

Information harvesting in MoK

Information retrieval using web crawlers

Giulio Crestani, Gianluca Spadazzi

8 novembre 2015

Indice

1	Crawler e Web Crawler	3
1.1	Proprietà importanti	3
1.2	Altre proprietà	3
1.3	Funzionamento	4
2	MoK - Molecules of Knowledge	4
2.1	Definizione	4
2.2	Obiettivi e Struttura	4
2.3	Ruolo del nostro software nell'architettura generale di MoK	6
3	CrawlerCommander	6
3.1	Componenti principali	6
3.1.1	Central Unit	6
3.1.2	Crawlers	9
3.1.3	Database Online	9
3.1.4	Front-End	9
3.2	Funzionamento di una ricerca senza sinonimi	9
3.3	Funzionamento di una ricerca con sinonimi	12
4	Librerie/Tecnologie usate	13
4.1	WordNet	13
4.2	Connessione al database	13
4.3	Ricerca con Google	13
4.4	Crawler4j	14
5	Passaggio al distribuito con relative proprietà e gestione delle problematiche	14
5.1	Central Unit di riserva	14
6	Gestione degli errori	15
6.1	Verifica disponibilità dei crawlers	15
6.2	Central Unit di riserva e crash di tutti i crawler connessi	15

7	Modello del sistema	15
7.1	Note	18
7.1.1	Descrizione entità: Central Unit	18
7.1.2	Descrizione entità: Crawler4j	18
7.1.3	Descrizione entità: Crawler Google	19
8	Architettura interna di Central Unit e Crawlers	19
9	Diagramma delle interazioni	23
9.1	Legenda dello scenario standard	23
9.2	Legenda dello scenario con la Central Unit di riserva	24
10	Motivazione per le scelte tecnologiche	25
11	Possibili sviluppi futuri	28
11.1	Problema di swap da una Central Unit all'altra	28
11.1.1	Possibili soluzioni al problema	28
11.1.2	Problemi che nascono con la possibile soluzione	28
11.2	Clonazione infinita della Central Unit	28
11.2.1	Possibile soluzione: tre, quattro, cinque, ..., illimitate Central Unit	29
11.2.2	Vari problemi di questo approccio	29
11.2.3	Altra possibile soluzione	29
11.3	Limitare il numero di interrogazioni al database	30
11.3.1	Possibile problema e relativa soluzione	30
11.4	Inserimento timeout nel caso di indisponibilità dei crawlers	30
A	Premesse per l'utilizzo di CrawlerCommander	31
A.1	Note sui due crawler utilizzati	31
A.2	Ricerca con sinonimi	31
A.3	Database online	31
A.4	Settare le varie informazioni	31
A.5	Tutorial seguiti	31
B	Tutorial per l'utilizzo di CrawlerCommander	32

1 Crawler e Web Crawler

Un *crawler* (detto anche spider o robot), è un software che analizza i contenuti di una rete (o di un database) in un modo metodico e automatizzato, in genere per conto di un motore di ricerca. È un tipo di bot (programma o script che automatizza delle operazioni). I crawler solitamente acquisiscono una copia testuale di tutti i documenti visitati e le inseriscono in un indice. Un uso estremamente comune dei crawler è nel Web. Sul Web, il crawler si basa su una lista di URL da visitare fornita dal motore di ricerca (il quale, inizialmente, si basa sugli indirizzi suggeriti dagli utenti o su una lista precompilata dai programmatori stessi). Durante l'analisi di un URL, identifica tutti gli hyperlink presenti nel documento e li aggiunge alla lista di URL da visitare. Il processo può essere concluso manualmente o dopo che un determinato numero di collegamenti è stato seguito. Inoltre i crawler attivi su Internet hanno la facoltà di essere indirizzati da quanto indicato nel file *robots.txt* posto nella root del sito. All'interno di questo file, è possibile indicare quali pagine non dovrebbero essere analizzate. Il crawler ha la facoltà di seguire i consigli, ma non l'obbligo. Alcuni esempi di Web Crawler dei principali motori di ricerca sono i seguenti:

Nome bot	Motore di ricerca
googlebot	Google
slurp	Inktomi - Yahoo!
Yahoo! Slurp	Yahoo
Romilda	Facebook

1.1 Proprietà importanti

Le proprietà fondamentali che un crawler deve fornire sono essenzialmente due:

- **Robustezza:** Il Web contiene server che generano pagine in grado di fuorviare i crawler tali da farli rimanere bloccati cercando di recuperare un numero infinito di pagine in un particolare dominio. I crawler devono essere progettati per resistere a queste “trappole”.
- **Gentilezza:** I server Web hanno delle politiche che regolano la velocità con cui un crawler può visitare una pagina. Queste politiche di cortesia devono essere rispettate.

1.2 Altre proprietà

Ulteriori proprietà che dovrebbero essere fornite da un crawler sono le seguenti:

- **Distribuzione:** Il crawler deve avere l'abilità di svolgere il suo compito in un ambiente distribuito attraverso macchine multiple.
- **Scalabilità:** L'architettura deve permettere un aumento della velocità di scansione con l'aggiunta di macchine extra e larghezza di banda.

- **Performance ed efficienza:** Il sistema deve fare un uso efficiente delle varie risorse di sistema, tra cui processore, memoria e larghezza di banda.
- **Qualità:** Siccome una parte significativa di tutte le pagine web sono di scarsa utilità, per servire le esigenze delle query lanciate dall'utente, il crawler deve essere in grado di recuperare le pagine più "utili" ai fini della ricerca.
- **Freschezza delle informazioni:** In alcune applicazioni i crawler devono operare in continuazione perciò si deve fare in modo di ottenere sempre delle copie aggiornate.
- **Estensibilità:** Un crawler deve essere in grado di evolvere nel tempo per coprire, ad esempio, nuovi formati di dati, nuovi protocolli ecc. Questo implica che l'architettura sia di tipo modulare.

1.3 Funzionamento

Il crawler inizia con uno o più URL che costituiscono il set iniziale (*seed set*). Sceglie un URL da questo insieme e seleziona la pagina sulla quale eseguire il *parsing*¹ in modo da estrarre sia il testo che i link contenuti.

Successivamente si analizza il testo delle pagine e per quanto riguarda i link estratti, essi vengono aggiunti ad un altro insieme (*URL frontier*) il quale è composto da tutti quei link le cui pagine devono ancora essere analizzate dal crawler. Inizialmente questo insieme è formato dal set iniziale.

Una volta che un URL viene utilizzato dal crawler, esso viene eliminato dal *URL frontier*.

2 MoK - Molecules of Knowledge

2.1 Definizione

MoK ²Modello di coordinazione che promuove l'auto-organizzazione della conoscenza in un sistema multi-agente. Cerca quindi di aggregare l'informazione in maniera autonoma.

2.2 Obiettivi e Struttura

Gli obiettivi principali di questo modello sono quindi la capacità di aggregare dati autonomamente in modo da costruirne di più complessi e la diffusione di queste informazioni verso i potenziali produttori o consumatori.

Un sistema MoK può essere visto come una rete di contenitori di informazioni condivisa e gli utenti possono interagire con il sistema attraverso delle *azioni*. Queste ultime rilasciano delle *tracce* nell'ambiente che vengono sfruttate dal sistema stesso (in base alle leggi di coordinazione di MoK) per guidare l'evoluzione dell'informazione, incrementando la loro concentrazione ed influenzando l'evoluzione della conoscenza in modo trasparente rispetto agli utilizzatori.

¹Parsing: processo che analizza un flusso continuo di dati in ingresso (input) (letti per esempio da un file, una tastiera o una pagina web) in modo da determinare la sua struttura.

²*Molecules of Knowledge: Self-Organisation in Knowledge-Intensive Environments* - Mariani, Seminar @ Distributed Systems Group - Wien, 2014

Il modello di MoK utilizza le seguenti astrazioni:

- **Seeds:** sono le sorgenti delle informazioni; essi iniettano dei nuovi dati (*atomi*) continuamente e spontaneamente all'interno dello spazio di lavoro.
- **Compartments:** sono i depositi delle informazioni.
- **Catalysts:** sono gli utenti del sistema ovvero sia produttori che consumatori delle informazioni. Essi sfruttano ed influenzano il servizio di auto-organizzazione.
- **Atoms:** sono le unità primitive delle informazioni e vengono sempre prodotte dai *seeds*.
- **Molecules:** sono unità composte di informazioni. Prodotte dalla reificazione³ di diversi atomi.
- **Enzymes:** sono l'unione delle *azioni* generate dai *catalysts*.
- **Reactions:** sono le “leggi” con cui si gestisce l'evoluzione dei *compartments*.

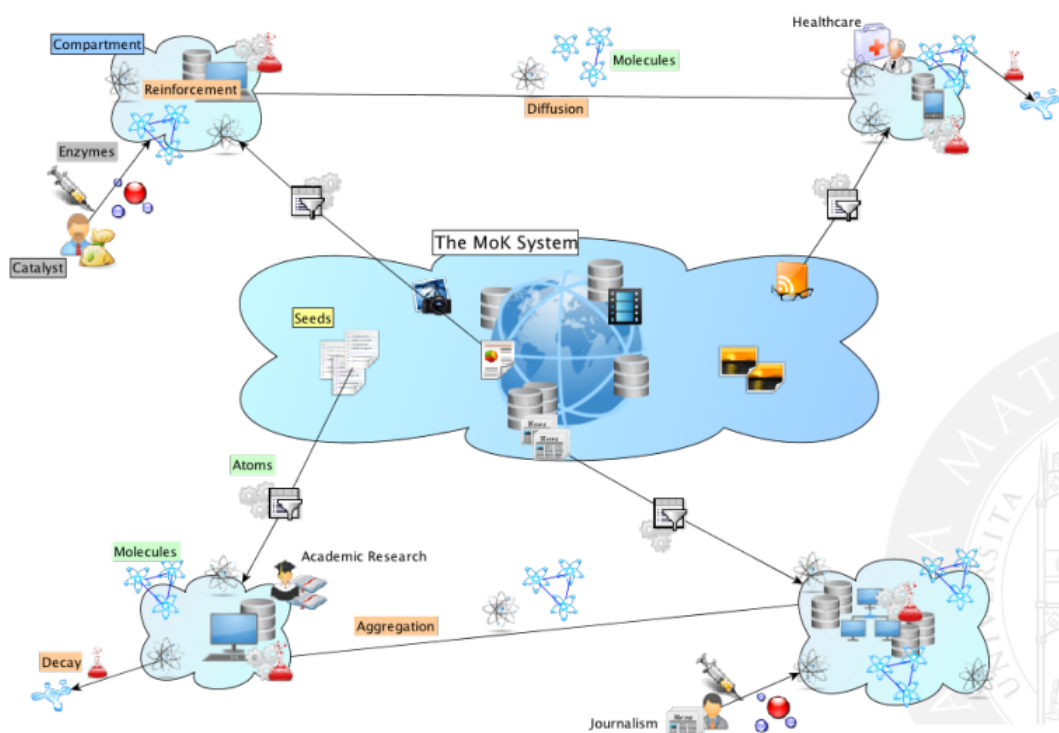


Figura 1: Sistema MoK.

In Figura 1 è possibile vedere un esempio del sistema.

³In informatica reificazione è il procedimento di creazione di un modello di dati basato su un concetto astratto predefinito. (Wikipedia)

2.3 Ruolo del nostro software nell'architettura generale di MoK

Siccome l'architettura complessiva di MoK è molto complessa ed ancora in fase di ricerca, si è deciso di *splittare* in diversi progetti riguardanti aree abbastanza distinte tra loro. Queste aree sono essenzialmente 3:

1. *Distributed information harvesting: from unstructured text to semantically enriched MoK atoms.*

Questa prima area riguarda la raccolta delle informazioni (anche metadati) in modo da estrarre più contenuto informativo possibile e quale sia il modo migliore di rappresentare l'informazione in un sistema "qualunque" basato su MoK per meglio supportare i suoi meccanismi di auto-organizzazione.

2. *Information persistency: scalable distributed storage solutions (e.g. key-value DBs, graph DBs, document-oriented DBs).*

Questa area si concentra sullo studio delle tecnologie che meglio si prestano a gestire una grandissima quantità di dati e a poter spostare/replicare tali dati tra diversi dispositivi, con altissima frequenza.

3. *Information management: decentralized self-* algorithms, semantic reasoning on resource-constrained devices, efficient in-memory DBs.*

In quest'ultima area si cercano di sperimentare diversi framework di comunicazione (message-based, tuple-based, stream-based) e si cerca di creare un middleware sempre per gestire la comunicazione tra sistemi pervasivi.

Il nostro progetto fa parte della prima area. L'idea è quella di sviluppare un componente distribuito in grado di navigare il web in cerca di fonti di informazione (preferibilmente testuali, strutturate e non) di qualche tipo. Per fare ciò si è creato un *Web Crawler* che navigasse e cercasse in rete. Se ne sono testati alcuni già noti in letteratura. Oltre a ciò si è creata un'unità centrale in grado di gestire tutti i crawler in modo da sincronizzarli affinché non si vadano a generare gli stessi risultati una volta trovati.

3 CrawlerCommander

L'idea alla base del software è piuttosto semplice: si utilizza un crawler di terze parti chiamato *crawler4j*⁴ e un crawler sviluppato da noi che si appoggia sulle API per la ricerca su google.

3.1 Componenti principali

I componenti principali sono riassumibili nella Figura 2.

3.1.1 Central Unit

Questo componente è la parte essenziale del sistema. Esso gestisce tutti i crawler facendoli partire al momento opportuno e controlla i risultati che arrivano da questi ultimi in modo da salvarli in un repository online affinché siano disponibili

⁴<https://github.com/yasserg/crawler4j>

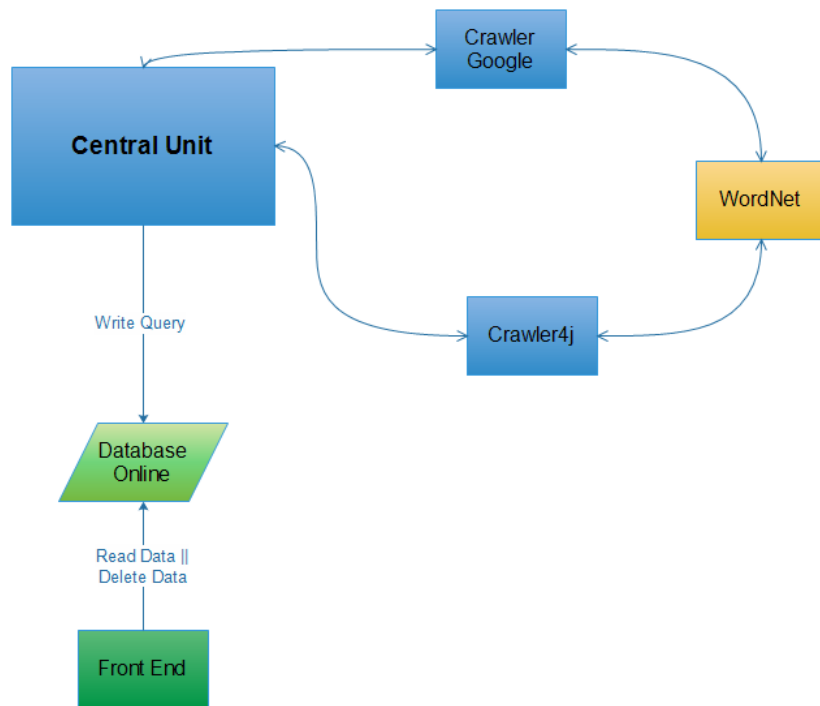


Figura 2: Riepilogo componenti di CrawlerCommander.

da diversi dispositivi. È presente un thread che gestisce la Central Unit e un thread per ogni crawler connesso. Nella Figura 3 è presente uno schema che riassume i suoi componenti interni.

- **Lista dei crawler**

Lista contenente il riferimento a tutti i web crawler presenti in rete che vogliono “unirsi” alla ricerca. Di ognuno di questi si conosce l’indirizzo IP.

- **Lista delle Web Pages**

Una *WebPage* è un oggetto che contiene tutti i dati utili di una determinata pagina web. Questi oggetti vengono creati dalla Central Unit e inseriti nel database online. Andando più nel dettaglio, per ogni pagina web che si incontra si tiene traccia di alcuni campi. In particolare:

1. **Keyword:** parola utilizzata nella ricerca dalla quale ne è uscita questa pagina come uno dei risultati. È possibile inoltre cercare una multi-keyword ovvero specificare più keyword nello stesso campo. Queste keyword devono essere separate da un trattino (-). Ad esempio se si volesse cercare le due keyword *albero* e *casa* si dovrebbe specificare come keyword: *albero-casa*;
2. **Title:** titolo delle pagine web;
3. **Link:** link della pagina;
4. **Snippet:** sono le prime righe contenute;

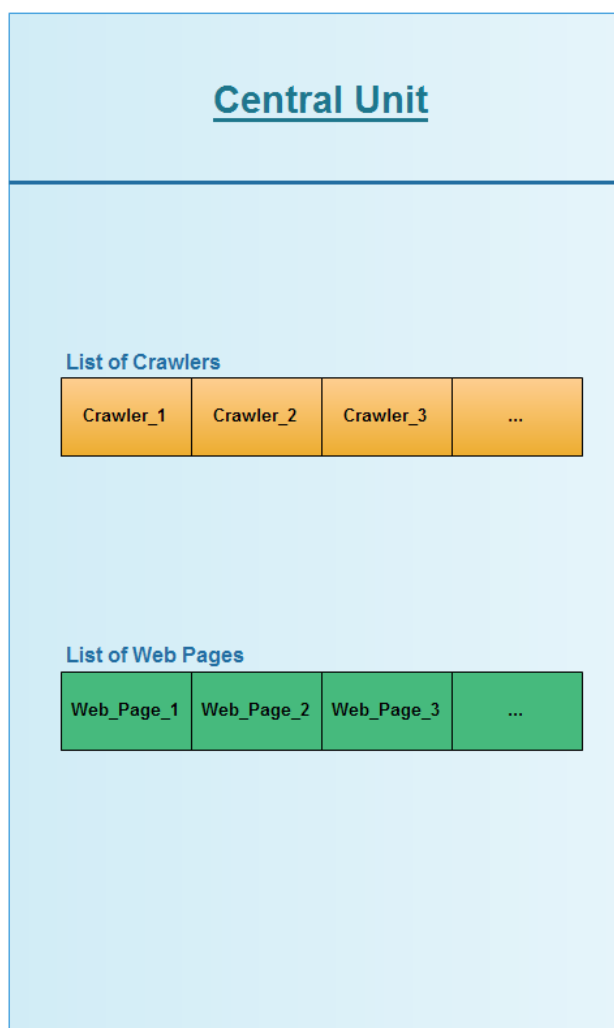


Figura 3: Riepilogo componenti della Central Unit.

5. **Length:** numero di caratteri contenuti.

La maggior parte di questi campi sono ricavati dalle funzionalità messe a disposizione dai web crawlers che abbiamo utilizzato. Oltre a queste ce ne sarebbero altre per le quali non si è tenuto traccia ma che, se in un futuro risulteranno utili ai fini di MoK, si potranno tranquillamente utilizzare. Per esempio in crawler4j è presente anche le funzionalità per scaricare, oltre al testo della pagina, tutte le immagini presenti e tutti i contenuti multimediali supportando diversi formati.

3.1.2 Crawlers

Componenti essenziali del sistema. Tramite essi si effettuano le ricerche delle pagine estrapolandone i dati indicati e che successivamente inviano alla Central Unit che le inserirà nel database. Un crawler, per identificarsi, deve registrarsi alla Central Unit inviando il suo indirizzo IP in modo che successivamente la Central Unit lo inserisca nella sua lista interna di crawlers.

3.1.3 Database Online

Per testare il nostro software si è usato un semplice database online. Si chiama *Free MySQL Hosting*⁵ che permette di avere 5MB di spazio libero (che per le nostre prove è più che sufficiente). Su questo database si interfaccia la Central Unit utilizzando semplici query per inserire i dati provenienti dai vari crawler.

Si è deciso di adottare la soluzione online perchè, siccome si vuole creare un sistema distribuito, si devono usare tutte le accortezze del caso e quindi, in primo luogo, il fatto di rendere disponibili i dati a tutti coloro che ne hanno bisogno, dovunque essi siano.

3.1.4 Front-End

Semplice interfaccia da utilizzare sui dispositivi mobili e non. Tramite esso è possibile visualizzare tutti i dati contenuti nel database online e gestirli. Quindi è possibile leggere e rimuovere tuple.

- **Per PC Desktop**

Semplice interfaccia grafica sviluppata in Java che permette di visualizzare in una tabella il contenuto del database online. Oltre a visualizzare permette anche di eliminare le tuple, aggiornare i dati e aprire il link della riga selezionata con il browser. Per queste occorrenze sono presenti diversi bottoni. (Figura 4)

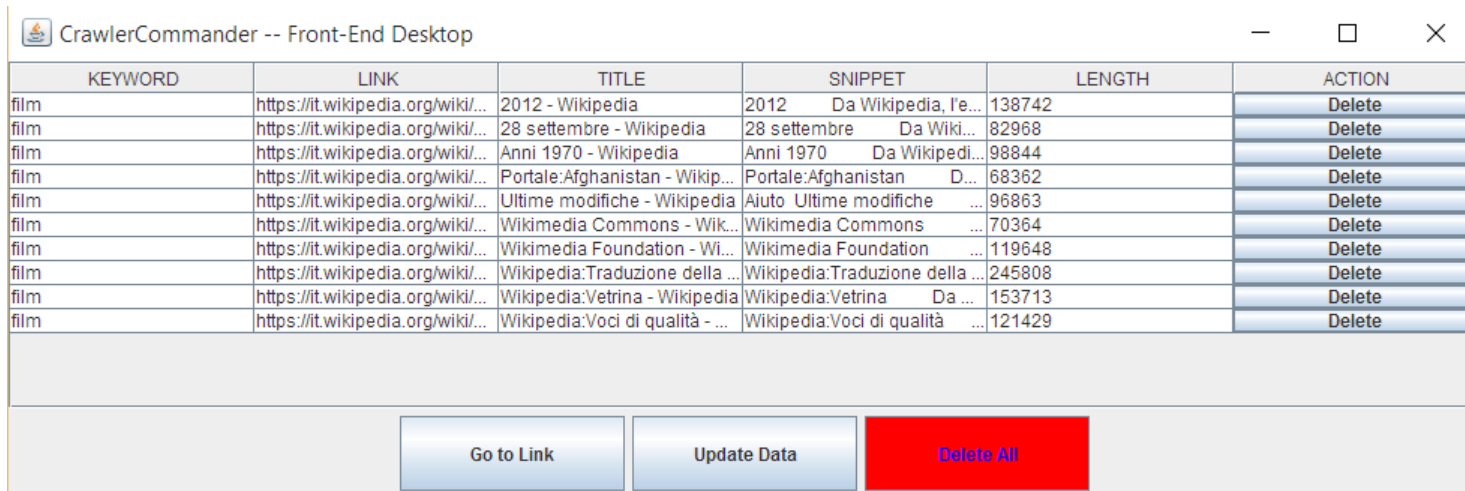
- **Per dispositivi mobili**

In un sistema distribuito che si rispetti i dati devono essere accessibili da qualunque posizione, anche in mobilità. Quindi si è realizzato un front end Android oltre a quello per pc. Appena l'applicazione viene aperta parte un *task* che si connette al database online, scarica i dati e con essi popola una lista. È possibile anche aggiornare manualmente la lista in un secondo momento premendo su un bottone. Premendo invece su una delle pagine listate, viene mostrata una finestra di dialogo che ci permette di scegliere se visitare il sito della pagina raccolta oppure di eliminarla. In quest'ultimo caso, parte un task che esegue una query sul database online che elimina la pagina, e appena terminato viene nuovamente eseguito il task che aggiorna la lista, affinché quest'ultima sia aggiornata in modo tale da non contenere più la pagina eliminata. Infine è anche possibile eliminare tutti i dati premendo sul bottone nella actionBar. (Figura 5)

3.2 Funzionamento di una ricerca senza sinonimi

Nel seguito si fa un esempio di ricerca eseguita con CrawlerCommander senza voler approfondire con i sinonimi.

⁵<http://www.freemysqlhosting.net/>



KEYWORD	LINK	TITLE	SNIPPET	LENGTH	ACTION
film	https://it.wikipedia.org/wiki/...	2012 - Wikipedia	2012 Da Wikipedia, l'e...	138742	Delete
film	https://it.wikipedia.org/wiki/...	28 settembre - Wikipedia	28 settembre Da Wiki...	82968	Delete
film	https://it.wikipedia.org/wiki/...	Anni 1970 - Wikipedia	Anni 1970 Da Wikipedi...	98844	Delete
film	https://it.wikipedia.org/wiki/...	Portale:Afghanistan - Wikip...	Portale:Afghanistan D...	68362	Delete
film	https://it.wikipedia.org/wiki/...	Ultime modifiche - Wikipedia	Aiuto Ultime modifiche ...	96863	Delete
film	https://it.wikipedia.org/wiki/...	Wikimedia Commons - Wik...	Wikimedia Commons ...	70364	Delete
film	https://it.wikipedia.org/wiki/...	Wikimedia Foundation - Wi...	Wikimedia Foundation ...	119648	Delete
film	https://it.wikipedia.org/wiki/...	Wikipedia:Traduzione della ...	Wikipedia:Traduzione della ...	245808	Delete
film	https://it.wikipedia.org/wiki/...	Wikipedia:Vetrina - Wikipedia	Wikipedia:Vetrina Da ...	153713	Delete
film	https://it.wikipedia.org/wiki/...	Wikipedia:Voci di qualità - ...	Wikipedia:Voci di qualità ...	121429	Delete

Go to Link
Update Data
Delete All

Figura 4: Screenshot applicazione Java per PC.

1. Si avvia CrawlerCommander e si attende che dei crawler si registrino inviando il loro indirizzo IP alla Central Unit. Come è già stato spiegato, all'interno della Central Unit è presente una lista di crawler per tenersi memorizzati coloro che si sono già registrati. Fisicamente non è altro che un *ArrayList* all'interno del quale vengono salvati dei dati; quindi la Central Unit può memorizzare un numero potenzialmente infinito⁶ di crawler. Se viene lanciata una ricerca senza che ci siano dei crawler connessi, verrà visualizzata una finestra popup di avviso.
2. L'utente scrive nell'interfaccia grafica la keyword e il numero di risultati che desidera ottenere. Successivamente fa partire la ricerca. Questa azione implica l'invio di una stringa da parte della Central Unit verso tutti i crawler della lista di questo formato:

$$TASK \quad \underbrace{film}_{\text{Keyword}} \quad \underbrace{20}_{\text{number of results}} \quad \underbrace{false}_{\text{with Synonyms?}}$$

3. Una volta inviata la suddetta stringa a tutti i crawler, viene fatta partire la ricerca vera e propria inviando una seconda stringa semplicemente con la scritta '*START*'. I crawler al loro interno hanno un *handler* che non fa altro che controllare quali stringhe vengono ricevute. A fronte dell'arrivo di questa stringa, tutti i crawler iniziano una ricerca interna specificando un numero di risultati infinito⁷. Si è utilizzata questa soluzione in modo

⁶Un limite è dato dalla capienza massima di un *ArrayList* in Java e quindi da quanta memoria è disponibile nella JVM. Un altro limite è dato dal numero di thread che la macchina è in grado di gestire siccome si va a creare un thread per ogni connessione.

⁷Il crawler di Google ha una limitazione di 100 ricerche da al più 100 risultati al giorno perchè si è utilizzata una versione gratuita.



Figura 5: Screenshot applicazione Android.

che se sono presenti tanti crawler alcuni più veloci di altri nel ricercare i risultati, si sfruttano al massimo e non si aspetta che quelli più lenti trovino i loro risultati. Sarà successivamente la Central Unit a stopparli come si vedrà in seguito.

4. Ogni volta che un crawler trova una pagina ne estrapola i dati interessanti generando una WebPage che successivamente invia alla Central Unit.
5. La Central Unit riceve questi dati e li elabora prima di inserirli nell'apposita lista di WebPage descritta nella Sezione 3.1.1:
6. Una WebPage viene elaborata nel seguente modo:
 - (a) Si tiene traccia delle pagine già cercate in una lista;

- (b) La Central Unit di continuo riceve le pagine cercate dai vari crawler e, prima di salvarle nella lista, controlla che non siano già state cercate nella sessione di ricerca corrente e che non siano già presenti nel database online;
 - (c) Una volta che vengono salvate nella lista un numero pari al numero di pagine da cercare specificato nell'interfaccia grafica dall'utente, queste vengono inviate al server online con una query.
7. Dopo che è stato raggiunto il numero di risultati specificato, la Central Unit invia come ultimo messaggio a tutti i crawler la stringa '*STOP*' e questi ultimi si arrestano.
 8. Una volta che l'utente desidera fermare la ricerca, deve semplicemente fermarla con il bottone nell'interfaccia grafica. In questo caso, se si termina la Central Unit attraverso il bottone e la ricerca non è completata, i dati raccolti fino a quel momento vengono comunque inviati al database online.

3.3 Funzionamento di una ricerca con sinonimi

Il principio che sta alla base della ricerca con i sinonimi è lo stesso rispetto a quella senza sinonimi. Ovviamente in questo caso i vari crawler si basano sulla libreria che permette di trovare i sinonimi di una determinata parola (descritto nella Sezione 4.1). Ci sono solo alcune differenze spiegate in seguito.

- La stringa che viene inviata ai crawler per settarsi è:

TASK film 10 true

indicando *true* nell'ultimo campo.

- Per quanto riguarda *Crawler4J* (uno dei crawler utilizzati e descritto nella Sezione 4.4) si è implementata una funzione di ricerca dei sinonimi. Semplicemente si controlla se nella pagina che si sta analizzando è presente anche la parola sinonima.
- Per il crawler di Google (Sezione 4.3), invece, si è deciso di cercare 5 pagine contenenti la keyword, 5 pagine contenenti il primo sinonimo, 5 pagine contenenti il secondo sinonimo, 5 pagine contenenti il terzo sinonimo e ricominciando da quelle contenenti la keyword⁸.
- Per fare in modo di dare spazio anche ai sinonimi all'interno del database, si è scelto di tenere la metà delle pagine risultanti provenienti da ricerche con la keyword mentre per quanto riguarda le rimanenti, assegnarle ai sinonimi. Questa gestione è lasciata alla Central Unit che controlla tutte le keyword delle pagine che riceve e decide se tenerle o se scartarle stavolta anche in base al fatto se sono parole sinonime o no.

⁸Si è deciso di fermarsi ai primi 3 sinonimi perché se si utilizzano troppi sinonimi dopo si fa troppa varietà. Ed inoltre se il numero di risultati voluto è basso non ha neanche molto senso cercare troppi sinonimi.

4 Librerie/Tecnologie usate

4.1 WordNet

Un crawler non si dovrebbe limitare a recuperare informazioni delle pagine relative alla keyword specificata nell'interfaccia grafica. Può essere utile effettuare ricerche supportando anche i sinonimi. Ad esempio, se l'utente specifica come keyword di ricerca la parola "albero" potrebbe essere utile ottenere risultati relativi anche alle pagine che troveremmo usando la keyword "pianta" o altro.

Per fare ciò ci si è appoggiati su *WordNet*⁹. Esso è un grande database lessicale che contiene diversi sinonimi per ogni parola. Grazie alla relativa libreria è possibile sfruttare questa risorsa anche offline (scaricando il database sul pc) e ottenere sinonimi relativi alle keyword specificate per la ricerca.

Quindi in questo caso il numero di risultati specificati dall'utente saranno suddivisi in parti in base sia alla keyword specificata ma anche in base ai suoi sinonimi. Per ora nell'applicazione è necessario cercare parole inglesi per sfruttare questa caratteristica perché WordNet fornisce solo i sinonimi in questa lingua.

4.2 Connessione al database

Per poter connettersi ad un database situato in rete si è avuto bisogno di utilizzare una particolare libreria. ***MySQL Connector/J***¹⁰ è il *JDBC (Java Database Connector)* driver ufficiale per MySQL. Questa libreria fornisce le API per connettersi a database SQL, necessarie per leggere e scrivere i dati su un database online.

4.3 Ricerca con Google

Google Custom Search¹¹ è una piattaforma fornita da Google che permette agli sviluppatori di sfruttare il famoso motore di ricerca per effettuare ricerche e recuperare dati dal web. Le API per sfruttare questo servizio sono disponibili per il linguaggio Java in un ampio set di librerie. Questo servizio è molto potente in quanto risulta molto efficiente dal punto di vista della velocità di ricerca. È stata usata la versione gratuita, che permette di effettuare al massimo 100 ricerche al giorno.

Per prima cosa è stato necessario creare un motore di ricerca personalizzato tramite un sito specializzato. Si è anche settata come pagina da cui cercare, Wikipedia in italiano. Una volta creato, è stato salvato il relativo *search id*, necessario per sfruttare le API. Successivamente è stato creato un nuovo progetto tramite la *Google Cloud Console* e salvata l'*API Key*, anch'essa necessaria per sfruttare le API. Dopo questi passaggi è stato possibile, appunto, usare le API di Google per fare ricerche su internet.

⁹<https://wordnet.princeton.edu/>

¹⁰<http://dev.mysql.com/doc/connector-j/en/connector-j-overview.html>

¹¹Link ai vari step da seguire: <http://weblog4j.com/2014/06/03/having-fun-with-google-custom-search-api-and-java/>

4.4 Crawler4j

È un crawler¹² open source scritto in Java che si può utilizzare tramite un'opportuna libreria. Si può specificare una pagina web che il crawler esaminerà raccogliendo dati su di essa, ma non si limiterà a questo: esaminerà allo stesso modo anche i link contenuti nella pagina e così via all'infinito a meno che venga specificato un numero massimo di pagine da cercare. Possiede diverse funzioni che possono essere sfruttate, come la possibilità di avere un crawler multithreaded o di scaricare le immagini delle pagine oltre agli altri dati.

Un problema di Crawler4j è che non è possibile avviarne più di uno sulla stessa macchina. Questo perché quando ricerca dalla rete, si appoggia ad una cartella che risiede in locale. Se se ne avviano più di uno, essi andranno ad usare gli stessi file.

5 Passaggio al distribuito con relative proprietà e gestione delle problematiche

Il sistema è stato pensato sin dall'inizio per poter lavorare in un ambito distribuito. Questo implica un ulteriore livello di gestione attraverso rete.

Si deve pensare ad uno scenario in cui sia la Central Unit che i crawler non risiedono in una rete locale ma sparsi nella rete. Il modo che si è scelto per connetterli tra loro è tramite il protocollo TCP/IP.

Inizialmente si è fatta la seguente *assunzione* ovvero che la Central Unit, essendo il cuore del sistema, stia in un server ad un indirizzo IP fisso e ad una porta noti a coloro che lanceranno i vari crawler. Questi ultimi, non appena lanceranno un crawler, si vedranno apparire una semplice interfaccia grafica dove si chiede di specificare indirizzo IP e porta della Central Unit principale ed un secondo IP e porta per identificare la Central Unit di riserva (descritta nella Sottosezione 5.1). In questo modo una volta che si lanceranno i vari crawler, andranno a connettersi correttamente e potranno incominciare a svolgere il loro compito anche se risiedono fisicamente da un'altra parte rispetto agli altri componenti.

5.1 Central Unit di riserva

Come già introdotto nella Sezione 5, si è deciso di “munire” il sistema di una seconda Central Unit (di riserva) pensata per la gestione di una particolare tipologia di evenienza. Si pensi al caso in cui c'è una ricerca che sta funzionando e quindi è presente un pull di crawler connessi alla Central Unit primaria e che le mandano i risultati che trovano. Ad un certo punto però, per un qualche motivo, la Central Unit primaria non è più disponibile. In uno scenario nel quale non è presente quella di riserva, si andrebbe ad annullare tutta la ricerca fatta. Invece, con l'introduzione della seconda Central Unit di riserva si evita questa falla.

La gestione che si è scelta per questo fatto è semplice: se la Central Unit principale si rende indisponibile, tutti i crawler connessi si connettono alla seconda (si ricorda che tutti i crawler hanno anche l'indirizzo IP e porta di quest'ultima) mandandole tutti i dati che l'utente iniziale ha inserito: *Keyword* e *Numero di risultati* in modo da informarla dei dati che le arriveranno e quando

¹²<https://github.com/yasserg/crawler4j>

deve terminare la ricerca dopo che ha raggiunto il numero giusto di risultati. Successivamente viene fatta ricominciare la ricerca come inizialmente e scritti i risultati sul database online. Un possibile sviluppo di questa soluzione è descritto nelle Sottosezioni 11.1 e 11.2.

Nel caso in cui un nuovo crawler voglia connettersi alla Central Unit e quest'ultima è indisposta, quello che si fa è semplicemente chiuderlo. Non facciamo in modo che si connetta a quella di riserva perché la gestiamo come un caso di emergenza quindi non vengono più accettati crawler e si aspetta solo che la ricerca che stava svolgendo si concluda. Una volta conclusa si chiude tutto il sistema.

Una possibile evoluzione a questa gestione potrebbe essere quella di accettare nuovi crawler anche se sta andando quella di riserva in modo da ottimizzare la ricerca o in modo che, se si disconnetterebbero tutti i crawler già attivi al momento dell'attivazione della seconda Central Unit, la ricerca possa continuare.

6 Gestione degli errori

In questa sezione si spiegano come si è pensato di gestire i vari errori che si potevano incontrare nell'utilizzo del software.

6.1 Verifica disponibilità dei crawlers

Può capitare che un crawler si connetta alla Central Unit e che, mentre attende che la ricerca venga fatta partire, si renda indisponibile per qualche malfunzionamento. Per risolvere a questo problema si è pensato di verificare lo stato della connessione tra la Central Unit e i crawler semplicemente effettuando una *write* di prova. Se questa va a termine (e quindi non lancia eccezioni) allora significa che il crawler è connesso ed è tutto a posto altrimenti significa che ci sono stati dei problemi e quindi lo si elimina dalla lista di crawler propria della Central Unit.

6.2 Central Unit di riserva e crash di tutti i crawler connessi

Ci si mette ora nel caso in cui stia funzionando la Central Unit di riserva con ad essa connessi dei crawler che le mandano i risultati che ottengono dalla rete e questi ultimi si rendono indisponibili. La soluzione proposta prevede che la Central Unit di riserva salvi al suo interno i risultati ottenuti fino a quel momento e che invierà al database insieme ai risultati del prossimo task che viene completato¹³. In questo modo si scongiura la perdita completa dei dati. Un possibile sviluppo futuro potrebbe essere quello specificato nella Sottosezione 11.4.

7 Modello del sistema

In questa sezione viene mostrato il modello del sistema in UML. Si sono divise in due immagini. (Figure 6 e 7)

¹³E quindi una nuova ricerca fatta partire da una Central Unit che “muore”.

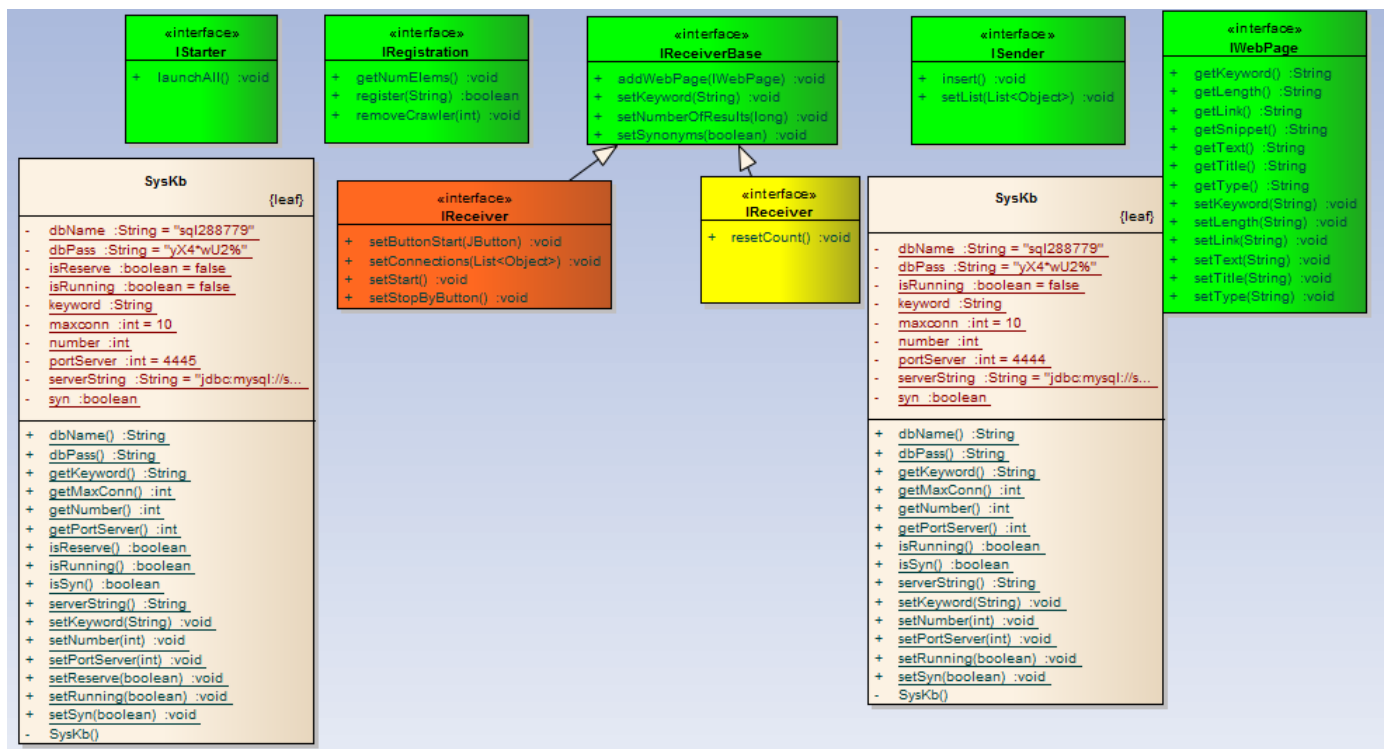


Figura 6: Rispettivamente Central Unit e Central Unit di riserva. Le interfacce verdi sono quelle in comune.

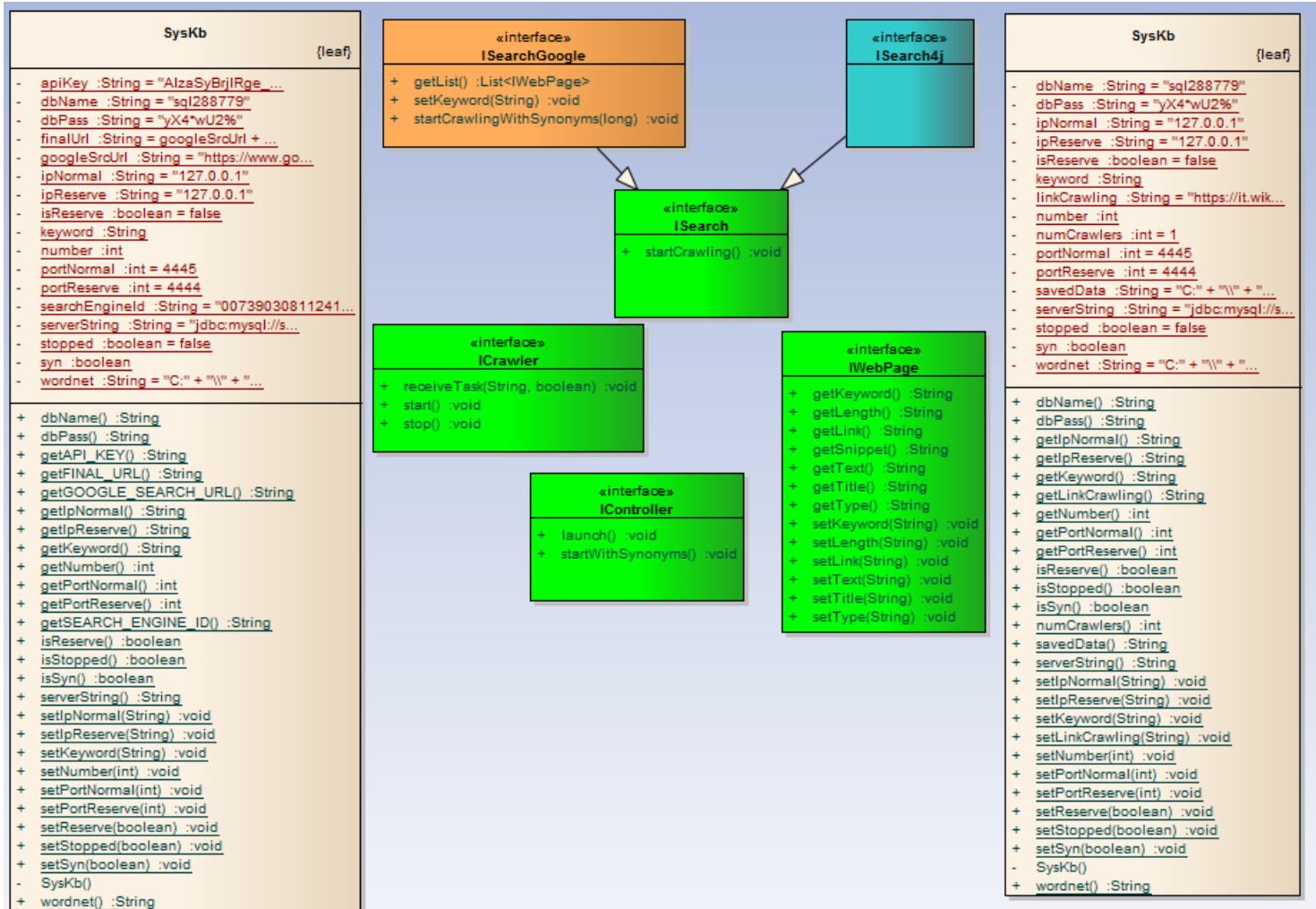


Figura 7: Rispettivamente Crawler Google e Crawler4j. Le interfacce verdi sono quelle in comune.

7.1 Note

Per prima cosa ci si sofferma sul fatto della gestione dei thread (che dalle immagini non si vede). La central unit ha un thread che sul quale ci fa girare l'applicazione vera e propria e viene creato un thread per ogni connessione con i crawler (fino ad un massimo specificato per non avere troppo carico). I crawler, invece, hanno un thread per la connessione e un thread per effettuare la ricerca, una volta che è stata fatta partire.

7.1.1 Descrizione entità: Central Unit

IWebPage Componente che modella le pagine web e le relative informazioni.

IRegistration Si occupa di tenere traccia degli indirizzi IP dei crawler che si connettono alla Central Unit. Per come si gestisce ci è totalmente inutile perché non viene usato ma siccome forse ci saranno sviluppi futuri, viene mantenuto.

IStarter Si occupa di inviare il messaggio “START” ai crawler per comunicargli che devono iniziare le loro operazioni di ricerca.

ISender Componente che si occupa di inviare la lista delle pagine con le relative informazioni al database online.

IReceiver Riceve e filtra le pagine web. Oltre a questo, invia il messaggio di “STOP” ai crawler. La ricerca deve fermarsi una volta che sono state raccolte tante pagine quanto specificato.

SysKb Classe statica che tiene traccia di alcuni *flag* usati per la gestione ed in particolare di informazioni sul sistema che possono essere recuperate (come ad esempio gli indirizzi IP, i numeri di porta e le informazioni necessarie per connettersi al database online). Modificando questi dati è possibile settare il sistema a proprio piacimento.

Central Unit di riserva Non fa altro che modellare una seconda Central Unit quindi si ha che ISender, IWebPage e SysKb rimangono inalterati mentre IReceiver è stato specializzato siccome ha un metodo diverso.

7.1.2 Descrizione entità: Crawler4j

Come spiegato brevemente nella Sottosezione 4.4, un problema di questo crawler è il fatto che non è possibile avviarne più di uno per ogni macchina.

IController Componente che funge da tramite tra ICrawler e ISearch4j. Si occupa di far partire la ricerca sia nel caso in cui sia con o senza sinonimi.

IWebPage Vedi Central Unit.

ISearch4j Si occupa di svolgere le operazioni di ricerca delle pagine web (non appena IController le comunica di farlo) e di inviarle alla Central Unit.

ICrawler Modella un crawler e le operazioni relative ad esso, come la partenza e l'arresto di una ricerca e la ricezione di un certo task. Nota: estende Thread.

SysKb Vedi Central Unit.

7.1.3 Descrizione entità: Crawler Google

IController Componente che funge da tramite tra ICrawler e ISearchGoogle. Si occupa di far partire la ricerca sia nel caso in cui sia con o senza sinonimi.

IWebPage Vedi Central Unit.

ISearchGoogle Come ISearch4j sopra descritta con l'aggiunta di qualche metodo.

ICrawler Come ICrawler sopra descritto.

SysKb Vedi Central Unit.

8 Architettura interna di Central Unit e Crawlers

Nella prima delle due prossime figure viene descritta l'architettura della Central Unit mentre nella seconda sono presenti i due crawler utilizzati.

Note sui comandi del diagramma CU Nel diagramma, i crawler sono indicati come client. Di seguito vengono spiegati alcuni significati che si sono dati ad alcuni elementi del diagramma.

- Le chiamate contrassegnate dalla guardia *[while searching]* vengono eseguite se la ricerca è già partita (è lo scenario in cui un crawler si connette quando la ricerca è già partita e gli viene comunque dato il task).
- Le chiamate contrassegnate dalla guardia *[first research]* vengono eseguite solo quando parte la ricerca la prima volta.
- Con la guardia *[stop pressed]* si indica l'atto in cui il pulsante stop è stato premuto.
- Nel fragment *[loop waituserinput]* si aspetta che l'utente prema start o stop.
- Nel fragment *[loop readmessagges]* si aspetta che i crawler mandino dei messaggi.
- Nel fragment *[loop waitconnections]* si aspettano le connessioni dai crawler.
- Nel fragment *[alt filerpages]* vengono filtrate le pagine ricevute coi criteri già specificati nella relazione.

Note sui comandi del diagramma dei crawler Il comportamento descritto riguarda il crawler di Google. Quello di *crawler4j* è praticamente identico a quest'ultimo con l'unica differenza che al posto dell'entità *SearchGoogle* c'è l'entità chiamata *Search4j*. Oltre a questo c'è una classe *Fetcher* su cui *crawler4j* si appoggia per le pagine.

- Nel fragment *[loop waituserinput]* si aspetta un input da interfaccia grafica.
- Nel fragment *[alt tryconnect]* si esegue un tentativo di connessione con la Central Unit.
- Nel fragment *[loop sendpages/receive commands]* si inviano pagine a CrawlerCommander e si ricevono comandi da parte di esso (e dalla Central Unit di riserva).
- Nel fragment *[loop fetchpages]* si effettua il crawling raccogliendo le Web-Page.

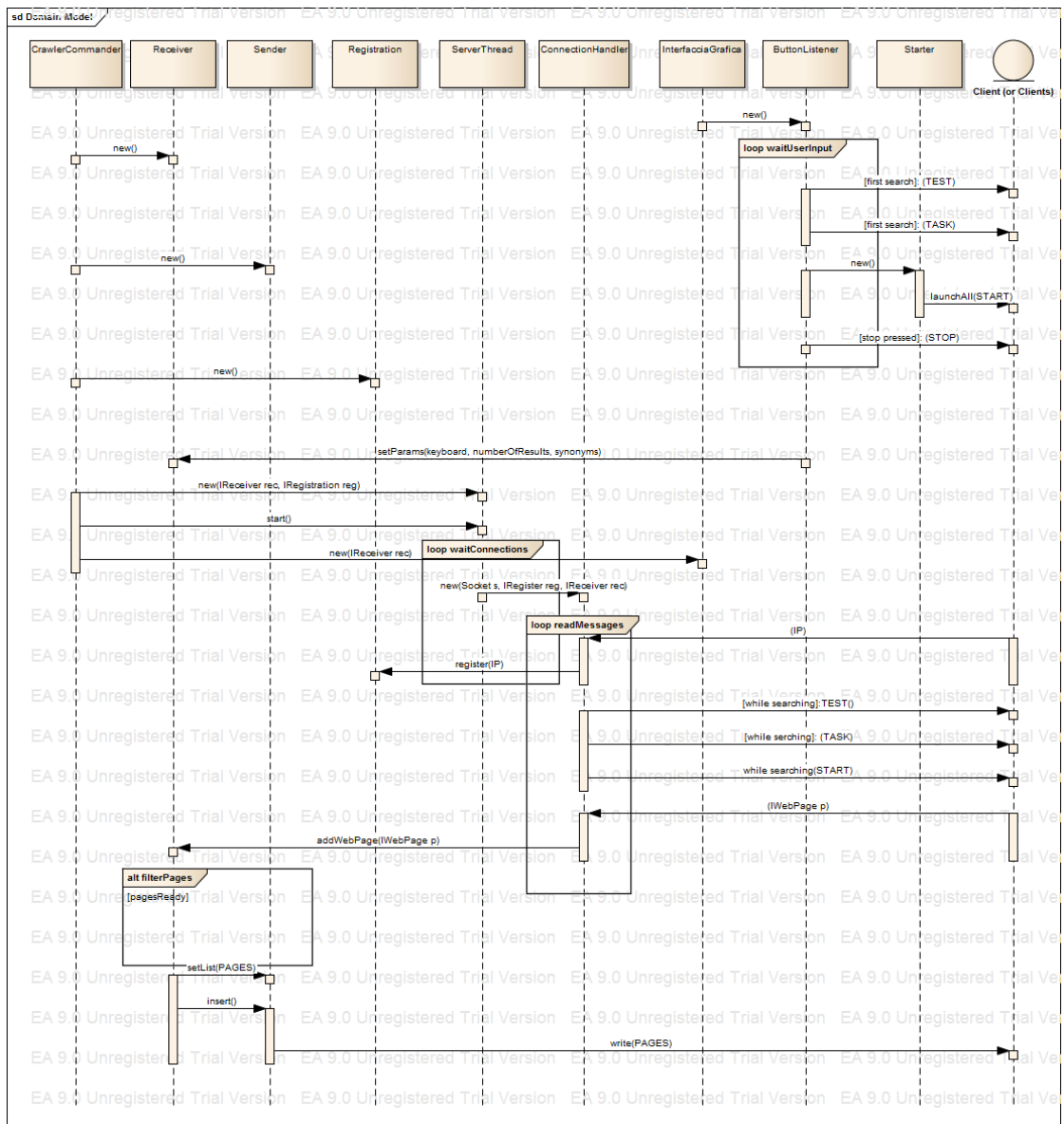


Figure 1: Architettura di funzionamento della Central Unit.

9 Diagramma delle interazioni

Ci si concentra ora sulle varie interazioni che si verificano tra le varie parti. Figure 8 e 9.

È importante precisare che i nomi presenti sulle varie interazioni (freccie) non rispecchiano i nomi reali dei metodi che si vanno a invocare ma si è cercato di utilizzare una rappresentazione astratta della logica del sistema. Quindi può darsi che dietro una semplice invocazione ci sia molto di più.

9.1 Legenda dello scenario standard

Central Unit Rappresenta la Central Unit primaria.

Crawler Rappresenta un generico crawler.

DatabaseOnline Database online in cui vengono salvati i dati delle ricerche.

FrontEnd Applicazioni desktop e mobile che permettono di visualizzare i dati contenuti nel database online ed eventualmente cancellarli o consultare le pagine web raccolte dai crawler.

User 1 Colui che fa partire la ricerca tramite l'interfaccia grafica della Central Unit.

User 2 Colui che tramite l'interfaccia grafica del crawler specifica indirizzi IP e numeri di porta di Central Unit principale e di riserva e che tramite un bottone fa partire i tentativi di connessione alla prima.

loop WaitConnections La Central Unit ha un loop infinito nel quale sempre in attesa di nuove connessioni da parte dei crawler. Il ciclo termina solamente quando quest'ultima viene terminata.

loop Receive e Filter Pages Quando parte la ricerca la Central Unit resta in attesa di pagine web inviate dai crawler che vengono filtrate ed eventualmente salvate. Il ciclo termina quando il task è completo ovvero è stato raccolto il numero di pagine specificato dall'utente.

loop attemptConnections Inizialmente i crawler cercano di connettersi alla Central Unit. Il ciclo termina non appena la connessione viene stabilita.

loop waitCommand Una volta connessi, i crawler restano in attesa di comandi da parte della central unit. Il ciclo termina solo quando il crawler viene terminato.

loop Crawling Una volta ricevuto il comando *TASK* subito dopo *START*, i crawler iniziano le operazioni di ricerca inviando le pagine alla Central Unit. Il ciclo termina quando viene ricevuto il comando *STOP*.

readOrDelete I front end possono leggere o cancellare i dati dal database online.

startSearch L'utente preme su un bottone dell'interfaccia grafica e la Central Unit fa partire la ricerca.

connectWithButton Alla pressione del bottone da parte di User 2, i crawler iniziano a cercare di connettersi alla Central Unit.

sendCommand Dopo che l'utente ha premuto sul bottone, la Central Unit invia il comando *TASK* (descritto nella Sottosezione 3.2) e poi il comando *START* per far partire i crawler connessi nella ricerca. Una volta che quest'ultima è terminata, verrà inviato ai crawler il comando *STOP*.

sendIP Una volta connessi alla Central Unit, i crawler le inviano il proprio indirizzo IP.

sendWebPage Via via che le pagine vengono recuperate dai crawler, esse vengono inviate alla Central Unit una alla volta.

sendData(pages) Quando il task è completo, la Central Unit invia le Web-Page al database online.

checkPage & updatePage Ogni volta che una pagina viene ricevuta dalla Central Unit, essa controlla che non sia già presente nel database online prima di salvarla. Se si accorge che è già presente esegue una query di *UPDATE* che aggiorna i campi della pagina in modo da mantenere i dati il più "freschi" possibili.

9.2 Legenda dello scenario con la Central Unit di riserva

Una volta che si rende indisponibile la prima Central Unit viene attivata quella di riserva. L'utente però di quest'ultima azione non si accorge di nulla perché non è presente un'interfaccia che lo mostri.

Di seguito vengono proposti tutti i componenti.

Crawler Vedi sopra.

CentralUnit Vedi sopra.

CentralUnit_ Reserve Simboleggia la Central Unit di riserva. Risiede su un altro server e non interagisce con l'utente. Serve come emergenza, ovvero terminare una ricerca nel caso in cui quella principale termini prima del suo completamento.

DatabaseOnline Vedi sopra.

loop Search Rappresenta la ricerca corrente della Central Unit principale, che in questo scenario termina prima del suo completamento.

break Crash La ricerca viene interrotta perché la Central Unit principale, per qualche motivo, termina.

loop Crawling Vedi sopra.

connect & sendData(TASK) Quando la Central Unit principale non è più raggiungibile, il crawler si connette quella di riserva e le manda il task che stava svolgendo.

sendWebPage Vedi sopra.

sendCommand(STOP) Vedi sopra.

loop Receive & Filter Pages Vedi sopra.

checkPage & updatePage Vedi sopra.

sendData(PAGES) Vedi sopra.

10 Motivazione per le scelte tecnologiche

Di seguito vengono indicate alcune motivazioni per le quali si è scelta questo metodo di soluzione del problema piuttosto che altri.

- **Linguaggio Java:** linguaggio a oggetti che oltre a essere ben collaudato e robusto è anche quello da noi conosciuto meglio, consentendoci quindi di risparmiare molto tempo nello sviluppo del software;
- **Crawler Google:** è il motore di ricerca più vasto e performante al mondo, quindi poterlo sfruttare è sicuramente vantaggioso;
- **Crawler4j:** oltre ad essere OpenSource è scritto in Java e quindi è molto più semplice da implementare e interfacciare con il nostro progetto in quanto come detto sopra, Java è il linguaggio con cui abbiamo lavorato;
- **WordNet:** database di sinonimi molto vasto che permette di essere scaricato in locale;
- **Database Online:** necessario per diversi motivi: è sicuramente più scalabile e più robusto che salvare i dati in un database locale. Inoltre permette di avere i dati sempre disponibili da qualunque postazione (mobile e non) essendo sempre attivo, diversamente dai pc personali.

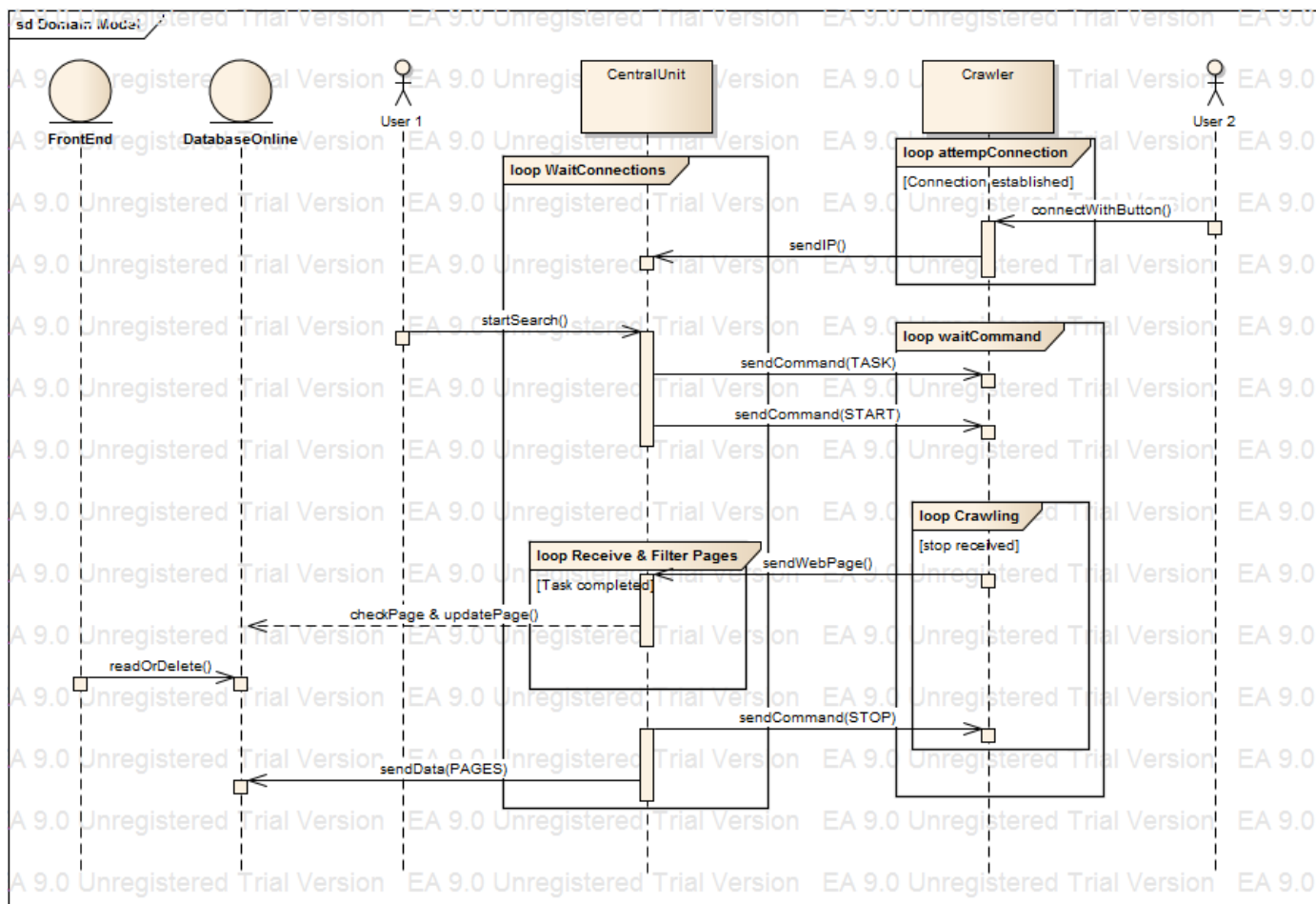


Figura 8: Scenario di interazione standard.

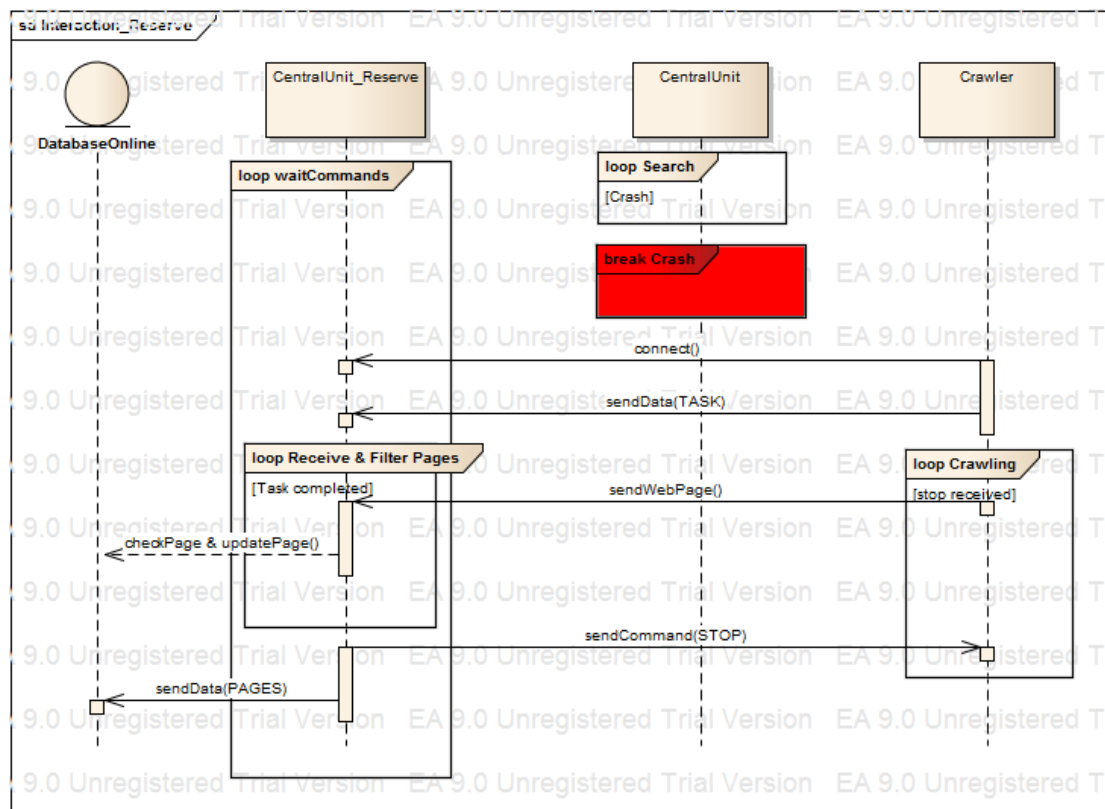


Figura 9: Scenario in cui si fa uso della Central Unit di riserva.

11 Possibili sviluppi futuri

In questa sezione vengono presentate delle possibili soluzioni o evoluzioni del sistema o parti di esso in modo da lasciare a coloro che si vorranno cimentare, un buon punto di ispirazione da cui partire.

11.1 Problema di swap da una Central Unit all'altra

Come si è descritto nella Sottosezione 5.1, quando la prima Central Unit si rende indisponibile per un qualche motivo, si è deciso di far connettere tutti i crawler ad una seconda Central Unit avente tutte le caratteristiche della prima e tramite la quale è possibile continuare, o meglio ricominciare, la ricerca.

Il punto critico è proprio questo ovvero il fatto che se si esegue uno swap di Central Unit si deve per forza ricominciare la ricerca eliminando tutti i risultati trovati fino a quel momento. Per esempio viene fatta partire una ricerca molto corposa e per qualche motivo, dopo qualche ora, la Central Unit si disconnette. Nel tempo passato dall'avvio della ricerca, quest'ultima ha ricevuto tutte le WebPage dai vari crawler connessi che andrà a perdere senza che nessuna tupla venga inserita nel database.

11.1.1 Possibili soluzioni al problema

Per evitare questo spiacevole equivoco si è pensato ad una soluzione abbastanza semplice: ogni x^{14} WebPage che la Central Unit colleziona, viene fatta partire una query di inserimento in modo che salvi il lavoro parziale fatto fino a quel momento. In questo modo si cerca di scongiurare il fatto di ricominciare ogni volta la ricerca.

11.1.2 Problemi che nascono con la possibile soluzione

Un problema nell'implementazione di questa soluzione deriva dal fatto che si è cercato di lasciare il più indipendenti possibile le due Central Unit e quindi tra di loro non c'è nessuna comunicazione. Questo non permette di informare la seconda Central Unit del numero di risultati a cui è giunta la prima. Una possibile soluzione è informare tutti i crawler del numero di pagine che la Central Unit ha ricevuto, tutte le volte che fa un inserimento nel database, in modo che questi ultimi informino la seconda Central Unit. Un'altra semplice soluzione potrebbe essere quella di abilitare la comunicazione tra le due Central Unit in modo da non dover sempre informare i vari crawler degli avvenimenti e lasciarli fare semplicemente il loro lavoro di ricerca.

11.2 Clonazione infinita della Central Unit

Se l'introduzione della seconda Central Unit non fosse abbastanza per gestire alcune problematiche di un sistema distribuito? Se per caso anche quest'ultima si rendesse indisponibile durante la sua esecuzione? Si andrebbero a perdere tutte le connessioni con i crawler e tutti i risultati finora trovati.

¹⁴ x può essere un numero intero a piacere. Ovviamente più è piccolo, più query di inserimento verranno eseguite in uno stesso arco temporale.

11.2.1 Possibile soluzione: tre, quattro, cinque, ..., illimitate Central Unit

La soluzione che subito si è pensata è semplice: perché limitarci ad avere solo una Central Unit di riserva? Perché non è possibile averne diverse? La soluzione, appunto, prevede l'utilizzo di un pull di Central Unit le quali ognuna svolge un compito uguale a quella di riserva specificata nella Sottosezione 5.1. Semplicemente se anche la seconda "cade", si ridirezionano tutti i crawler verso la terza; se anche questa "cade", si ridirezionano tutti i crawler verso la quarta; ecc. Una soluzione limite potrebbe essere quella di far replicare una Central Unit in una sua copia in modo da averne un numero illimitato. Si deve solo capire come gestire il fatto di utilizzare delle diverse connessioni per ogni copia perché se la copia rimane con lo stesso IP o sotto lo stesso dominio e quest'ultimo per un qualche motivo si disconnette, è chiaro che anche la copia non sarebbe molto utile.

11.2.2 Vari problemi di questo approccio

- Un primo problema di questo approccio è il seguente: come faccio a sapere a priori gli indirizzi IP di tutte le Central Unit? Se queste sono in un numero abbastanza limitato allora è possibile inserire gli indirizzi in una lista che, nel caso in cui una Central Unit venga meno, si ridirezionano tutti i crawler verso un'altra della lista. Ma se il numero di queste ultime fosse potenzialmente illimitato si capisce subito che questa soluzione non porta a nessuna conclusione.
- Un altro problema è se un altro crawler volesse accedere al sistema e partecipare alle ricerche. A priori quest'ultimo prova ad accedere alla prima Central Unit ma essendo indisponibile si trova tagliato fuori. Si dovrebbe cercare un modo per poter informare tutti i nuovi crawler che vogliono entrare nel sistema di indirizzarli verso la Central Unit attiva in quel momento.

11.2.3 Altra possibile soluzione

Un'altra possibile soluzione invece di avere un pull di Central Unit è la seguente: installare una sorta di Central Unit all'interno di ogni crawler connesso. In questo modo si evitano alcuni problemi come il fatto di avere le Central Unit di riserva.

In sostanza inizialmente si ha una Central Unit con vari crawler connessi. Se però questa viene a mancare si demanda tutto il suo lavoro ad uno dei crawler connessi (che avrà già al suo interno tutto l'occorrente per gestire questa eventualità). Quindi quest'ultimo farà tutto il compito della prima Central Unit ovvero comunicare con gli altri crawler e connettersi al database online in modo da rendere il sistema come era inizialmente. Se poi anche questo crawler dovesse "morire", allora si demanderebbe tutto il lavoro ad un altro dei crawler connessi, anch'esso avente già tutto l'occorrente per gestire questa evenienza.

Un problema di questa soluzione è che, se una ricerca rimane senza crawler connessi, si andrebbe in uno stato di default. Questo perché nel caso in cui uno dei crawler stesse svolgendo la funzione di Central Unit e rimanesse senza crawler connessi, se anche un nuovo crawler volesse partecipare alla ricerca, non

saprebbe a quale indirizzo connettersi perché non verrebbe informato da nessuno circa il cambiamento di Central Unit.

11.3 Limitare il numero di interrogazioni al database

Nella soluzione implementata, ogni volta che un crawler manda una WebPage alla Central Unit, quest'ultima prima di inserirla nella lista di WebPage descritta nella Sezione 3.1.1 controlla che non sia già stata cercata nella sessione di ricerca corrente e che non sia già presente nel database online.

Questo implica un elevato numero di interrogazioni al database che rallentano pesantemente tutto il sistema.

Una possibile soluzione per scongiurare queste interrogazioni potrebbe essere quella di scaricare una copia in locale di tutte le chiavi primarie delle pagine già presenti nel database. In questo modo viene eseguita una sola lettura. Successivamente la Central Unit, ogni volta che riceve una WebPage, non deve fare altro che controllare se quest'ultima non sia già stata cercata nella sessione corrente di ricerca (ovvero se non è già presente nella lista di WebPage) o se la chiave primaria della WebPage ricevuta non sia già presente nella lista salvata in locale contenente tutte le chiavi primarie.

In questo modo si va ad abbattere il numero di interrogazioni limitando l'interazione che si ha con il database solamente alla lettura iniziale e all'inserimento o cancellazione di tuple.

11.3.1 Possibile problema e relativa soluzione

Un possibile problema di questa soluzione potrebbe essere quello del passaggio della lista di chiavi primarie alla Central Unit di riserva o chi per lei. Le possibili soluzioni sono tre:

1. Inviare direttamente la lista alla Central Unit di riserva (ammesso che ci sia una connessione tra loro);
2. Passare attraverso i crawler; se non si vogliono avere connessioni dirette tra le Central Unit, una possibile soluzione è quella di mandare la lista ai crawler che, nel momento in cui si andranno a connettere alla seconda Central Unit, le passino anche questa lista;
3. La nuova Central Unit non fa altro che interrogare nuovamente il database scaricandosi una nuova lista in locale delle chiavi primarie delle tuple già presenti nel database.

11.4 Inserimento timeout nel caso di indisponibilità dei crawlers

Nel caso in cui tutti i crawlers connessi alla Central Unit di riserva si rendano indisponibili è possibile inserire un timeout in modo che, passati un certo numero di secondi, la Central Unit di riserva esegua una query di inserimento al database con i risultati che ha collezionato fino a quel momento. Questo viene fatto perché nella soluzione descritta nella Sottosezione 6.2 si aspetta che un'altra Central Unit cada e che venga riattivata quella di riserva. Se questo non capitasse si entrerebbe in un loop infinito.

A Premesse per l'utilizzo di CrawlerCommander

Di seguito vengono specificate alcune premesse prima di spiegare come eseguire una ricerca con CrawlerCommander.

A.1 Note sui due crawler utilizzati

Si è scelto di utilizzare due crawler differenti non per differenziare le ricerche che vengono fatte ma per evidenziare che, anche se sono semplicemente due tecnologie differenti, si è cercato lo stesso di mantenere, dove possibile nel codice, la stessa struttura. Possono essere usati da soli e insieme perché raccolgono gli stessi dati e svolgono i medesimi compiti.

A.2 Ricerca con sinonimi

Se si vuole utilizzare la ricerca con sinonimi si deve installare in locale WordNet. Successivamente si deve specificare il path della cartella dove andare a prendere i sinonimi (di solito viene posizionata in C:\ProgramFiles(x86)\WordNet\2.1\dict).

A.3 Database online

Un'altra premessa riguarda il database online che si è usato. Se ne è scelto uno che va bene per il nostro scopo ovvero salvare online non una gran mole di dati. Però sappiamo bene che ne esistono di molto migliori in circolazione. Un'altra cosa che viene richiesta è il fatto di dover indicare una volta a settimana che si sta ancora usando il database. Semplicemente arriva una mail per confermare ciò.

A.4 Settare le varie informazioni

Per i setting vari si deve modificare la classe `SysKb.java` dei vari componenti: Central Unit principale, quella di riserva ed i due crawler. Per le specifiche dei vari campi è stata creata la javadoc che indica cosa rappresentano. (Figura 10).

A.5 Tutorial seguiti

La guida che si è seguita per poter utilizzare Google è la seguente:

```
http://weblog4j.com/2014/06/03/  
having-fun-with-google-custom-search-api-and-java/
```

Per quanto riguarda WordNet invece, il tutorial che si è eseguito è il seguente:

```
http://lyle.smu.edu/~tspell/jaws/
```

Infine per crawler4j si è usato questo:

```
https://github.com/yasserg/crawler4j
```

```
// dati relativi al motore di ricerca creato, non vanno modificati
private static String googleSrcUrl = "https://www.googleapis.com/customsearch/v1?";
private static String apiKey = "AizaSyBrjIRge_qdYx-0Xtx948k_KH2lxli5_5g";
private static String searchEngineId = "007390308112418266202:xga2fmt6xd8";
private static String finalUrl = googleSrcUrl + "key=" + apiKey + "&cx="
    + searchEngineId;
```

Figura 10: Dati del crawler di Google nel file *crawlergoogle.SysKb.java*.

B Tutorial per l'utilizzo di CrawlerCommander

Di seguito vengono precisati i passaggi per eseguire una ricerca con CrawlerCommander ed i vari crawler.

1. Per prima cosa si deve avviare la Central Unit (file `centralunit.CrawlerCommander.java`), la Central Unit di riserva (file `CentralUnitRiserva\centralunit.CrawlerCommanderReserve.java`) e i vari crawlers (file: `crawler4j.CrawlerViewer.java` e `crawlergoogle.CrawlerViewer.java`). Si ricordi che due istanze di `crawler4j` non è possibile averle su una stessa macchina contemporaneamente (si veda la Sottosezione 4.4 per chiarimenti a riguardo).
2. Successivamente si deve indicare nelle varie interfacce dei crawler l'indirizzo IP e la porta dove reperire la Central Unit principale e quella di riserva. Poi se si vuole connettere il crawler alla Central Unit si deve spingere sul tasto *Launch!*.
3. Dopo (o anche prima) aver connesso i vari crawler è possibile inserire la *keyword* e il numero di risultati che si vuole ottenere nell'interfaccia grafica della Central Unit e, quando si vuole far partire la ricerca, spingere il tasto *Search*.
4. A questo punto viene eseguita la ricerca descritta in questo documento (se si osserva nella console dei messaggi si noteranno delle stringhe informative). Una volta terminata si riattiverà il tasto *Search* e si disattiverà il tasto *Stop*.
5. Se si volessero verificare le tuple inserite, basta lanciare il front end desktop (nel progetto `FrontEndDesktop` e avviare `tabella.Main.java`) oppure dall'app mobile per Android (installando l'app presente nella cartella `apk` del progetto `CrawlerReader`).
6. Nel caso in cui si volesse testare la Central Unit di riserva è possibile farlo nel seguente modo. Una volta fatta partire una ricerca (con un numero non troppo basso di risultati) si termina la Central Unit primaria (tramite il tasto di stop di Eclipse o chi per lui) e si vedrà nella console delle stringhe che indicano la "migrazione" dei crawler verso quella di riserva e il continuo della ricerca.