Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Ambito di interesse

Il lavoro di tesi rientra nel contesto della collaborazione tra l'*Università degli Studi di Brescia* e *Birkbeck, University of London.*

L'argomento di studio principale di questo lavoro è il web semantico. Questo termine, coniato dal suo ideatore Tim Berners Lee, si riferisce alla trasformazione del World Wide Web in un ambiente in cui i documenti pubblicati sono associati ad informazioni e dati (metadati) che ne specificano il contesto semantico. Questo richiede che i documenti pubblicati (pagine HTML, file, immagini, ecc...) siano definiti in un formato adatto all'interrogazione e l'interpretazione e, più in generale, all'elaborazione automatica.

Nel caso specifico, siamo interessati alle possibilità di effettuare ricerche molto più evolute delle attuali, basate sulla presenza nel documento di parole chiave, e alla costruzione di reti di relazioni e connessioni tra documenti secondo logiche più elaborate del semplice collegamento ipertestuale.

1.2 Obbiettivi e modalità di lavoro

L'obbiettivo principale del lavoro svolto è quello di realizzare un prototipo per il sistema **Real Food Trade** (**RFT**) definito nei tesi dei dottori L.

Menichetti[1], L. Nardini[2], M. Ungania[3] presso l'Università degli Studi Roma Tre. Si tratta di una piattaforma per la compravendita di prodotti ittici, il cui compito fondamentale è quello di consentire ai pescatori di organizzare punti di vendita temporanei per il consumatore finale, saltando di fatto gli intermediari.

In questo contesto, sfrutteremo le potenzialità del web semantico per implementare tecniche di ricerca che sfruttino la semantica delle specie ittiche gestite dal sistema.

Il primo passo è stato lo studio della piattaforma definita nelle tesi precedenti. RFT è un sistema complesso, oltre alla ricerca semantica e la gestione dei punti vendita, mette a disposizione diverse funzionalità, tra cui la gestione degli ordini e le ricerche geografiche. Nella realizzazione del prototipo si è resa necessaria una selezione delle funzionalità a cui dare la precedenza di implementazione e quali lasciare agli sviluppi futuri.

1.3 Struttura della Tesi

Capitolo 1 : Introduzione

Capitolo introduttivo del lavoro di tesi.

Capitolo 2: Web Semantico

Vengono descritte le caratteristiche del Web semantico e i principali passi dello sviluppo del Web verso questa direzione. Viene introdotto il concetto di Linked Data e le tecnologie necessarie alla sua implementazione, come RDF e SPARQL.

Capitolo 3 : Real Food Trade

In questo capitolo viene descritto il sistema di riferimento che prende il nome di Real Food Trade (RFT). Si tratta di un sistema sviluppato in tesi precedenti, l'obbiettivo del presente lavoro è quello di realizzare un prototipo

di RFT per l'implementazione delle funzionalità che richiedono la ricerca semantica.

Capitolo 4 : Progettazione del prototipo

Mostreremo le scelte progettuali del prototipo. In particolare vengono illustrate le funzionalità di RFT implementate, il modello dei dati e i casi d'uso.

Capitolo 5 : Modello semantico dei dati

RFT si basa su un modello semantico dei dati che rappresenta la tassonomia di riferimento per le specie ittiche. In questo capitolo mostreremo le caratteristiche di questo modello e da quali dataset prende le informazioni.

Capitolo 6 : Ricerca per nome

La ricerca per nome è quella che fornisce dei risultati in base ad una stringa fornita dall'utente che rappresenta la chiave di ricerca. Nel capitolo vengono descritti i dettagli di questa funzionalità.

Capitolo 7 : Ricerca per similarità

La ricerca per similarità è quella che fornisce le specie simili ad una specie di riferimento indicata dall'utente. Nel capitolo vengono descritti i dettagli di questa funzionalità.

Capitolo 8 : Implementazione

Vengono descritte le caratteristiche tecniche del back-end e del front-end del prototipo implementato. Vengono descritte nel dettaglio alcune delle tecnologie utilizzate.

Capitolo 9 : Sviluppi futuri

Nell'ultimo capitolo, vengono elencate alcune funzionalità di possibile sviluppo futuro.

Capitolo 2

Web semantico

2.1 Dal Web al Web semantico

Il Web

La data di nascita del *Worl Wide Web* viene comunemente indicata nel 6 Agosto 1991, giorno in cui Tim Berners-Lee pubblicò il primo sito web. L'idea del WWW era in realtà nata due anni prima, nel marzo del 1989 quando Tim Berners-Lee, allora giovane ricercatore presso il CERN di Ginevra, insieme al collega Robert Calliau propose al suo suepervisore lo sviluppo di un sistema per la condivisione in formato elettronico di documentazione scientifica.
I principali standard attravero i quali il Web è implementato sono:

- HTML: il linguaggio di markup con cui sono scritte e descritte le pagine web;
- HTTP: il protocollo di rete (livello di applicazione del modello ISO/OSI) su cui è basato il Web;
- URL: lo schema di identificazione, e quindi di rintracciabilità, dei contenuti e dei servizi del Web.

Il Web è un'enorme rete di documenti collegati tra loro da link ipertestuali. I dati pubblicati possono essere utilizzati sia da utenti umani, che da agenti software.

Limiti del Web

Il Web descritto fino a questo punto è limitato all'interpretazione delle informazioni da parte dell'utente umano. Il linguaggio HTML, che ha contribuito alla diffusione del fenomeno del Web, ha un forte limite: esso si occupa solo ed esclusivamente della formattazione dei documenti, tralasciando del tutto il significato dei contenuti. Questo pone delle difficoltà al reperimento e fruizione delle informazioni da parte delle macchine.

Una prima soluzione è stata proposta dal creatore stesso del Web, con la definizione dello standard XML (eXtensible Markup Language). Si tratta di un metalinguaggio che consente la creazione di unovi linguaggi di markup. La caratteristica innovatica è la possibilità di aggiungere informazioni semantiche ai contenuti attraverso la definizione di opportuni tag. A questo punto le macchine sono in grado di eseguire ricerche migliori grazie alla possibilità di accedere ad informazioni nascoste nell'oscurità contestuale.

Le specifiche XML hanno però una lacuna molto importante: non definiscono alcun meccanismo univoco e condiviso per specificare relazioni tra informazioni espresse sul web, condizione necessaria per l'elaborzione automatica.

Anche in questo caso la soluzione al problema è venuta dal W3C di Berners-Lee, attraverso la formalizzazione del web semantico.

Il Web Semantico

Con il termine web semantico, termine coniato dal suo ideatore, Tim Berners-Lee, si intende la trasformazione del World Wide Web in un ambiente dove i documenti pubblicati (pagine HTML, file, immagini, e così via) sono associati ad informazioni e dati (metadati) che ne specificano il contesto seman-

tico in un formato adatto all'interrogazione e l'interpretazione (es. tramite motori di ricerca) e, più in generale, all'elaborazione automatica.

L'enfasi viene spostata dalla sola sintassi, interpretabile esclusivamente dall'essere umano, alla semantica, ovvero al contenuto dei documenti, per consentire l'interpretazione di questi anche ai computer. Il tentativo di rendere comprensibile i contenuti alle applicazioni ha lo scopo di permettere ricerche molto più evolute delle precedenti, che erano basate solo sulla presenza di parole chiave, e altre operazioni specialistiche come la costruzione di reti di relazioni e connessioni tra documenti secondo logiche più elaborate del semplice collegamento ipertestuale.

Alla creazione di una pagina web, le informazioni in esse contenuta andranno dichiarate sencondo precise regole semantiche (da qui il termine Web Semantico). Nasce dunque l'esigenza di definire un nuvo modo di strutturare i dati. È stato sviluppato a questo scopo un insieme di principi e tecnologie, conosciuti come Linked Data.

2.2 Linked Data

Il temrine *Linked Data* indica un insieme di *best practices* per la pubblicazione di dati strutturati che possono essere tra loro collegati e in questo modo diventare frubili per interrogazioni semantiche. Questo termine fu coniato da Tim Berners-Lee nel *design note* del 2006[4] riguardante il progetto del web semantico. I Linked Data vennero poi presentati alla conferenza TED[1] del 2009.

Componenti

I componenti che vengono utilizzati per implementare i Linked Data sono:

- URI (Uniform Resource Identifier): stringa che identifica univocamente una risorsa;

[1]TED (Technology Entertainment Design) è una conferenza che si tiene ogni anno ed abbraccia una vasta gamma di arogmenti che comprendono scienza, arte, politica e altro.

- HTTP: il protocollo di rete usato nel Web;
- RDF: strumento per la codifica, scambio e riutilizzo di metadati strutturati attraverso grafi;
- Formati serializzabili: formati per la rappresentazioni fisica di un grafo RDF.

Principi

Tim Berners-Lee descrisse quattro principi sui Linked Data:

1. Usare URI per identificare oggetti.
2. Usare HHTP URI in modo che questi oggetti possano essere referernziati e cercati sia da persone che da user agent.
3. Fornire informazioni utili sull'oggetto quando la sua URI è deferenziata, usando formati standard come RDF.
4. Includere link ad altre URI relative ai dati esposti per migliorare la ricerca di altre informazioni relative nel Web.

Il primo principio dei Linked Data sostiene l'utilizzo di URI per identificare, non solo documenti Web e contenuti digitali, ma anche oggetti appartenenti al mondo reale o concetti astratti. Questo può includere sia cose tangibili, come prodotti o persone, o cose più astratte come ad esempio relazioni di conoscenza o parentela tra persone.

Il protocollo HTTP è il meccanismo universale di accesso al Web. Nel Web classico, gli URI HTTP sono usati per combinare identificatori globalmente univoci con un semplice e ben comprensibile meccanismo di recupero. Così, il secondo principio, sostiene l'uso degli URI HTTP per identificare oggetti e concetti astratti, permettendo a questi URI di essere dereferenziati, attraverso il protocollo HTTP, in una descrizione dell'oggetto o concetto identificato.

Al fine di permettere ad un'ampia gamma di applicazioni di processare contenuti Web, è importante accordarsi sui formati standard per la pubblicazione dei contenuti. Il terzo principio perciò sancisce l'uso di un singolo modello dei dati per pubblicare dati strutturati sul Web: RDF.

Il quarto principio afferma che l'uso dei collegamenti ipertestuali serve non solo per collegare documenti, ma qualsiasi tipo di oggetto. Per esempio un collegamento ipertestuale potrebbe essere posto tra due persone, oppure tra una persona e una località. In contrasto con quanto accade nel Web classico, i collegamenti ipertestuali nei Linked Data sono classificati da un tipo che descrive la relazione esistente tra gli oggetti. Questi collegamenti ipertestuali vengono chiamati RDF link, per poterli distinguere con i normali link del Web. Nel contesto dei Linked Data, se un link RDF connette URI appartenenti a differenti namespace, significa che sta connettendo risorse appartenenti a diversi insiemi di dati. Così come i collegamenti ipertestuali nel Web collegano documenti in un singolo spazio globale delle informazioni, i Linked Data usano gli RDF link per connettere i dati in un singolo spazio dei dati globale. Questi link permettono alle applicazioni di navigare all'interno dello spazio dei dati. Per esempio un'applicazione Linked Data che ha identificato un URI e recuperato i dati RDF che descrivono una persona, potrebbe seguire dei link che conducono da quei dati ad altri dati presenti su altri server, i quali potrebbero descrivere il posto in cui questa persona vive o la compagnia per cui lavora. Grazie al fatto che tutta l'infrastruttura si basa su standard e un data model comune, diventa possibile implementare generiche applicazioni che operano su tutto lo spazio dei dati. In poche parole i principi dei Linked Data posano le fondamenta per estendere il Web con uno spazio globale dei dati basato sugli stessi principi architetturali del Web classico.

Linked Data Cloud

Uno degli obbiettivi del W3C è quello di estendere il Web con informazioni condivise pubblicando diversi dataset in formato Linked Data e collegando le

risorse appartenteni a dataset diversi. Quello che si ottiene è una rete enorme ed in continua espansione di dati, costituita da tutti i dataset pubblicati in formato Linked Data che prende il nome di Linked Data Cloud.

In Figura 2.2 e in 2.1 viene evidenziata questa espansione, mostrando la Linked Data Cloud rispettivamente nel luglio del 2009 e nell'agosto del 2014.

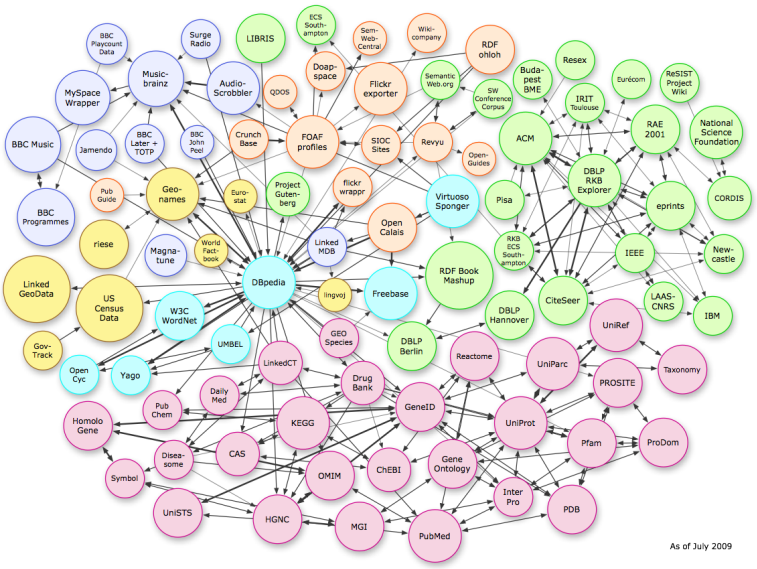

Figura 2.1: Linked Data Cloud, Luglio 2009

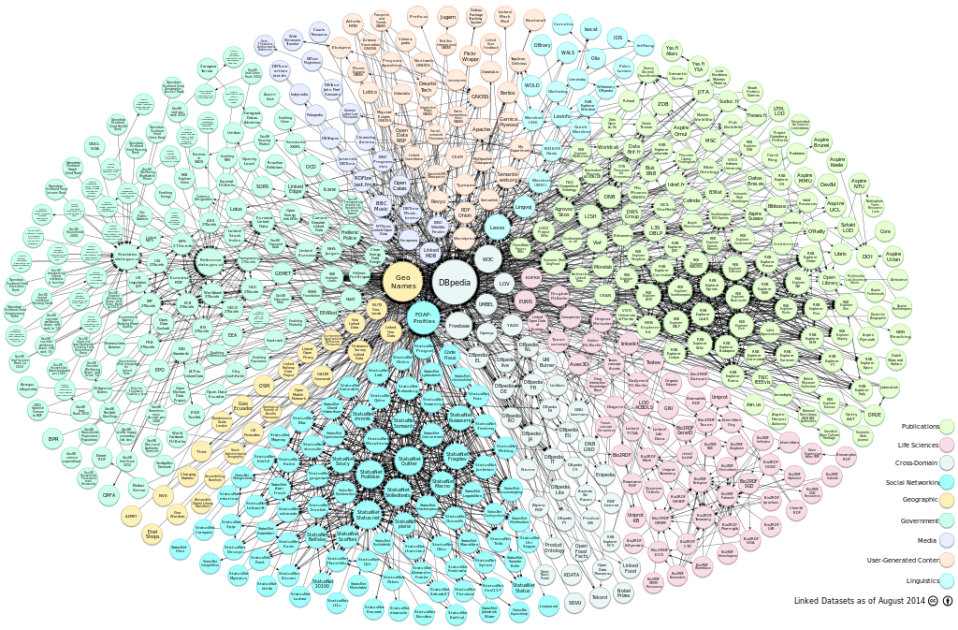

Figura 2.2: Linked Data Cloud, Agosto 2014

Il numero di dataset è cresciuto a dismisura e continuano ad aumentare anche le interconessioni tra gli stessi, realizzando una rete di informazioni correlate tra loro sempre più vasta.

2.3 Resoruce Description Framework

Il Resource Description Framework (RDF) è un linguaggio, proposto dal W3C, per rappresentare informazioni sulle risorse nel Web. Viene descritto nei dettagli nel documento delle specifiche RDF Primer[5], di cui questa sezione vuole esserne un riassunto.

L'intento principale era quello di rappresentare i metadati di risorse Web, come per esempio il titolo, l'autore o la data dell'ultima modifica di una pagina Web. Generalizzando il concetto di "risorsa Web", RDF può essere utilizzato per rappresentare informazioni riguardanti cose che possono essere *identificate* nel Web, anche quando queste non sono direttamente recuperabili nella rete.

RDF nasce per essere utilizzato in quei contesti in cui le informazioni devono essere processati da applicativi, invece che essere semplicemente mostrati agli utenti umani. Viene messo a disposizione un framework comune per esprimere queste informazioni, in modo tale da poterle condividere tra diverse applicazioni senza perdita di significato. In particolare questo significa che le informazioni possono essere messe a disposizione di programmi diversi da quelli per i quali erano state originariamente create.

L'idea base di RDF è identificare gli oggetti tramite URI e descrivere le risorse in termini di proprietà.

Questo consente di rappresentetare statement sulle risorse come un grafo di nodi ed archi etichettati.

RDF è costituito da due componenti:

- RDF Model: espone la struttura del modello RDF e descrive una possibile sitassi.
- RDF Schema: espone la sitassi per definire schemi e vocabolari per i metadati.

RDF Model

RDF si basa sull'idea che gli oggetti descritti hanno delle proprietà che hanno dei valori e le risorse vengono descritte tramite statements che specifiano quali proprietà e quali valori posseggono. Uno statement RDF è una tripla costituita da:

- soggetto: è lo URI che identifica la risorsa cui lo statement fa riferimento;
- predicato: è lo URI che identifica la proprietà, indica quale relazione esiste tra il soggetto e l'oggetto.
- oggetto: identifica il valore assunto dalla proprietà.

Gli URI appartengono a set di URI che prendono il nome di vocabolari. Generalmente gli URI in questi vocabolari sono organizzati in modo tale

da poter essere rappresentati come un insieme di QName usando un prefisso comune. Questo significa che gli URI appartenenti ad uno stesso vocabolario condividono lo stesso namespace.

A seconda del tipo di oggetto, si distinguono due tipi di triple RDF:

- triple letterali : l'oggetto è un RDF literal, ovvero un valore costante come una stringa o un intero. Vengono utilizzate per descrivere proprietà come l'età o il nome di una persona. I literal possono essere plain o typed. Un plain literal è costituito da una stringa e da un tag per che identifica la lingua. Un typed literal è costituito da una stringa e da un datatype URI che ne identifica il datatype. Per rappresentare interi,float o altri tipi numerici possiamo usare i datatpye definiti da XML schema.

- link RDF: l'oggetto è uno URI. Vengono utilizzate per descrivere la relazione tra due risorse. Un link RDF può essere interno od esterno. Un link interno mette in relazione due risorse appartenenti alla stessa sorgente Linked Data, mentre un link esterno due risorse appartenenti a differenti namespace.

La seguente frase in italiano:
`http://www.esempio.org` ha un **autore** che si chiama **Mario Rossi**
potrebbe essere rappresentata dalla tripla RDF:

- soggetto: `http://www.esempio.org`
- predicato: `http://purl.org/dc/elements/1.1/creator`
- oggetto: `http://www.esempio.org/staffid/211090`

Come si può notare, gli URI non vengono utilizzati solo per rappresentare il soggetto, ma anche il predicato e l'oggetto. La tripla in questione può essere rappresenta dal grafo in Figura 2.3. Un insieme di triple può essere visto quindi come un grafo, chiamato grafo RDF in cui gli oggetti e i soggetti sono i nodi, mentre i predicati sono gli archi.

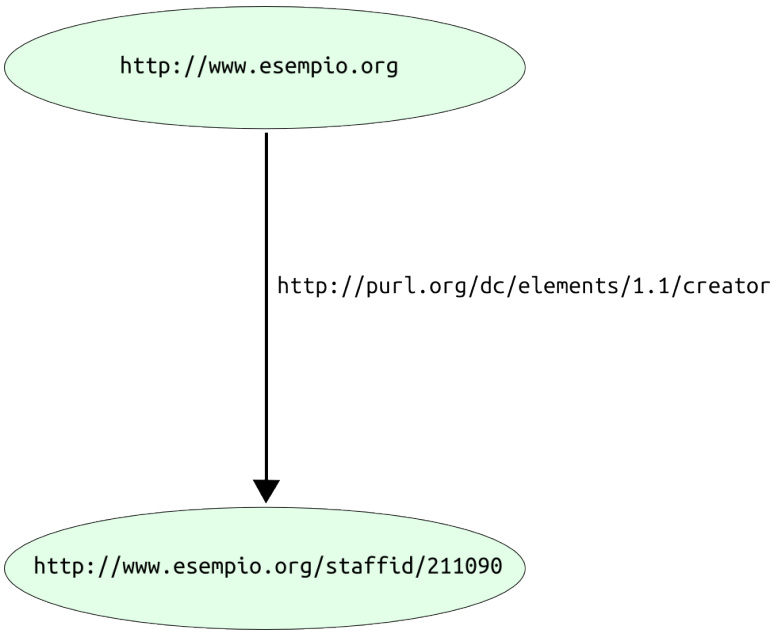

Figura 2.3: Un esempio di tripla RDF

RDF Schema

Le risorse descritte possono essere classificate in categorie, questa idea è simile al concetto di Classi e Oggetti nei linguaggi di programmazione orientati agli oggetti. Questo concetto viene supportato da RDF tramite una proprietà predefinita, `rdf:type`. Quando una risorsa RDF è descritta con una proprietà `rdf:type`, il valore di quella proprietà è una risorsa che rappresenta la classe del soggetto descritto. Il soggetto invece è da considerarsi una instanza di quella classe.

RDF consente di esprimere semplici triple sulle risorse, usando proprietà con un nome e valori. Tuttavia vi è la necisstà da parte degli utilizzatori di definire i vocabolari che intendono usare nelle loro triple, in particolare per indicare che stanno descrivendo specifiche classi di risorse trami tel'utilizzo di specifiche proprietà. RDF non permette di definire classi e proprietà, queste invece andranno descritte come un vocabolario RDF, usando le estensioni fornite dal *RDF Vocabulary Description Language 1.0: RDF Schema*, al quale ci si riferisce semplicemente con *RDF Schema*.

RDF Schema consente di descrivere classi e proprietà e di indicare quali di queste saranno usate insieme. In altre parole, RDF Schema fornisce un *type system* per RDF. Questo type system è simile, sotto certi punti di vista, con quello dei linguaggi orientati agli oggetti, come Java. Per esempio, RDF Schema consente alle risorse di essere instanze di più classi. Inoltre consente una organizzazione gerarchica delle classi. Le classi RDF, però sono diverse da quelle dei linguaggi di programmazione. Una classe RDF non forza le informazioni in una particolare struttura, ma si limita invece ad aggiungere informazioni supplementari per le risorse RDF che descrive.

Serializzazione

Con il termine serializzazione ci si riferisce alla rappresentazione fisica di un grafo RDF. Esistono diversi tipi di serializzazione, tra cui:

- Turtle: un formato compatto e *human-friendly*;
- Notation3 (N3): tecnica non standard di serializzazione, molto simile a Turtle, ma con caratterstiche aggiuntive, come la possibilità di definire delle regole di inferenza;
- RDF/XML: è il primo formato standard per serializzare RDF, usa un sintassi *XML-based*.

2.4 OWL

OWL (**O**ntology **W**eb **L**anguage) è un linguaggio di markup per rappresentare esplicitamente significato e semantica di termini con vocabolari e relazioni tra gli stessi. La rappresentazione dei termini e dele relative relazioni è chiamata *ontologia*.

OWL consente di effettuare deduzioni sulle ontologie che permette di descrivere, integrandole con i contenuti delle pagine Web.

Sottolinguaggi

Esistono tre sottolinguaggi OWL, in ordine di crescente espressività:

- OWL Lite: fornisce l'espressività base per una classificazione gerarchica e vincoli semplici;
- OWL DL: contiente tutti i costrutti OWL, ma possono essere usati solo entro certe restrizioni. Sin rivolge agli utenti che desiderano la massima espressività mantendendo la completezza computazionale e la decidibilità;
- OWL Full: fornisce la massima espressività e la libertà sintattica sintattica di RDF, ma senza garanzie computazionali. Non esiste un software in grado di supportare il ragionamento completo su tutte le caratteristiche di OWL Full.

2.5 SPARQL

SPARQL (acronimo ricorsivo per **S**PARQL **P**rotocol **a**nd **R**DF **Q**uery **L**anguage) è un linguaggio di interrogazione per basi di dati RDF. Per estrarre informazioni dalle basi di dati distribuite sul Web, è necessario interroagre uno SPARQL endpoint. Uno SPARQL endpoint è un servizio conforme al protocollo SPARQL che consente l'interrogazione di una base di dati RDF tramite la sottomissione di query in questo linguaggio.

Triple pattern

SPARQL consente la definizione di query basate su triple pattern, congiunzioni, disgiunzioni e pattern opzionali. Il costrutto triple pattern prevede la definizione di una tripla costituita da soggetto, predicato e oggetto, che rispecchia la struttura delle asserzioni RDF. Uno più elementi di questa tripla possono essere indicati come variabili, diventando le incognite dell'interrogazione.
Questo è un esempio di triple pattern:

```
soggetto  predicato    oggetto
?fish     dct:subject  dbc:Fish_of_Asia
```

In questo esempio, l'unica variabile è il soggetto, mentre il predicato e l'oggetto sono costanti. Le query vengono poi risolte per pattern matching, questo significa che tra le triple RDF del data set al quale l'endpoint fa riferimento, quelle che trovano riscontro nel triple pattern specificato, assoceranno i propri valori alle incognite corrispondenti.

Sintassi

La sintassi utilizzata è Turtle, un'estensione di N-triples. Una query SPARQL è la seguente:

```
PREFIX ...
SELECT ...
FROM ...
WHERE{
...
}
```

`PREFIX` consente di dichiarare i prefissi per i namespace, in modo da alleggerire la scrittura e lettura delle query.
`SELECT` precede l'elenco delle variabili di ricerca da includere nei risultati.
`FROM` specifica il dataset da interrogare.
La clausola `WHERE` consente di definire il criterio di selezione, specificando una o piu triple pattern tra parentesi graffe.
In coda alla query, si possono aggiungere i cosiddetti *query modifier*, come per esempio il costrutto `ORDER BY che` consente di ordinare i risultati secondo il criterio desiderato.

Risultati

I risultati sono forniti in diversi formati:

- XML: utilizzando un particolare vocabolario XML;

- JSON: versione JSON del vocabolario XML, utile per applicazioni Web;
- RDF: utilizzandon una delle serializzazioni;
- HTML: visualizzazione human-friendly di uno dei formati precedenti (tipicamente XML).

Esempio

Il seguente esempio è un'interrogazione SPARQL che modella la domanda: "Quali sono tutte le capitali in Africa?".

```
PREFIX abc:  <http://example.com/exampleOntology#>
SELECT ?capital ?country
WHERE {
?x abc:cityname ?capital ;
abc:isCapitalOf ?y .
?y abc:countryname ?country ;
abc:isInContinent abc:Africa .
}
```

Capitolo 3

Il caso di studio : RFT

Il sistema del caso di studio prende il nome di **Real Food Trade** (**RFT**). RFT consente ai pescatori di vendere i propri prodotti direttamente al consumatore finale, tramite la pubblicazione di mercati temporanei. Questi mercati temporanei rappresentano dei punti di incontro, identificati da un orario ed un luogo nello spazio, tra produttore e consumatore che rendono possibile la vendita diretta. Questo sistema è stato progettato ed implementato nelle tesi dei dottori L. Menichetti, L. Nardini, M. Ungania (Università degli Studi Roma Tre).

L'obbiettivo della presente tesi è quello di sfruttare i Linked Data per creare tecniche di ricerca semantica nel contesto di Real Food Trade. È stato dunque necessario studiare il sistema preesistente prima di creare il nuovo prototipo che ne rappresenta una porzione limitata al settore di interesse. In questo capitolo sono descritte le funzionalità e i requisiti del sistema completo ed in seguito le funzionalità e le scelte progettuali adottate per la realizzazione del prototipo. Si è cercato di mantenere una continuità con la nomenclautra adottata nei lavori precedenti.

3.1 Utenti

Segue una lista degli attori che svolgono funzioni nel sistema.

- **User** È un qualsiasi utilizzatore del sistema;
- **Customer** È una specializzazione di *User*. Rappresenta un consumatore, ovvero un utente che utlizza l'applicazione per acquistare prodotti messi in vendita dai pescatori;
- **Seller** Rappresenta un pescatore, ovvero un utente il cui compito principale è quello di vendere i propri prodotti. Un venditore ha anche la facoltà di comprare prodotti (non quelli messi in vendita da se stesso), per questo motivo è una specializzaione di *Customer*;
- **Admin** È una specializzazione di *User*. Rappresenta una figura amministrativa del sistema che ha la funzione di creare e gestire i *Flash Stand*.

3.2 Definizioni

Segue una lista di definizioni che descrivono gli oggetti modellati dal sistema.

- **Flash Stand** È il mercato temporaneo, caratterizzato da un orario di apertura e dalle coordinate geografiche. Appartiene ad un *Seller*;
- **Stand Product** È un prodotto messo in vendita da un *Seller* in un suo *Flash Stand*. Fa riferimento ad una specie ittica precisa;
- **Flash Market** Rappresenta un luogo scelto da un *Admin* in cui i *Seller* possono creare i propri *Flash Stand*;
- **Order** È l'insieme delle prenotazioni di *Flash Product* di uno stesso *Flash Stand* effettuate da un *Customer*;
- **Feedback** È una votazione che *Seller* e *Customer* si danno reciprocamente, dopo la conclusione, ovvero il ritiro e pagamento, di un *Order*.

3.3 Funzionalità

In questa sezione illustriamo le principali funzionalità del sistema completo.

Pubblicazione di un Flash Stand

Un *Seller* può pubblicare un *Flash Stand*, ovvero un proprio mercato temporaneo indicandone l'orario di validità e la posizione geografica. Un *Flash Stand* può essere associato dal *Seller* proprietario ad un *Flash Market*.

Il pescatore potrà poi definire gli *Stand Product* che metterà in vendita nel prorpio *Flash Stand*, indicandone quantità e prezzo. Vengono gestite due tipi di quantità: stimata e reale. In questo modo, il *Seller* può pubblicare un prodotto prima di iniziare una battuta di pesca, indicandone la quantità stimata. Durante la pesca potrà ricevere prenotazioni dagli utenti e regolarsi di conseguenza. Al termine della pesca, aggiornerà il valore iniziale con la quantità reale.

Il pescatore può associare una posizione geografica relativa al luogo di provenienza di un certo pescato, garantendo un maggiore livello di trasparenza al consumatore.

Fondamentale per la futura ricerca semantica è l'appartenenza di un prodotto in vendita ad una precisa specie ittica. Per questo motivo, all'atto dell'inserimento di un nuovo *Stand Product*, il venditore deve selezionare una specie dalla tassonomia di riferimento usata da RFT.

Ricerca di un prodotto

L'utente *Customer* utilizza il sistema con lo scopo di effettuare degli ordini che verrano poi ritirati personalmente presso i *Flash Stand*. Per questa ragione, deve essere in grado di navigare i prodotti pubblicati.

Vi sarà una funzione di ricerca, tramite la quale l'utente sarà in grado di selezionare una specie della tassonomia. Il sistema mostrerà gli *Stand Product* disponibili per quella specie e, sfruttando le tecniche di ricerca semantica, per le specie simili.

Gestione degli ordini

Una volta che un cliente ha trovato il prodotto desiderato, può effettuare un'ordine, ovvero prenotare una data quantità del prodotto in questione, per poterlo poi ritirare presso il rispettivo *Flash Stand*. Il sistema deve essere in grado di gestire questi ordini.

Per prima cosa le prenotazioni di un cliente relativi a più prodotti dello stesso *Flash Stand* devono essere raggruppati in un unico ordine per ottimizzare l'interazione *Customer*/*Seller*.

Il sistema deve essere in grado di gestire le quantità in tempo reale, decurtando dalla disponibilità di un prodotto le quantità prenotate. Nel caso in cui i clienti effettuino ordini per un prodotto la cui quantità è stimata, potrebbe accadere che la quantità reale, inserita a posteriori dal venditore, non copra le richieste già ricevute. Per risolvere questa situazione, il sistema dovrà essere dotato di un meccanismo di scheduling degli ordini che scelga, con un certo criterio, quali ordini rifiutare. Il cliente quindi è consapevole del rischio di un eventuale impossibilità dell'evasione di un ordine nel momento in cui prenota un prodotto con quantità prevista e non reale. Si rende quindi necessario anche un meccanismo di notifica per la conferma, o il rifiuto, degli ordini una volta inserita la quantità pescata dal venditore.

Per ora, non è prevista la gestione dei pagamenti, RFT consente di prenotare prodotti che poi verranno ritirati e pagati di persona presso un *Flash Stand*. È compito quindi del venditore indicare quali ordini sono stati ritirati e che quindi si possono considerare conclusi con successo e quali invece sono non lo sono.

Feedback

RFT consente ai *Customer* e *Seller* di lasciarsi un feedback reciproco relativo ad ogni ordine. Questo feedback è espresso da un voto su una scala prestabilita ed un eventuale messaggio testuale.

Una volta chiuso un ordine, il sistema chiede ai due attori coinvolti di lasciare il feedback. Il cliente potrà giudicare la qualità dei prodotti, a

favore di tutti gli utilizzatori. Il venditore potrà invece indicare se il cliente ha effettivamente ritirato l'ordine. Questi voti costruiscono un ranking degli utilizzatori, molto utile ad entrambe le categorie di utenti. I clienti potranno scegliere tra i pescatori giudicati migliori dalla comunità di RFT, mentre i pescatori potranno rifiutare gli ordini effettuati da utenti che spesso non si sono presentati al ritiro e che quindi avranno un ranking basso.

Ricerche geografiche

RFT nasce per essere utilizzato su dispositivi mobili, come smartphone o tablet. La mobilità è una caratteristica fondamentale di questo sistema, grazie ad essa i pescatori potranno modificare i dati del proprio pescato direttamente in mare, mostrando ai clienti le quantità in tempo reale. Essi potranno inoltre tenere traccia delle prenotazioni ricevute, adeguando di conseguenza le quantità da pescare. Utilizzando questi dispositivi si può sfruttare anche la localizzazione geografica per migliorare alcune funzionalità.

Ogni *Flash Stand* è caratterizzato da una posizione geografica espressa in coordinate di latitudine e longitudine. Quando un utente effettua la ricerca di un prodotto, passa per la selezione di una specie della tassonomia di riferimento. Il sistema poi, mostra i *Flash Product* disponibili per quella specie ittica. A questo punto il sistema mostra prima i *Stand Product* venduti nei *Flash Stand* più vicini, sulla base della localizzazione del dispositivo del cliente.

I dati geografici possono essere utilizzati per migliorare l'esperienza utente in vari modi. Nel caso di un *Customer* che abbia prenotato più ordini in diversi *Flash Stand*, RFT può calcolare il percorso migliore per il ritiro fisico della merce. Ovviamente, in questo calcolo, bisogna tenere conto anche degli orari di apertura e chiusura di ciascun *Flash Stand*.

3.4 Lo scenario Cileno

Come caso di studio per RFT è stato analizzato il mercato ittico in una zona del Cile, nei pressi di Santiago. Il potere contrattuale dei pescatori, al momento della vendita del pescato, è molto limitato e ciò riduce molto il loro margine di guadagno. Questa situazione è dovuta al fatto che l'unico acquirente dei prodotti ittici è il grossista ed egli, grazie a questa posizione privilegiata, ha un forte peso nella decisione del prezzo.

RFT fornisce una valida alternativa a questa situazione, offrendo un canale di vendita costituito dal consumatore finale. Vendendo direttamente al cliente, RFT elimina la figura del grossista, o almeno ne limita l'importanza sottraendogli parte del mercato. Vendendo i propri prodotti al dettaglio, i pescatori possono alzare il prezzo garantendolo comunque inferiore a quello proposto al dettaglio dopo il passaggio dal grossista. È chiaro come ci guadagni sia il pescatore che il cliente.

Va chiarito che l'intenzione di RFT non è quella di eliminare definitivamente ogni forma di intermediazione. A questo proposito, infatti va evidenziato come l'utente *Seller* sia un estensione dell'utente *Customer*. Questo significa che un venditore può acquistare, e quindi rivendere successivamente, prodotti da altri venditori della comunità RFT. Entrando in un mercato come quello del caso Cileno, RFT avrà potenzialmente la capacità di eliminare l'intermediazione senza valore aggiunto, ovvero quella rappresentata da figure che si garantiscono un costo di acquisto alla fonte favorevole, grazie a posizioni di privilegio. Questi intermediari, rivendendo ai dettaglianti, detengono la maggior parte dei guadagni del mercato in questione, senza aver aggiunto valore alla merce, come può fare invece un intermediario che rivende in una zona diversa, assumendosi il costo del trasporto. RFT propone nuovi canali di vendita più convenienti ai produttori, limitando quindi la posizione di privilegio detenuta dai grossisti.

3.5 Prototipo

In questa sezione mostriamo le funzionalità minime implementate nella progettazione del prototipo e i casi d'uso individuati. Estendendo poi il prototipo si potrà giungere ad una versione completa del sistema, così come definita nel presente capitolo.

3.5.1 Funzionalità implementate

Il sistema si appoggia su una tassonomia delle specie ittiche, la cui architettura viene descritta nel prossimo capitolo. Questa tassonomia è l'oggetto di riferimento per la ricerca semantica.

La funzione di ricerca si rende necessaria durante diverse operazioni. Quando un venditore inserisce un prodotto, deve specificare a quale specie appartiene, in questo modo i clienti possono raggiungere un prodotto indicando una specie della tassonomia. Gli utenti indicano una specie, passando dalla ricerca semantica. Emerge come sia necessario implementare le funzionalità di inserimento e gestione dei *Flash Stand* e *Stand Product* da parte dei *Seller* e di ricerca di questi da parte dei *Customer*.

Ovviamente nasce l'esigenza di differenziare l'esperienza utente sulla base del ruolo. Vengono quindi implementate funzioni base di registrazione ed autenticazione. Gli utenti sono quelli definiti nel capitolo precedente, ai fini del prototipo l'utente *Admin* non compare come attore di alcuna attività.

3.5.2 Casi d'uso

Ai fini del lavoro di tesi risulta superflua l'implementazione di tutte le funzionalità di RFT. Di seguito sono riportati i casi d'uso individuati necessari all'implementazionde delle funzionalità minime necessarie.

Registrazione Un nuovo utente si registra nel sistema come *Seller* o *Customer*.

Pre-condizioni	L'utente non è loggato.
Post-condizioni	Il nuovo utente viene inserito nel sistema.
Scenario principale	1. L'utente indica che vuole registrarsi. 2. Il sistema chiede l'inserimento di: • username; • password; • email; 3. Il sistema chiede di indicare il ruolo (*Seller* o *Customer*). 4. L'utente conferma la registrazione.

Login Un utente si autentica nel sistema.

Pre-condizioni	L'utente è registrato.
Post-condizioni	L'utente è autenticato.
Scenario principale	1. L'utente indica che vuole effettuare il login. 2. Il sistema chiede l'inserimento di username e password. 3. L'utente conferma il login.

Creazione di un nuovo Flash Stand Un utente *Seller* inserisce un nuovo *Flash Stand* nel sistema.

Pre-condizioni	L'utente è loggato come *Seller*.
Post-condizioni	Un *Flash Stand* è stato correttamente inserito nel sistema.
Scenario principale	1. Il *Seller* indica che vuole creare un nuovo *Flash Stand*. 2. Il sistema chiede l'inserimento di: • latitudine e longitudine; • data e ora di inizio e di fine. 3. Il *Seller* conferma la creazione del FS.

Aggiunta di un nuovo Stand Product Un utente *Seller* pubblica un nuovo *Stand Product* in un proprio *Flash Stand*.

Pre-condizioni	L'utente è loggato come *Seller*. L'utente ha almeno un FS attivo.
Post-condizioni	Il nuovo SP è correttamente inserito nel FS specificato.
Scenario principale	1. Il *Seller* seleziona il *Flash Stand* al quale vuole aggiungere uno *Stand Product*. 2. Il *Seller* indica che vuole inserire un nuovo *Stand Product*. 3. Il sistema chiede all'utente di indicare una specie ittica dalla tassonomia. 4. Il sistema chiede l'inserimento di quantità e prezzo. 5. Il *Seller* conferma l'inserimento di uno SP.

Ricerca di un Flash Stand Un utente (tipicamente un *Customer*, ma potrebbe essere anche un *Seller*) cerca un FS che venda un certo prodotto.

Pre-condizioni	Nel sistema sono presenti FS attivi con SP inseriti.
Post-condizioni	Nessuna.
Scenario principale	1. L'utente indica la volontà di cercare un prodotto. 2. Il sistema chiede all'utente di indicare una specie ittica presente nella tassonomia. 3. Il sistema mostra gli SP che fanno riferimento a quella specie ed alle specie simili. 4. L'utente seleziona uno SP e vengono visualizzati i dettagli dello stesso e del FS di appartenenza.

Capitolo 4

Progettazione del prototipo

4.1 Modello dei dati

Segue il modello dei dati che modellizza le entità coinvolte dalle funzionalità implementate nel prototipo.

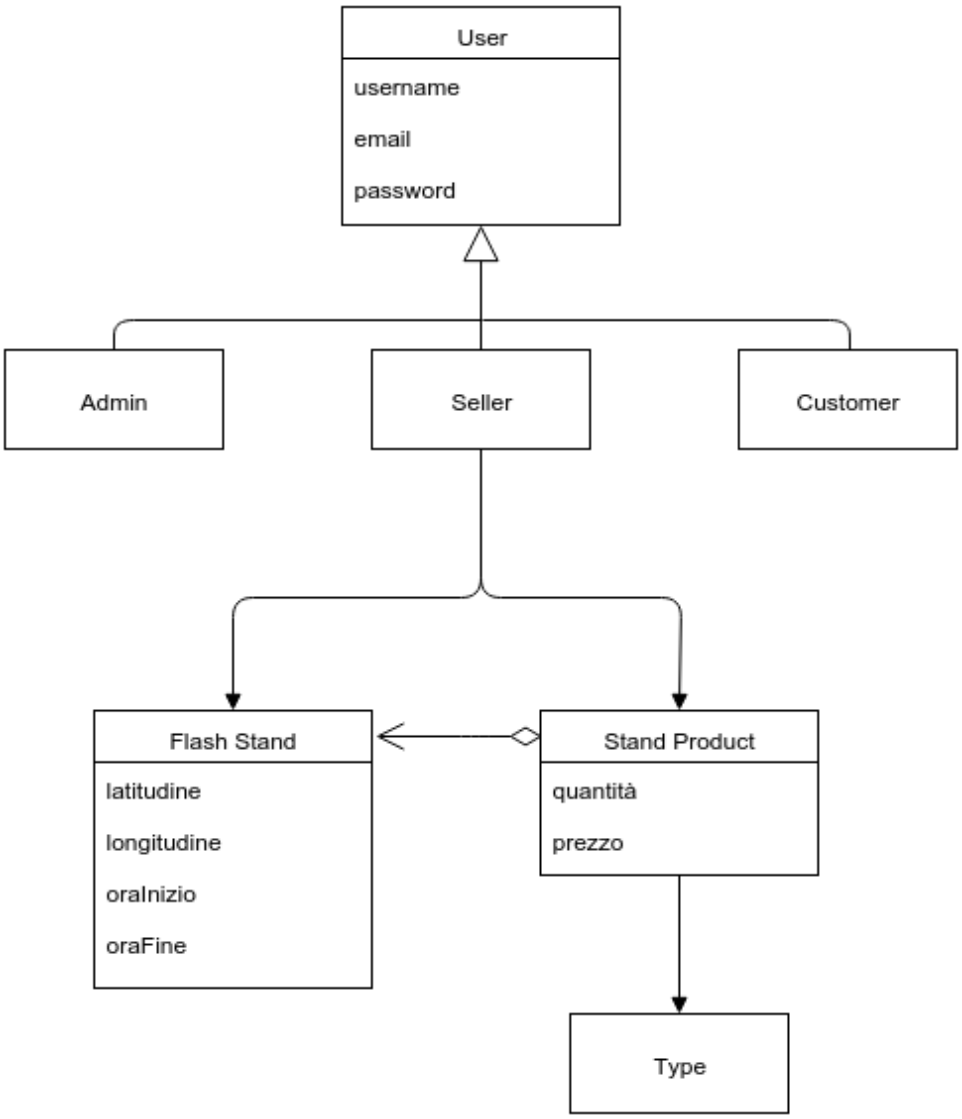

Figura 4.1: Modello dei dati del prototipo RFT

Flash Stand

Rappresenta un mercato temporaneo creato da un *Seller* e ad esso appartiene. Un *Seller* non può avere più di un *Flash Stand* con la stessa posizione geografica e fascia oraria. Un FS può contenere uno o più *Stand Product*.

Gli attributi che rappresentano la posizione geografica sono *latitudine* e *longitudine*.

Gli attributi *oraInizio* e *oraFine* specificano la fascia oraria di attività. Non è obbligatoria una durata minima, l'importante è che sia rispettata la condizione:
$oraFine \geq oraInizio$. Con riferimento all'istante attuale t, un FS può trovarsi in tre stati. Se $t < oraFine$ si dice che il *Flash Stand* è "**attivo**", mentre si dice "**scaduto**" se $t \geq oraFine$. Un FS si dice "**in corso**" se
$oraInizio < t < oraFine$ (un FS "in corso" è anche "attivo").

Stand Product

Rappresenta un prodotto pubblicato da un *Seller* in un suo *Flash Stand*. Ad esso è associato un *Type*, che indica il tipo di prodotto, ovvero la specie ittica alla quale fa riferimento nella tassonomia.

L'attributo *quantità* indica la quantità disponibile alla vendita in una particolare unità di misura (chilogrammi, litri, inità,ecc...). La quantità specificata può essere "**prevista**" o "**reale**".

Ogni prodotto deve avere un *prezzo*, specificato nella quantità di misura. Il prezzo dovrebbe essere espresso nella moneta corrente della nazione in cui il FS è localizzato.

Type

Rappresenta una specie nella tassonomia di riferimento. È l'entità che crea un collegamento tra una precisa specie ittica e un prodotto pubblicato su RFT.

User

Entità che modellizza gli utenti di RFT. Le categorie di utenti sono quelle già definite nella descrizione del sistema completo: *Admin*, *Seller*, *Customer*.

Capitolo 5

Modello semantico dei dati

RFT si basa su un modello semantico dei dati, che di seguito chiameremo *indice*. Ogni elemento dell'indice rappresenta una specie ittica ed ogni prodotto pubblicato su RFT deve essere mappato con un elemento dell'indice. In questo modo si ha una corrispondeza univoca tra prodotto e specie. L'indice inoltre tiene traccia di informazioni aggiuntive che consentono la ricerca semantica dei prodotti.

In questo capitolo illustreremo le scelte progettuali dell'indice, i dataset di partenza utilizzati per la sua creazione e le potenzialità di questa struttura dati.

5.1 Dataset di partenza

Le informazioni utilizzate per la costruzione dell'indice provengono da due fonti: la tassonomia definita dalla FAO e DBpedia.org.

5.1.1 Tassonomia ASFIS-FAO

Una tassonomia è una classificazione dei concetti in questione, le specie viventi nel nostro caso, in una gerarchia di unità tassonomiche annidate. Un'unità tassonomica, o taxa, è un raggruppamento di organismi reali, distinguibili morfologicamente e geneticamente da altri e riconoscibili co-

me unità sistematica, posizionata all'interno della struttura gerarchica della classificazione scientifica.

La fonte primaria è l'ontologia delle specie ittiche distribuita dall'ASFIS[1] a cura della FAO[2]. Tale ontologia contiene informazioni riguardo la classificazione tassonomica di 12600 entità biologiche e viene distribuita in formato OWL.

Struttura Una classificazione tassonomica rigorosa parte dal rango più in alto denominato "vita" fino agli strati più bassi "specie" e "sottospecie". La tassonomia in questione invece, non riporta tutte le informazioni tassonomiche, in quanto l'obbiettivo non è quello di fornire una tassonomia rigorosa, bensì quello di elencare le specie di pesci, crostacei, mammiferi e alghe che teoricamente possono essere target di pescatori[3]. L'ontologia definisce inizialmente i *gruppi*, un *gruppo* rappresenta una macro-categoria come pesci o molluschi. Ogni gruppo è rappresentato da un nodo nella tassonomia. Dai gruppi discendono gli *ordini*, composti a loro volta dalle *famiglie*, da cui seguono le *specie*, ognugno dei quali costituisce un nodo tassonomico. Per ogni *gruppo* quindi, la tassonomia definisce un albero a quattro livelli con la radice in *gruppo*. La tassonomia ASFIS è costituita da 7 alberi che rappresentano i seguenti gruppi: PISCES, CRUSTACEA, MOLLUSCA, MAMMALIA, AMPHIBIA-REPTILIA, INVERTEBRATA AQUATICA, PLANTAE AQUATICAE.

Proprietà Esistono legami tra i nodi che compongono la tassonomia che vengono esplicitati dalle seguenti proprietà:

- `hasDirectHigherRank` : mette in relazione un nodo tassonomico con il nodo di rango maggiore direttamente collegato nella tassonomia;

[1] Aquatic Science and Fisheries Information System

[2] Food and Agriculture Organization of the United Nations

[3] L'ontologia definisce tutto ciò che è "pescabile" indipendentemente dal fatto che sia legale o meno. Sarà compito del sistema RFT controllare che la legge sia rispettata dagli utilizzatori.

- `hasHigherRank` : mette in relazione un nodo tassonomico con i nodi di rango maggiore;
- `hasDurectLowerRank` : mette in relazione un nodo tassonomico con il nodo di rango inferiore direttamente collegato nella tassonomia;
- `hasLowerRank` : mette in relazione un nodo tassonomico con i nodi di rango inferiore;

Campi Ogni nodo è costituito da diversi dettagli. Segue una lista dei campi che descrivono un taxon:

- **Nome scientifico**
  Questo campo contiene il nome scientifico nella nomenclatura binomiale;
- **Nomi in lingua inglese, francese, spagnolo**
  Nomi comuni nelle tre lingue indicate, non sempre sono presenti;
- **Taxonomic code**
  Si tratta di un codice di dieci cifre per la classificazione. In alcuni casi sono utilizzate tre cifre aggiuntive. Ogni gruppo di cifre si riferisce ad un rango della tassonomia, secondo il seguente schema:
  - 1^a cifra: Gruppo;
  - 2^a-3^a cifra: Ordine;
  - 4^a-5^a cifra: Famiglia;
  - 6^a-8^a cifra: Genere;
  - 9^a-10^a cifra: Specie;
- **3-alpha code**
  È un codice identificativo interno all'agenzia ASFIS costituito da tre lettere. Generalmente i tre caratteri sono scelti in maniera casuale, solo in alcuni casi sono legati al nome scientifico o inglese;

- **ID**

  Codice numerico univoco.

5.1.2 DBpedia

DBpedia è un progetto nato nel 2007 e tuttora in corso, per l'estrazione di informazioni strutturate da Wikipedia e per il rilascio di queste informazioni sul Web come Linked Data in formato RDF. Il progetto nacque da una collaborazione tra le due università *Free University of Berlin* e *University of Leipzig* e la società americana *OpenLink Software*.

DBpedia consente agli utenti di effettuare query semantiche sulle relazioni e proprietà associate con le risorse di Wikipedia, compresi link ad altri dataset. DBpedia è stato descritto da Tim Berners-Lee come una delle parti più famose delgli sforzi per il Linked Data decentralizzato.

Dataset Gli articoli pubblicati su Wikipedia sono costituiti per la maggior parte da testo, ma includono anche informazioni strutturate, come informazioni di categorizzazione, immagini, coordinate geografiche e link a siti esterni. L'obbiettivo di DBpedia è quello di estrarre queste informazioni per inserirle in un dataset uniforme che può essere poi interrogato.

Il dataset di DBpedia contiene 4,58 milioni di entità, di cui 4,22 sono classificati in un'ontologia consistente. Per avere un idea della vastità del dataset, si pensi che e risorse descritte comprendono 1.445.000 persone, 735.000 luoghi, 123.000 album musicali, 241.000 organizzazioni e 251.000 specie. Sono riportate informazioni in 125 lingue differenti, 25,2 milioni di link ad immagini e 29,8 milioni di link a pagine web esterne. Il grafo RDF che rappresenta le informazioni estratte è costituito da 3 miliardi di triple e contiene circa 50 milioni di link ad altri dataset RDF.

5.2 Indice delle specie

Chiamiamo *indice* o *indice delle specie* la struttura dati che rappresenta la tassonomia di riferimento per Real Food Trade. Ci riferiremo con il termine

nodo invece, per indicare un elemento, ovvero una specie, dell'indice.

Questa struttura dati è costruita estraendo informazioni dalle fonti appena descritte: la tassonomia fornita dalla FAO e DBpedia. Non è definito un link diretto tra una risorsa FAO e la corrispondete in DBpedia (se esite), si è dovuto quindi individuare un campo per realizzare questo mapping. È stato scelto il nome scientifico, in quanto univoco e presente in entrambi i dataset di partenza. Il mapping è necessario per l'estrazione di informazioni dalle due fonti eterogenee e la loro unificazione nella nuova struttura dati.

Si noti che non tutte le specie definite nella tassonomia FAO hanno una corrispondente risorsa in DBpedia. Al momento della creazione dell'indice, vengono inserite tutte le specie presenti nell'ontologia FAO, arricchendole con le informazioni estraibili da DBpedia, se presenti.

5.2.1 Informazioni rappresentate

L'indice di fatto è una collezione di nodi, implementata in Java tramite la struttura dati ArrayList. Per ogni specie vengono estratte le seguenti informazioni:

- **ID FAO**
- **URI FAO**
  URI nella tassonomia FAO.
  Esempio: `http://www.fao.org/aims/aos/fi/species_taxonomic.owl#ID_31005_2118`;
- **URI DBpedia**
  URI della risorsa DBpedia.
  Esempio:`http://dbpedia.org/resource/Coho_salmon`;
- **Nome scientifico**
- **Nomi comuni**
  Vengono estratti i nomi in inglese, francese e spagnolo dalla tassonomia FAO. Da DBpedia vengono estratti tutti i nomi rappresentati

dalla proprietà `foaf:name`. Per ogni nome viene memorizzato anche il linguaggio tramite un *language tag* (it,en,fr,ecc...);

- **Immagine**
  URL dell'immagine nel rispettivo articolo Wikipedia. Viene riportato nella risorsa DBpedia dalla proprietà `foaf:depiction`.

Proprietà estratte Ai fini delle tecniche di ricerca semantica, si rende necessario l'estrazione dei valori assunti da specifiche proprietà dei grafi di partenza. Dopo un'attenta analisi delle possibili scelte, si è deciso di estrarre le tre proprietà seguenti:

- `taxonomic:hasHigherRank`
- `taxonomic:hasDirectHigherRank`
- `dct:subject`
  Mette in relazione una risorsa nel grafo DBpedia con una seconda risorsa, che ne rappresenta una categoria.

Capitolo 6

Ricerca per nome

Per implementare la ricerca di un prodotto o di una specie in RFT si rende necessario un meccanismo per effettuare una ricerca sull'indice data una stringa. La ricerca per nome quindi torna il nodo dell'indice con il nome più simile alla chiave di ricerca.

6.1 Similarità tra stringhe

La similarità tra stringhe viene calcolato con una metrica pubblicata in un articolo online[1] da Simon White. L'algoritmo è stato guidato da tre requisiti:

- **Somiglianza lessicale**
  Stringhe con piccole differenze dovrebbero essere riconosciute come simili. In particolare, una sotto-stringa significativa in comune dovrebbe garantire un alto grado di similarità;

- **Robustezza nell'ordinamento delle lettere**
  Due stringhe che ontengono le stesse parole, ma in un ordine differente, dvrebbero essere riconosciute come simili. D'altro canto, se una stringa è solo un anagramma casuale dei caratteri contenuti nell'altra, dovrebbe essere riconosciuta come dissimili;

[1] http://www.catalysoft.com/articles/StrikeAMatch.html

- **Indipendenza dal linguaggio**

  L'algoritmo non dovrebbe lavorare solo in inglese, ma in più lingue;

Metrica

La metrica è basata sul numero di coppie di caratteri adiacenti presenti in entrambe le stringhe. Questo premia sia la presenza di sottostringhe comuni, che l'ordinamento comune di queste sottostringhe.

Supponiamo di avere due stringhe *s1* e *s2*. Sia *pairs* la funzione che genera l'insieme delle coppie di caratteri adiacenti in una stringa. La formula utilizzata per il calcolo della similarità tra *s1* e *s2* è la seguente:

$$sim(s1, s2) = \frac{2 \cdot |pairs(s1) \cap pairs(s2)|}{|pairs(s1)| + |pairs(s2)|}$$

La similarità tra le due stringhe è data dal doppio del numero di coppie di caratteri in comune diviso per la somma delle coppie di caratteri nelle due stringhe.

Si noti che la formula fornisce un valore di similarità nullo per una coppia di stringhe completamente dissimili, ovvero senza alcuna coppia di caratteri in comune. D'altro canto, la similarità di una stringa (non vuota) con se stessa è 1. L'algoritmo quindi torna sempre un valore compreso tra 0 e 1, il che rende naturale esprimere la similarità in percentuale.

Esempio

Illustriamo il funzionamento della metrica descritta tramite un esempio. Supponiamo di voler confrontare le stringhe 'France' and 'French'. Per prima cosa si passa alle corrispondenti stringhe in caratteri maiuscoli, ciò rende l'algoritmo case insensitive, dopo di che le si dividono nelle coppie di caratteri:
FRANCE = {FR,RA,AN,NC,CE}
FRENCH = {FR,RE,EN,NC,CH}
L'intersezione è data dalle coppie FR,NC. La similarità è calcolata con la

formula precedentemente definita:

$$sim(FRANCE, FRANCE) = \frac{2 \cdot |\{FR, NC\}|}{|\{FR, RA, AN, NC, CE\}| + |\{FR, RE, EN, NC, CH\}|}$$

$$= \frac{2 \cdot 5}{5 + 5} = 0.4$$

Possiamo quindi concludere che le due stringhe in questione hanno una similarità del 40

6.2 Algoritmo

L'utente inserisce una stringa e indica la lingua nella quale vuole effettuare la ricerca. Per ogni nodo dell'indice vengono considerati solo i nomi nella lingua utente ed il nome scientifico. Per ogni nome viene calcolata la similarità con la chiave di ricerca e la maggiore di queste viene associata al nodo. Al termine viene tornato il nodo con la similarità maggiore.

Capitolo 7

Ricerca per similarità

La ricerca per similarità è un meccanismo che selezionato un nodo dell'indice, fornisce i nodi ad esso simili. La prima scelta da fare è definire cosa si intende per similarità. Tralasciando temporaneamente i formalisimi, possiamo dire che due specie sono simili se :

- sono in una posizione simile nell'albero tassonomico, per esempio condividono lo stesso nodo di rango superiore;
- condividono caratterisitiche indipendenti dalla tassonomia, per esempio sono entrambi pesci locali della stessa zona.

Entrambe queste caratteristiche sono descritte dalle tre proprietà memorizzate nell'indice, che ricordiamo essere:

- `taxonomic:hasHigherRank`
- `taxonomic:hasDirectHigherRank`
- `dct:subject`

Le prime due rappresentano le informazioni riguardanti la tassonomia, la terza le caratteristiche ad essa estranee.

L'algoritmo utilizzato per il calcolo della similarità è il Vector Space Model (VSM). Questo modello utilizza una rappresentazione vettoriale delle proprietà coinvolte.

7.1 Vector Space Model

Vector Space Model (VSM) è un modello algebrico per la rappresentazione di documenti testuali in forma vettoriale. La similarità tra due documenti viene calcolata computando il coseno di similitudine tra i due vettori che li rappresentano.

Coseno di similitudine Il coseno di similitudine è una tecnica euristica per la misurazione della similitudine tra due vettori effettuata calcolando il coseno tra di loro. Dati due vettori di attributi numerici, A e B, il livello di similarità tra di loro è espresso utilizzando la formula:

$$\frac{A \cdot B}{|A| \cdot |B|}$$

In base alla definizione del coseno, dati due vettori si otterrà sempre un valore di similitudine compreso tra -1 e +1, dove -1 indica una corrispondenza esatta ma opposta (ossia un vettore contiene l'opposto dei valori presenti nell'altro) e +1 indica due vettori uguali.

VSM e RDF Nel caso di studio, non siamo interessati ai documenti testuali, bensì al calcolo della similarità tra risorse RDF. Per questo motivo, consideriamo la versione proposta nell'articolo[6] pensata per risorse RDF in un sistema di raccomandazione.

Proponiamo di seguito l'algoritmo originale presentato nell'articolo sopracitato e gli adattamenti che si sono ritenuti necessari per il caso in questione.

7.1.1 Algoritmo originale

Il modello utlizzato si rifà allo schema TF-IDF (Term Frequency - Inverse Document Frequency) che nasce per i documenti di testo. Questa versione di VSM ha lo sopo di calcolare la similarità tra risorse appartenenti allo stesso grafo RDF, considerando solo proprietà il cui oggetto è una risorsa e non un letterale.

Rappresentazione vettoriale Sia p una proprietà e sia t la cardinalità del suo range, ovvero il numero di risorse distinte che compaiono come oggetto di una terna in cui il predicato sia p. Sia r_i l'i-esima risorsa dell'indice, essa sarà descritta, rispetto alla proprietà p, dal vettore

$$W_{i,p} = (w_{1,i,p}, w_{2,i,p}, ..., w_{t,i,p})$$

Dove

$$w_{n,i,p} = f_{n,i,p} * log\left(\frac{M}{a_{n,p}}\right) \qquad \text{con } n : n\text{-esimo elemento del range di } p$$

$$f_{n,i,p} = \begin{cases} 1 & \text{se esiste la terna } < r_i\ p\ n > \\ 0 & \text{altirmenti} \end{cases}$$

$$M = \text{numero di risorse del grafo}$$

$$a_{n,p} = \text{numero di risorse collegate a } n \text{ da } p$$

Calcolo della similitudine La similitudine tra due risorse r_i e r_j rispetto alla proprietà p è calcolata computando il coseno di similitudine tra i due vettori:

$$sim^p(r_i, r_j) = \frac{\sum_{n=1}^{t} w_{n,i,p} \cdot w_{n,j,p}}{\sqrt{\sum_{n=1}^{t} w_{n,i,p}^2} \cdot \sqrt{\sum_{n=1}^{t} w_{n,j,p}^2}}$$

La formula per la similarità rispetto a tutte le proprietà non viene riportata in quanto fa riferimento a entità tipiche di un sistema di raccomandazione, che non interessano il nostro sistema.

7.1.2 Adattamenti

Il nostro caso viola le premesse dell'algoritmo originale in due punti:

- Consideriamo anche proprietà che hanno sia risorse che letterali come oggetto;
- Per ogni proprietà (eccetto quelle definite nel grafo FAO) entrano in gioco due grafi: quello della FAO e quello al quale la proprietà appartiene.

Sulla base di queste differenze sono state fatte le seguenti scelte:

- Nella costruzione del range di una proprietà consideriamo oggetti differenti due letterali con lo stesso valore lessicale me differente *language tag*;
- Nella costruzione del range di p e nel successivo calcolo di $a_{n,p}$ consideriamo solo le terne in cui il soggetto sia presente in entrambi i grafi;
- M = numero di risorse appartenenti ad entrambi i grafi.

Calcolo della similitudine Manteniamo la formula vista per il calcolo tra due risorse rispetto ad una proprietà. La similarità tra due risorse r_i e r_j rispetto ad un set P di proprietà viene calcolata come la media aritmetica delle similarità delle diverse proprietà:

$$sim(r_i, r_j) = \frac{\sum_{p \in P} sim^p(r_i, r_j)}{|P|}$$

7.2 Risultati sperimentali

I test condotti hanno lo scopo di dimostrare l'effettiva utilità delle tecniche di ricerca implementate. Particolare enfasi è posta sull'utilizzo di proprietà non legate alla tassonomia, nel caso specifo è stata utilizzata solo la proprietà `dct:subject` . La modularità del codice, poi consente la facile inclusione di nuove proprietà, anche derivanti da grafi linked data diversi.

Definizioni Si rendono necessarie alcune definizioni per poter illustrare i risultati dei test.

- **Higher Rank**
  Chiamiamo *Higher Rank* (*HR*) di una specie, una seconda specie che si trova in un livello superiore delle tassonomia.

- **Higher Rank diretto**
  Chiamiamo *Higher Rank diretto* di una specie, la specie che si trova nel livello immediatamente superiore della tassonomia.

Nei test eseguiti abbiamo misurato il numero di Higher Rank in comune tra i risultati di una ricerca semantica e la specie rispetto alla quale la ricerca fa riferimento. Per semplicità di notazione riportiamo solo gi ID dei nodi.
I prefissi utilizzati sono:

- `taxonomic = http://www.fao.org/aims/aos/fi/species_taxonomic.owl#`
- `dbr = http://dbpedia.org/resource/`
- `dct = http://purl.org/dc/terms/`
- `dbc = http://dbpedia.org/resource/Category:`

7.2.1 Riordinamento dei risultati

Utilizzando solo le prime due proprietà definite dalla ontologia FAO, l'unica metrica utilizzata per il calcolo della similarità di due specie sarebbe la loro posizione nella tassonomia.

Data una specie in particolare, i risultati di una ricerca semantica vedrebbero al primo posto tutti i pesci con il maggior numero di Higher Rank in comune con la specie in questione, prediligendo quelli che condividono anche l'Higher Rank diretto. Questi nodi avranno una similarità pressochè identica a parità di Higher Rank in comune.

L'introduzione di una proprietà indipendente dalla tassonomia consente un loro riordinamento.

Esempio Consideriamo il nodo `14545`, uri DBpedia: `dbr/Maroon_clownfish`.

Ricerca semantica per il nodo `14545` utilizzando solo le proprietà FAO:

# Risultato	ID	HR comuni	Similarità
1	10112	4	1.00
2	11407	4	1.00
...	...	...	...
36	12825	4	1.00
...	...	...	...
40	11394	4	1.00

I primi risultati sono 40 nodi, tutti con 4 Higher Rank in comune col nodo `14545` e tutti con similarità massima. La similarità è massima perchè questi nodi hanno la stessa tassonomia del nodo `14545` ed utilizzando solo queste proprietà risultano identici al nodo di riferimento.

Introduciamo ora la terza proprietà.
Ricerca semantica per il nodo `14545` utilizzando tutte le proprietà :

# Risultato	ID	HR comuni	Similarità
1	12825	4	0.82
2	13113	4	0.66
3	15433	4	0.66

La tendenza rimane la stessa, ovvero i primi nodi rimangono quelli con Higher Rank pari a 4. Tra questi, però emerge il nodo `12825` che, con l'introduzione della terza proprietà, ottiene una similarità maggiore rispetto agli altri nodi con il massimo numero di Higher Rank in comune. Questo è dovuto al fatto che il nodo `12825` fa riferimento alla risorsa `dbr/Garibaldi_(fish)` che condivide con il nodo `14545` la categoria `dbc:Monotypic_fish_genera`. Gli altri risultati invece non condividono alcuna categoria, nel caso specifico non hanno proprio una risorsa corrispondente in DBpedia.

7.2.2 Indipendenza dalla tassonomia

Un altro vantaggio dell'utilizzo di proprietà non legate alla tassonomia, è la capacità di invertire la tendenza descritta precedentemente. In alcuni casi, infatti una specie potrebbe essere più simile ad un'altra non strettamente legata nella tassonomia. Questa similarità però è impossibilitata ad emergere utilizzando solo le proprietà definite dalla FAO, che usano come unica metrica la posizione nell'albero tassonomico.

Esempio Consideriamo il nodo `2933`, uri DBpedia: `dbr/Chinook_Salmon`. Esso appartiene alle seguenti categorie:

- `dbc:Commercial_fish`
- `dbc:Fish_of_the_United_States`
- `dbc:Megafauna_of_Eurasia`
- `dbc:Salmon`
- `dbc:Symbols_of_Oregon`
- `dbc:Oncorhynchus`
- `dbc:Freshwater_fish_of_Japan`
- `dbc:Animals_described_in_1792`
- `dbc:Arctic_freshwater_fish`
- `dbc:Fish_of_the_Great_Lakes`
- `dbc:Fish_of_the_Pacific_Ocean`
- `dbc:Fly_fishing_target_species`

Risultati della ricerca semantica:

# Risultato	ID	HR comuni	Similarità
1	2118	9	0.89
2	2116	8	0.86
3	2931	9	0.83

Come possiamo vedere dalla tabella, il secondo risultato ha un Higher Rank in comune in meno rispetto al terzo, ciò nonostante la similarità è maggiore. Questo è dovuto al fatto che il nodo `2116` condivide con il nodo di riferimento 9 categorie, mentre il nodo `2931` solo 5. Di seguito i dettagli delle risorse DBpedia di questi due nodi.

- Nodo `2116`
  uri DBpedia: `dbr/Pink_salmon`
  Categorie in comune con il nodo `2933`:
  - `dbc:Commercial_fish`
  - `dbc:Fish_of_the_United_States`
  - `dbc:Salmon`
  - `dbc:Oncorhynchus`
  - `dbc:Animals_described_in_1792`
  - `dbc:Arctic_freshwater_fish`
  - `dbc:Fish_of_the_Great_Lakes`
  - `dbc:Fish_of_the_Pacific_Ocean`
  - `dbc:Fly_fishing_target_species`
- Nodo `2931`
  uri DBpedia: `dbr/Chum_salmon`
  Categorie in comune con il nodo `2933`:
  - `dbc:Salmon`
  - `dbc:Oncorhynchus`
  - `dbc:Arctic_freshwater_fish`

- `dbc:Fish_of_the_Pacific_Ocean`
- `dbc:Fly_fishing_target_species`

7.2.3 Indipendeza dalla sintassi

In questo test si vuole dimostrare come la ricerca basata su proprietà semantiche fornisca dei risultati migliori di quelli ottenibili con una ricerca basata semplicemente sulla sintassi. In una ricerca di questo tipo vengono mostrati i nodi i cui nomi corrispondono maggiormente ad una stringa scelta dall'utente come chiave della ricerca. Con questo primitivo approccio non verranno visualizzati i nodi simili ma con nomi dissimili.

Esempio Si tratta di due ricerche fondamentalmente diverse, in quanto la ricerca sintattica prevede la definizione di una stringa, mentre la ricerca semantica la selezione di un nodo della tassonomia. Per confrontare i risultati si è proceduto nel seguente modo.

Consideriamo il nodo `2933`, uri DBpedia: `dbr/Chinook_salmon`. Ad esso sono associati i seguenti nomi (consideriamo i nomi inglesi):

- Oncorhynchus tshawytscha
- Chinook salmon
- Chinook king salmon

Vogliamo trovare i nodi simili a questo tramite le due tecniche di ricerca. Eseguiamo una ricerca sintattica utilizzando come chiave la stringa "Chinook salmon". Nella tabella sottostante vi sono i risultati di questa ricerca. Riportiamo la similarità semantica di ogni nodo col nodo di riferimento.

# Risultato	ID	Nomi	Similarità
1	2931	Oncorhynchus keta, Chum salmon	0.83
2	2116	Oncorhynchus gorbuscha, Pink salmon	0.86
3	13130	Leptobrama muelleri, Beachsalmon	0.0004
4	2927	Salmonoidei	0.24
5	15758	Trachinotus mookalee, Indian pompano	0.04

Eseguiamo ora la ricerca semantica rispetto al nodo `2933`:

# Risultato	ID	Nomi	Similarità
1	2118	Oncorhynchus kisutch, Coho salmon	0.89
2	2116	Oncorhynchus gorbuscha, Pink salmon	0.86
3	2931	Oncorhynchus keta, Chum salmon	0.83
4	2117	Oncorhynchus nerka, Sockeye salmon	0.82
5	2934	Smithichthys fucorum, Leafy klipfish	0.76

Andiamo ora a confrontare le due tabelle. Utilizzando la ricerca sintattica otteniamo dei risultati simili nel nome, ma lontani semanticamente. Ad esempio il terzo risultato della prima tabella ha una similarità col nodo di riferimento pari a 0.0004. Utilizzando invece la ricerca semantica proposta, si ottengono risultati semanticamente vicini, anche se con nomi diversi. Ad esempio il quinto risultato della seconda tabella ha una similarità molto alta (0.76), ma non sarebbe mai comparso nei risultati di una ricerca sintattica.

7.2.4 Conclusioni

Dai test riportati emerge come la tecnica di ricerca proposta sia in grado di fornire risultati migliori di quelli ottenibili tramite tecniche più semplici come quelle basate esclusivamente sulla tassonomia o sui nomi.

I primi due test mostrano i miglioramenti riscontrati rispetto ad una ricerca semantica basata esclusivamente sulla tassonomia. La nuova tecnica infatti si è dimostrata in grado di consentire un ordinamento dei nodi nella

stessa posizione nell'albero tassonomico e di dare la giusta priorità alle specie simili anche se più lontante nella tassonomia rispetto ad altre.

Il terzo test mostra come questa tecnica consente di superare i limiti imposti da una ricerca sintattica basata esclusivamente sui nomi, dimostrando di essere in grado di riconoscere la similarità tra due specie anche se con nomi completamente dissimili.

Capitolo 8

Implementazione

In questo capitolo vengono descritti i dettagli implementativi di RFT, dividendo la trattazione in Back End e Front End. Vengono descritte nel dettaglio le tecnologie utilizzate maggiormente rilevanti.

8.1 Back End

Il back end del sistema RFT è rappresentato da un Web Service, realizzato secondo l'architettura REST. Il Web Service mette a disposizione delle API per accedere alle operazioni sulle risorse RESTful. Esso si appoggia sull'indice delle specie e su un database mySQL.

8.1.1 Architettura REST

REpresentational State Transfer (REST) è un'architettura software che definisce dei vincoli da applicare al design dei componenti in un sistema distribuito di ipermedia.

Un sistema conforme a tali vincoli si dice RESTful. Tipicamente, ma non sempre, un sistema RESTful comunica tramite protocco HTTP, utilizzando le stesse azioni (GET, POST, PUT, DELETE, ecc...) che i browser usano per raggiungere pagine Web e inviare dati a server remoti.

REST è stato definito da Roy Fielding, uno dei principali autori delle speicifiche del protocollo HTTP, nella sua tesi di dottorato[7].

Vincoli L'approccio architetturale REST è definito dai seguenti sei vincoli applicati ad una architettura, mentre si lascia libera l'implementazione dei singoli componenti.

- **Client-server**

  Un insieme di interfacce uniformi separa i client dai server. Questa separazione di ruoli significa che, per esempio, il client non si deve preoccupare del salvataggio delle informazioni, che rimane all'interno di ogni singolo server, in questo modo la portabilità del codice del client ne trae vantaggio. I server non si devono preoccupare dell'interfaccia grafica o dello stato dell'utente, in questo modo i server sono più semplici e maggiormente scalabili. Server e client possono essere sostituiti e sviluppati indipendentemente fintanto che l'interfaccia non viene modificata.

- **Stateless**

  La comunicazione client-server è ulteriormente vincolata in modo che nessun contesto client venga memorizzato sul server tra le richieste. Ogni richiesta di ogni client contiene tutte le informazioni necessarie per richiedere il servizio, e lo stato della sessione è contenuto sul client.

- **Cacheable**

  Come nel World Wide Web, i client possono fare caching delle risposte. Le risposte devono in ogni modo definirsi implicitamente o esplicitamente cacheable o no, in modo da prevenire che i client possano riusare stati vecchi o dati errati. Una gestione ben fatta della cache può ridurre o parzialmente eliminare le comunicazioni client-server, migliorando scalabilità e performance.

- **Layered system**

  Un client non può dire se è connesso direttamente ad un server di livello più basso od intermedio, i server intermedi possono migliorare la scalabilità del sistema con load-balancing o con cache distribuite. Layer intermedi possono offrire inoltre politiche di sicurezza

- **Code on demand** (opzionale)

  I server possono temporaneamente estendere o personalizzare le funzionalità del client trasferendo del codice eseguibile. Ad esempio questo può includere componenti compilati come Applet Java o linguaggi di scripting client side come JavaScript. Code on demand è l'unico vincolo opzionale per la definizione di un'architettura REST.

- **Uniform interface** Un'interfaccia di comunicazione omogenea tra client e server permette di semplificare e disaccoppiare l'architettura, la quale si può evolvere separatamente. I quattro vincoli per l'uniformità dell'interfaccia sono:

  - **Identificazione delle risorse**

    Le risorse sono individuate univocamente utilizzando gli URI. Le risorse sono concettualmente separate dalle rappresentazioni che vengono tornate ai client. Per esempio, il server potrebbe inviare dati estratti dal proprio database in formato HTML, XML o JSON, dei quali nessuno è la rappresentazione interna al server;

  - **Manipolazione delle risorse**

    Quando un client accede alla rappresentazione di una risorsa, incluso ogni metadata ad essa collegato, ha abbastanza informazioni per modificare o eliminare la risorsa;

  - **Messaggi auto-descrittivi**

    Ogni messaggio comprende l'informazione necessaria per processarlo;

  - **Hypermedia as the Engine of Application State (HATEOAS)**

    I client effettuano transazioni solo attraverso azioni che sono dinamicamente identificate attraverso ipermedia dal server. Fatta eccezione per alcuni semplici entry point prefissati all'applicazione, un client non presume che una particolare azione sia disponibile per una particolare risorsa fatta eccezione al di là di

quelle descritte nelle rappresentazioni precedentemente ricevute dal server.

Proprietà architetturali REST influenza diverse proprietà architetturali di un sistema. Segue un elenco delle stesse:

- Performance
- Scalabilità
- Semplicità dell'interfaccia
- Modificabilità dei componenti
- Visibilità delle comunicazioni
- Portabilità dei componenti
- Affidabilità

Riportiamo gli effetti sulle sopracitate proprietà, così come evidenziati da Fielding. La separazione Client/Server semplifica l'implementazione dei componenti, riduce la complessità dei connettori semantici, migliora l'efficacia dell'ottimizzazione delle performace e incrementa la scalabilità dei componenti server. La strutturazione a livelli consente agli intermediari, come proxy, gateway e firewall, di essere introdotti in vari punti della comunicazione senza cambiare le interfaccie tra i componenti. Ciò facilita la traduzione delle comunicazioni e consente di migliorare le performance su larga scala.

Risorse Un concetto importante in REST è l'esistenza di risorse (fonti di informazioni), a cui si può accedere tramite un identificatore globale (un URI). Per utilizzare le risorse, le componenti di una rete (componenti client e server) comunicano attraverso una interfaccia standard (ad es. HTTP) e si scambiano rappresentazioni di queste risorse (il documento che trasmette le informazioni). Ad esempio, una risorsa cerchio potrebbe accettare

e restituire una rappresentazione che specifica un punto per il centro e il raggio, formattati nel formato SVG, ma potrebbe anche accettare e restituire una rappresentazione che specifica tre punti distinti qualsiasi lungo la circonferenza nel formato CSV.

Un numero qualsiasi di connettori (client, server, cache, tunnel ecc.) può mediare la richiesta, ma ogni connettore interviene senza conoscere la ?storia passata? delle altre richieste. Di conseguenza una applicazione può interagire con una risorsa conoscendo due cose: l'identificatore della risorsa e l'azione richiesta - non ha bisogno di sapere se ci sono proxy, gateway, firewalls, tunnel, ecc tra essa e il server su cui è presente l'informazione cercata. L'applicazione comunque deve conoscere il formato dell'informazione (rappresentazione) restituita, tipicamente un documento HTML, XML o JSON, ma potrebbe essere anche un'immagine o qualsiasi altro contenuto.

8.1.2 JAX-RS

Per l'implementazione di RFT come sistema RESTful è stata utilizzata la libreria JAX-RS (Java API for RESTful Web Services) . È stata introdotta in Java SE 5, per semplificare le fasi di sviluppo e pubblicazione di Web Service secondo l'architettura REST. A partire dalla versione 1.1, JAX-RS è componente ufficiale di Java EE. Esistono diverse implementazioni di JAX-RS, nel caso di studio è stato utilizzato il framework Jersey.

Caratteristica fondamentale di queste API è l'utilizzo di annotazioni Java. Un'annotazione è una forma sintattica per esprimere metadati che può essere aggiunta al codice Java. Possono essere annotati classi, metodi, variabili, parametri e package. Le annotazione JAX-RS che consentono di mappare una classe con una risorsa Web sono le seguenti:

- `@Path`

  Specifica il path relativo di una risorsa, che può essere una classe o un metodo;

- `@GET,@PUT,@POST,@DELETE,@HEAD`
  Specifica il tipo di richiesta HTTP di una risorsa;
- `@Produces`
  Specifica il Media Type di una risposta;
- `@Consumes`
  Specifica il Media Type accettato.

8.1.3 Operazioni CRUD

Una volte definite le risorse bisogna stabilire le operazioni che sono permesse su queste risorse. In informatica, con l'acronimo CRUD ci si riferisce alle quattro funzioni base dello storage permanente: Create, Read, Update e Delete.

Nel contesto di un Web Service RESTful, l'utilizzo dell'approccio CRUD ci impone di sfuttare i metodi (o verbi) predefiniti del protocollo HTTP, e cioè GET, POST, PUT e DELETE. In un sistema RESTful sappiamo già a priori come ottenere la rappresentazione di una risorsa. In altre parole, questo principio REST stabilisce una mappatura uno a uno tra le tipiche operazioni CRUD (creazione, lettura, aggiornamento, eliminazione di una risorsa) e i metodi HTTP.

Metodo HTTP	**Operazione CRUD**	**Descrizione**
POST	Create	Crea una nuova risorsa
GET	Read	Ottiene una risorsa esistente
PUT	Update	Aggiorna una risorsa o ne modifica lo stato
DELETE	Delete	Elimina una risorsa

8.1.4 Individuazione delle Risorse

Il primo passo da fare consiste nell'individuare le risorse da esporre tramite il Web service. È importante evidenziare che in questa fase dobbiamo pensare

a come le risorse vengono viste da un ipotetico client e non a come vengono implementate internamente. Occorre infatti separare l'interfaccia verso l'esterno dai dettagli implementativi. Riportiamo le risorse individuate per l'implementazione del prototipo. Omettiamo

- **User** : rappresenta un utente del sistema;
- **Flash Stand** : rappresenta un *Flash Stand*;
- **Stand Product** : rappresenta uno *Stand Product*;
- **Node** : rappresenta un nodo dell'indice;
- **Language** : rappresenta una lingua utilizzata dai nomi delle specie estratte dai dataset di partenza.

8.1.5 API Rest

Riportiamo gli URL delle API fornite dal Web Service implementato. Tutti i parametri sono opzionali, i valori di default sono quelli tra parentesi quadre. I parametri comuni alla maggior parte delle API sono:

- `lang` : *language tag* ("en,"it,ecc...)
- `size` : numero di risultati

I risultati sono forniti in formato JSON.

Ricerca per nome

- (GET) `api/search/syntax/{key}?start=[0]&size=[5]&lang=[en]`
  - `start` : posizione iniziale dei risultati
  - `key` : stringa chiave per la ricerca

  Ritorna i nodi i cui nomi nella lingua specificata, o il nome scientifico, matchano maggiormente con la chiave di ricerca. Vengono valutati i nodi dell'indice e vengono tornati i primi `size` nodi a partire dalla posizione `start`.

Ricerca per similarità

- (GET) `api/search/semantic/{ID}?lang=[en]&size=[5]`
  Ritorna i nodi con maggiore similarità semantica rispetto al nodo rappresentato dall'ID specificato. Il *language tag* serve per indicare quali nomi estrarre per visualizzare i risultati nella lingua utente, ma non ha alcun valore per la ricerca semantica.

Nodi

- `api/nodes/{ID}?lang=[en] (GET)`
  Ritorna il nodo con ID specificato.
- `api/nodes/size (GET)`
  Torna il numero di nodi presenti nell'indice.
- `api/nodes/{ID}/standproducts (GET)`
  Torna gli *Stand Product* presenti associati a quel nodo.
- `api/nodes/{ID}/flashstands (GET)`
  Torna i *Flash Stand* che abbiano uno *Stand Product* associato al nodo.

Flash Stand

- `api/flashstands`
  - `GET`: Torna tutti i *Flash Stand*.
  - `POST`: Consente la creazione di un nuovo *Flash Stand*.
    Sicurezza
    Ruolo ammesso : *Seller*
    HTTP body
    `[Content-Type:application/json]`
    {
    `longitude :`*`double`*`,`
    `latitude:`*`double`*
    }

- `api/flashstands/{ID}`
  - `GET`: Torna il *Flash Stand* specificato.
  - `PUT`: Consente di modificate il *Flash Stand* specificato.

    Sicurezza

    Ruolo ammesso : *Seller*

    Il *Flash Stand* deve appartenere all'utente autorizzato.

    HTTP body

    `[Content-Type:application/json]`

    {

    `longitude :`*`double`*`,`

    `latitude:`*`double`*

    }
  - `DELETE`: Consente di eliminare il *Flash Stand* specificato.

    Sicurezza

    Ruolo ammesso : *Seller*

    Il *Flash Stand* deve appartenere all'utente autorizzato.

- `api/flashstands/{ID}/standproducts?id_index=[all]`
  - `GET`: Torna gli *Stand Product* presenti nel *Flash Stand* specificato. È possibile applicare il filtro su un particolare pesce presente nell'indice tramite il parametro `id_index`.
  - `POST`: Consente la creazione di un nuovo *Stand Product* per il *Flash Stand* specificato.

    Sicurezza

    Ruolo ammesso : *Seller*

    Si effettua il controllo che il *Flash Stand* appartenga all'utente autorizzato.

    HTTP body

    `[Content-Type:application/json]`

    {

`id_index :int,`

`price:double`

}

Stand Product

- `api/standproducts (GET)`
  Torna tutti gli *Stand Product*
- `api/standproducts/{ID}`
  - `GET`: Torna lo *Stand Product* specificato
  - `PUT`: Consente di modificate lo *Stand Product* specificato.
    Sicurezza
    Ruolo ammesso : *Seller*
    Lo *Stand Product* deve appartenere all'utente autorizzato.
    HTTP body
    `[Content-Type:application/json]`
    {
    `id_index :int,`
    `price:double`
    }
  - `DELETE`: Consente di eliminare lo *Stand Product* specificato.
    Sicurezza
    Ruolo ammesso : *Seller*
    Lo *Stand Product* deve appartenere all'utente autorizzato.

Utenti

- `api/users?role=[all] (GET)`
  Torna tutti gli utenti appartenenti al ruolo specificato. Il ruolo può essere `Admin,Seller,Customer` o `all`(Tutti gli utenti).

Sicurezza

Ruolo ammesso : *Admin*

- `api/users/{ID} (GET)`

  Torna l'utente specificato, senza dati sensibili (password).

- `api/users/{ID}/flashstands (GET)`

  Torna i *Flash Stand* dell'utente specificato.

Varie

- `api/languages (GET)`

  Ritorna i *language tag* per le le lingue presenti nel nell'indice.

- `api/authentication (POST)`

  Torna una *HTTP Response* con *Status=200* (ok) in caso di credenziali corrette.

- `api/registration (POST)`

  Consente di creare un nuovo utente e torna l'ID assegnato in automatico.

  Sicurezza

  Si possono creare solo utenti con ruolo *Seller* o *Customer*. La mail o lo username non devono essere già in uso. Il controllo della robustezza della password viene delegata al client.

  HTTP body

  {
  `username :`*`string`*`,`
  `mail:`*`string`*`,`
  `password:`*`string`*`,`
  `role:`*`string`*
  }

8.1.6 Sicurezza

Ai fini della realizzazione del prototipo la questione relativa alla sicurezza non è stata posta al centro dell'attenzione. Ciò nonostante è stato implementato un sistema minimo di sicurezza che garantisse i processi di autenticazione e autorizzazione. L'**autenticazione** è il processo di verifica dell'identità dell'utente tramite il controllo delle credenziali fornite. Se le credenziali sono valide, il processo di autorizzazione può iniziare. L'**autorizzazione** è il processo in cui viene concesso ad utente autenticato di accedere a determinate risorse, controllando quali privilegi l'utente ha sul sistema. Il meccanismo utilizzato è *HTTP basic access authentication*.

HTTP Basic Access Authentication Nel contesto di una transazione HTTP, basic access authentication è un metodo per uno user agent per fornire username e password quando effettua una richiesta. L'implementazione di questo metodo è la tecnica più semplice per consentire un controllo degli accessi a risorse Web in quanto non richiede cookie o id di sessione.

Basic Authentication, indicata anche come autenticazione BA, non prevede protezione per le credenziali trasmesse. Esse vengono banalmente codificate con base64 ma non criptate o crittografate con hash. Viene solitamente utilizzato in HTTPS.

Il meccanismo in questione richiede solo lo username e password, ma in ogni richiesta HTTP. Per inviare le proprie credenziali, il client utilizza l'header HTTP *Authorization*, il quale è costruito nel seguente modo:

1. Username e password sono uniti nella stringa "username:password"
2. Il risultato è codificato con base64
3. Il metodo di autorizzazione ("Basic") e uno spazio sono inseriti all'inizio della stringa codificata.

Ad esempio, se il client utilizza '`nicolo`' come username e '`password1234`' come password, l'header è formato nel seguente modo:

```
Authorization:  Basic bmljb2xvOnBhc3N3b3JkMTIzNA==
```

8.1.7 Esempi di richieste

Seguono due brevi esempi di richieste alle API fornite dal sistema RESTful. Riportiamo una schematizzazione delle richieste HTTP con il body (se presente) e delle risposte. Il campo `Host` è l'indirizzo target delle richieste.

Richiesta degli Stand Product di un Flash Stand Supponiamo di voler ottenere tutti gli *Stand Product* associati al *Flash Stand* con id `17`. La richiesta HTTP sarà:

```
GET /RealFoodTradeWS/api/flashstands/17/standproducts HTTP/1.1
Host: localhost:8080
```

Se il *Flash Stand* specificato ha *Stand Product* pubblicati, viene tornata una risposta HTTP con codice 200 (OK) e i dettagli degli *Stand Product* nel body.

```
HTTP/1.1 200 Ok
Content-Type: application/json
{
  "results": [
    {
      "id": "27",
      "id_index": "2993",
      "id_flashstand": "17",
      "price": "33.00"
    },
    {
      "id": "28",
      "id_index": "2470",
      "id_flashstand": "17",
      "price": "11.00"
    }
  ]
```

```
}
```

Se la richiesta viene inoltrata ad un *Flash Stand* vuoto, viene tornata una risposta HTTP con codice 404 (Not Found) e un messaggio di errore nel body.

```
HTTP/1.1 404 Not Found
Content-Type: application/json
{
  "errorCode": 404,
  "errorMessage": "There are not Flash Product for Flash Stand with id 18"
}
```

Creazione di un Flash Stand Per poter creare un *Flash Stand* l'utente deve essere autenticato come *Seller*, per questo bisogna specificare correttamente l'header `Authorization`. Nel body andrà rappresentato il *Flash Stand* da aggiungere, il quale è caratterizzato dalle coordinate geografiche.

```
POST /RealFoodTradeWS/api/flashstands HTTP/1.1
Host: localhost:8080
Content-Type: application/json
Authorization: Basic username:password
{
    "longitude":-71.708667,
    "latitude":-33.425555
}
```

Nel caso di successo, viene tornata una risposta HTTP con codice 200 (OK) riportante l'id del nuovo *Flash Stand* appena creato.

```
HTTP/1.1 200 Ok
Content-Type: application/json
{
  "id": 123
}
```

Nel caso di impossibilità nel creare la risorsa desiderata, viene tornata una risposta HTTP con codice 403 (Forbidden), con un messaggio indicante il tipo di errore, che potrà essere:

- Credenziali non valide
- Header Authorization non corretto

8.2 Front End

Capitolo 9

Sviluppi Futuri

Proponiamo delle possibili estensioni dell'applicativo sviluppato. Essendo il sistema in questione un prototipo, gli sviluppi futuri naturali saranno costituiti dall'implementazione delle funzionalità necessarie a soddisfare i requisiti definiti per il sistema completo.

9.1 Ricerche geografiche

9.2 Gestione degli ordini

9.3 Integrazione col Deep Web

9.4 OAuth

9.5 Estensione generale

Le funzionalità proposte da RFT possono risultare positive anche in contesti diversi da quelli del mercato ittico. Una prima generalizzazione può essere quella di utilizzare questo sistema in altri mercati alimentari. Se per esempio consideriamo il settore agricolo

Bibliografia

[1] L. Menichetti, "Gestione di dati georeferenziati tramite cloud computing," Master's thesis, Università degli Studi Roma Tre, 2012/2013.

[2] L. Nardini, "Mapping di query visuali su dati georeferenziati," Master's thesis, Università degli Studi Roma Tre, 2012/2013.

[3] M. Ungania, "Ricerca semantica di dati georeferenziati tramite mobile cloud computing," Master's thesis, Università degli Studi Roma Tre, 2012/2013.

[4] T. Berners-Lee, "Linked Data - Design Issue," 2006. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html

[5] F. Manola and E. Miller, "W3C RDF Primer," no. 4, 2004. [Online]. Available: http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[6] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker, "Linked open data to support content-based recommender systems," in *Proceedings of the 8th International Conference on Semantic Systems*, ser. I-SEMANTICS '12. New York, NY, USA: ACM, 2012, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2362499.2362501

[7] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, UNIVERSITY OF CALIFORNIA, IRVINE, 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm