Thywin – Happlan

Geschreven door: Bobby Bouwmann, Thomas Gerritsen, Thomas Kooi, Erwin Janssen, Imre Woudstra

Datum: 17-4-2014

Project begeleider: Alice de Groot

Opdrachtgever: Joost Kraaijeveld

Inleiding

Dit plan is gemaakt door projectgroep Thywin voor de projectfase van MPNA, 2014. Het doel van het project betreft een gespecialiseerde zoek engine en crawler. Het gespecialiseerde gedeelte zal in ons geval vijf onderwerpen bevatten: probabilistic, logic, discrete, event en simulation.

In dit document wordt het project beschreven en hoe wij als groep dit zullen uitvoeren.

Versiebeheer

Versie	Naam	Datum	Bijzonderheden
0.1	Bobby	15-4-2014	Eerste opzet Hanplan + Projectmanagementmethode + Projectoverzicht toegevoegd
0.2	Thomas K	15-4-2014	Projectorganisatie toegevoegd
0.3	Thomas G	15-4-2014	Inleiding + Projectoverzicht toegevoegd
0.4	Imre	15-4-2014	Risicoanalyse toegevoegd
0.5	Erwin	16-4-2014	Kwaliteitseisen toegevoegd
1.0	Bobby	16-4-2014	Planning toegevoegd + controle document
1.1	Thomas	17-4-2014	Zinsopbouw aangepast
1.2	Bobby	25-4-2014	Risico's + codekwaliteit begewerkt
1.3	Bobby	8-5-2014	Feedback verwerkt Joost en Alice

Inhoudsopgave

Inleiding 2
Versiebeheer 2
Inhoudsopgave 3
1. Projectoverzicht 5
4.1 Doel 5
4.2 Randvoorwaarden 5
4.3 Hulpmiddelen / tooling 5
2. Projectmanagementmethode 6
3. Functionele requirements 7
4. Planning 8
5. Projectorganisatie 9
5.1 Afspraken 9
5.2 Organisatie Structuur 9
5.3 Rollen 10
5.4 Communicatie 10
6. Kwaliteitseisen 11
6.1 Meetbare kwaliteit 11
6.2 Definition of Done 12
6.3 Niet meetbare kwaliteit 12
6.4 Huisstijl 13
7. Risicoanalyse 14
8. Plannen voor betere relevantie bepaling 15
9. Analyse 16
9.1 Analyse Datastore 16
9.2 Analyse Indexering 17
9.3 Analyse Dedicated Crawler en Parser vs CrawlerParser 18
10. Bijlage 20
10.1 Bijlage 1 (JSF Code Styles) 20

11. Bibliografie 22

1. Projectoverzicht

4.1 Doel

Het doel van dit project is om zoekmachine te bouwen die, op basis van een aantal steekwoorden, relevante pagina's van het internet verzameld.

4.2 Randvoorwaarden

- Elk lid beschikt over een computer/laptop
- Er zijn tenminste 3 Raspberry pi's beschikbaar
- Opdrachtgever is beschikbaar voor feedback

4.3 Hulpmiddelen / tooling

Software

- Git (Github)
- Raspbian
- Eclipse IDE for C/C++ Developers
- Cygwin
- Gnu tool chain

Hardware

- Raspberry pi's
- Ethernet kabels
- Switch voor meerdere connecties
- Externe Harddrive

2. Projectmanagementmethode

In het project maken wij gebruik van de projectmanagent methode HAP. HAP staat voor Highperformance Agile Projectmanagementmethode en is gevormd uit de projectmanagementmethode Scrum en EVO. Het voordeel van HAP is dat de projectgroep zelf kan kiezen welke onderdelen van Scrum en EVO gebruikt worden tijdens het project. Hierdoor is HAP heel erg flexibel.

Net zoals Scrum en Evo wordt er gewerkt met korte iteraties, hierdoor is er veel overleg met de opdrachtgever. In dit project is het uitgangspunt iteraties van één week.

Aan het eind van elke iteratie vindt er een oplevering plaats waar de opdrachtgever feedback geeft en er afspraken worden gemaakt over de volgende oplevering en wat daar wordt opgeleverd. Na elke iteratie zal er een retrospective gehouden worden, hierbij komen de goede en slechte punten naar voren.

3. Functionele requirements

De requirements zijn gemaakt op basis van de opdracht van de opdrachtgever. Tijdens het eerste gesprek met de opdrachtgever zijn een aantal belangrijke punten naar voren gekomen. Deze zijn verwerkt in requirements en geprioriteerd met MoSCoW.

Nummer	Beschrijving	Prioriteit(MoSCoW)
1	Crawlen op een specifiek onderwerp op het internet	Must
2	Webinterface zoekemachine	Must
3	Indexeren van opgehaalde pagina's	Must
4	Opslaan van geindexeerde data	Must
5	Verwerken van HTML document	Must
6	Systeem draait op Rasbian	Must
7	Robot.txt	Should
8	Zoekresultaten zijn binnen 10 seconden geladen	Must

4. Planning

De globale planning geeft een overzicht van de geplante hoofdactiviteiten per iteratie.

Iteratie	Activiteiten	Week
1	PSU Analyse datastore Analyse Indexering Hap-plan Github organiseren Inrichten ontwikkelomgeving Inrichten raspberry pi's	1 t/m 2
2	Opzet parser Opzet crawler Opzet master Document queue URL queue Document vergelijker met vectoren Communicatie crawler > master Communicatie master > parser	3
3	Database opzetten Links extracten van document Communicatie Protocol Design Document Communicatie protocol implementeren Content-Type filter	4
4	Database implementeren Exception handling Indexeren URL queue uitbreiden en database klaarmaken Document queue uitbreiden en database klaarmaken Optimaliseren parser	5
5	Zoeken (website) Optimaliseren parser Testen (test-cases)	6 t/m 7
6	Oplevering	8

5. Projectorganisatie

5.1 Afspraken

1. Werktijden:
    a. Start: 9:00
    b. Eind: 16:00, indien niet op schema tenminste 17:00
    c. Pauze: 12:00 t/m 12:30
2. Elke ochtend om 9:00 wordt er begonnen met een daily stand-up. Deze zal niet meer dan 15 minuten duren en wordt uitgevoerd volgends de Scrum project management methode.
3. Indien een groepslid te laat dreigt te komen, na 9:00, zal dit groepslid de rest van de groep hier zo snel mogelijk van op de hoogte brengen.
4. Er wordt binnen de groep, maar uiteraard ook naar buiten, respectvol met elkaar omgegaan.
5. Telefoneren gebeurt niet in de werkruimte.
6. De werk locatie is D1.03d. Indien deze ruimte niet beschikbaar is, kan er uitgeweken worden naar het gereserveerd lokaal of wordt er een andere plek op de HAN gezocht.
7. Voor eventuele apparatuur, zoals de Raspberry pi's, is er een kluis beschikbaar op de HAN. Hierin wordt het materiaal geplaatst dat altijd aanwezig moet zijn.
8. Indien een groepslid een gehele dag gepland afwezig zal zijn, wordt dit minimaal een dag van te voren tijdens de daily stand-up gemeld.
9. De analyse, beslissingen en eventuele onderzoeken worden gedocumenteerd.

5.2 Organisatie Structuur

De organisatie binnen de projectgroep is los opgebouwd volgens het scrum model. Dit betekend dat iedereen binnen de projectgroep gelijkwaardig is. Eventuele verdere aanvullen is te vinden in het onderdeel projectmanagementmethode.

Binnen de projectgroep zijn er 3 rollen: Scrum master, Product Owner by Proxy, en Team lid.

- De scrum master doet de daily standup leiden en de planning bewaken. Hiernaast is de scrum master ook nog gewoon team lid.
- De rol van product owner by proxy zal binnen dit project vallen bij de gehele projectgroep.
- Iedereen binnen de projectgroep is teamlid.

5.3 Rollen

De rollen binnen de project groep zijn:

Naam	Rol
Bobby Bouwmann	Scrummaster
Erwin Janssen	Sfeerbewaker
Thomas Gerritsen	Sfeerbewaker
Imre Woudstra	Contactpersoon
Thomas Kooi	Contactpersoon

Verantwoordelijkheden

Andere verantwoordelijkheden binnen het project zijn die van huisstijlbewaker. Deze is opgesplitst in documenten en code huisstijl.

De code zal worden bewaakt door Erwin Janssen en Imre Woudstra. Bobby Bouwmann zal de huisstijl bewaking over documenten doen.

Onder huisstijl verstaan wij: Opmaak en layout.

5.4 Communicatie

De interne communicatie, binnen de groep, zal verlopen via telegram. Eventuele back-up communicatie loopt via: Whatsapp, sms of email.

De externe communicatie wordt afgehandeld door de contactpersonen van het project (Imre Woudstra & Thomas Kooi).

Naam	Email	Telefoon
Bobby Bouwmann	bobbybouwmann@gmail.com	06-49669828
Erwin Janssen	erwinjanssen@outlook.com	06-40214006
Thomas Gerritsen	Thomas.gerritsen@live.nl	06-22289546
Imre Woudstra	Imre.woudstra@gmail.com	06-29491627
Thomas Kooi	thomasskooi@live.nl	06-34871262
Alice de Groot	alice.degroot@han.nl	06-55328629

Competenties

Naam	Gekozen Domein competentie	Verplicht
Bobby Bouwmann	Analyseren	Realiseren
Erwin Janssen	Ontwerpen	Realiseren
Thomas Gerritsen	Analyseren	Realiseren
Imre Woudstra	Analyseren	Realiseren
Thomas Kooi	Ontwerpen	Realiseren

6. Kwaliteitseisen

In dit hoofdstuk zal worden besproken welke kwaliteitseisen er gesteld zullen worden aan de producten die tijdens dit project opgeleverd zullen worden. Hierbij zal ook vermeld staan op welke manier de kwaliteit van de betreffende eis zal worden gewaarborgd.

Er wordt een onderscheid gemaakt tussen twee verschillende producten: documenten en code. Er wordt daarnaast ook een onderscheid gemaakt tussen twee soorten kwaliteit: meetbaar en niet meetbaar.

6.1 Meetbare kwaliteit

Meetbare kwaliteit is datgene wat hard vast te stellen is en niet onderworpen is aan de subjectiviteit van de opdrachtgever of de projectgroep.

Documenten

Kwaliteitseis	Bewaking
Documenten bevatten geen spelfouten.	Spellingscontrole van de tekstverwerker, controle door de Huisstijlbewaker Documenten en controle door procesbegeleider.
Documenten voldoen aan de huisstijlregels zoals gedefinieerd onder de kop huisstijl.	De Huisstijlbewaker Documenten zal documenten nakijken en controleren op de huisstijl.

Code

Kwaliteitseis	Bewaking
Code in de repository bevat geen errors.	De Huisstijlbewaker Code zal alle code die in de repository staat compileren en controleren op errors.
Code compileert, met het hoogste warning niveau, zonder warnings.	De Huisstijlbewaker Code zal alle code die in de repository staat compileren en controleren op warnings.
Code kwaliteit is hoog.	Gebruik van tools die code op kwaliteit controleert. De opmerkingen van deze tool worden verwerkt zodat de kwaliteit van de code zo hoog mogelijk wordt. De Huisstijlbewaker Code kan eventueel aanpassingen in deze controle maken. Afhankelijk van de gebruikte tool wordt een cijfer / percentage als doel gesteld.
De applicatie is snel / efficiënt.	Tijdsmetingen uitvoeren en de resultaten hiervan langs de performance requirements houden.
Code voldoet aan de huisstijlregels zoals gedefinieerd onder de kop huisstijl.	De Huisstijlbewaker Code zal code nakijken en controleren op de huisstijl.
Code voldoet aan de door ons gekozen eisen uit de JSF standaard	Zie bijlage 1 (JSF Code styles) voor de eisen.

6.2 Definition of Done

Crawler

De crawler is klaar op het moment dat deze met succes een uri kan ontvangen van de master en deze verwerken. Deze uri gebruikt de crawler vervolgens op een pagina te downloaden van die url. Deze pagina wordt samen met de uri opgestuurd naar de master. Deze actie blijft gebeuren tot de master geen uri's meer stuurd.

Parser

De parser is klaar op het moment dat deze de html pagina kan verwerken en kan bepalen op relevantie. Alle uri's worden uit het html bestand gehaald. Daarna wordt gekeken of het bestand relevant is of niet, zo ja dan gaan we deze indexeren en andere wordt het html bestand verwijderd. Als de relevantie is bepaald worden de uri's opgeslagen met deze relevantie.

Master

De master is klaar op het moment dat deze een lijst met uri's en een lijst met documenten kan bijhouden. Ook wel queue's genoemd. De master kan uri's of documenten versturen naar een crawler of parser op het moment dat hierna een request wordt gedaan. De master moet ook zoek requesten kunnen verwerken van de website af.

6.3 Niet meetbare kwaliteit

Documentatie

Kwaliteitseis	Bewaking
Documenten zijn geschreven met consequent woord- en taalgebruik.	De Huisstijlbewaker Documenten leest alle documenten door en zorgt voor een consequent woord- en taalgebruik.
Documenten zijn bevatten formele en professionele teksten.	Controle door Huisstijlbewaker Documenten en procesbegeleider.

Code

Kwaliteitseis	Bewaking
Code is leesbaar en bevat structuur	Wekelijkste codereview door andere projectleden en controle door Huisstijlbewaker Code.
Code is goed gedocumenteerd	Wekelijkste codereview door andere projectleden en controle door Huisstijlbewaker Code.

6.4 Huisstijl

Om consistentie aan te brengen in de producten die gedurende dit project opgeleverd gaan worden, zijn in deze paragraaf huisstijl regels gedefinieerd over de opmaak en inhoud van de documenten en code.

Opmaak

Teksttype	Font	Tekstgrootte	Tekstkleur
Kop 1	Calibri	24	RGB(138,9,18)
Kop 2	Calibri	18	RGB(138,9,18)
Kop 3	Calibri	14	RGB(138,9,18)
Kop 4	Calibri	12	RGB(138,9,18)
Standaardtekst	Calibri	11	Zwart

Aan de kop en voet het documenten zijn gekleurde balken te vinden. De vulkleur van de balk is RGB(138,9,18), de randkleur is RGB(183,138,14). Het gebruikte logo is de onderstaande afbeelding.

Taalgebruik

- Formeel taalgebruik in de derde persoon. Er wordt gesproken over “de projectgroep” en “de opdrachtgever” in plaats van “wij” , “jij”, “u”.

Code

- Accolades op aparte regel.
- Defines en contanten volledig in hoofdletters en liggend streepje tussen de woorden. (VOORBEELD_NAAM)
- Voor public variabelen en functies maken wij gebruik van PascalCase (VoorbeeldNaam)
- Voor private variabelen en functies maken wij gebruik van CamelCase (voorbeeldNaam)

7. Risicoanalyse

Risico	Impact	Kans	Uitwijkstrategie
Niet alle requirements zijn vanaf het begin duidelijk	Middel	Middel	Prioriteit ligt bij de bekende requirements.
Requirements die technisch moeilijk te implementeren zijn	Middel	Middel	De requirement op een andere manier beschrijven en ontwikkelen of helemaal weglaten, besluit in overleg met de opdrachtgever.
De grote en de complexiteit van het systeem	Middel	Middel	De ontwikkeling beperken, bepaalde features of requirements achterwegen laten.
De complexiteit van het testen van het systeem	Klein	Klein	Unit-testen beperken. Wel testen op verwachte resultaten.
Het falen van gebruikte libraries en code van andere	Middel	Klein	Een alternatieve library of alternatieve code zoeken.
Tijdsbeperking voor het unit-testen van code	Klein	Middel	In overleg met de opdrachtgever bepaalde features achterwegen laten zodat er getest kan worden op belangrijkere punten.
Ontwikkelsysteem is anders dan het doelsysteem	Groot	Middel	Ontwikkelen op de raspberry pi's. Dit vertraagt de ontwikkeling vanwege geen GUI.
Ontwikkelsysteem ondersteunt niet alle aspecten van het programma	Groot	Klein	Ontwikkelen op de raspberry pi's. Dit vertraagt de ontwikkeling vanwege geen GUI.
Huidige kennis van teamleden valt niet samen met onderwerp van het project	Klein	Groot	Teamleden moeten zich eerst verdiepen in de nieuwe stof. Dit kan wel extra vertraging opleveren.
Teamleden hebben geen kennis van de tools die gebruikt worden	Klein	Middel	Teamleden moeten zich eerst verdiepen in de nieuwe stof. Dit kan wel extra vertraging opleveren.

8. Plannen voor betere relevantie bepaling

Om een betere relevantie van data te bepalen hebben wij een aantal mogelijkheden bedacht. Deze onderdelen worden hieronder uitgewerkt. Vaak zijn de onderdelen geschikt voor het parsen van het gecrawlde document en voor de zoekterm van de gebruiker op de website.

Relevantie bepalen met betrekking tot het onderwerp

Hiermee willen wij relevantie van het onderwerp (de definieerde woorden) meenemen in het resultaat van het zoeken. Dit kan bijvoorbeeld met een dergelijke formule.

$$\frac{(ratio1 * subjectrelevance) + (ratio2 * queryRelevance)}{(ratio1 + ratio)}$$

'subjectRelevance' = document vector van het onderwerp.
'queryRelevance' = document vertor van zoekterm.

Aantal externe referenties

Het aantal externe links dat naar een pagina wijst, kan iets zeggen over deze pagina. Deze waarde kan gebruikt worden bij het bepalen van de relevantie.

Relevantie op basis van paginaonderwerp

Bij deze vorm van relevantie kunnen we de meta data, het title element en de html kop tags gebruiken. Ook kan er in de uri relevante informatie staan.

Bepalen van de belangrijkste zoekterm

In de searchquery zijn bepaalde woorden belangrijker dan anderen, zo zijn in de meeste gevallen lidwoorden minder belangrijk dan zelfstandig naamwoorden.

Zoekterm bepalen aan de hand van de geschiedenis

Als een gebruiker bijvoorbeeld binnen een bepaalde (korte) tijd twee zoek termen gebruikt bijvoorbeeld "Computer" en "Chips" dan zou de gebruiker "Computer chips" verwachten en niet "Lays patato chips".

Rating systeem voor websites

Dit is gebaseerd op het aantal klikken op een bepaalde uri. Hoe vaker een bepaalde link is geklikt hoe belangrijker deze link mogelijk is voor andere gebruikers. Dit zou ook handmatig kunnen door de gebruiker met een "like" systeem.

9. Analyse

9.1 Analyse Datastore

Deze analyse gaat over het opslaan van gecrawlde data. Deze data bevat relevante informatie over een webpagina en wat hier wel en niet aan belangrijk is.

Flat file

Flat file in de simpelste vorm van het opslaan van data. Het voordeel hiervan is dat het makkelijk gebruikt kan worden op andere systemen. Hier is alleen een tekst editor voor nodig. Het is makkelijk om data toe te voegen, aangezien je gewoon aan het einde kan bijschrijven. Het nadeel van dit systeem is dat zoeken hierin lastig is, aangezien er geen gebruik gemaakt wordt van relaties tussen bestanden. Er kunnen geen van wiskunde Het meest voorkomende formaat is CSV (Comma Separated Variable).

Databases

Een database is daarentegen niet geheel simpel. Het is wel een flexibele en efficiënte manier van het opslaan van data. Ook kunnen er query's worden uitgevoerd op de database. Hiermee wordt er een vraag aan de database gesteld en de database stuurt deze data dan terug. Als het goed ontworpen is, is er weinig kans op dubbele data. Nadelen van databases zijn de techniek en de kennis die je ervoor moet bezitten. Als het niet goed ontworpen is kan het de efficiëntie verlagen.

Vele relationele database management systemen kunnen gebruikt worden. Hieronder een aantal punten om voor die database te kiezen.

PostgreSQL

Ook wel postgres. Postgres is ontwikkeld met het oog op features en standaarden. De nieuwere versies van postgres maken gebruik van compressie van data, waardoor er minder IO nodig is en het dus sneller is. Door de vele features kan postgres gemakkelijk geconfigureerd worden waardoor de performance kan toenemen. Sinds postgres 9.2 is het ook sneller met de COUNT(*) functie, dit komt door de index-only scan support. Postgres staat ook bekend als de open source versie van Oracle.

MySQL

Mysql is ontwikkeld met het oog op snelheid. MySQL maakt bijvoorbeeld gebruik van query cache. Hierdoor kunnen recente query's sneller worden terug gegeven. MySQL kan gebruik maken van 'INSERT IGNORE'. Deze functie voegt een rij toe als deze niet bestaat. Ook kan MySQL gebruik maken van 'REPLACE'. Deze functie vervangt de huidige rij met een nieuwe rij. Postgres heeft deze functionaliteit niet.

Conclusie

Voor beide is genoeg support om het simple werkend te krijgen op de Raspberry Pi. Postgres is een stuk verder ontwikkeld dan MySQL. Ook speelt het mee dat de opdrachtgever een groot fan is van postgres. De keuze ligt dus bij postgres.

9.2 Analyse Indexering

Mogelijke indexeringen:

Er zijn twee mogelijke manieren voor het indexeren van een HTML pagina, genaamd inverted indexing en forward indexing.

Inverted Index

- Er wordt bij gehouden welke documenten / HTML pagina's, horen bij een bepaald word.
- Dit kan alleen vast stellen dat een wordt binnen een document voor komt, er is geen informatie over hoe vaak en de positie (zo genaamd boolean index).
- Er is een update bottleneck
- Minder storage benodigd
- Form van hash tabel

Word	Documents
laptop	Document 1, Document 3, Document 4, Document 5, Document 7
desktop	Document 2, Document 3, Document 4
search	Document 5
engine	Document 7

Forward Index

- Er wordt voor elke document bij gehouden welke woorden er in voor komen
- Goed voor asynchronous systemen
- Makkelijk te converteren naar inverted index

Document	Words
Document 1	Laptop, Desktop
Document 2	Toetsenbord, kabel
Document 3	Desktop, lamp

Document Parsing

- Documenten opbreken in aparte componenten (WORDEN / TOKENS).
- Woorden worden tokens genoemd
- Document formats
    - Welke format is de opgehaalde document maakt uit voor hoe deze wordt geparsed.
    - Abusing document formatting : Spam dexing. Bijv; worden die verborgen zijn op een computer scherm maar niet voor een indexer.

Prioriteit van tokens

- Bij HTML kan er prioriteit van tokens worden gebruikt
- H1, H2, etc. <title>, <p>, etc..

Hiermee kan de relevantie van een token bepaald worden. Een token in een H1 tag is mogelijk belangrijker als een zelfde token binnen een P tag.

Hierbij kunnen extra punten voor relevantie binnen het document bepaald worden.

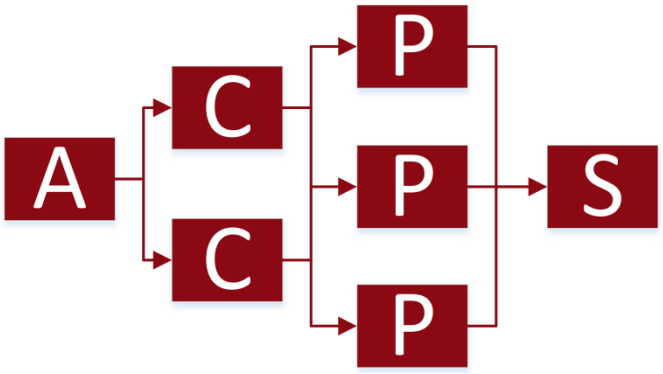
9.3 Analyse Dedicated Crawler en Parser vs CrawlerParser

Voor de verdeling van taken crawlen en parsen ontstond de keus tussen een opstelling met dedicated crawlers en parsers en een opstelling waarin een Raspberry Pi de rol van zowel crawler als parser op zich nam. In deze analyse wordt de keuze gemaakt tussen deze twee opties.

Legenda: A = Aansturing, C = Crawler, P = Parser, S = storage, CP = CrawlerParser

Dedicated Crawler en Parser

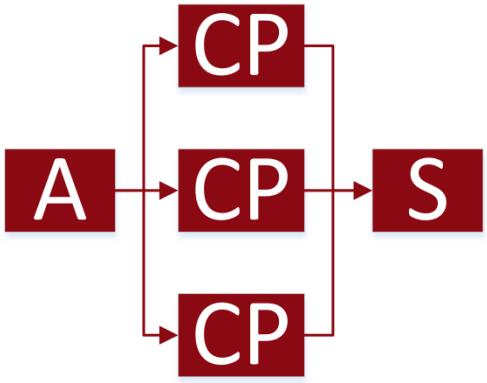
In deze opstelling zijn er Raspberry Pi's wiens enige taak crawlen of parsen is. De architectuur die hierbij hoort wordt met de afbeelding hiernaast beschreven. De kenmerken van deze opstelling staan hieronder.



- Elke Raspberry Pi is dedicated in zijn taak.
- Modulariteit is hoog.
- Externe load balancer is nodig.
- Meer netwerkverkeer.
- Het systeem is afhankelijker van de crawlers, vooral als er slecht een enkele crawler is.
- Ongelijke verhouding tussen de crawlers en de parsers.
- Crawlers en parsers moeten potentieel op elkaar wachten.

CrawlerParser

In deze opstelling vervullen de Raspberry Pi's de functie van zowel crawler als parser. De architectuur die hierbij hoort staat in de afbeelding hiernaast. De kenmerken van deze opstelling staan hieronder.



- Verhouding tussen het aantal crawlers en het aantal parsers is 1:1
- Pi's hebben meerdere functies en moeten hiertussen schakelen.
- Minder netwerkverkeer, meer intern verkeer.
- Interne load balancer.
- Hoeft meestal niet te wachten op een crawler of parser.

Conclusie

Uiteindelijk is gekozen voor de losse crawler en losse parser aanpak. Een belangrijke rede was dat deze methode modulair is. Bij de CrawlerParser opstelling wordt intern aan loadbalancing gedaan, hierdoor hoeft er ook niet nagedacht te worden over de verhouding tussen de crawlers en de parsers en kunnen er CrawlerParsers worden toegevoegd. Bij de opstelling van een dedicated crawler en parser kunnen er momenten zijn dat de crawler moet wachten op een parser, omdat de parser data niet zo snel kan verwerken als de crawler deze kan aanleveren. Omdat deze methode modulair is kunnen er gemakkelijk meerdere parsers of crawlers aangezet worden. Bij de opstelling van een dedicted crawler en parser moet er ook rekening gehouden worden met het aantal crawlers vs het aantal parsers, of er moet een externe loadbalancer zijn die een Raspberry Pi laat crawlen of parsen.

10. Bijlage

10.1 Bijlage 1 (JSF Code Styles)

De regels en blz. verwijzen naar http://www.stroustrup.com/JSF-AV-rules.pdf

Nr.	Blz.	Rule
1	12	Any one function (or method) **will** contain no more than 80 logical source lines of code (L-SLOCs).
2	12	There **shall not** be any self-modifying code.
14	18	Literal suffixes **shall** use uppercase rather than lowercase letters.
30	21	The *#define* pre-processor directive **shall not** be used to define constant values. Instead, the *const* qualifier **shall** be applied to variable declarations to specify constant values.
34	22	Header files **should** contain logically related declarations only.
35	22	A header file **will** contain a mechanism that prevents multiple inclusions of itself.
37	22	Header (include) files **should** include only those header files that are required for them to successfully compile. Files that are only used by the associated .cpp file should be placed in the .cpp file—not the .h file.
39	23	Header files (*.h) **will not** contain non-const variable definitions or function definitions.
42	24	Each expression-statement **will** be on a separate line.
49	25	All acronyms in an identifier **will** be composed of uppercase letters.
53	26	Header files **will** always have a file name extension of ".h".
54	27	Implementation files **will** always have a file name extension of ".cpp".
56	27	The name of an implementation file **should** reflect the logical entity for which it provides definitions and have a ".cpp" extension (this name will normally be identical to the header file that provides the corresponding declarations.) At times, more than one .cpp file for a given logical entity will be required. In these cases, a suffix should be appended to reflect a logical differentiation.
57	27	The public, protected, and private sections of a class **will** be declared in that order (the public section is declared before the protected section which is declared before the private section).
58	27	When declaring and defining functions with more than two parameters, the leading parenthesis and the first argument **will** be written on the same line as the function name. Each additional argument **will** be written on a separate line (with the closing parenthesis directly after the last argument).
59	28	The statements forming the body of an *if, else if*, *else, while, do…while* or *for* statement **shall** always be enclosed in braces, even if the braces form an empty block.
60	28	Braces ("{}") which enclose a block **will** be placed in the same column, on separate lines directly before and after the block.
61	28	Braces ("{}") which enclose a block **will** have nothing else on the line except comments (if necessary).
62	28	The dereference operator '*' and the address-of operator '&' **will** be directly

		connected with the type-specifier.
63	28	Spaces **will not** be used around '.' or '->', nor between unary operators and operands.
64	29	A class interface **should** be complete and minimal.
69	30	A member function that does not affect the state of an object (its instance variables) **will** be declared *const*. Member functions should be *const* by default. Only when there is a clear, explicit reason should the *const* modifier on member functions be omitted.
73	31	Unnecessary default constructors **shall not** be defined.
84	33	Operator overloading **will** be used sparingly and in a conventional manner.
98	38	Every nonlocal name, except main(), **should** be placed in some namespace.
108	40	Functions with variable numbers of arguments **shall not** be used.
115	41	If a function returns error information, then that error information **will** be tested.
120	43	Overloaded operations or methods **should** form families that use the same semantics, share the same name, have the same purpose, and that are differentiated by formal parameters.
125	44	Unnecessary temporary objects **should** be avoided.
126	44	Only valid C++ style comments (//) **shall** be used. For functions we use (///).
127	45	Code that is not used (commented out) **shall** be deleted.
129	45	Comments in header files **should** describe the externally visible behavior of the functions or classes being documented.
136	46	Declarations **should** be at the smallest feasible scope.
143	47	Variables **will not** be introduced until they can be initialized with meaningful values.
152	49	Multiple variable declarations **shall not** be allowed on the same line.
160	51	An assignment expression **shall** be used only as the expression in an expression statement.
162	51	Signed and unsigned values **shall not** be mixed in arithmetic or comparison operations.
182	55	Type casting from any type to or from pointers **shall not** be used.
183	55	Every possible measure **should** be taken to avoid type casting.
184	55	Floating point numbers **shall not** be converted to integers unless such a conversion is a specified algorithmic requirement or is necessary for a hardware interface.
186	56	There **shall** be no unreachable code.
187	56	All non-null statements **shall** potentially have a side-effect.
189	56	The *goto* statement **shall not** be used.
192	56	All *if*, *else if* constructs **will** contain either a final *else* clause or a comment indicating why a final *else* clause is not necessary.
197	57	Floating point variables **shall not** be used as loop counters.
198	57	The initialization expression in a *for* loop **will** perform no actions other than to initialize the value of a single *for* loop parameter. Note that the initialization expression may invoke an accessor that returns an initial element in a sequence
199	58	The increment expression in a *for* loop **will** perform no action other than to change a single loop parameter to the next value for the loop.

11. Bibliografie

MySQL vs PostgreSQL. (2014, 4 18). Opgeroepen op 4 23, 2014, van WikiVS: http://www.wikivs.com/wiki/MySQL_vs_PostgreSQL