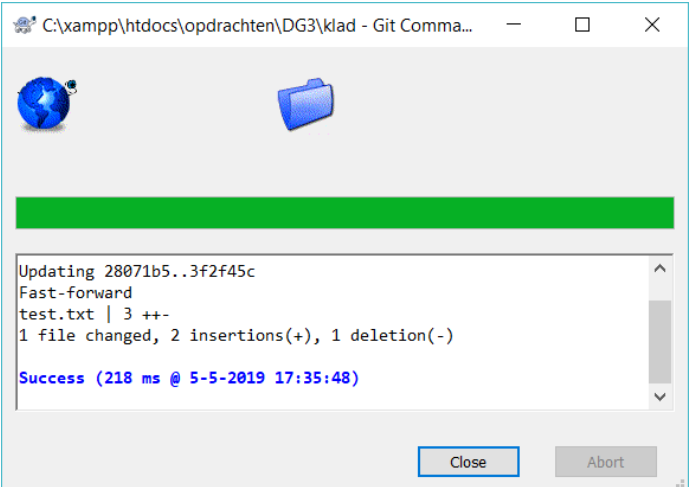
Een branch kiezen

Als je (zoals hierboven) een branch hebben aangemaakt, zal je twee branches hebben. Want standaard heb je altijd al een master-branch. Dit betekent dat je dus nu kunt kiezen tussen de twee branches. Dat doe je met checkout.

	<p>Ga in de command prompt naar de map waar je project staat, en voer deze opdracht uit:</p> <pre>git checkout issue1</pre> <p>Deze opdracht zorgt ervoor dat je omschakelt naar de branch "issue1"</p>
	<p>Klik rechts op de map waar je project staat en kies TortoiseGit-> Switch/Checkout</p> <p>Je krijgt dan dit scherm te zien, je kiest dan in het branch vakje voor "issue1" en als je op OK klikt schakel je om naar de branch "issue1"</p> <div data-bbox="639 920 1401 1512">  </div>

Een branch samenvoegen met een andere branch

Als je in branches werkt, komt er ook een moment dat je code wilt samenvoegen in een andere branch. Je schakelt daarvoor om naar de branch waar je je code **in** wilt samenvoegen en vervolgens doe je afhankelijk van het programma dat je gebruikt dit (of iets vergelijkbaars in andere GIT clients)

	<p>Ga in de command prompt naar de map waar je project staat en schakel om naar de "master" branch. En daarna:</p> <pre>git merge issue1</pre> <p>Dit zorgt ervoor dat je alle wijzigingen die je gedaan hebt in de branch "issue1" verwerkt worden in de branch "master"</p>
	<p>Klik rechts op de map waar je project staat en kies TortoiseGit->Switch/Checkout en schakel om naar de "master" branch. Vervolgens klik je op TortoiseGit->Merge en je krijgt dit scherm:</p>  <p>Hier kies je in de Branch pulldown voor "issue1" en drukt op OK. Je krijgt dan dit scherm te zien, waarin je kunt zien of de merge goed gegaan is.</p> 

Oefening 2

Nu gaan we oefenen met branches en merge.

- Maak een map in je eigen webserver en noem die map Branches. Ga naar die map toe en start hier een nieuwe repository (zie eventueel DG1 hoe dat moet)
- Je begint in de master branch en maakt daar een standaard HTML bestand (**index.html**), die er als hieronder uit ziet. Er zit een foutje in(<<head>), maar die gaan we straks verbeteren. Commit dit.

```
<html>
  <<head>
</head>
  <body>
    <h1>Startpagina</h1>
    <p>Hieronder komen een aantal links te staan</p>
    <ul>
    </ul>
  </body>
</html>
```

- Je bent van plan om een Contactpagina toe te voegen, en dus maak je een branch "contactpagina" aan en schakel om naar die branch.
- Voeg een pagina **contact.html** toe, maak hier ook een nette HTML pagina van met adresgegevens en commit dit.
- Maak in index.html de link aan tussen en naar het bestand dat je net gemaakt hebt en commit dat ook.
- Vervolgens ga je de fout in **index.html** aanpassen. Dus maak je een nieuwe branch aan ("bugfix1") en ga je in die branch de fout verbeteren.
- Daarna ga je beide branches weer samenvoegen in de master-branch.
- Er moeten nog een aantal links toegevoegd worden, dus maak je een nieuwe branch "links_in_index" en gaat daar naar toe. Voeg in index.html de links naar de portal en magister toe. Commit dit.
- Schakel om naar de master branch, en pas daar in index.html de <p> aan naar <h2>
- En tot slot merge links_in_index naar de master branch.

DG3: SAMEN PROJECTEN DOEN II



Ga in de command prompt naar de map waar je project staat en schakel om naar de "master" branch. En daarna:

```
git log
```

Je ziet dan dit:

```
commit 5ec4ccae9b653027b3e3019efb6799ae248a2b29 (HEAD -> master)
Merge: d7a36c6 ffc51b7
Author: Erik Mast <e.mast@drenthecollege.nl>
Date: Sun May 5 18:36:42 2019 +0200
```

```
Merge branch 'links_in_index'
```

```
commit d7a36c6d14c968610b9f94a0f863b4b8aa9929cb
Author: Erik Mast <e.mast@drenthecollege.nl>
Date: Sun May 5 18:35:45 2019 +0200
```

```
p aangepast naar h2
```

```
commit ffc51b79c41782df10c7e0555cfd13cc04b6ac12 (links_in_index)
Author: Erik Mast <e.mast@drenthecollege.nl>
Date: Sun May 5 18:35:01 2019 +0200
```

```
Links toegevoegd
```

```
commit 40d7b9722fb3d36ca5a739f37203f5cfbda81dca
Merge: 80c9985 f30b4a3
Author: Erik Mast <e.mast@drenthecollege.nl>
Date: Sun May 5 18:27:20 2019 +0200
```

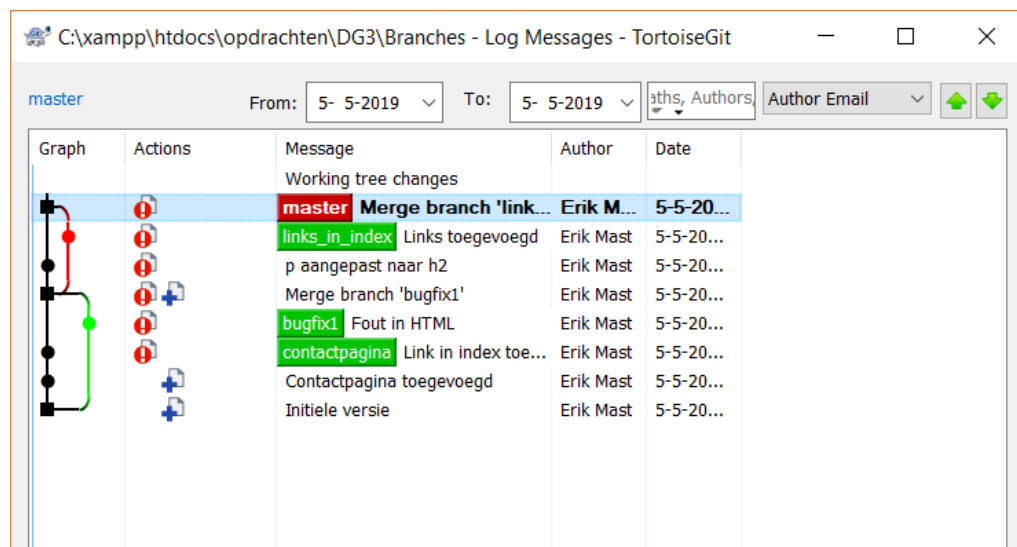
Je zien hier niet het hele overzicht, maar met

```
git log | more
```

zie wel alles. Onderstaande manier is echter overzichtelijker en dus aan te raden.



Klik rechts op de map waar je project staat en kies TortoiseGit-> Show Log. Je zult dat dit zien:



Dit is een soort kaart van wat er gebeurd is in jouw repository.

Eindopdracht

Deze opdracht moet je met minimaal twee en maximaal vier teamleden doen. Je kunt kiezen voor een eigen project (die aan de eisen voldoet) òf de alternatieve opdracht doen.

Je hoeft dus niet beide opdrachten te doen!

Eigen project als opdracht

Je kunt kiezen voor een eigen project of onderstaand project doen. Deze opdracht kun je ook heel goed combineren met bijvoorbeeld B3 (relationele databases) en/of A3 (Java of PHP) maar dan moet je opdracht wel groter maken dan de eindopdracht uit die modules.

Alternatieve opdracht

Je gaat een simpele Social Media site maken, een simpele versie van Facebook:

In deze video legt de opdrachtgever uit wat hij wil:

[Youtube Module DG3](#)

Eisen

- Er moeten minimaal 8 functionele blokken beschreven worden (dus bijvoorbeeld Inloggen, lijst van gebruikers, gebruikersprofiel, forumoverzicht etc) in een definitiestudie.
- Deze moeten gemaakt worden in een door jou gekozen techniek.
- Deze blokken moeten door het team verdeeld worden in 2 (of meer) sprints, op basis van prioriteit.
- Binnen een sprint moeten de functionele blokken aangegeven worden in branches in Git. Aan het eind van elke sprint, moeten alle branches die belangrijk zijn voor het product weer in de hoofdbranche samengevoegd zijn, én je moet een tag toevoegen in Git zodat te zien is dat dit het resultaat van de sprint is.
- Code moet netjes en leesbaar zijn

Voor deze opdracht heb je ongeveer 16 uur (voor een simpel project) – 40 uur (voor een groter project bijvoorbeeld in combinatie met een andere module) de tijd.

De opdracht lever je in in Magister. Zorg altijd de website ingepakt verstuurd wordt (zipfile). Als je niet weet hoe je een zipfile maakt: Klik rechts op de map waar jouw bestanden staan, en kies daarna voor “Kopiëren naar”->Gecomprimeerde (gezipte) map. Er wordt dan een zipfile gemaakt die je kunt opsturen.

Als je combineert met een andere module, moet de zip bij alle modules ingeleverd worden. Elk teamlid moet dit zelf doen.

Bronnen

- ✓ Waterval methode
<https://nl.wikipedia.org/wiki/Watervalmethode>
- ✓ Waterval of Scrum
<https://www.insyde.nl/blog/watervalmethode-vs-scrum/item325>
- ✓ Video voor definitiestudie
<https://youtu.be/Ah-dwcR6dQQ>