Test Document

Het testen van de functionaliteit zal aan de hand van de use cases gebeuren. Omdat deze use cases erg technisch van aard zijn, gebruiken wij hiervoor de gegenereerde log files van de processen.

Aan de hand hiervan, wordt er gekeken of de output overeenkomt met wat er volgends de use cases als post conditie staat. Vanwege het feit dat de verschillende processen veel onderlinge communicatie hebben zullen enkele functionaliteit testen overlappen.

Functionaliteit	Test methode
Opslaan van een URI • Master • Parser	• Opstarten van een Master & Parser proces, er voor zorgend dat er tenminste 1 document in de document queue (database) staat. • Indien geen document beschikbaar, ook een Crawler opstarten.
Ophalen van URI uit de Queue • Master • Crawler	• Opstarten van een Crawler & Master proces en kijken in de database of de URI queue verminderd
Opslaan van een document • Master • Crawler	• Opstarten van een Master & Crawler • Kijken in de database of er een nieuw document is toegevoegd die als URI heeft wat er eerst uit de URI queue was gehaald

Parsen van een document
• Parser
	• Opstarten Master & Parser
	• Kijken in de logs van de parser wat de output van het geparsed document is:
	  ○ Kijken naar hoeveel URIs gevonden
	  ○ Kijken naar hoeveel worden gevonden
	• Kijken in de database of er een nieuwe index is toegevoegd
	• Kijken in de database of er nieuwe URIs zijn toegevoegd

Ophalen van een document vanaf het web
• Crawler
	• Bekijken van de logs van de crawler
	  ○ Bekijk verkregen URI
	  ○ Bekijk resultaat van gecrawlede URI

Zoeken op website • Search Engine	*Kan alleen worden gedaan indien er indices staan in de database* • Typ word in de zoekbalk • Wacht tot resultaten terug komen • Vergelijk resultaten met een zelfde query op het getypde word in de database

Opslaan van een URI in de queue, ophalen van een URI uit de queue

Om een URI op te halen uit de queue, moet er een master en een crawler aanstaan. In eerst instantie is er gemeten hoeveel URI's er in de queue staan. Dit wordt gedaan met de volgende query:

SELECT COUNT(uri_id) FROM uri_queue

Het resultaat hiervan is 758 op het moment van schrijven. Vervolgens vraagt het crawler proces een URI op. De URI wordt dan verwijderd uit de URI queue. De URI queue staat dan op 757.

Om de URI queue aan te vullen wordt er een document door de parser heen gehaald. Deze haalt alle URI's eruit. Om dit in een test te doen wordt het document van URI, die in het proces hierboven gecrawld is, door de parser gehaald. Deze haalt vervolgens 41 URI's uit het document. Deze 41 URI's worden weer toegevoegd aan de URI queue. Als we dan weer een count doen op de database met de query die hierboven is gebruikt komen we uit op 794. Dit staat ook letterlijk in de database op het moment van schrijven.

Tellen van het aantal woorden

Nu kunnen we aan de hand van de indices behalen hoeveel unieke woorden er gevonden zijn op de webpagina. Door deze query uit te voeren weten hoeveel woorden een bepaalde URI bevat:

SELECT uris.uri, count(indices.keyword) FROM indices INNER JOIN uris ON indices.uri_id = uris.uri_id GROUP BY uris.uri_id

Het resultaat 194 unieke woorden bij URI: "http://www.cs.mun.ca/~donald/msc/node11.html"

Opslaan van een document

Om een document op te slaan moet er een crawler en een master draaien. De crawler vraagt een URI op en crawlt deze vervolgens. Hieruit komt een document. Dit document wordt met bijbehorende URI naar de master gestuurd. De master slaat vervolgens het document op.

De crawler bevat nu een log na het crawlen van een URI.

[2014-06-11 12:54:30] INFO: Starting to crawl using ip: 192.168.100.14
[2014-06-11 12:54:31] INFO: Sent valid link: http://en.wikipedia.org/wiki/Mathematical_model

Nu kunnen we kijken of de document_queue een document bevat. Hiervoor wordt deze query gebruikt:

SELECT uris.uri, document_queue.content FROM document_queue INNER JOIN uris ON uris.uri_id = document_queue.uri_id

Hier komt vervolgens een URI en een document uit. Voor de leesbaarheid is het document aan helemaal als laatste te vinden in de bijlage. Dit document is niet volledig toegevoegd omdat het meer dan 40 pagina's bevat. Vandaar een klein begin en een klein einde. Nu weten we ook dat de crawler met succes een document van het internet heeft gehaald.

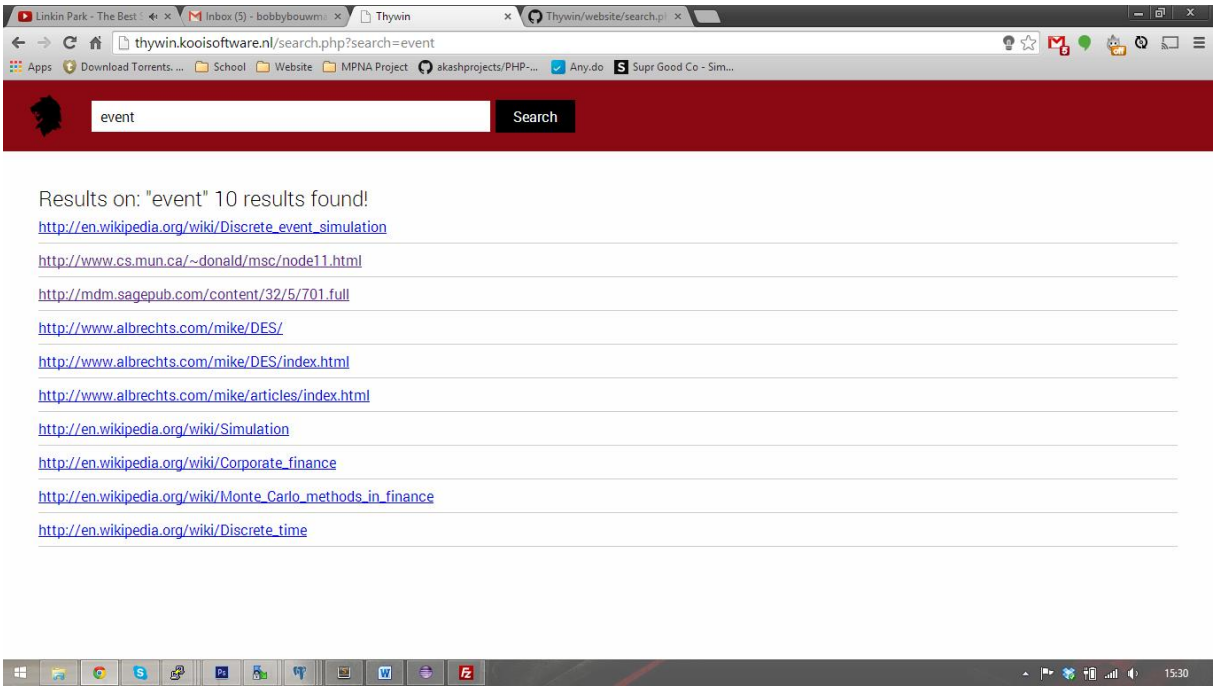
Zoeken op de website

De datbase is nu gevuld met indices, nu kan er gezocht worden op de website door een zoekterm in te typen. Voor het testen is de database niet heel erg vol, maar zo kunnen wij de resultaten nog overzien. We gaan nu het woord “event” zoeken op de website. Dit kunnen wij ook doen in de database met een eigen query:

SELECT uri FROM indices INNER JOIN uris ON indices.uri_id = uris.uri_id WHERE indices.uri_id IN (SELECT uri_id FROM indices WHERE keyword = 'event') GROUP BY uris.uri_id

http://www.cs.mun.ca/~donald/msc/node11.html
http://mdm.sagepub.com/content/32/5/701.full
http://en.wikipedia.org/wiki/Discrete_event_simulation
http://www.albrechts.com/mike/DES/
http://en.wikipedia.org/wiki/Simulation
http://en.wikipedia.org/wiki/Discrete_time
http://www.albrechts.com/mike/articles/index.html
http://www.albrechts.com/mike/DES/index.html
http://en.wikipedia.org/wiki/Monte_Carlo_methods_in_finance
http://en.wikipedia.org/wiki/Corporate_finance

Dit doet de searchengine ook zo ongeveer, alleen wordt deze nog gesorteerd op relevantie.



Bijlage

<!DOCTYPE html>
<html lang="en" dir="ltr" class="client-nojs">
<head>
<meta charset="UTF-8" />
<title>Mathematical model - Wikipedia, the free encyclopedia</title>

…

<script
src="//bits.wikimedia.org/en.wikipedia.org/load.php?debug=false&lang=en&modules=site&only=scripts&skin=vector&*"></script>

<script>if(window.mw){

mw.config.set({"wgBackendResponseTime":261,"wgHostname":"mw1180"});

}</script>

</body>
</html>