Alma Mater Studiorum · Università di Bologna

Dipartimento di Informatica: Scienza e Ingegneria (DISI)
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Indicizzazione di siti Web in JADE

Corso di Sistemi Distribuiti

Diego Alicata

Anno Accademico 2012/13

Indice

Capitolo 1

Introduzione

La seguente relazione vuole documentare la metodologia di analisi, ricerca e sviluppo seguita per la realizzazione del progetto. Nel contempo, entro ragionevoli limiti di sinteticità, vuole anche fornire la documentazione necessaria per l'utilizzo dello stesso da parte degli utenti finali e di possibili altri studenti o ricercatori che vogliano estendere il lavoro effettuato.

1.1 Specifiche

L'analisi di interi siti web ed in particolare di domini di secondo o terzo livello è un servizio sempre più richiesto da un'ampia platea di addetti ai lavori (webmaster, specialisti seo, analizzatori di contenuti). Le pagine di un sito dicono infatti molto sulla natura dello stesso e permettono di rispondere a molte domande inerenti i suoi contenuti. Appare tuttavia impensabile la creazione di un sistema capace di indicizzare una parte significativa del web a causa degli altissimi costi e di tutte le risorse che sarebbero necessarie. Lo stesso Google analizza solo una parte di tutto il web visibile e ne memorizza una parte ancora più piccola.

Appare quindi vantaggiosa ed interessante l'idea di realizzare un sistema distribuito capace di effettuare una indicizzazione ON DEMAND. Considerando un campione di 100, 1000 pagine, alla richiesta dell'utente il sistema sarà capace di distribuire il carico computazione dell'analisi di tale mole di dati tra tanti agenti Jade in grado di comunicare e di organizzarsi.

Il progetto potrà avere le seguenti caratteristiche:

- Basato su infrastruttura Jade per quel che riguarda la computazione, la coordinazione e l'estrapolazione di features relative alle pagine. Per il prototipo saranno utilizzate due o tre macchine su cui gireranno gli agenti Jade;

- Dotato di un'interfaccia web dalla quale richiedere il report dei siti (tale interfaccia può far uso di tecnologie quali Ajax per il lato client allo scopo di poter mostrare anche risultati parziali e lo stato dell'indicizzazione in corso). Anche il servizio web di interfaccia potrebbe essere distribuito su più macchine, per il prototipo basterà realizzarlo su una macchina;

- Dotato di un database. Il database fornisce i dati all'interfaccia web di report e contiene tutti i dati indicizzati dagli agenti. Permette di accedere ai dati memorizzati anche se il sottosistema computazionale di indicizzazione non è disponibile. Per il prototipo si può pensare di sviluppare tale DB in Mysql tenendo presente che in ambito commerciale, al fine di rendere il sistema nella sua interezza distribuito e maggiormente fault tolerant, si potrebbe utilizzare Oracle.

1.2 Web api

Al fine di disaccoppiare il più possibile il sottosistema di scansione ad agenti dal database e per rendere maggiormente flessibile il sistema nel suo complesso si può pensare di realizzare delle web *API*[1] (di tipo RESTful). Esse trovano sempre maggiore diffusione e costituiscono il fulcro di molti sistemi distribuiti in ambito commerciale. Grandi realtà quali Twitter o Amazon forniscono liberamente agli sviluppatori le proprie web api, per facilitare la comunicazione e l'integrazione di sistemi eterogenei.

Le web api (come del resto i web service RESTful in generale) presentano l'ulteriore vantaggio di poter funzionare correttamente anche in presenza di firewall e blocchi di sicurezza aziendali. Esse infatti usano di default la porta 80: tale porta è, normalmente, una delle poche (se non l'unica) lasciata aperta dai sistemi firewall. Non necessitano quindi di modifiche alle configurazioni di sicurezza.

Le web api promuovono intrinsecamente la openness del sistema e possono favorire sviluppi futuri del progetto. Al tempo stesso permettono di proteggere e validare le richieste al sistema fornendo così un ulteriore livello di sicurezza.

1.3 Finalità del progetto

Il presente progetto vuole quindi approfondire ed esplorare alcuni temi importanti nell'ambito dei Sistemi Distribuiti quali:

- Agenti: comunicazione ed organizzazione;

- Web api: le web api a supporto degli agenti nella realizzazione di un sistema distribuito;

[1] Application Programming Interface.

- Ajax come mezzo per la scalabilita architetturale: demandare al client parte dell'elaborazione ottenendo un'interfaccia più reattiva e migliorando anche la user experience;

- Interoperabilità tra tecnologie e strumenti diversi: come un sistema distribuito possa essere costituito da parti e sottosistemi eterogenei.

Capitolo 2

Analisi di siti web: principali problemi e soluzioni

Nel presente capitolo viene inquadrato il problema dell'analisi di siti web, vengono mostrati i principali concetti ad esso associati e analizzate le principali possibili problematiche.

2.1 L'estrapolazione di features da pagine web

Ogni pagina web ha nel suo contenuto HTML una serie di informazioni che è possibile estrapolare tramite tecniche di parsing e di text mining. Molte di tali informazioni sono contenute all'interno del tag head, altre è possibile estrarle dal body della pagina.

Le principali features estraibili da una pagina web sono:

- Il titolo della pagina: fornisce informazioni importanti sull'argomento oggetto della pagina;
- La descrizione: è costituita da una serie di frasi che descrivono nel dettaglio il contenuto. Viene specificata esplicitamente in un metatag dell'head dell'HTML. Rappresenta la volontà esplicita del webmaster o dell'autore della pagina nei riguardi di come vuole che essa venga mostrata dai motori di ricerca;
- Le keywords: sono parole chiavi relative al contenuto. Possono essere esplicite quando specificate in un metatag dell'head dell'HTML, oppure più frequentemente implicite. In quest'ultimo caso è possibile estrapolarle dal testo riconoscendo le frasi maggiormente ricorrenti.
- I link interni ed esterni con relativo testo di ancoraggio: mostrano come i dati sono collegati fra loro e rappresentano il segnale più importante preso in considerazione dai motori di ricerca.

2.2 Gli spider e l'indicizzazione

Seguendo la metafora del web inteso come ragnatela di siti interconnessi, viene chiamato spider (o crawler) un software che scarica ed analizza i contenuti presenti in una o più pagine web. Lo spider ha capacità di interpretazione dei diversi tipi di encoding utilizzati dai server web ed è anche capace di valutare eventuali codici di risposta forniti dai web server in conseguenza del suo passaggio.

Il frutto del lavoro dello spider è l'indicizzazione. Allorchè lo spider estrapola features di interesse, esse vengono memorizzate tramite un sistema di indicizzazione e risultano da quel momento disponibili per la consultazione.

2.3 Politeness e uso delle risorse

Ogni sito web risiede in uno o più web server. La disponibilità di banda e di risorse di computazione a seconda dei casi possono essere più o meno ampie, ma sono sempre limitate. Al fine di non saturare le risorse del sito web che visita, ogni spider deve quindi fare richieste rispettando dei vincoli di politeness.

Uno spider viene definito *polite*[1] quando non causa problemi al web server e non si dimostra invasivo. Non solo un singolo spider deve rispettare vincoli di politeness, ma qualsiasi sistema composto da un'insieme di spider nella sua interezza dovrà rispettarli.

Per rispettare tali vincoli lo spider dovrà limitare la quantità di richieste per unità di tempo ad un sito web. Ciò permetterà di evitare involontari *denial of service*[2] e allo stesso tempo permetterà al sistema di utilizzare meglio le proprie risorse di indicizzazione. Se infatti un sito tarda a rispondere, non è conveniente perseverare nelle richieste, esse causerebbero ulteriori ritardi e favorirebbero problematiche. La disponibilità di risorse di computazione può allora essere utilizzata per effettuare altre richieste ad altri siti in coda.

Un sistema di indicizzazione distribuito dovrà tener conto dei requisiti di politeness a livello globale, in tal modo le richieste dei vari spider distribuiti non risulteranno invasive ed il sistema si mostrerà nella sua interezza maggiormente prevedibile ed affidabile.

2.4 Il file robots.txt e il Robots Exclusion Protocol

Ogni sito web può decidere, a seconda delle proprie esigenze, di permettere o non permettere l'indicizzazione dello stesso da parte di spider.

Il Robots Exclusion Protocol rappresenta uno standard de facto che permette di rendere esplicita e comprensibile da qualsiasi spider tale volontà. Il

[1] Termine inglese dal significato: gentile, garbato, quindi non invasivo.

[2] Malfunzionamento solitamente conseguenza di un attacco informatico in cui si esauriscono le risorse di un sistema informatico saturandole.

protocollo fa affidamento su un file nominato robots.txt che può essere piazzato dal webmaster nella root del proprio sito web. La direttiva disallow correlata ad uno o più user-agent esplicita quali parti del sito web non vadano indicizzate.

E' buona norma che ogni spider rispetti le direttive del sito web, onde evitare possibili problemi legali inerenti la violazione di sistemi informatici.

L'esempio seguente mostra il file robots.txt di un sito che desidera che nessun contenuto sia indicizzato da parte di nessuno spider.

```
User-agent: *
Disallow: /
```

L'esempio seguente mostra invece il file robots.txt di un sito che non vuole la directory docs indicizzata da parte di Google.

```
User-agent: Googlebot
Disallow: /docs
```

All'indirizzo http://www.useragentstring.com/pages/useragentstring.php è possibile recuperare una lista degli user-agent relativi ai più popolari spider.

Capitolo 3

Analisi e progettazione del sistema

Nel presente capitolo viene mostrata l'analisi del problema e la progettazione del sistema.

3.1 Casi d'uso

In primo luogo, al fine di strutturare al meglio il sistema, è possibile osservarlo dall'esterno e caratterizzarlo a partire dalle funzionalità e dai casi d'uso dei principali attori coinvolti.

In considerazione delle specifiche di progetto il sistema deve fornire al generico utente alcune funzionalità e in particolare:

L'utente può:

- Richiedere rapporti su un sito;
- esaminare i rapporti su un sito.

Inoltre per meglio gestire il sistema, si può pensare di dare la possibilità all'utente di:

- Mettere in pausa o far riprendere l'indicizzazione di un sito;
- vedere lo stato dei container jade.

L'utente, per poter accedere alle funzionalità fornite dal sistema, deve prima autenticarsi tramite login.
L'utente godrà quindi, nel presente prototipo, di privilegi amministrativi. Nel prodotto finale per questioni di sicurezza si può pensare di scorporare tali privilegi dando origine alla figura dell'amministratore.

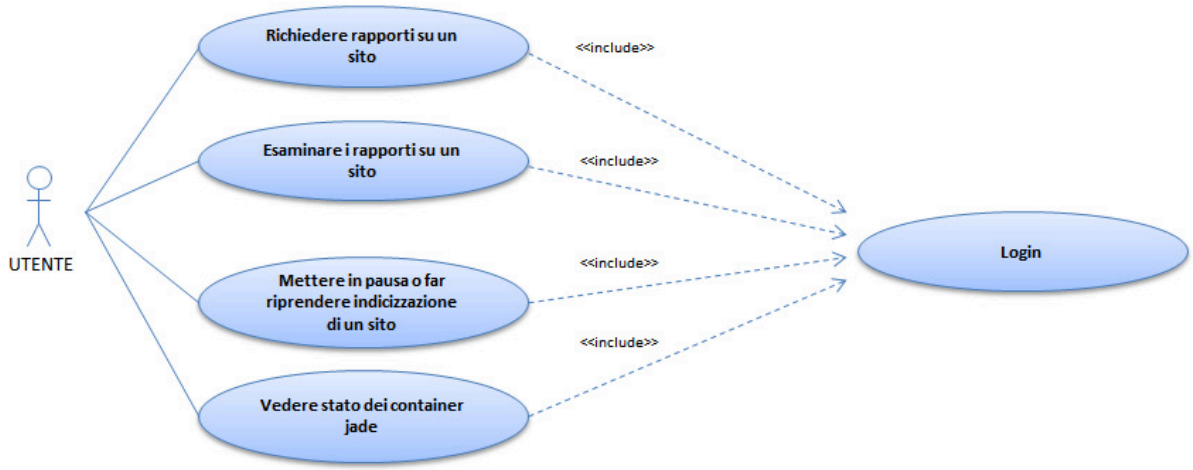


Figura 3.1: Il diagramma dei casi d'uso

3.2 L'architettura del sistema

E' possibile strutturare il sistema in differenti sottosistemi dando luogo a una architettura a tre livelli formata da:

- Il livello di presentazione che offre l'interfaccia web al client;
- il livello intermedio che si occupa della business logic;
- il livello dati rappresentato dal database.

Il livello business può essere ulteriormente organizzato in: web api e agenti Jade. Gli agenti effettuano la scansione dei siti web, estrapolano le features e comunicano tramite web api con il sottostante livello dati. Le web api partecipano attivamente alla business logic e forniscono il servizio di supporto agli agenti. Il livello di presentazione otterrà i dati da mostrare interrogando le web api. In figura 3.2 è possibile visionare uno schema dell'architettura del sistema delineata.

Tale architettura migliora il disaccoppiamento, l'intercambiabilità e la modificabilità dei sottosistemi. Creando ad esempio un applicativo alternativo di interfaccia per Android o iPhone è possibile aggiungere funzionalità di presentazione dei dati su applicazioni ottimizzate per cellulari senza bisogno di modifiche agli altri sottosistemi.

E' questo solo uno degli esempi possibili che la openness di un sistema così strutturato può fornire.

E' possibile notare che ogni sottosistema può essere distribuito su più macchine per migliorare la fault tolerance, la scalabilità e fornire load balancing.

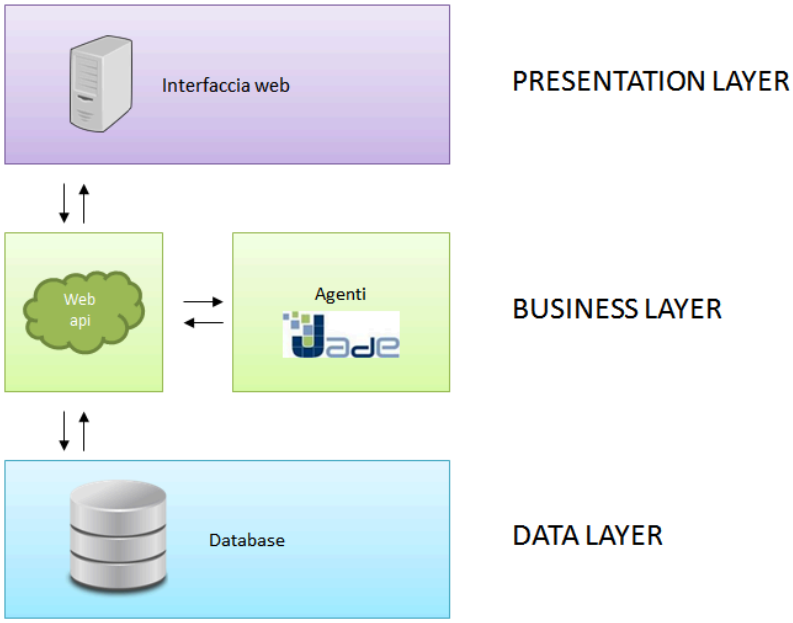


Figura 3.2: L'architettura del sistema

Nel presente progetto, come da specifiche, solo il sottosistema di scansione basato su agenti Jade sarà distribuito su più macchine. Mentre il livello di presentazione, le web api e il database saranno fisicamente su una macchina ognuno.

Nelle successive parti verranno analizzati nel dettaglio i sottosistemi.

Il database

Il database dovrà essere in grado di registrare e fornire le seguenti informazioni:

- Siti indicizzati e da indicizzare;
- pagine indicizzate e da indicizzare;
- keywords, collegamenti interni ed esterni;
- stato del sistema;
- utenti e loro credenziali.

Gli agenti Jade

Allo scopo di rendere il sottosistema di scansione maggiormente fault tolerant, minimizzare le richieste e aumentare la capacità indicizzativa possono essere progettati diversi tipi di agenti:

- Un ManagerAgent che presiede alla coordinazione locale del container jade e che sfrutta le web api come supporto per la coordinazione globale del sistema e per la memorizzazione delle features;
- uno o più SpiderAgent capaci di analizzare pagine web. Comunicano col manager per stabilire quale pagina scansionare e per inoltrare i risultati della scansione;
- un CheckerAgent capace di istanziare gli agenti e che a seguito di periodici controlli è in grado di reistanziare il manager se terminato inaspettatamente. Si tratta di un agente che svolge un ruolo primario durante l'inizializzazione e ausiliario durante la fase di vita del container. Esso permette inoltre di migliorare la fault tolerance.

Le web api

Le web api forniscono supporto agli agenti e vengono utilizzate dall'interfaccia per richiedere i risultati dell'indicizzazione. Le funzionalità quindi che devono fornire sono le seguenti:

- Permettere la registrazione delle features relative alle pagine scansionate;
- registrare lo stato di attività degli agenti;
- fornire a richiesta uno o più url che gli agenti possono indicizzare a secondo delle necessità del sistema e tenuti in considerazione gli obbiettivi di politeness che si vogliono raggiungere;
- fare un controllo preventivo del sito da indicizzare controllando l'eventuale file robots.txt;
- permettere la lettura di tutti i risultati dell'indicizzazione;
- permettere la lettura dello stato degli agenti.

Le web api hanno un ruolo principale nella coordinazione globale e fanno in modo che gli spider non effettuino richieste eccessive nell'unità di tempo ad uno stesso sito.

L'interfaccia web

L'interfaccia web risiede in un web server. Il web server fornisce al browser le pagine html relative al contenuto presentazionale del sistema. L'interfaccia web comunica con le web api.

Allorchè un utente richiede informazioni su un sito web l'interfaccia:

- Se i dati sono disponibili li mostra.
- Altrimenti comunica tramite web api la richiesta di indicizzazione di nuovi dati a partire dall'home page. Le web api accedono al file robots.txt del nuovo sito se presente e se il sito non nega la possibilità di essere indicizzato registrano l'url. A questo punto gli agenti Jade attivi, allorchè richiedono al Manager cosa possono scansionare, iniziano a ricevere la notifica di scansionare gli url del nuovo sito. Dopo qualche secondo l'interfaccia inizia ad ottenere tramite web api i nuovi dati disponibili.

Capitolo 4

Implementazione

Nel presente capitolo viene mostrato in dettaglio il sistema realizzato.

4.1 Tecnologie utilizzate

Al fine di creare un sistema costituito da sottosistemi eterogenei sono state utilizzate le seguenti tecnologie:

- Java e Jade per gli agenti;
- C# per le web api e per il web server di interfaccia;
- Mysql e il motore di memorizzazione InnoDB con supporto alle transazioni per quanto riguarda il database;
- Html 5, Ajax e Css come tecnologie delle quali fa uso l'interfaccia web.

4.2 Metodologia

La separazione in sottosistemi ha facilitato la loro specializzazione nei riguardi dei compiti preposti permettendo di astrarre dai dettagli implementativi degli altri sottosistemi. Allo scopo di migliorare la manutenibilità ogni sottosistema è stato sviluppato facendo largo uso di commenti, dando nomi esplicativi ad i metodi implementati e tenendo in grande considerazione la pulizia del codice e la leggibilità dello stesso.

4.3 Gli agenti Jade nel dettaglio

Il sottosistema di scansione basato su agenti Jade è stato realizzato tramite ambiente di sviluppo NetBeans 7.2.

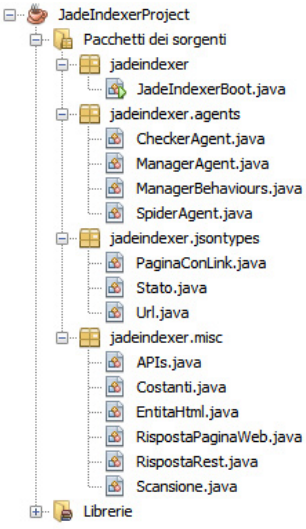


Figura 4.1: Il progetto Jade

La classe di avvio JadeIndexerBoot permette l'avvio del framework Jade e di un main container con un numero di default di spider sia direttamente dall'ambiente di sviluppo che dal jar di release.

Sono state creare due classi di utilità: APIs e Scansione. La prima offre metodi per la comunicazione con le web api, la seconda metodi per la scansione ed il reperimento delle features da pagine web.

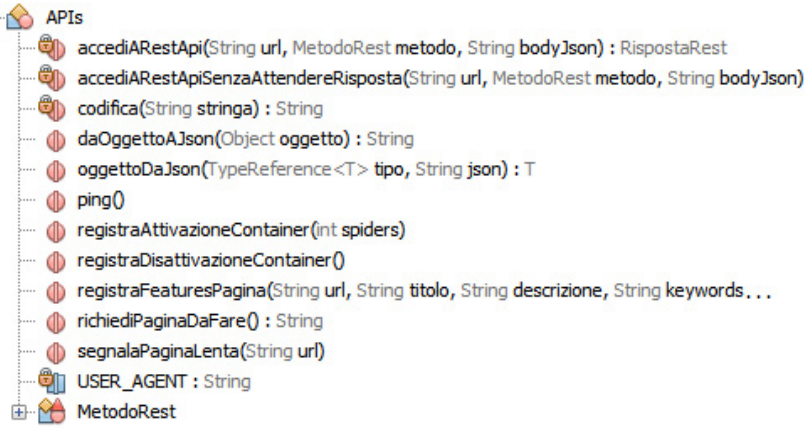


Figura 4.2: I membri della classe APIs

Figura 4.3: I membri della classe Scansione

In fase di avvio viene inizializzato il container Jade e caricato il CheckerAgent. Esso avvia il ManagerAgent e poi il numero specificato di SpiderAgent.

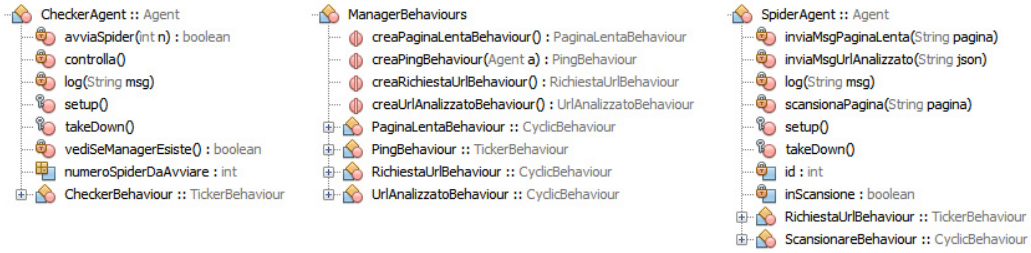


Figura 4.4: I vari agenti implementati con i relativi behaviours e la classe factory per i behaviours del Manager

Gli spider una volta attivi chiedono periodicamente al manager la prossima pagina da scansionare. Il manager comunica con le web api e reperisce tale informazione che viene poi inoltrata allo spider richiedente. A questo punto lo spider scansiona la pagina ed invia i risultati della computazione in formato JSON al Manager. Il manager può così inviare i risultati alle web api. L'esistenza del manager come unico comunicatore con le api riduce il numero di richieste e può permettere una serie di ulteriori ottimizzazioni future. Ad esempio il manager potrebbe aspettare un certo numero di secondi ed inviare in blocco l'esito della scansione di una quantità di pagine alle web api minimizzando ulteriormente il numero di richieste.
Si invita a fare riferimento alla documentazione JavaDoc allegata per i dettagli completi dell'implementazione del sistema ad agenti.

4.4 L'interfaccia web

L'interfaccia web gestisce l'interazione con l'utente e comunica con le web api per le richieste di servizio.

La scelta della separazione tra elaborazione e presentazione dell'informazione ha portato all'utilizzo di codice codebehind per l'esecuzione delle elaborazioni e di pagine ASPX che presentano i dati.

Il codice html all'interno delle pagine è stato opportunamente ottimizzato e infine validato con il plugin per Firefox Tidy allo scopo di ottenere pagine aderenti al *doctype*[1] HTML 5 e visualizzabile senza difetti nei principali browser.

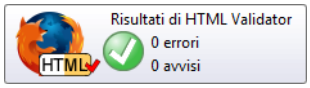


Figura 4.5: Risultati validazione Tidy

L'utilizzo estensivo dei *CSS*[2] ha permesso di separare ulteriormente nell'html la presentazione dei dati dal contenuto. Come risultante di ciò le pagine html contengono principalmente il testo da mostrare, mentre il layout è demandato ad i css e la gestione di controlli lato client ad appositi file javascript associati alle pagine che ne hanno bisogno.

Per fornire un esperienza utente superiore si è fatto uso intensivo di Ajax e di alcune librerie javascript per la visualizzazione grafica in tabelle dei dati. In particolare sono stati usati jQuery [14], DataTables [15] e Google Chart Tools [16]. E' il client browser che effettua grazie al javascript molti dei compiti che altrimenti sarebbero stati prerogativa del web server di interfaccia. Il client richiede l'aggiornamento dei dati con cadenza periodica ed è sempre il client a ordinare i risultati ottenuti secondo le esigenze dell'utente in tal modo non vi è necessità da parte del server di creare html per tabelle o immagini di grafici per i dati. Il risparmio è in termini di tempo, banda e risorse. Tutto ciò inoltre migliora complessivamente la scalabilità del sistema.

Ulteriori dettagli implementativi sono disponibili nella relativa documentazione allegata.

[1] Lo scopo di un Document Type Definition è quello di definire le componenti ammesse nella costruzione di un documento XML o HTML.

[2] Cascading Style Sheet.

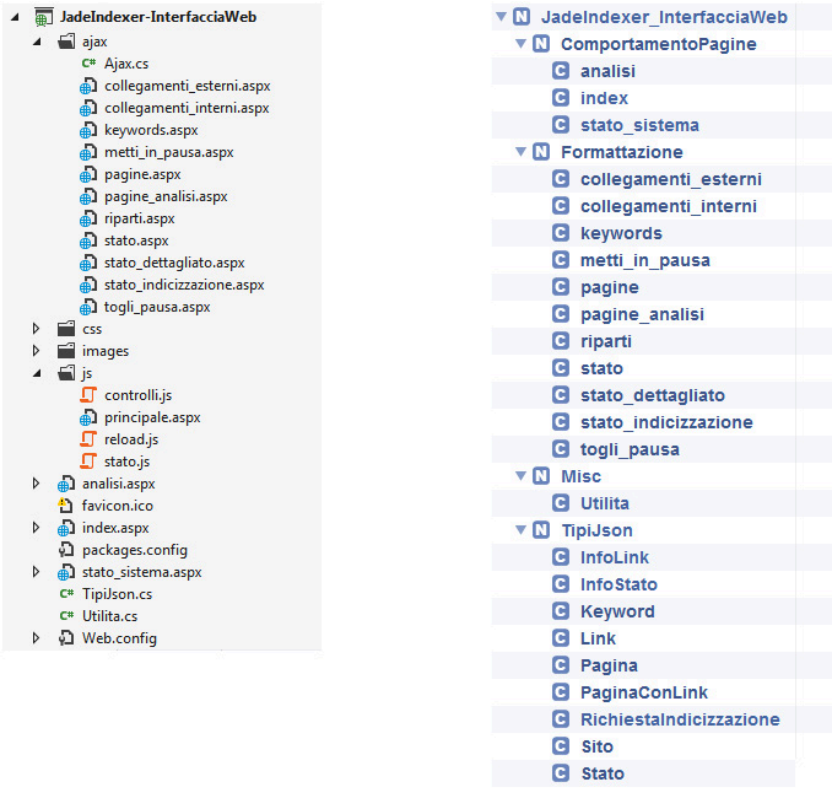


Figura 4.6: La struttura del progetto di interfaccia web a sinistra. A destra i namespace e le classi sviluppate

4.5 Le web api

Le web api sono state sviluppate in C#.
Come formato di scambio dati è stato scelto il formato *JSON*[3] che, per la sua compattezza, permette di risparmiare banda trasmissiva durante lo scambio di grosse mole di dati.
Segue la lista delle web api implementate con il dettaglio su come chiamarle e con quali parametri.

- **http://ji-api.diegoalicata.it/siti/**:

  Metodo POST Input body: TipiJson.Sito Inserisce un sito da indicizzare

- **http://ji-api.diegoalicata.it/siti/<url>**:

  Metodo GET Restituisce informazioni su un sito indicizzato

[3]JSON: JavaScript Object Notation. Per approfondimenti: JSON [8] - The Fat-Free Alternative to XML

Metodo DELETE Elimina le informazioni di indicizzazione relative ad un sito

- **http://ji-api.diegoalicata.it/pagine/**:

  Metodo GET Parametro: dom Restituisce tutte le pagine indicizzate relative al dominio dom

  Metodo POST Input body: TipiJson.PaginaConLink Registra una pagina con tutte le features e i link interni ed esterni associati

- **http://ji-api.diegoalicata.it/pagine_da_scansionare/**:

  Metodo GET Restituisce la prossima pagina da scansionare

- **http://ji-api.diegoalicata.it/link_interni/**:

  Metodo GET Parametro: dom Restituisce i link interni relativi al dominio dom

- **http://ji-api.diegoalicata.it/link_esterni/**:

  Metodo GET Parametro: dom Restituisce i link esterni relativi al dominio dom

- **http://ji-api.diegoalicata.it/keywords/**:

  Metodo GET Parametro: dom Restituisce le keywords principali relative al dominio dom

- **http://ji-api.diegoalicata.it/stato/**:

  Metodo GET Restituisce lo stato dei container di scansione

  Metodo POST Input body: TipiJson.Stato Registra o modifica lo stato dei container (anche usato per il ping)

- **http://ji-api.diegoalicata.it/stato/<ip>-<computer>**:

  Metodo DELETE Elimina lo stato del container in oggetto

- **http://ji-api.diegoalicata.it/info_stato/**:

  Metodo GET Restituisce lo stato dei container di scansione fornendo il numero totale di container attivi e di spider attivi

- **http://ji-api.diegoalicata.it/pagina_lenta/**:

  Metodo POST Input body: String Permette la segnalazione di una pagina lenta

- **http://ji-api.diegoalicata.it/autenticazione_utente/<utente>**:

  Metodo GET Parametro: MD5password Restituisce codice 200 (OK) se i dati di autenticazione sono corretti, altrimenti 401 (Unauthorized).

Ulteriori dettagli implementativi sono disponibili nella relativa documentazione allegata.

Di seguito viene mostrato un estratto del controller Pagine con l'implementazione del metodo GET. Il .Net Framework 4.0 permette di definire in C# in modo nativo le chiamata web api e con estrema eleganza è possibile specificarne il comportamento. Si può notare che non c'è alcuna indicazione sul formato di scambio dati. Il return di una lista di pagine viene automaticamente convertito dal sistema in una lista di pagine JSON come da configurazione.

```
public class PagineController : ApiController
{
    Utilita.AccessoDB DB = new Utilita.AccessoDB();

    // GET /pagine
    public IEnumerable<Pagina> Get([FromUri] string dom)
    {
        List<Pagina> risultati = new List<Pagina>();

        int idSito = -1;
        DB.apri();

        idSito = Utilita.getIdSito(dom, DB.mySqlConnection);

        if (idSito == -1)
        {
            DB.chiudi();
            throw new HttpResponseException(HttpStatusCode.NotFound);
        }
        else
        {
            DB.mySqlCommand.CommandText = "SELECT Pagine.Pagina, Titolo, Keywords,
                Numero_Link_Interni, Numero_Link_Esterni, Descrizione, Dimensione,
                DocType FROM Pagine WHERE SitoId = '" + idSito + "' AND
                DaAggiornare = 0 AND Aggiornamento <> '0000-00-00 00:00:00'";
            DB.myDataReader = DB.mySqlCommand.ExecuteReader();

            while (DB.myDataReader.Read())
            {
                Pagina singolaPaginaLetta = new Pagina();

                singolaPaginaLetta.url = DB.myDataReader.GetString(0);
                singolaPaginaLetta.titolo = DB.myDataReader.GetString(1);
                singolaPaginaLetta.keywords = DB.myDataReader.GetString(2);
```

```
                singolaPaginaLetta.interni = DB.myDataReader.GetInt32(3);
                singolaPaginaLetta.esterni = DB.myDataReader.GetInt32(4);
                singolaPaginaLetta.descrizione = DB.myDataReader.GetString(5);
                singolaPaginaLetta.dimensione = DB.myDataReader.GetDouble(6);
                singolaPaginaLetta.docType = DB.myDataReader.GetString(7);

                risultati.Add(singolaPaginaLetta);
            }
            DB.myDataReader.Close();
            DB.chiudi();

            return risultati;
        }
    }
```

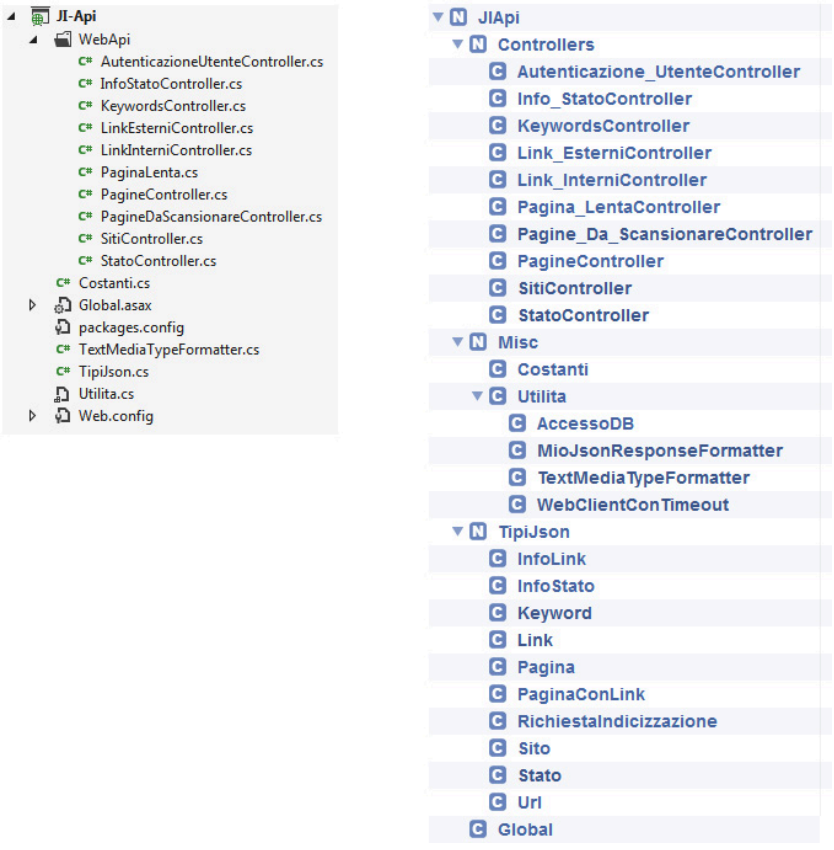


Figura 4.7: La struttura del progetto delle web api a sinistra. A destra i namespace e le classi sviluppate

4.6 Implementazione della politeness

Le web api sono time aware e sono state sviluppate in modo da permettere di gestire la politeness. Le web api forniscono ai vari manager che ne fanno richiesta un nuovo url che è possibile scansionare. Tale url viene scelto in modo che siano rispettati i vincoli di politeness.

In particolare di default ogni url relativo ad un singolo sito viene attribuito ad un manager se è trascorso un TEMPO_ATTESA_DEFAULT (300 ms) dall'ultima attribuzione di un url dello stesso sito. Se uno spider nota un tempo di risposta di una pagina superiore a SOGLIA_PAGINA_LENTA comunica al manager che tale pagina potrebbe essere lenta. La lentezza potrebbe essere un fatto relativo, in quanto potrebbe essere lo spider lento, sovraccarico oppure il container potrebbe disporre di banda limitata. Come ulteriore controllo sono quindi le web api che, contattate da un manager che segnala una pagina lenta, controllano a loro volta la lentezza della pagina. Se si ha un riscontro le web api portano il tempo di attesa relativo al sito da TEMPO_ATTESA_DEFAULT (300 ms) a TEMPO_ATTESA_MAX (3000 ms) in tal modo verrano ridotte le richieste effettuate verso il sito web lento. Trascorso un TEMPO_RITORNO_A_DEFAULT (120 secondi) TEMPO_ATTESA_DEFAULT viene ripristinato.

Il sistema è così in grado di analizzare una situazione origine di possibili problemi ed adattarsi dinamicamente.

4.7 Implementazione della lettura del file robots.txt

Allorchè l'utente richiede tramite interfaccia web la scansione di un sito web le web api, come prima cosa, leggono l'eventuale presenza del file robots.txt per vedere se l'indicizzazione è permessa e altresì si accertano che il sito sia accessibile. Non viene inserito come sito da scansionare un sito che fallisce uno qualsiasi di questi controlli.

4.8 Il database

Particolare attenzione è stata data al problema delle operazioni concorrenti su database. Le transazioni garantiscono una semantica di tipo all-or-nothing: o tutte le operazioni appartenenti ad una transazione hanno successo, oppure nessuna lo ha. L'utilizzo di Mysql 5.6.4 o successivi con motore InnoDB garantisce il supporto alle transazioni e permette la creazione di indici fulltext per il reperimento veloce delle informazioni. Sono state definite le dipendenze foreign key in modo tale che se si cancella un sito vengano eliminati i collegamenti, le keywords e ogni features relativa.

Il database è stato creato secondo il seguente estratto di codice SQL:

```
CREATE TABLE `CollegamentiEsterni` (
  `SitoId` int(11) unsigned NOT NULL,
  `DaPagina` int(11) NOT NULL,
  `DominioEsterno` varchar(255) NOT NULL,
  `DataRilevamento` datetime NOT NULL,
  PRIMARY KEY (`SitoId`,`DaPagina`,`DominioEsterno`),
  KEY `SitoId` (`SitoId`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `CollegamentiInterni` (
  `SitoId` int(11) unsigned NOT NULL,
  `IdPagina1` int(10) unsigned NOT NULL,
  `IdPagina2` int(10) unsigned NOT NULL,
  `DataRilevamento` datetime NOT NULL,
  PRIMARY KEY (`SitoId`,`IdPagina1`,`IdPagina2`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Keywords` (
  `SitoId` int(11) unsigned NOT NULL,
  `Keyword` varchar(255) NOT NULL,
  `Quantita` int(11) NOT NULL,
  PRIMARY KEY (`SitoId`,`Keyword`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Pagine` (
  `Id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `SitoId` int(11) unsigned NOT NULL,
  `Pagina` varchar(255) NOT NULL,
  `Titolo` varchar(255) NOT NULL,
  `Descrizione` varchar(255) NOT NULL,
  `Keywords` varchar(255) NOT NULL,
  `Numero_Link_Interni` int(11) NOT NULL DEFAULT '-1',
  `Numero_Link_Esterni` int(11) NOT NULL DEFAULT '-1',
  `Aggiornamento` datetime NOT NULL,
  `DaAggiornare` tinyint(4) NOT NULL DEFAULT '0',
  `Dimensione` double NOT NULL,
  `Codice_Risposta` smallint(5) unsigned NOT NULL,
  `Assegnata` datetime NOT NULL,
  PRIMARY KEY (`Id`),
  UNIQUE KEY `SitoId` (`SitoId`,`Pagina`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

CREATE TABLE `Siti` (
  `Id` int(10) unsigned NOT NULL AUTO_INCREMENT,
```

```
  `Url` varchar(255) NOT NULL,
  `Data_Inserimento` datetime NOT NULL,
  `Data_UltimoAggiornamento` datetime NOT NULL,
  `Indicizzazione_Attiva` tinyint(4) NOT NULL DEFAULT '0',
  `Ultima_Attribuzione` bigint(20) NOT NULL,
  `Attesa_Politeness_ms` int(11) NOT NULL,
  `Ultima_Variazione_Politeness` datetime NOT NULL,
  `Indicizzazione_Permessa` tinyint(4) NOT NULL DEFAULT '0',
  PRIMARY KEY (`Id`),
  UNIQUE KEY `Url` (`Url`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;

CREATE TABLE `Stato` (
  `Ip` varchar(255) CHARACTER SET latin1 NOT NULL,
  `Computer` varchar(255) CHARACTER SET latin1 NOT NULL,
  `NumeroSpider` tinyint(4) NOT NULL,
  `UltimaAttivita` datetime NOT NULL,
  `OS` varchar(255) CHARACTER SET latin1 NOT NULL,
  PRIMARY KEY (`Ip`,`Computer`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Utenti` (
  `NomeUtente` varchar(255) NOT NULL,
  `Password` varchar(255) NOT NULL,
  PRIMARY KEY (`NomeUtente`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- Foreign keys:

ALTER TABLE `CollegamentiEsterni`
  ADD CONSTRAINT `CollegamentiEsterni_ibfk_1` FOREIGN KEY (`SitoId`)
  REFERENCES `Siti` (`Id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `CollegamentiInterni`
  ADD CONSTRAINT `CollegamentiInterni_ibfk_1` FOREIGN KEY (`SitoId`)
  REFERENCES `Siti` (`Id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `Keywords`
  ADD CONSTRAINT `Keywords_ibfk_1` FOREIGN KEY (`SitoId`)
  REFERENCES `Siti` (`Id`) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE `Pagine`
  ADD CONSTRAINT `Pagine_ibfk_1` FOREIGN KEY (`SitoId`)
  REFERENCES `Siti` (`Id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
-- Indici:

CREATE INDEX ‘SitoId_2‘ ON Pagine (SitoId) USING BTREE;

CREATE INDEX ‘Pagina‘ ON Pagine (Pagina) USING BTREE;

CREATE INDEX ‘Aggiornamento‘ ON Pagine (Aggiornamento) USING BTREE;

CREATE INDEX ‘Keyword‘ ON Keywords (Keyword) USING BTREE;


-- Indici FULLTEXT richiede MySql 5.6.4 o successivi:

CREATE FULLTEXT INDEX ‘Titolo‘ ON Pagine (Titolo);

CREATE FULLTEXT INDEX ‘Descrizione‘ ON Pagine (Descrizione);
```

4.9 Note sulla sicurezza

Nel presente progetto, trattandosi di un prototipo, è stato implementato un controllo delle credenziali a livello di interfaccia web. In particolare le password vengono controllate tramite hash MD5. Nel prodotto commerciale sarà bene estendere tale sistema di controllo e prevedere delle credenziali anche per accedere alle web api magari fornendo il supporto al protocollo OAUTH [19] che è ormai uno standard di autenticazione per l'uso delle web api.

Capitolo 5

Il sistema in azione

Nel presente capitolo viene mostrato come accedere al sistema e vengono mostrate alcune schermate inerenti i risultati forniti a seguito della richiesta di indicizzazione di siti di differente tipologia e caratteristiche.

Il sottosistema ad agenti è stato testato sui sistemi operativi Windows XP, Windows 7, Linux Ubuntu e Mac OS X 10.6.8 Snow Leopard. Per Snow Leopard è stato necessario installare Java 7 tramite una procedura alternativa, in quanto la Apple fornisce solo Java 6. L'interfaccia web è stata testata con i browser Internet Explorer, Mozilla Firefox, Chrome su PC e con Safari su iPad e iPhone.

Figura 5.1: La home page di accesso

5.1 Accesso all'interfaccia

L'interfaccia web è disponibile presso:

`http://jadeindexer.diegoalicata.it`

E' possibile accedere con le seguenti credenziali:

```
NOME UTENTE: adminjade
PASSWORD: jade-2013a
```

5.2 Avvio degli agenti Jade

Figura 5.2: Lo stato degli agenti nell'interfaccia

Dopo il login lo stato in alto a destra indica il numero di container e di spider attivi. Se non ce ne sono disponibili basterà avviare JI-Agenti.jar

Il jar JI-Agenti.jar (disponibile in JI-Agenti-jar.zip), contiene già tutte le librerie necessarie (compreso Jade) ed è avviabile seguendo la seguente sintassi (richiede solo Java 7):

```
java -jar JI-Agenti.jar -spiders:3
```

Ciò istanzierà un container Jade con 3 spider.

Se si vuole anche avviare la GUI di controllo Jade:

```
java -jar JI-Agenti.jar -spiders:3 -avviaGUI
```

Non c'è necessità di ulteriori configurazioni. Gli spider risulteranno disponibili dopo pochi secondi nell'interfaccia web.

5.3 Comandare il sistema

E' possibile inserire l'indirizzo del sito web da esaminare nell'interfaccia web. Il sommario, a questo punto, mostra in tempo reale lo stato di indicizzazione. E' possibile in qualsiasi momento mettere in pausa o far ripartire da zero l'indicizzazione del sito.

Figura 5.3: Il sommario

Cliccando su stato sistema è sempre possibile vedere i container jade attivi con i rispettivi spider istanziati, l'indirizzo ip, il nome del computer e il suo sistema operativo:

Figura 5.4: Lo stato dettagliato del sottosistema ad agenti

Di seguito i principali risultati mostrati per alcuni siti di interesse:

5.4 Esempio per www.unibo.it

Sommario | Pagine | Keywords | Links interni | Links esterni | Analisi pagine

Pagina	Titolo	Keywords	L. int.	L. est.
/	Home Page - Università di Bologna	studenti, online, bologna, maggio, di bologna	88	11
/almaenglish/	AlmaEnglish	almaenglish, corsi, test, progetto, studenti	25	3
/CILTA/AlmaEnglish /Corsiaderenti.htm	Corsi aderenti	ingegneria, facoltà, facoltà di, corsi, scienze	21	2
/CILTA/CorsieLearning/	Corsi in e-learning	learning, corsi in, corsi, idoneità, prove	24	3
/CILTA/Progettiricerca/	Progetti e ricerche	progetti, centro, ricerche, unibo, il centro	23	3
/CILTA/Progettiricerca /AssociazioniCentriLinguistici.htm	Associazioni di Centri Linguistici	linguistici, centri linguistici, centri, universitari, linguistici universitari	20	3
/CMSUniboWeb /UniboSearch/Rubrica.aspx	UniboRubrica - Università di Bologna	università, ricerca, di bologna, bologna, università di bologna	5	0
/Magazine/Attualita/2013/05 /07/Due_grandi_eventi.htm	Due grandi eventi per la campagna 5 per mille Unibo	della, per mille, campagna, mille, sono	28	3
/Magazine/Eventi/	Eventi - UniboMagazine	maggio, seminari, convegni, eventi, bologna	99	1
/Magazine/Eventi/2002/	Eventi - UniboMagazine	giugno, eventi, maggio, della, bologna	71	1
/Magazine/Eventi/2002/06 /24/La+morte+del+paesaggio.htm	La morte del paesaggio	della, morte, come, paesaggio, del paesaggio	64	4

Figura 5.5: Le pagine scansionate con url relativo, titolo, keyword principali, numero link interni ed esterni.

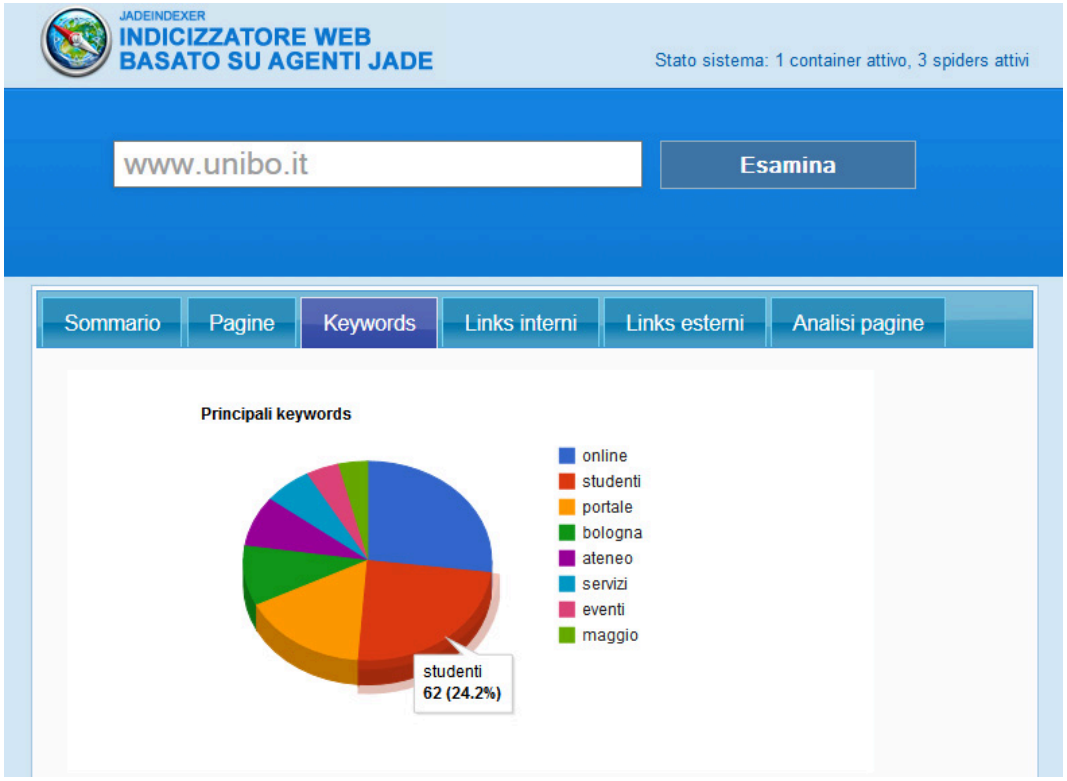
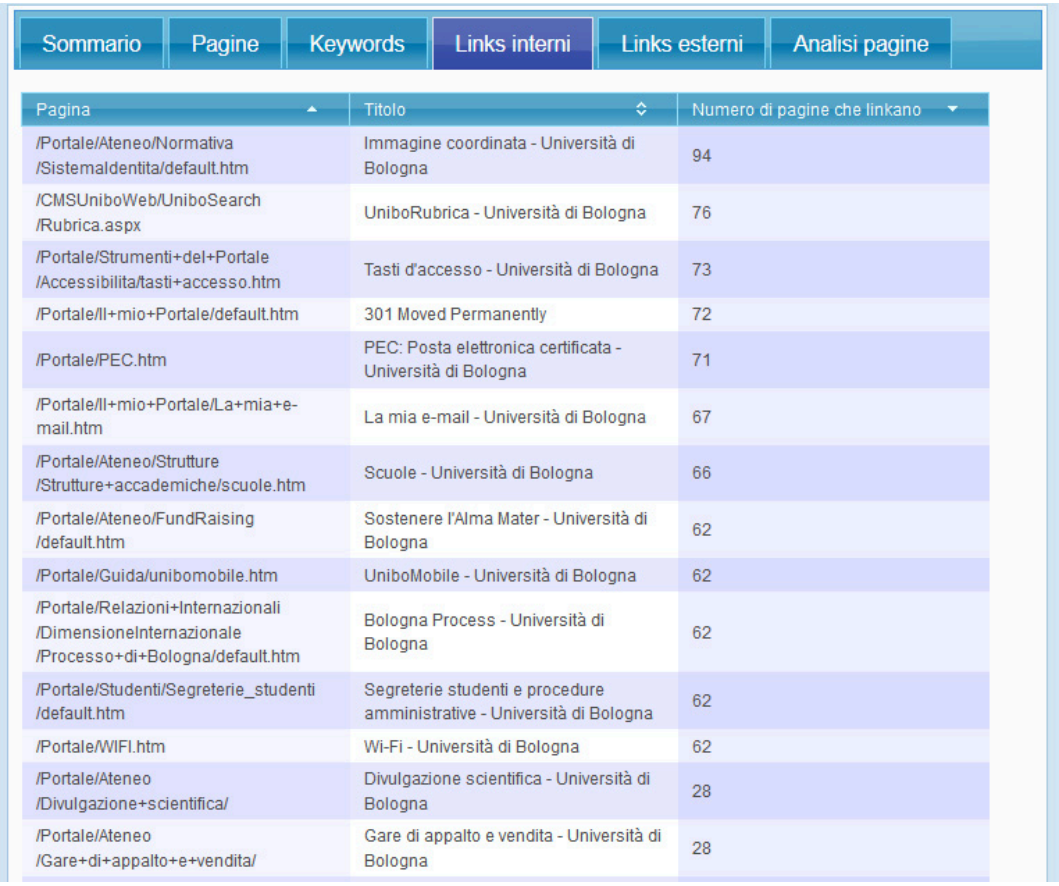


Figura 5.6: Le principali keywords di www.unibo.it considerato il campione di 100 pagine.

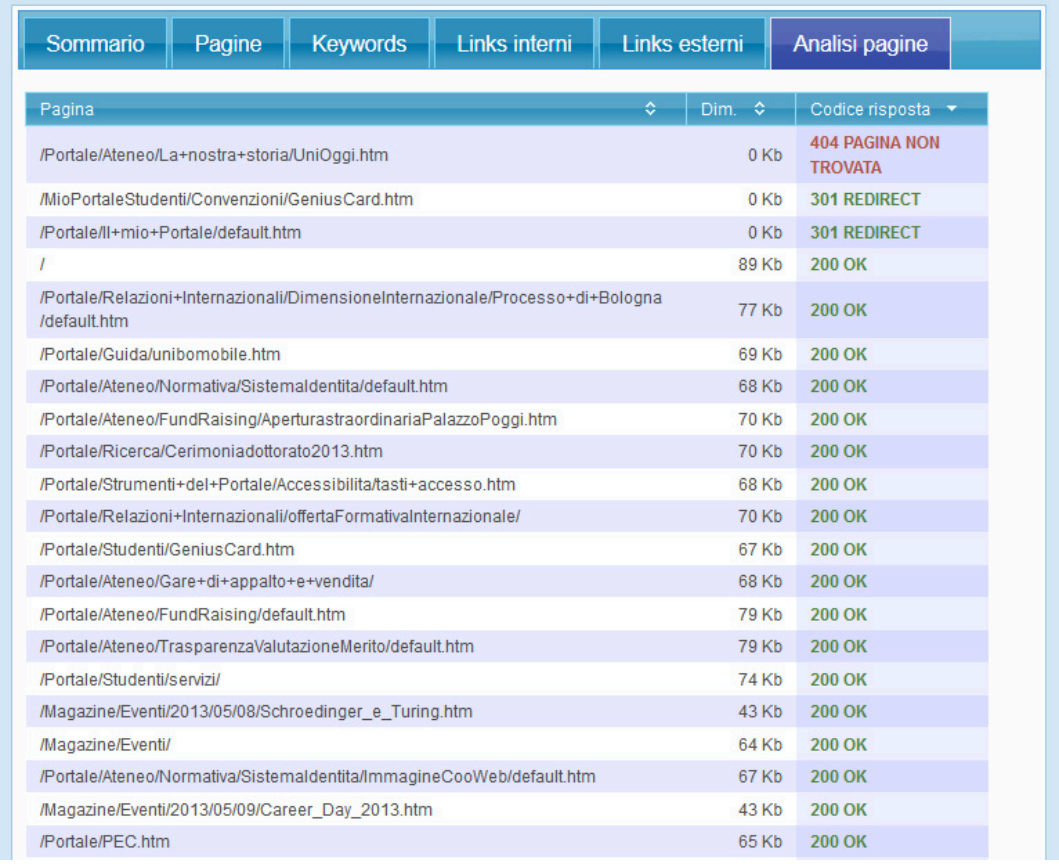


Pagina	Titolo	Numero di pagine che linkano
/Portale/Ateneo/Normativa /Sistemaldentita/default.htm	Immagine coordinata - Università di Bologna	94
/CMSUniboWeb/UniboSearch /Rubrica.aspx	UniboRubrica - Università di Bologna	76
/Portale/Strumenti+del+Portale /Accessibilita/tasti+accesso.htm	Tasti d'accesso - Università di Bologna	73
/Portale/Il+mio+Portale/default.htm	301 Moved Permanently	72
/Portale/PEC.htm	PEC: Posta elettronica certificata - Università di Bologna	71
/Portale/Il+mio+Portale/La+mia+e-mail.htm	La mia e-mail - Università di Bologna	67
/Portale/Ateneo/Strutture /Strutture+accademiche/scuole.htm	Scuole - Università di Bologna	66
/Portale/Ateneo/FundRaising /default.htm	Sostenere l'Alma Mater - Università di Bologna	62
/Portale/Guida/unibomobile.htm	UniboMobile - Università di Bologna	62
/Portale/Relazioni+Internazionali /DimensioneInternazionale /Processo+di+Bologna/default.htm	Bologna Process - Università di Bologna	62
/Portale/Studenti/Segreterie_studenti /default.htm	Segreterie studenti e procedure amministrative - Università di Bologna	62
/Portale/WIFI.htm	Wi-Fi - Università di Bologna	62
/Portale/Ateneo /Divulgazione+scientifica/	Divulgazione scientifica - Università di Bologna	28
/Portale/Ateneo /Gare+di+appalto+e+vendita/	Gare di appalto e vendita - Università di Bologna	28

Figura 5.7: I link interni più linkati. Normalmente i link maggiormente linkati sono anche quelli più importanti.

Figura 5.8: I link esterni a www.unibo.it.

Il sistema è stato in grado di identificare una pagina non trovata (codice di risposta HTTP: 404) linkata dalla pagina: http://www.unibo.it/Portale/Ateneo/Normativa/SistemaIdentita/documenti.htm
L'analisi pagine è in grado di identificare problemi nella struttura del sito.



Pagina	Dim.	Codice risposta
/Portale/Ateneo/La+nostra+storia/UniOggi.htm	0 Kb	404 PAGINA NON TROVATA
/MioPortaleStudenti/Convenzioni/GeniusCard.htm	0 Kb	301 REDIRECT
/Portale/Il+mio+Portale/default.htm	0 Kb	301 REDIRECT
/	89 Kb	200 OK
/Portale/Relazioni+Internazionali/DimensioneInternazionale/Processo+di+Bologna/default.htm	77 Kb	200 OK
/Portale/Guida/unibomobile.htm	69 Kb	200 OK
/Portale/Ateneo/Normativa/Sistemaldentita/default.htm	68 Kb	200 OK
/Portale/Ateneo/FundRaising/AperturastraordinariaPalazzoPoggi.htm	70 Kb	200 OK
/Portale/Ricerca/Cerimoniadottorato2013.htm	70 Kb	200 OK
/Portale/Strumenti+del+Portale/Accessibilita/tasti+accesso.htm	68 Kb	200 OK
/Portale/Relazioni+Internazionali/offertaFormativaInternazionale/	70 Kb	200 OK
/Portale/Studenti/GeniusCard.htm	67 Kb	200 OK
/Portale/Ateneo/Gare+di+appalto+e+vendita/	68 Kb	200 OK
/Portale/Ateneo/FundRaising/default.htm	79 Kb	200 OK
/Portale/Ateneo/TrasparenzaValutazioneMerito/default.htm	79 Kb	200 OK
/Portale/Studenti/servizi/	74 Kb	200 OK
/Magazine/Eventi/2013/05/08/Schroedinger_e_Turing.htm	43 Kb	200 OK
/Magazine/Eventi/	64 Kb	200 OK
/Portale/Ateneo/Normativa/Sistemaldentita/ImmagineCooWeb/default.htm	67 Kb	200 OK
/Magazine/Eventi/2013/05/09/Career_Day_2013.htm	43 Kb	200 OK
/Portale/PEC.htm	65 Kb	200 OK

Figura 5.9: Analisi delle pagine con dimensione e codice di risposta.

5.5 Esempio per apice.unibo.it

Seguono alcune schermate relative ad apice.unibo.it:

Figura 5.10: Le principali keyword per apice.unibo.it su un campione di 100 pagine.

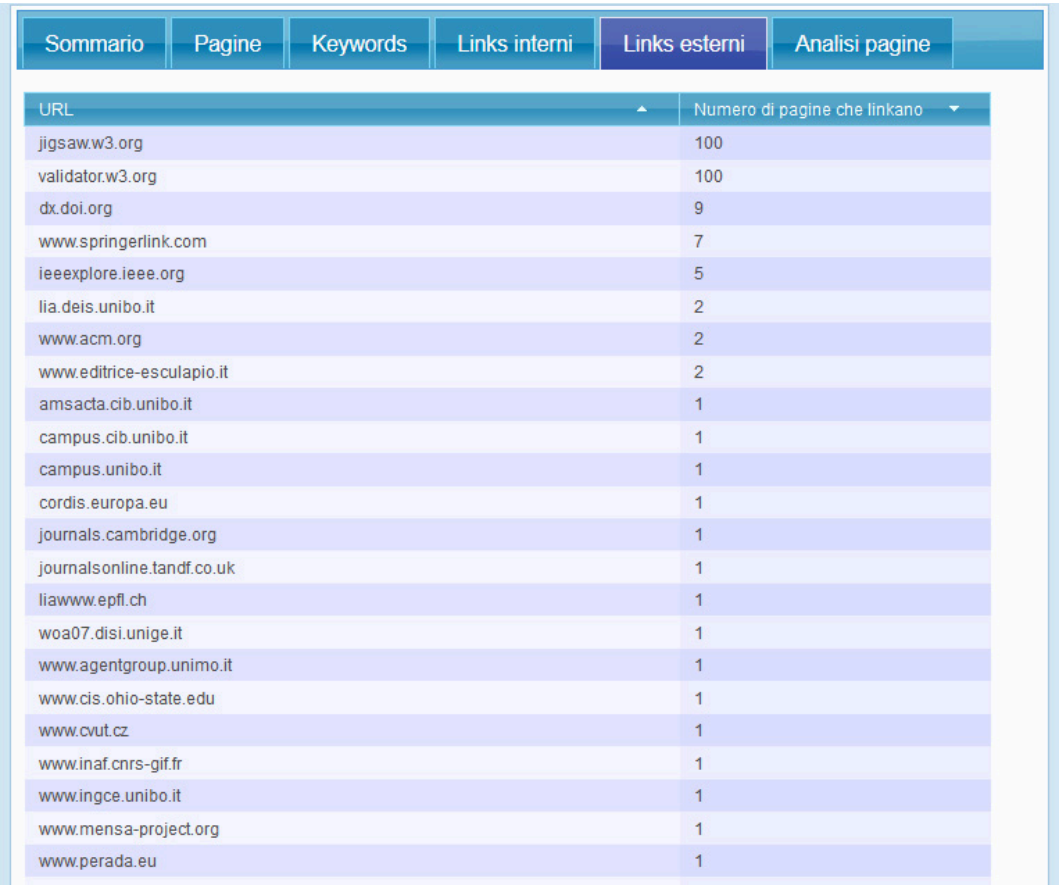
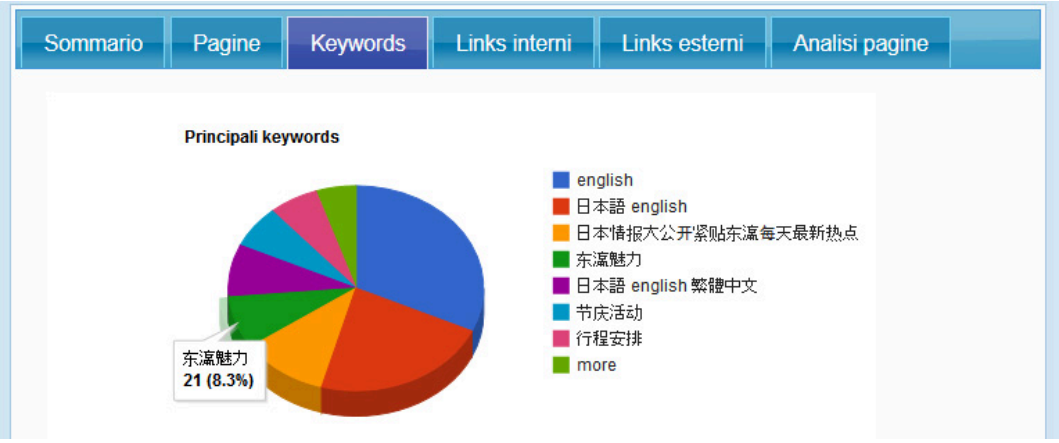


Figura 5.11: I link esterni a apice.unibo.it.

5.6 Esempio per www.welcome2japan.cn

Il sistema supportando UTF8 non ha problemi con siti web con caratteri non latini. Ecco un esempio:



5.7 L'output da linea di comando

Segue un esempio di output di un container Jade durante una fase di attività:

```
[checker]: Avviato CheckerAgent
[checker]: Avvio ManagerAgent
[manager]: Avviato ManagerAgent
[spider0]: Avviato spider0
[manager]: Registrata attivazione container
[spider1]: Avviato spider1
[spider2]: Avviato spider2
[spider0]: Richiesta di url da scansionare
[spider0]: Inizio scansione della pagina: http://www.tim.it/tariffe/abbonamento/
tutto-compreso-300-nuovi-clienti/
[spider1]: Richiesta di url da scansionare
[spider1]: Inizio scansione della pagina: http://www.tim.it/119-self-service/
info-e-supporto/info-tariffe/
[spider0]: Finita scansione di: http://www.tim.it/tariffe/abbonamento/
tutto-compreso-300-nuovi-clienti/
[spider0]: Tempo di risposta: 591 ms
[spider2]: Richiesta di url da scansionare
[spider1]: Finita scansione di: http://www.tim.it/119-self-service/
info-e-supporto/info-tariffe/
[spider1]: Tempo di risposta: 472 ms
[spider2]: Inizio scansione della pagina: http://www.tim.it/estero/
dall-estero-per-chiamare/TIM-in-Viaggio/
[spider2]: Finita scansione di: http://www.tim.it/estero/
dall-estero-per-chiamare/TIM-in-Viaggio/
[spider2]: Tempo di risposta: 335 ms
```

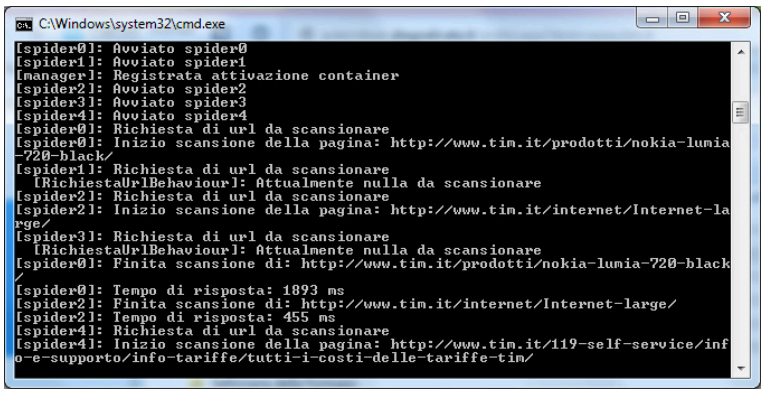


Figura 5.12: Gli agenti su Windows 7

Figura 5.13: Gli agenti su Mac OS X 10.6.8 Snow Leopard (con Java 7)

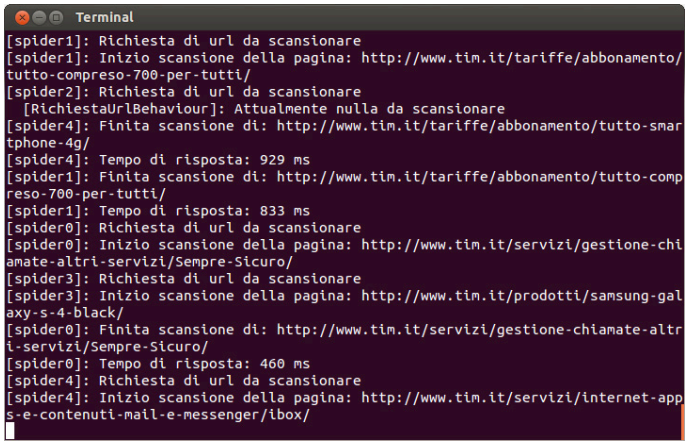


Figura 5.14: Gli agenti su Linux Ubuntu 12.04

Capitolo 6

Ulteriori istruzioni per l'utilizzo del sistema realizzato

6.1 Interrogazione delle web api

Le web api vengono interrogate dall'interfaccia e usate dagli agenti Jade. Non è quindi necessario accedere direttamente ad esse. Sono comunque disponibili presso:

`http://ji-api.diegoalicata.it`

E' possibile fare richieste a scopo di debug seguendo la documentazione relativa fornita al capitolo 4.5 usando un qualsiasi client REST.
Ad esempio risulta molto pratico interrogare le api con l'estensione Rest Client per Firefox [5] oppure con l'estensione Advanced Rest Client per Chrome [6]:

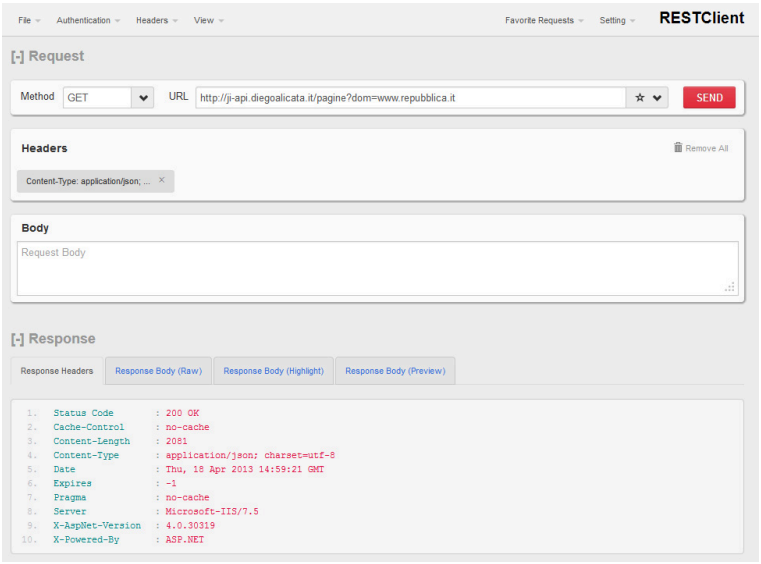


Figura 6.1: Rest Client per Firefox

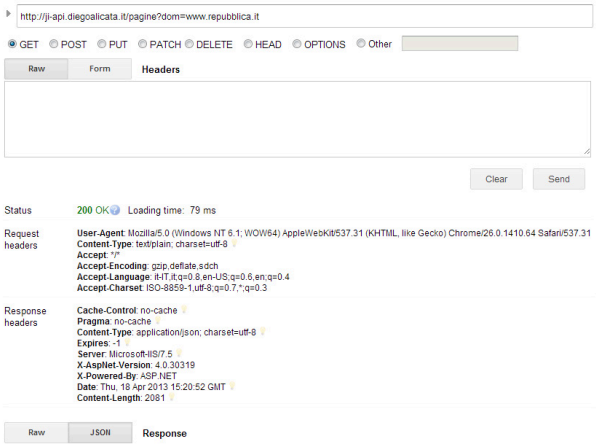


Figura 6.2: Advanced Rest Client per Chrome

6.2 Deployment dai sorgenti

L'interfaccia, il database e le web api, come precedentemente indicato, sono disponibili online. E' possibile comunque, se necessario, fare il deployment dai sorgenti seguendo le istruzioni che seguono:

- I sorgenti relativi all'interfaccia sono in JI-InterfacciaWeb.zip Il deployment dell'applicazione web può avvenire semplicemente copiando tutto il contenuto della cartella JadeIndexer-InterfacciaWeb in un server che

supporti il Net Framework 4.0 o successivi. Nel web.config il parametro 'urlwebapi' punta alla root del servizio di web api. E' possibile modificare tale valore se necessario. Il progetto è editabile tramite Visual Studio 2012 Express per il web [18] (scaricabile gratuitamente).

- Il database è possibile crearlo a partire dal file SQL fornito in JI-DB.zip. Usare Mysql 5.6.4 o successivi con supporto InnoDB.

- I sorgenti relativi alle web api sono in JI-Api.zip Il deployment delle web api può avvenire semplicemente copiando tutto il contenuto della cartella JI-Api in un server che supporti il Net Framework 4.0 o successivi. Modificare il parametro 'mysql' nel file web.config per specificare la stringa di connessione al database precedentemente creato.

- I sorgenti relativi al progetto NetBeans degli agenti Jade sono in JI-Agenti.zip Nei sorgenti jade è possibile modificare l'url delle web api modificando la costante URL_API.

Capitolo 7

Conclusioni e possibili sviluppi futuri

Lo sviluppo del presente progetto ha permesso di:

- Far vedere come sia possibile l'interoperabilità tra tecnologie e strumenti diversi: come un sistema distribuito possa essere costituito da parti e sottosistemi eterogenei in grado di agire come un unicum all'utente finale;
- come sia possibile creare un sistema basato su agenti che possano comunicare a livello locale tramite Jade e a livello globale tramite Web Api;
- come le web api possano agire a supporto degli agenti sia per la comunicazione, che per il controllo del sistema;
- come l'uso della tecnologia Ajax nell'interfaccia web permetta la scalabilita architetturale, facendo in modo che sia il client a realizzare parte dell'elaborazione relativa all'interfaccia che altrimenti sarebbe stata interamente a carico del web server.

Il sistema sviluppato costituisce un prototipo il più completo possibile nei riguardi dei requisiti proposti. Tuttavia molte sono le migliorie ed estensioni che si potrebbero ideare e a tal riguardo la strutturazione del sistema in differenti sottosistemi viene in aiuto facilitando la sua estensibilità:

- Si potrebbe ad esempio pensare di creare un sottosistema alternativo di interfaccia basato su applicativo Android o iOS. Lo sviluppo di tale sottosistema sarebbe relativamente semplice in quanto basterebbe implementare le chiamate alle Web Api dall'applicativo.
- Il campione di features estratte rappresenta un esempio delle capacità del sistema e può essere integrato con differenti altre tipologie. Si potrebbe pensare di reperire ed indicizzare informazioni relative a parametri

di popolarità nei social network, o ancora di inserire features relative alla validazione del contenuto HTML oppure informazioni relative alla velocità di risposta del web server nel fornire le singole pagine web.

- Il sistema potrebbe essere esteso per far uso di alcuni indicatori presi in esame dalle ricerche del Prof. Denti sulle reti complesse per evidenziare le pagine più critiche o più importanti. Ad esempio supponendo che un percorso di visita parta dalla homepage e proceda poi per una serie di pagine, identificare quale sia più opportuno ottimizzare nei tempi di risposta, può costituire un informazione di grande valore per system optimizers che vogliano aumentare il traffico del proprio sito web o i profitti da esso generati.

La richiesta da parte di webmaster, sviluppatori web e ottimizzatori SEO di sistemi di analisi di siti web come quello realizzato è alta. Inoltre è utile sottolineare che un sistema come quello realizzato reperisce e memorizza le informazioni che possono essere utili per la creazione di motori di ricerca specializzati. In tale ottica il progetto acquisisce quindi una duplice valenza e crescono le opportunità per ulteriori sviluppi.

Bibliografia

[1] The robots exclusion standard *http://en.wikipedia.org/wiki/Robots_exclusion_standard*

[2] Web Crawler *http://it.wikipedia.org/wiki/Crawler*

[3] List of User Agent Strings *http://www.useragentstring.com/pages/useragentstring.php*

[4] Architettura REST e Web api *http://en.wikipedia.org/wiki/Representational_state_transfer*

[5] Rest Client per Firefox *http://restclient.net/*

[6] Advanced Rest Client per Chrome *https://chrome.google.com/ webstore/ detail/advanced-rest-client/ hgmloofddffdnphfgcellkdfbfbjeloo*

[7] Jade Documentation *http://jade.tilab.com/doc/index.html*

[8] JSON: The Fat-Free Alternative *http://www.json.org/xml.html*

[9] Introducing JSON *http://www.json.org/*

[10] Gianluca Pengo, Crawling del Web Italiano, Università degli Studi di Padova *http://tesi.cab.unipd.it/26028/1/Crawling_del_Web_Italiano.pdf*

[11] J.B. Keith Humphreys, PhraseRate An HTML Keyphrase Extractor, University of California *http://138.23.89.181/ projects/ publications/ Humphreys-2002-PhraseRate.pdf*

[12] Timothy C. Craven, HTML Tags as Extraction Cues for Web Page Description Construction, University of Western Ontario *http://inform.nu/Articles/Vol6/v6p001-012.pdf*

[13] Tomaso Minelli, Estrazione ed Elaborazione di Contenuti Informativi dal Web, Università degli Studi di Padova *http://tesi.cab.unipd.it/554/1/Minelli.pdf*

[14] jQuery Documentation *http://www.jquery.com/*

[15] Data Tables (table plug-in for jQuery) *http://www.datatables.net/*

[16] Google Chart Tools *https://developers.google.com/chart/*

[17] Accesso ai RESTful Web Services in Java *http://www.slideshare.net/ AndrewRiot/ lezione-11-accesso-ai-restful-web-services-in-java*

[18] Visual Studio 2012 Express Download gratuito

http://www.microsoft.com/ visualstudio/ ita/products/ visual-studio-express-products

[19] OAUTH *http://it.wikipedia.org/wiki/OAuth*

[20] Michael Flarup, Icona World Icon, Liberamente utilizzabile per fini non commerciali *http://www.veryicon.com/icons/internet–web/safari-1/world-6.html*