Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Nome corso di Laurea

Esplorando i Linked Open Data con RSLT

Relatore:
Chiar.mo Prof.
Fabio Vitali

Presentata da:
Juan Manuel Felix

Sessione I
Anno Accademico 2019-2020

Indice

Elenco delle figure

Introduzione

Sono trascorsi ormai vent'anni da quando, nel numero di maggio 2001 della rivista Scientific American, è apparso un pezzo, firmato da Tim Berners-Lee, James Hendler e Ora Lassila, intitolato "The Semantic Web". Al suo interno, gli autori, evidenziavano come le il Web dell'epoca fosse ancora carente di molte funzionalità che lo stesso Berners-Lee, suo creatore, aveva immaginato per esso.

Il Web of Documents – è stato definito - era rivolto esclusivamente agli esseri umani e le informazioni in esso contenute viaggiavano ignare all'interno dei documenti. Per le macchine, d'altro canto, il documento così com'era, era un fulcro ancora troppo fragile per potervici basare sopra, guardandovi attraverso, la propria conoscenza del mondo.

È così che, sullo slancio delle intuizioni contenute nello scritto dei tre ricercatori e grazie allo sviluppo e alla diffusione di nuove tecnologie pensate per aderire alla visione di un Web semantico, la pubblicazione di dati semantici è diventata un'esigenza reale sempre più condivisa e, pian piano, un nuovo modello di architettura dell'informazione sta prendendo piede e porta il nome di Linked Open Data (LOD).

Se è vero che i LOD accessibili sul "nuovo" Web sono ormai tantissimi, è vero anche che la situazione iniziale pare ribaltata: la visione/implementazione di un Web of Data è quella di un ambiente dove le macchine possono leggere, processare e scambiarsi i singoli dati – quasi li intendessero – ma la fruizione di questi dati da parte di utenti umani è una questione più che mai spinoza.

Il proposito di questo lavoro è quello mostrare come sia possibile tra-

durre, ed integrare all'interno di un'applicazione, dei dati RDF – che è un formato della famiglia LOD – provenienti da un endpoint SPARQL, attraverso l'utilizzo di un'apposita libreria denominata **RSLT** e progettata per facilitare tale processo. RSLT offre la possibilità di definire, all'interno di un normale documento HTML, dei template di trasformazione per i dati, derivati direttamente dalla sintassi XSLT e implementati grazie al supporto di AngularJs.

Per raggiungere gli scopi prefissati si è scelto di creare due diversi servizi Web che, utilizzando RSLT, permettessero di visualizzare le informazioni contenute su due diversi triplestore che, per la tipologia di dati contenuti, sono stati giudicati di particolare interesse.

Si è scelto di dare un nome ai due prototipi :

OSCR – che permette di consultare i dati aggiornati sull'andamento dell'epidemia di Covid-19 del 2020 in Italia diffusi dal Dipartimento della Protezione Civile e pubblicati in formato RDF dal progetto Open Data Sicilia.

FOOD Semantic Browser – che consente di navigare i dati presenti sul triplestore messo a disposizione dal progetto FOod in Open Data e che contiene i dati relativi alle certificazioni ufficiali in ambito agroalimentare emanati dal Ministero delle Politiche Agricole, Alimentari e Forestali.

La dissertazione inizierà introducendo alcuni concetti chiave per la comprensione dell'ambito nel quale il progetto ha preso forma, come il Semantic Web e i Linked Open Data. Verranno, quindi, passate in rassegna alcune delle tecnologie che si occupano della visualizzazione dei dati in questo ambito. Un capito a parte verrà dedicato all'approfondimento di RSLT, la libreria pivot delle applicazioni realizzate. Solo a questo punto verranno prima introdotti, poi analizzati nel dettaglio, OSCR e FOOD Semantic Browser. Chiuderà la dissertazione una valutazione delle soluzioni trovate e una breve conclusione per spunti futuri.

Capitolo 1

Semantic Web

A inizio millennio, il World-Wide-Web (WWW) – che all'epoca aveva compiuto poco più di dieci anni - appariva al suo inventore – Tim Berners-Less - come un gigantesco archivio pensato per essere consultato esclusivamente da degli esseri umani[BHO01].

Con la diffusione dei personal computer, a partire dalla seconda metà degli anni '90, il Web aveva conosciuto un fortissimo incremento della produzione delle informazioni in esso disponibili. Una pletora di nuovi utenti – nuove comunità e quindi di nuovi ambiti di sapere – l'aveva invaso e aveva iniziato a pubblicarvi documenti – formalmente detti risorse -, raggiungibili per via di un indirizzo – chiamato URI - che li identificava in maniera univoca.

L'aumento esponenziale della conoscenza disponibile, però, non era stato accompagnato da una organizzazione dei contenuti pedissequa e per i computer, le informazioni contenute nei documenti, non avevano nessun significato – nessuna *semantica*. Eppure, quando nel 1989, Berners-Lee aveva illustrato per la prima volta il suo progetto ai laboratori del CERN di Ginevra, aveva posto già da subito l'enfasi su tutta una serie di informazioni che avrebbero corredato una risorsa, permettendo di esprimere caratteristiche come chi ne è l'autore, l'argomento di cui tratta, a quali altre risorse è collegata, ecc. Dare peso a questi metadati – dati sui dati –, trovando la maniera migliore per esprimerli ed utilizzarli, sarebbe stata la strada da seguire per dotare il Web

un'aura – per così dire - cogitante.

Va detto che i documenti avevano formalizzato la propria struttura sulla Rete sin dall'inizio, attraverso l'adozione del linguaggio di mark-up HTML e il browser, dal canto suo, sfruttava questa organizzazione esclusivamente per scopi presentazionali – per l'impaginazione – e per cercare collegamenti verso altri documenti – attraverso i link ipertestuali. L'individuazione e l'elaborazione dei contenuti, all'interno di un documento votato alla marcatura tipografica, restava comunque molto faticosa: un motore di ricerca non poteva appoggiarsi su di un qualche tipo di definizione (semantica) per identificare un concetto ma era costretto a praticare una ricerca testuale (sintattica) che quasi mai è agevole e priva di ambiguità. Anche con l'introduzione di metodi espliciti per la definizione di metadati all'interno dei documenti HTML – meta tag - non aveva dato i frutti sperati. In questo modo, le potenzialità del Web – almeno agli occhi del suo fondatore – rimanevano in larga parte inespresse.

C'è voluto tutto l'impegno del W3C – il World-Wild-Web Consortium - affinché delle nuove pratiche, capaci di aggiungere significato alla miriade di informazioni presenti sul Web, vedessero la luce. Il consorzio – fondato nel 1994 da un Bersners-Lee che da poco aveva lasciato il CERN per il MIT portandosi appresso la propria creatura – ha contribuito in maniera fondamentale allo sviluppo e alla diffusione di tecnologie di programmazione per supportare la realizzazione del cosiddetto Semantic Web: la visione, cioè, di un Web pensato per essere fruito anche e soprattutto dalle macchine – machine processable -, le quali siano messe in grado, non semplicemente di cercare le informazioni in maniera più efficiente ma di "ragionare" sui dati stessi, producendo nuove informazioni, dette inferenze. Le tecnologie sulle quali il Semantic Web si fonda sono:

1. un modello sintattico per la rappresentazione dei dati in maniera strutturata: RDF

2. un meccanismo formale che fornisca un lessico per caratterizzarli e metterli in relazione: OWL

e verranno presentate nelle prossime sezioni.

1.1 RDF - Resource Description Framework

Un passo fondamentale verso la gestione automatizzata dell'informazione veicolata dal Web è stata l'individuazione di specifiche convenzioni sintattiche, semantiche e strutturali per la rappresentazione e la condivisione dei dati. Il *Resource Description Framework* (RDF) è lo strumento base per la codifica, lo scambio e il riutilizzo dei dati sul WWW, sviluppato dal W3C. È definito da due componenti fondamentali:

- RDF Model and Syntax: definisce un modello per i dati e la sua codifica XML;
- RDF Schema: permette di definire specifici vocabolari per i metadati.

Il modello dei dati – data model – RDF è basato su tre tipi di oggetti:

1. Risorse – Qualunque cosa descritta con RDF prende il nome di risorsa ed è individuabile da un URI.
2. Proprietà – Una proprietà descrive un aspetto specifico di una risorsa, come una sua caratteristica o una relazione con altre risorse. Ogni proprietà ha un significato specifico, un insieme di valori ammissibili, i tipi di risorse a cui può essere associata, e le sue relazioni con altre proprietà.
3. Statements – Una risorsa viene associata ad un valore, per mezzo di una proprietà, all'interno di un enunciato (statement). Uno statement è quindi una tripla composta da soggetto (la risorsa), predicato (una proprietà) e un oggetto che può essere un'altra risorsa oppure una stringa di testo.

RDF rappresenta, quindi, una maniera semplice e flessibile di scomporre la complessità della conoscenza in elementi atomici che possono facilimente essere rappresentati mediante grafi nei quali le risorse vengono identificate come

nodi, le proprietà come archi etichettati e orientati, e i valori come rettangoli, se corrispondenti a sequenze di caratteri, o come altri nodi, se corrispondenti ad altre risorse. Una rappresentazione grafica della composizione di un enunciato RDF è quello definito qui sotto:

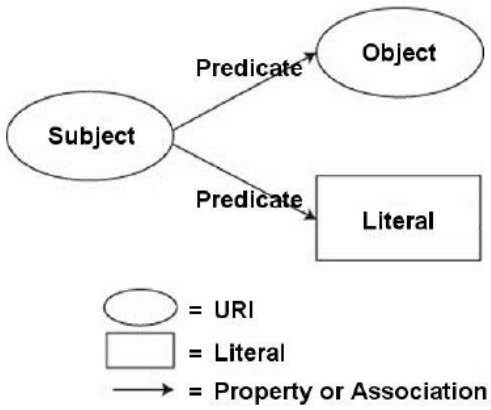

Figura 1.1: rappresentazione grafica di due statement RDF aventi lo stesso soggetto ma oggetti diversi

Attraverso questo schema, è possibile organizzare la conoscenza nel seguente modo:

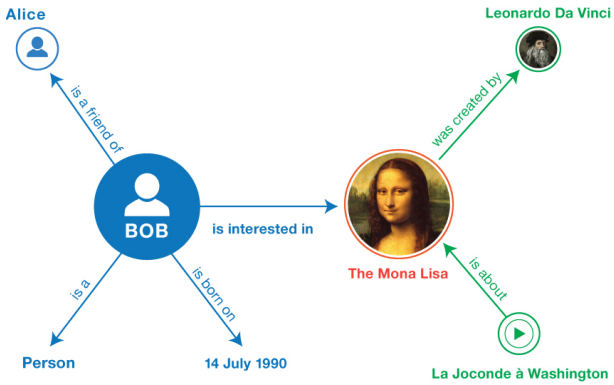

Figura 1.2: esempio di grafo RDF

Mentre il data model RDF permette di strutturare i dati e descrivere le risorse attraverso l'enunciazione di loro proprietà e caratteristiche, RDF Schema (RDFS) è quella componente che permette di specificare un vocabolario per poter esprimere proprietà - come "è di tipo" "è un sottogenere di" - all'interno di un determinato domino del sapere. Con RDFS si definiscono il significato, le caratteristiche e le relazioni di uno specifico insieme di proprietà, compresi i vincoli sul dominio e sui valori delle singole proprietà.

Inoltre, implementando il concetto di classe e sottoclasse, RDFS consente di stabilire una gerarchia tra concetti (tassonomia) - e di conseguenza tra risorse - che sta alla base della cosidetta inferenziazione: ossia la possibilità per le macchine di "dedurre" nuovi statements percorrendo questa gerarchia.

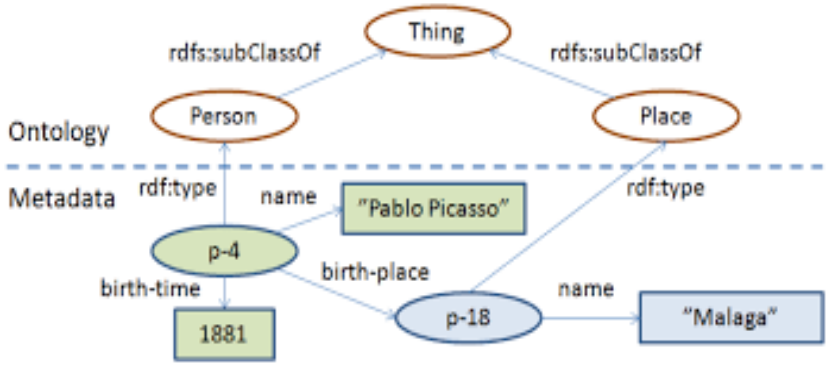

Figura 1.3: esempio di gerarchia creata utilizzando RDFS

RDFS definisce un insieme ridotto di proprietà che, da solo, non è sufficiente a fornire un vocabolario completo per trattare i vari campi della conoscenza. A questo scopo è stato sviluppato un linguaggio decisamente più espressivo: OWL.

1.2 OWL – Web Ontology Language

Il Web Ontology Language (OWL) è un linguaggio pensato per la definizione di ontologie per il Web. In informatica, quando si parla di ontologia, si

fa riferimento a una descrizione formale, fatta di regole interpretabili da una macchina, di un particolare domininio della conoscenza.

La produzione di un'ontologia è un passo delicato e importante in quanto la descrizione prodotta dovrebbe rispecchiare una concettualizzazione del dominio di riferimento quanto più possibile condivisa dai futuri utilizzatori.

I componenti principali di un'ontologia sono:

- Concetti, chiamati anche Classi o Tipi - Rappresentano entità astratte, con caratteristiche specifiche, che possono essere istanziati come Individui (o Istanze);
- Individui - Sono gli oggetti concreti della descrizione e sono detti istanze di una Classe.
- Relazioni - Sono predicati binari che si applicano per asserire qualcosa su degli Individui o dei Concetti.

Un'ontologia, in definitiva, crea uno strumento ancora più potente di RDFS per poter "descrivere" la conoscenza del mondo reale ad una macchina.

All'interno di questo elaborato incontreremo le ontologie di riferimento dei due dataset presi in esame ma, prima, verrà presentato l'utimo strumento chiave del paradigma semantico: il linguaggio per la manipolazione dei dati.

1.3 SPARQL

Una volta che sono stati definiti il modello di rappresentazione dei dati e il linguaggio per creare vocabolari, è arrivato il momento di presentare un linguaggio per selezionare e recuperare questi dati dai documenti e dagli store in cui sono custoditi.

SPARQL Protocol and Rdf Query Language – o semplicemente SPARQL – è il linguaggio standard del W3C per l'integrazione di dati RDF.

Basato sul riconoscimento di patter su grafo, SPARQL permette di effetturare ricerche anche molto complesse mediante l'utilizzo, per esempio, di

filtri, ordinatori, limitatori e pattern opzionali. Le query SPARQL seguono la struttra SELECT-FROM-WHERE come quelle del più noto linguaggio per database relazionali SQL. In particolare:

- SELECT – specifica la proiezione, ovvero i dati da ricercare, il loro numero e il loro ordine;
- FROM – opzionale, è utilizzato per specificare la sorgente in cui cercare;
- WHERE – impone dei vincoli, attraverso pattern o verifiche booleane, che devono essere soddisfati dalla soluzione.

Una semplice interrogazione – query – SPARQL assume la seguente sintassi:

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
SELECT *
WHERE {
    ?person foaf:name ?name .
    ?person foaf:mbox ?email .
}
```

Figura 1.4: esempio di query con SPARQL

Servendosi del supporto di uno SPARQL-endpoint, cioè di un database che consente l'accesso a dati RDF per mezzo del protocollo standard HTTP e di query SPARQL, è possibile richiedere e utilizzare informazioni altamente strutturate e interconnesse, come descritto fin'ora.

Le tecnologie che si occupano di trattare questi dati, per essere visualizzati una volta recuperati, verranno illustrate nel capitolo 2. Alla fine di questo capitolo introdurremo alcuni concetti, relativi alla pubblicazione e alla diffusione di dati, che si sono evoluti assieme agli standard del Semantic Web.

1.4 Linked Open Data

Quotidianamente, soggetti pubblici e privati producono una grande quantità di dati che rischiano di rimanere "intrappolati" nell'anonimato. La pubblicazione e lo scambio di questie informazioni sul Web è un tema assai importante nella società contemporanea.

Grazie alle tecnologie messe a disposizione dal Semantic Web, i dati possono essere pubblicati in maniera accessibile e strutturata in modo tale da essere messi in relazione tra loro e con altri altri dati, altrettanto strutturati ma provenienti da fonti diverse. In tale modo si facilita la contestualizzazione di queste informazioni ma soprattuto è possibile, per le macchine, processarle ed estrarre conoscenza aggiuntiva – inferenze - da esse.

Nel 2006, Berners-Lee ha tracciato quattro regole per la pubblicazione di dati sul Web[1]:

1. Utilizzare gli URI per identificare le risorse
2. Utilizzare URI HTTP in modo tale che sia possibile accedere alle risorse
3. Quando viene richiesto una risorsa, fornire inormazioni utili per mezzo di uno standard riconosciuto (RDF, SPARQL)
4. Includere link verso altre risorse, in modo che sia possibile la scoperta di nuove informazioni

I dati che seguono questo paradigma sono detti Linked Data e mirano alla trasformazione del Web in un ambiente completamente accessibile e processabile dalle macchine: il Web of Data.

Nel 2010, ai precenti punti si aggiunge una scala fatta di cinque gradini per misurare la conformità, dei dati pubblicati sul Web, ai precetti LOD. La scala procede da una fino a cinque stelle, assegnando una stella aggiuntiva per ogni gradino scalato.

[1] https://www.w3.org/DesignIssues/LinkedData

Partendo dal gradino più basso, un dato deve essere:

1. Disponibile sul Web in qualunque formato ma con licenza libera (es. jpg);
2. Disponibile come dato strutturato leggibile da una macchina (es. xls);
3. Come sopra ma con formato non proprietario (es. xml);
4. Come sopra ma utilizza standard W3C come RDF e SPARQL (es. rdf);
5. Come sopra, in più è collegato con altri dati provenienti da altre fonti (Linked Open Data).

I dati, per essere definiti Open Data, possono trovarsi su qualunque gradino di questa scaletta. La specificazione Open fa riferimento più ad un aspetto di policy, nella pubblicazione e nella diffusione di queste informazioni, che dovrebbe privilegiare formati non-proprietari e licenze che non ne limitino il riutilizzo e la diffusione.

Prima di addentrarci maggiormente nel mondo dei Linked Open Data, sarà utile illustrare le tecnologie che stanno alla base della loro visualizzazione.

Capitolo 2

Visualizzazione RDF

Con la diffusione delle pratiche per la formattazione dei dati ispirate dal Semantic Web, la pubblicazione e la consultazione di questi dati è diventata un'esigenza più che mai centrale nella società moderna. Come viene spiegato qui [PAL14], automatizzare la pubblicazione di informazioni è un approccio comune tra i professionisti di molti settori per esporre facilmente enormi quantità di file e documenti al pubblico consumo. Allo stesso tempo, però, la vastità delle tematiche che questi dati investono, rende il problema della loro visualizzazione automatizzata un ambito più che mai centrale per incentivare la loro diffusione.

Nel caso specifico dei dati RDF, il modello soggetto-predicato-oggetto degli statement, ne ha da sempre facilitato la rappresentazione per mezzo di grafi, come con LODlive[1]. Altro approccio largamente utilizzato, segnalato sempre da [PAL14] è quello della visualizzazione in formato tabulare, come possiamo vedere, ad esempio, in Pubby[2].

Queste tecniche sono molto utili per comprendere la struttura dei dati ma, dato che trattano campi della conoscenza molto eterogenei tra loro, sono pensate per essere molto generiche nella loro resa visuale e non offrono la versatilità necessaria ad essere integrate all'interno di applicazioni Web

[1] http://lodlive.it/

[2] http://wifo5-03.informatik.uni-mannheim.de/pubby/

indipendenti.

Esistono, anche, strumenti pensati per la visualizzazione di informazioni specifiche di un dato ambito della conoscenza. Alcuni esempi citati da [PAL14] sono: FoaF Explorer, map4rdf, LinkedGeoData browser. Ma anche queste soluzioni non offrono grandi margini di integrabilità all'interno di altre applicazioni.

Un software che si occupi di tradurre dati RDF per un pubblico ampio dovrebbe permettere l'integrazione di questi dati all'interno di applicazioni Web in maniera da poter definire delle visualizzazioni personalizzate per la propria utenza.

Esistono numerosi tool [CFG17] [BLH08] [RP15] [BLP05] [SKJ12], scritti in vari linguaggi, pensati appostitamente per indirizzare queste esigenze e prima di proseguire ne passeremo in rassegna qualcuno.

STTL [3] è un linguaggio che si basa sull'utilizzo di template scritti estendendo la sintassi SPARQL per trasformare gli RDF. Una trasformazione espressa in STTL descrive le regole per trasformare un grafico sorgente in un risultato testuale in un qualche altro formato per i dati semantici (XML, N-Triples, Turtle, RDFa, TriG, N-Quads, JSON-LD). In qualche modo STTL è per RDF ciò che XSLT è per XML [dal sito]. STTL offre grande libertà nella definizione dei template [Quadrelli] e, quindi, delle possibili trasformazioni per i dati oltre alla possibilità di utilizzare documenti nel formato preferito. Ciononostante, le trasformazioni STTL generano dei contenuti statici: non è possibile la generazione di un template all'interno di un altro. Questo comportamento non lo rende la scelta ideale nello sviluppo di applicazioni interattive .

XSLT+SPARQL [4] nasce come estensione di XSLT, il linguaggio di trasformazione per XML, per l'esecuzione di query SPARQL su grafi RDF e permette di manipolarne i risultati per mezzo del linguaggio XPATH,

[3] https://ns.inria.fr/sparql-template/

[4] https://berrueta.github.io/research/xsltsparql

linguaggio di selezione per XML, pensato per operare su alberi. Con XSLT+SPARQL è possibile scrivere trasformazioni completamente dichiarative che accedono ai dati espressi in RDF, indipendentemente dalla loro serializzazione (RDF/XML, N3, etc.). Inoltre, per mezzo di alcune funzioni, è possibile eseguire query verso endpoint SPARQL 1.0. La versione 1.1 del linguaggio di interrogazione non è supportata in quanto la gestione del formato di risposta – json – non è processabile attraverso XSLT [BLH08].

RSLT, la libreria adottata per lo sviluppo di questo progetto, fa parte di una famiglia di soluzioni che implementano strategie di tipo dichiarativo – in contrapposizione a quelle di tipo procedurale – per la trasformazione dei dati. Questi software, pur traendo tutti spunto dalle tecnologie della famiglia XML, si differenziano tra loro per il tipo di approccio adottato:

- in alcuni casi si è tentato di aggiungere il supporto per RDF e SPARQL a linguaggi di trasformazione preesistenti (XSPARQL e XSLT+SPARQL);
- in altri, l'integrazione di funzionalità per la definizione di template dichiarativi, all'interno dei succitati standard per la gestione dei dati semantici, è parsa la strategia migliore (STTL e Fresnel);
- in altri casi ancora, invece, si è optato per definire nuovi paradigmi che rendessero il processo di trasformazione più agevole e meno vincolato a tecnologie pensate per altri scopi, a vantaggio di un maggior potere espressivo dei linguaggi così ottenuti (OWL-PL e RSLT).

RSLT [PV15], nello specifico, è una libreria Javascript giunta alla sua versione 1.1 da qualche anno [QUA15] ma finora raramente integrata all'interno di applicazioni reali: [BCD15] [MAS15]. Per questa ragione, questa esposizione si pone come scopo quello di mettere alla prova RSLT attraverso la realizzazione di due prototipi di servizi per la visualizzazione dei dati contenuti in triplestore preselezionati. Lo sviluppo di applicazioni del genere, fortemente condizionate dai dati trattati, obbliga ad un accurato studio preliminare del

contenuto dei dataset ma offre la possibilità di fornire, all'utente finale, un ambiente confortevole per la ricerca e la consultazione dei dati.

Nel prossimo capitolo verrà illustrata la libreria nel dettaglio.

Capitolo 3

RSLT

RDF Stylesheet Language Template (RSLT) è una libreria, scritta in JavaScript con il supporto del framework Angularjs, che definisce un meccanismo per la visualizzazione di dati RDF basato sulla definizione di template dichiarativi associati alle risorse restituite dalle query SPARQL, per la creazione di pagine HTML.

Gli obiettivi che questa libreria si propone di soddisfare sono i seguenti:

- Semplicità di integrazione all'interno di applicazioni Web
- Semplicità nella rappresentazione tanto dei semplici statement RDF quanto di entità ontologicamente complesse
- Presentazione degli IRI in maniera "leggibile"
- Costruzione di rappresentazioni complesse a partire dalla combinazione di più rappresentazioni semplici

Come accennato nell'introduzione, RSLT è un adattamento del linguaggio di templating per XML: XSLT, al modello RDF. I costrutti di XSLT sono implementati all'interno di RSLT come delle direttive di AngularJs.

La definizione delle query avviene all'interno della direttiva applytemplate – figlia dell'apply-template di XSLT - con una notazione simile a quella SPARQL:

```
<applytemplate select="?person foaf:name 'Tim Berners-Lee'">
```

Tramite questa notazione è, però, possibile - definendo la variabile all'interno della query con uno o due punti interrogativi – distinguere, rispettivamente, tra la richiesta di un insieme di statement che soddisfino il pattern della query e la richiesta di entità complete, che è l'insieme di tutti gli statement che fanno riferimento ad ogni soggetto che soddisfa il pattern, suddiviso per soggetti. Per fare un esempio: l'interrogazione mostrata sopra, darà per risposta tutte le triple in cui è presente il predicato foaf:name e che hanno come oggetto il testo 'Tim'. Aggiungendo, invece, un secondo punto interrogativo alla variabile nel pattern nel seguente modo:

```
<applytemplate select="??person foaf:name 'Tim Berners-Lee'">
```

La query restituirà anche tutte le triple che condividono il soggetto con quelle già trovate. Attraverso questa seconda opzione è, dunque, possibile ricavare tutte le proprietà per un determinato soggetto (o gruppo di soggetti) e averle a disposizione, in ogni momento, all'interno del template.

Oltre ad applytemplate, RSLT implementa delle altre direttive ereditate da XSLT, che sono state utilizzate all'interno di questo progetto:

applytemplate – che viene utilizzata per invocare un template, specificato attraverso l'attributo name;

foreach - che permette di applicare comodamente un sotto-pattern per colezioni all'interno di un template;

template - che definisce un template che viene in invocato direttamente, da calltemplate, o indirettamente attraverso l'attributo match che permette di attivare un template ogni qual volta un'entità soddisfa l'espressione in esso contenuta.

In base al contenuto dell'attributo match, possiamo definire tre tipi di template:

1. associati a statement RDF:

```
<template match="?person foaf:name ‘Tim Berners-Lee’">...</template>
<template match="?person foaf:name ?string">...</template>
```

2. associati a risorse specifiche derefenziabili attraverso un URI:

```
<template match="http://example.com/person/Tim_Berners-Lee">...<template>
```

3. associati ai tipi delle risorse restituite dalle query:

```
<template match="?Tim -> foaf:Person>...</template>
```

All'interno di un template tutte le variabili vengono associate alle relative entità di RSLT e le proprietà ad esse legate sono disponibili per essere mostrate attraverso il supporto di Angularjs e degli strumenti di markup preferiti:

```
<template match="?person -> foaf:Person>
     <h1>Hi! My name is {{person[‘foaf:name’]}}.</h1>
     <p>My friends are:</p>
     <ul>
     <applytemplate select="?person foaf:knows ??friend>
     </ul>
</template>
<template match="?friend -> foaf:Person">
     <li>{{friend[‘foaf:name’]}}</li>
</template>
```

Si noti, nell'esempio sopra, come, per visualizzare i valori delle proprietà associate ad una risorsa che compare come oggetto di uno statement – in questo caso la proprietà foaf:name della risorsa friend -, occora prima recuperare questi valori tramite l'invocazione di un template aggiuntivo.

Quando si eseguono applytemplate e foreach, RSLT controlla prima se le entità richieste sono già presenti all'interno dell'applicazione. Se non vengono trovate, allora l'algoritmo procede nell'invio della query verso l'endpoint. Per evitare un'invio eccessivo di richieste, ogni volta che si incontra una nuova entità, RSLT mette a disposizione l'attributo preload che permette di specificare una serie di pattern aggiuntivi per la query SPARQL:

```
<applytemplate select="??person foaf:name ‘Tim Berners-Lee."
    preload="?person foaf:knows ??friend">
    ...
</applytemplate>
```

Oltre le direttive di ispirazione XSLT, RSLT definisce anche:

rslt - definibile una sola volta all'interno del documento, funge da contenitore dei template. Permette di specificare l'endpoint da utilizzare per le query tramite l'attributo tripletore.

prefix - utilizzabile all'interno di rslt ogni qual volta si voglia definire un prefisso per le query SPARQL e per i nomi delle proprietà delle entità RSLT.

Data la differenza strutturale tra un documento XML, che presenta sempre un elemento root – radice -, e un grafo RDF, che non ha un inizio, subito dopo la definizione dei prefissi compare, nell'elemento rslt, la definizione di un template di start, dal quale si propaga l'attivazione di tutti gli altri.

Essendo RSLT utilizzabile come modulo di Angularjs, è possibile sfruttare tutte le potenzialità del noto framework per realizzare facilimente applicazioni Web complesse come quelle che verranno illustrate nei prossimi capitoli.

Capitolo 4

Linked Open Data - Casi di studio

In questo capitolo verranno presentati quelli che definiremo "casi di studio", ovvero due progetti volti a formalizzare secondo i precetti dei LOD alcuni dati rilasciati in formati diversi.

Nello scenario degli Open Data si inserisce una gamma di dati, sempre più vasta, messa a disposizione da Governi e Enti Pubblici. Questi dati riguardano i vari ambiti che entrano in gioco nella gestione della Cosa Pubblica: si va dai dati demografici a quelli sulla produzione agricola, passando per quelli geografici. La "liberazione" di informazioni di pubblico interesse è molto importante sia per quanto riguarda il tema della trasparenza della Pubblica Amministrazione sia per la possibilità, che questi dati offrono, di essere riutilizzati all'interno di servizi costruiti ad-hoc per una loro fruizione da parte di un pubblico quanto più vasto. Purtroppo però, quasi sempre queste informazioni vengono ancora pubblicate utilizzando formati che non ne facilitano il riuso e la diffusione. Per fortuna, esistono diverse iniziative e gruppi di lavoro, istituzionali o meno, che si occupano della traduzione e (ri)pubblicazione di questi dati secondo i canoni dei Linked Data.

Per omaggiare il lavoro svolto da queste iniziative si è deciso di selezionarne due in particolare per provare a definire dei servizi di che, facendo uso

di RSLT, creassero un ambiente ideale per la consultazione dei dati offerti. Le iniziative prese in esame sono:

COVID-19 Sicilia che pubblica i dati rilasciati quotidianamente dal Dipartimento della Protezione Civile e dalla regione Sicilia sull'attuale pandemia di COVID-19 in Italia;

FOod in Open Data che mette a disposizione modelli di standardizzazione e ontologie di riferimento per la rappresentazione dei contenuti dei disciplinari relativi ai marchi di qualità del prodotti agroalimentari italiani.

Di seguito presenteremo più nel dettaglio i due progetti.

4.1 COVID-19 Sicilia

Covid-19 Sicilia[1] è un progetto di Open Data Sicilia[2], un gruppo che ha come scopo quello di diffondere e far conoscere la cultura dell'Open Government e le prassi degli Open Data.

Sin dall'inizio dello stato di emergenza dichiarato dal Governo Italiano il 31 Gennaio 2020 e dovuto alla diffusione del nuovo Coronavirus, il Dipartimento della Protezione Civile ha attivato attraverso il proprio portale Web una rete per il monitoraggio dell'andamento della pandemia nel paese italiano[3]. Giornalmente, viene presentata una scheda con i dati aggiornati a partire da quelli che vengono inviati dalle singole Regioni al Ministero della Salute e all'Istituto Superiore di Sanità. I dati sono visualizzabili in maniera interattiva all'interno di una dashboard[4] che presenta mappe e grafici e sono, inoltre, disponibili per il download in formato CSV e PDF.

[1] https://github.com/opendatasicilia/COVID-19_Sicilia

[2] http://opendatasicilia.it/

[3] http://www.protezionecivile.gov.it/attivita-rischi/rischio-sanitario/emergenze/coronavirus

[4] http://opendatadpc.maps.arcgis.com/apps/opsdashboard/index.html#/b0c68bce2cce478eaac82fe38d4138b1

L'esigenza, esplicitata dal gruppo in una lettera aperta[5], di un dataset con i dati relativi all'andamento dei casi in Sicilia nasce dalla mancanza di tutte le informazioni sulle singole province, all'interno dei bollettini giornalieri del Dipartimento. La regione Sicilia – sostengono gli attivisti - comunica quotidianamente queste informazioni in formato semplicemente testuale o per immagini ma finora non era ancora stata predisposta una loro traduzione in Linked Data e una loro integrazione nei repository della Protezione Civile.

Nel triplestore messo a disposizione da Covid-19 Sicilia[6], le entità di riferimento che contengono i dati sono modellate sull'ontologia RDF Data Cube Vocabulary[7], che offre il vocabolario adatto per dati di natura statistica.

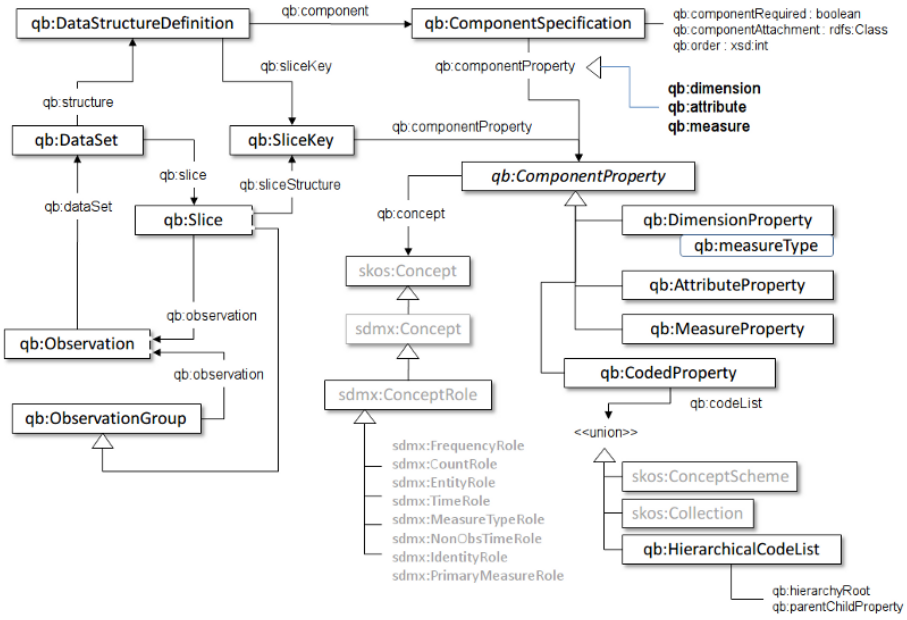

Figura 4.1: diagramma delle classi di RDF Data Qube Vocabulary

[5]http://opendatasicilia.it/2020/03/23/lettera-aperta-alla-regione-siciliana-per-la-pubblicazione-in-formato-machine-readable-dei-dati-sulla-covid19/

[6]http://45.62.242.205:8890/sparql

[7]https://www.w3.org/TR/vocab-data-cube/

Seguendo questo modello, ogni aggiornamento rilasciato dal Dipartimento della Protezione Civile è definito come un'Osservazione (rdf:type qb:Observation) e per ciascuna di esse sono disponibili le seguenti proprietà:

- Dataset di riferimento

Le dimensioni spaziali e temporali:

- Data dell'osservazione
- Area geografica di riferimento

Una serie di valori di riferimento per l'andamento del contagio:

- Ricoverati con sintomi
- Ricoverati in terapia intensiva
- Totale degli ospedalizzati
- Attuali in isolamento domiciliare
- Totale dei positivi
- Variazione totale dei positivi
- Nuovi Positivi
- Dimessi guariti
- Deceduti
- Tamponi effettuati
- Totale dei casi

4.2 FOod in Open Data (FOOD)

Il progetto FOod in Open Data (FOOD)[8] nasce dalla collaborazione tra l'Istituto di Scienze e Tecnologie Cognitive del CNR – presso il Laboratori di Tecnologie Semantiche (STLab)[9] - e l'Agenzia per L'Italia Digitale per standardizzare i dati emanati dal Ministero delle Politiche Agricole, Ambientali e Forestali e disponibili in formato PDF[10].

Le finalità del progetto – si legge sul sito - riguardano la creazione di onotologie per la rappresentazione dei contenuti dei disciplinari relativi ai marchi di qualità dei prodotti agroalimentari e la produzione, a partire dalle ontologie create, di dati aperti espressi secondo il paradigma Linked Data.

Le ontologie definite riguardano i marchi di qualità DOP e IGP e 20 diversi tipi di prodotto. Per la definizione delle materie prime dei prodotti DOP e IGP è stata impiegata l'ontologia AGROVOC[11].

Le ontologie, come tutti i dati standardizzati, sono liberamente accessibili all'indirizzo di un apposito triplestore[12].

4.3 Considerazioni

Grazie ad entrambi questi progetti, che nascono da contesti diversi, i dati precedentemente messi a disposizione dalle istituzioni hanno fatto un salto di qualità nella scala LOD che abbiamo visto prima: il formato RDF, infatti, è lo standard consigliato al quarto gradino, mentre PDF e CSV raggiungono, rispettivamente, il primo e il terzo gradino di questa scala. Nel prossimo capitolo presenteremo i prototipi implementati.

[8] http://etna.istc.cnr.it/food/
[9] https://www.istc.cnr.it/it/group/stlab
[10] https://www.politicheagricole.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/309
[11] http://aims.fao.org/vest-registry/vocabularies/agrovoc
[12] http://etna.istc.cnr.it/food-sparql/

Capitolo 5

OSCR

Open Sicilia Covid-19 RSLT application (OSCR) è il nome del servizio, realizzato all'interno di questo progetto di tesi, per permettere la consulatazione dei dati, connessi all'epidemia di Covid-19 in Italia, forniti dal Dipartimento della Protezione Civile e messi a disposizione dal gruppo Open Data Sicilia su un apposito SPARQL-endpoint.

I dati in questione, data la loro natura principalmente numerica, sono presentati, all'interno di documenti HTML, per mezzo di grafici che ne facilitano la lettura e l'analisi e ne scandiscono l'andamento col trascorrere del tempo.

Questo tipo di servizio è simile a quello che si può osservare andando sull'apposita dashboard presente sul sito del Dipartimento della Protezione Civile. I dati a partire dai quali i due servizi sono realizzati sono però differenti, in quanto la suddetta dashboard utilizza il formato CSV, mentre il servizio qui proposto recupera gli RDF da un triplestore.

5.1 Panoramica

Dall'home page di OSCR è possibile selezionare un dataset dal quale recuperare le informazioni, tramite un comodo menù a tendina. Le prime tre opzioni - nazionale, regionale e provinciale - raggruppano i dati ufficiali forniti dal Dipartimento della Protezione Civile. Nel quarto sono presenti i dati completi per la regione Sicilia forniti dalla Regione ma non presenti per intero sugli altri dataset.

Selezionato il dataset, e l'eventuale regione o provincia, si richiede al servizio il recupero dei dati relativi che, una volta disponibili, saranno facilimente consultabili all'interno del servizio per mezzo di un grafico lineare che espone l'andamento di vari indicatori (nuovi contagi, ospitalizzati, decessi, guarigioni,...) nel tempo.

Ad esempio, selezionando il dataset nazionale, si ottiene una visualizzazione del genere:

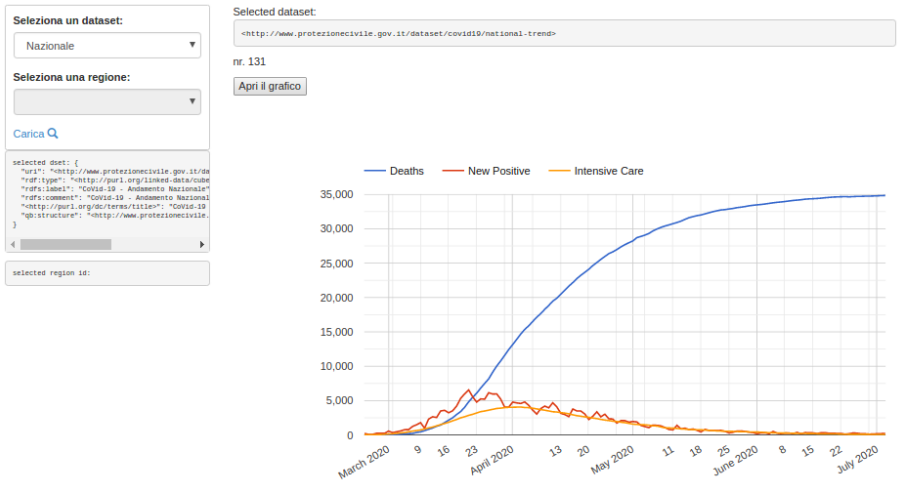

Figura 5.1: visualizzazione di dati relativi all'andamento nazionale con OSCR

Oltre al menù di selezione, a sinistra viene mostrata, in formattazione json, la risorsa dataset selezionata. Mentre assieme al grafico vengono

mostrati il numero di osservazioni recuperate e, anche per queste, c'è la possibilità di visualizzazione in formattazione json.

Questa, invece, è la visualizzazione che si ottiene selezionando il dataset regionale e una regione dall'elenco:

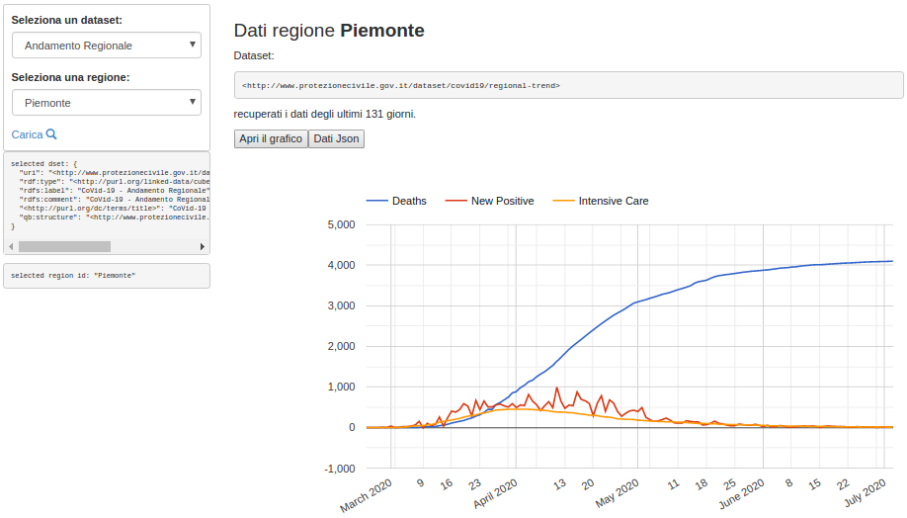

Figura 5.2: visualizzazione di dati relativi alla regione Piemonte

5.2 Dettagli Implementativi

Per la realizzazione di OSCR a supporto delle funzionalità di RSLT, che nella versione 1.1 integra il modulo *ng-Route* di Angularjs, è stata affiancata una libreria per il disegno di grafici: angular-google-chart[1]. Questa libreria nasce come ridefinizione della ben più nota Google Charts tramite direttive Angularjs.

Di seguito verranno illustrati i template e le funzioni Javascipt utilizzati nell'implementazione di OSCR.

[1]https://github.com/angular-google-chart/angular-google-chart

Il primo template eseguito – invocato dal template di start – è quello che si occupa di recuperare i nomi di dei dataset e di visualizzarli all'interno di un menù a tendina contenuto in una form:

```
<template name="dsSelectionMenu">
  <foreach select="??datasets rdf:type qb:Dataset" collected>
    <form class="searchBoxes ">
      <label for="sel1">Seleziona un dataset:</label>
        <select class = "form-control"
         id = "sel1"
         ng-model = "selectedDataset"
         ng-options = "dset['rdfs:label'] for dset in datasets"
         ng-init = "selectedDataset =
                    selectedDataset || datasets[0]">
        </select>
   </form>
  </foreach>
</template>
```

Il template 'dsSelectionMenu' fa utilizzo della direttiva foreach con l'attributo collected specificato che fa sì che il resto del template venga applicato una volta soltanto. In questo modo è possibile creare il menù a tendina della form ciclando sulla collezione di entità legate alle varibili tramite la direttiva ng-option di Angularjs. Viene specificato anche un modello, tramite ng-model, per poter sfruttare i meccanismi di binding delle variabili Angularjs in modo da avere a disposizione nel controller dell'applicazione il valore selezionato nel menù.

L'elemento anchor a fine template specifica, nell'attributo href, il pattern di formattazione con i modelli contenenti i valori selezionali che saranno intercettati dal route provider del modulo ng-Route, per definire gli indirizzi locali all'applicazione. Le opzioni di routing specificate dal provider non definiscono template ma solo un controller che passa i parametri ottenuti – che contengono i valori attuali delle selezioni della form - ad una funzione, defini-

ta nel controller principale, che prima aggiorna il valore delle variabili interne al controller sincronizzate sui model e poi fa una chiamata calltemplate per attivare il template relativo al dataset di cui caricare e mostrare i dati.

Questi template non si differenziano soltanto per le ovvie soluzioni di layout adottate ma hanno alla base due tipi di query differenti. I template per i dati nazionali recupera le osservazioni in questo modo:

```
<foreach
  select="??osservazioni qb:dataset {{selectedDataset}}"
  collected>
```

dove selectedDataset contiene il valore dell'URI del dataset selezionato, aggiornato poco prima nel controller.

I template per i dati regionali e provinciali sono obbligati a fare qualche acrobazia in più:

```
<foreach
  select = '??osservazioni sdmx-dimension:refArea ?area'
  preload='??area l0:name ?areaName.
  FILTER(regex(?areaName,"^{{selectedRegion}}","i"))'
  collected>
```

dove selectedRegion contiene il valore della label della regione selezionata, aggiornata dal controller. L'utilizzo di una query piuttosto complessa si deve al fatto che le entità regioni e province sono definite con una ontologia che utilizza il protocollo HTTPS mentre il parser interno a RSLT non riconosce come validi gli URI HTTPS se esplicitati nei pattern di una query.

I template per la visualizzazione dei dati inseriscono, infine, un bottone per visualizzare un grafico, che quando viene cliccato compie nell'ordine le seguenti azioni:

1. chiama calltemplate per attivare il template 'drawChartTemplate' che – utilizzando la direttiva google-chart di angular-google-chart – definisce un div in modalità nascosta e lo binda tramite l'attributo chart a una

variabile, specificata all'interno della funzione che costruirà il grafico, che conterrà i dati da visualizzare nel grafico;

2. attiva la visualizzazione per il div appena creato in modalità nascosta

3. chiama la funzione drawChart che si occupa di costruire il grafico recuperando i dati necessari dal servizio triplestore, offerto da RSLT, che registra tutte le entità caricare all'interno dell'applicazione e le rende disponibili all'interno dell'applicativo. Una volta che i dati sono pronti vengono passati a un prototipo di grafico definito come oggetto Javascript che infine viene assegnato alla variabile specificata nell'attributo chart del div creato dal template drawChartTemplate.

A questo punto il grafico con i dati raccolti viene mostrato all'interno della pagina.

Capitolo 6

FOOD-SB

FOOD Semantic Browser (FOOD-SB) è il nome dell'altro servizio realizzato, sempre con il supporto di RSLT 1.1, per permettere una rapida e compatta visualizzazione dei dati messi a disposizione dal progetto FOOD[1] nell'ambito delle certificazioni agroalimentari.

Con FOOD-SB è possibile consulare l'elenco dei prodotti italiani certificati e, per ciascun prodotto, visualizzare una scheda che ne elenca le caratteristiche come la denominazione ufficiale, il luogo di produzione, la composizione e le caratteristiche organolettiche.

Questo tipo di servizio è simile a quello che si può osservare andando sul portare Dop Igp[2] del Ministero delle Politiche Agricole, Ambientali e Forestali. A quanto risulta (sul portale Dop Igp non viene specificata la natura dei dati. Le definizioni dei disciplinari sono disponibili in formato PDF), però, questi dati non raggiungo più di tre stelle nella scala di valore dei LOD. Andremo, ancora una volta, ad implementare un servizio che faccia uso degli standard RDF e SPARQL per l'utilizzo di questi dati.

[1] http://etna.istc.cnr.it/food

[2] https://dopigp.politicheagricole.it/

6.1 Panoramica

FOOD-SB permette di esplorare in maniera facile e intuitiva il triplestore messo a disposione dal progetto FOOD. Sulla home viene visualizzato, sulla sinistra, l'elenco dei prodotti disponibili, suddivisi per categorie. Selezionando una categoria, un sotto-menù a fisarmonica viene aperto per permettere di scegliere un prodotto specifico da visualizzare. A seguito di questa seconda selezione, viene caricata sulla destra la scheda descrittiva del prodotto selezionato che ne specifica:

- Denominazione ufficiale
- Tipologia
- Caratteristiche organolettiche

L'immagine sotto mostra la visualizzazione di un prodotto con FOOD-SB.

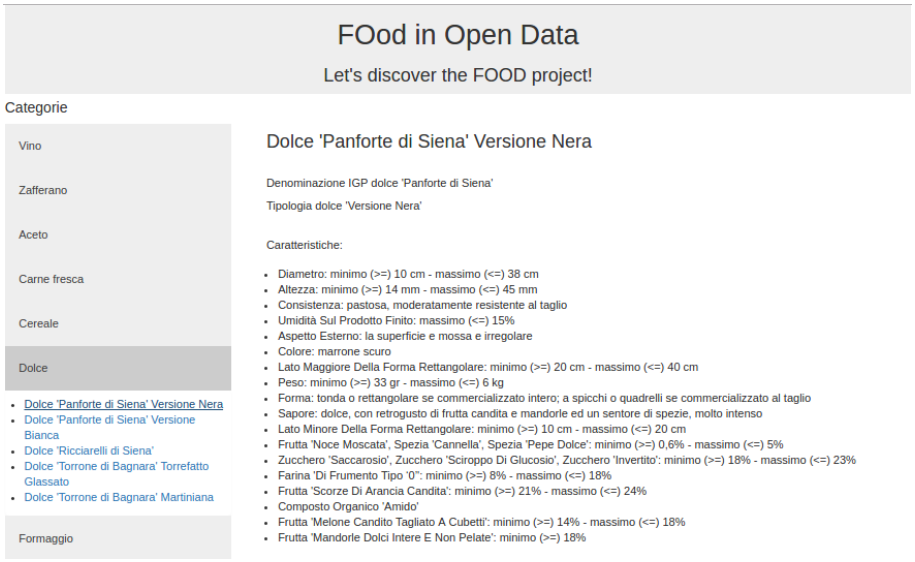

Figura 6.1: visualizzazione prodotto Panforte di Siena

I dettagli dell'implementazione di FOOD-SB verrano introdotti nel paragrafo seguente.

6.2 Dettaglia implementativi

In questo paragrafo verranno analizzati i dettagli dell'implementazione di FOOD-SB.

All'avviso del servizio, lo starter invoca il template categoryList:

```
<template match='$start'>
    <calltemplate name="categoryList" renderTo="#left">
    </calltemplate>
</template>
<template name="categoryList">
    <h4>Categorie</h4>
    <at select= "??category rdfs:subClassOf upper:Prodotto">
    </at>
</template>
```

In questo modo vengono recuperate tutte le categorie di prodotto presenti nel dataset. Per ogni categoria restituita, un apposito template viene invocato per poter estrarne l'etichetta (rdfs:label) da inserire nel menù principale e una seconda interrogazione verso l'endpoint viene eseguita, per cercare tutti i prodotti che fanno parte della categoria selezionata:

```
<template match="?category -> owl:Class">
  <button class="accordion" onclick="loadProducts(this)">
  {{category['rdfs:label'] | uppercase}}</button>
    <ul>
      <at select="??product rdf:type ?category" mode="simple">
      </at>
    </ul>
</template>
```

Per ogni prodotto restituito viene, infine, inserita, nel sotto-menù della propria categoria, l'etichetta corrispondente:

```
<template match="?product -> upper:Prodotto" mode="simple">
    <li>
      <a href="#product/{{product['uri']}}">
        {{product['rdfs:label']}}
      </a>
    </li>
</template>
```

Cliccando su un prodotto, il pattern contenuto nell'attributo href viene passato al servizio di routing di Angularjs che, sulla base dei due parametri passati, setta una variabile nello scope – definita col nome di *uri* - con l'uri del prodotto e invoca il template product:

```
<template name="product">
    <at select="{{$root.uri}}"></at>
</template>
```

L'applyTemplate all'interno di product, infine, utilizza la variabile uri per selezionare il prodotto che viene intercettato da un altro template sulla base del tipo (rdf:type):

```
<template match="?product -> upper:Prodotto">
    <h3>{{product['rdfs:label']}}</h3>

    <foreach select= "?product upper:haDenominazione ??denom">
      <p>{{denom['rdfs:label']}}</p>
    </foreach>
    <at select= "?product upper:haTipologia ??tipo"></at>

    <p>Caratteristiche:</p>
    <ul>
```

```
        <foreach select="?product upper:haDescrizione ??descrCar">
          <li>{{descrCar['rdfs:label']}}</li>
        </foreach>
      </ul>
  </template>
```

Questo è il secondo template, seguendo il nostro ciclo di esecuzione, con matching sul tipo upper:Prodotto. Come si può notare, però, grazie alla specificazione dell'attributo mode si può legare una direttiva applyTemplate con uno specifico template, a patto che i vincoli sul tipo siano comunque rispettati.

Capitolo 7

Valutazioni

OSCR e FOOD-SB forniscono dei servizi per la visualizzazione dei dati sicuramente meno sofisticati di quelli già messi a disposizione dalle istituzioni ma centrano l'obiettivo, attraverso l'utilizzo degli standard RDF e SPARQL, di essere dei servizi in linea con i principi LOD.

In realtà, ai dati offerti da Open Data Sicilia, manca ancora una stella per essere definiti Linked ma è in cantiere l'ambizione di linkare questi dati verso altri dataset. Mentre i dati disponibili sul triplestore del progetto FOOD sono linkati al dataset di Dbpedia. Non è stato, però, possibile accedere al dataset di DBpedia per recuperare informazioni inferite poiché risulta che l'endoind messo a disposizione non implementi una policy di Cross-Origin Resource Sharing (CORS), bloccando tutti i tentativi di recuperare dati da altri dataset. Questo comportamento è decisamente limitante se si vuole implementare appieno il paradigma LOD nei servizi sviluppati con l'ausilio della libreria.

Nella progettazione dei prototipi si è cercato di porre l'enfasi sul meccanismo di routing offerto da Angularjs, per facilitare la navigazione e contemporaneamente fornire una visualizzazione leggibile degli URI. Si è cercato di sfruttare al meglio il binding sulle variabili per elaborare i dati nel controller ed aggiornare facilmente la view.

Un aspetto da tenere in considerazione, quando si utilizza RSLT, è dato

dal tempo di esecuzione delle query. Se i dati da recuperare sono molti, i lunghi tempi di attesa possono diventare sconvenienti. Una soluzione, non sempre praticabile, sta nel cercare di riscrivere la query con pattern più stringenti. Un esempio di notevole rallentamento si può osservare durante il caricamento del menù principale di FOOD-DB: la categoria Vini contiente 5921 prodotti e il loro recupero finisce spesso per freez-are l'intero servizio per qualche secondo.

Inoltre RSLT, come SPARQL, permette di impostare un limite di triple da restituire per una data interrogazione. Se il numero delle risposte ricevute è uguale al limite vuol dire che, probabilmente, ci sono ancora altri dati da caricare; a questo punto si procede con un nuovo invio della query, specificando un offset dal quale riprendere ad inviare i dati. Si è notato che questo approccio non funziona se l'endpoint ha impostato un proprio limite, minore di quello del servizio. In questo caso, infatti, il numero delle risposte ricevute sarà sempre minore del limite atteso da RSLT, che però non si accorgerà di aver ottenuto un numero di risultati parziale poiché tale numero non risulterà mai uguale al limite interno al servizio.

Altro aspetto limitante dello sviluppo di applicazioni con RSLT è dato dalla stringente sintassi implementata dal parser interno alla libreria. Il parser è stato modificato per permettere di utilizzare, nelle query SPARQL, nomi di prefissi e proprietà che fanno uso caratteri non alfanumerici (_, - , /).

Non è, inoltre, consentito dal parser l'utilizzo esplicito di URI https nei pattern delle query in questa maniera:

```
?observation qb:dataset <https://example>
```

Conclusioni

Attraverso questa dissertazione abbiamo visto come la pubblicazione di dati conformi agli standard Linked Open Data apporti diversi vantaggi. Quello che è bene sottolineare è che, fra i diversi obiettivi c'è anche quello di rendere tali dati "aperti", ossia interoperabili e riutilizzabili facilmente da terzi. Un aspetto senza dubbio essenziale in un epoca in cui è di notevole vantaggio produrre progetti - software e non - non concepiti a compartimenti stagni ma in grado di essere compresi, manipolati e riutilizzati in contesti nuovi.

OSCR e FOOD-SB rappresentano infatti un modello valido per l'utilizzo di questa tipologia di dati che permette, anche ad un pubblico digiuno e poco competente in ambito di Semantic Web, di visualizzare informazioni riferite a specifici ambiti della conoscenza: rispettivamente i numeri sull'andamento dell'epidemia di Covid-19, pubblicati come CSV e PDF dal Dipartimento della Protezione Civile e messi a disposizione in formato RDF dal progetto Open Sicilia, e tutti i prodotti agroalimentari italiani che hanno ricevuto un marchio di certificazione, pubblicati dal Ministero delle Politiche Agricole, Alimentari e Forestali in formato PDF e successivamente convertiti in RDF dal progetto FOOD.

L'implementazione di questi servizi è stata possibile grazie all'utilizzo della libreria RSLT, con il suo linguaggio che agevola la trasformazione di dati RDF per una loro presentazione in linguaggio naturale. Una caratteristica fondamentale di RSLT è data dal fatto che può essere utilizzata come un modulo AngularJS. Nell'organizzazione di un codice, infatti, non si può fare a meno di un meccanismo in grado di raggruppare funzionalità secondo

criteri dettati dalla struttura stessa o dal suo utilizzo. I moduli favoriscono la separazione dei compiti definendo un'interfaccia pubblica e limitando la visibilità del funzionamento. AngularJS propone un proprio sistema che consente un'ampia flessibilità nell'organizzazione del codice stesso. Grazie a questo aspetto è stato possibile usufruire, nei servizi proposti, di alcune delle funzionalità offerte dal noto framework. Nello specifico, all'interno del presente progetto di testi, si è ricorso all'utilizzo di due differenti moduli: l'ng-Route, con l'intento di fornire una navigazione intuitiva (oggi di fondamentale importanza per garantire una user experience in linea con le aspettative dell'utente e per offrire un servizio valido all'interno del "mercato") dei vari servizi e il modulo angular-google-chart, che, attraverso l'utilizzo di SVG, nei browser standard come Chrome, Firefox, Safari ecc, permette la visualizzazione dei dati su OSCR sotto forma di specifici grafici.

Sebbene parliamo di visualizzazioni meno sofisticate e accattivanti rispetto agli standard visivi a cui oggi siamo soggetti, questi software utilizzano formati per i dati conformi agli standard del Sematic Web e dei LOD proposti e sviluppati dal W3C. È innegabile che delle migliorie possano e devono esserci, sia in ambito presentazionale che applicativo, poiché gioverebbero ai servizi implementati. Degli sviluppi futuri potrebbero riguardare l'aggiunta di direttive ad-hoc per l'aggiunta di elementi visivi, come i grafici utilizzati all'interno di OSCR o l'integrazione, all'interno dei servizi, di dati provenienti da altri dataset per aderire definitivamente al paradigma LOD. Per raggiungere quest'ultimo obiettivo, però, occorrerà che gli endpoint abilitino delle policy che ammettano Cross-Origin Resource Sharing.

Bibliografia

[PAL14] Oscar Peña, Unai Aguilera et Diego López-de-Ipiña."Linked Open Data Visualization Revisited: A Survey". Semantic Web Journal, 2014.

[QUA15] Davide Quadrelli. "RSLT: trasformazione di Open Linked Data in testi in linguaggio naturale tramite template dichiarativi", 2015.

[MAS15] Lorenzo Massimiliani. "OCEB: Un'applicazione web per la visualizzazione della rete citazionale di OpenCitations", 2015

[BCD15] A.Bagnacani, B.Ciancarini, A.Di Iorio, A.Nuzzolese, S.Peroni et F.Vitali. "The Semantic Lancet Project: a Linked Open Dataset for Scholarly Publishing", 2015.

[CFG17] O.Corby, C.Faron Zucker et F.Gandon, "STTL Transformations", 2017.

[BLH08] D.Berrueta, J.E.Labra et I.Herman. "XSLT+SPARQL: Scripting the Semantic Web with SPARQL embedded into XSLT stylesheets", 2008.

[PV15] Silvio Peroni et Fabio Vitali. "RSLT (RDF Stylesheet Language Transformations)", 2015.

[RP15] Petar Ristoski et Heiko Paulheim. "Visual Analysis of Statistical Data on Maps using Linked Open Data", 2015.

[BLP05] C.Bizer, R.Lee et E.Pietriga. "Fresnel - A Browser-Independent Presentation Vocabulary for RDF". Second International Workshop

on Interaction Design and the Semantic Web @ ISWC'05, Nov 2005, Galway, Ireland. ffinria-00001057.

[SKJ12] Martin G. Skjæveland. "Sgvizler: A JavaScript Wrapper for Easy Visualization of SPARQL Result Sets". The Semantic Web: ESWC 2012 Satellite Events, 2012.

[BHO01] T.Berners-Lee, J.Hendler et O.Lassila. "The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities". 2001. ScientificAmerican.com.