UNIVERSITÀ DEGLI STUDI DI FIRENZE
Facoltà di Ingegneria
Dipartimento di Elettronica e Telecomunicazioni

Dottorato di Ricerca in
Telematica e Società dell'Informazione - XXV Ciclo
ING-INF/05

Architettura di riferimento e modelli semantici per sistemi di gestione di documenti strutturati distribuiti in rete

Tesi di Dottorato di Ricerca di
Riccardo Billero

Coordinatore:
Prof. Dino Giuli
Università degli Studi di Firenze

Supervisori:
Prof. Franco Pirri
Università degli Studi di Firenze

Dr. Ing. Maria Chiara Pettenati
ICON Foundation

Dr. Ing. Federica Paganelli
Consorzio Nazionale Interuniversitario
per le Telecomunicazioni

31 dicembre 2012

Ringraziamenti

Questi ultimi tre anni sono passati davvero in fretta ed eccomi ora a presentare i risultati del mio percorso di ricerca. Sono stati anni intensi, con un susseguirsi di eventi che mi hanno arricchito ed aiutato a crescere come persona.

E mentre mi sembra di avere discusso ieri la mia tesi di laurea, mi accorgo tuttavia che la mia vita è completamente cambiata da allora e questo anche grazie alle persone che mi sono state accanto.

Un ringraziamento immenso va a mia moglie Cristina, che ho sposato proprio all'inizio di questo percorso; grazie tesoro, perché mi ami gratuitamente per quello che sono. La tua presenza al mio fianco mi aiuta ad affrontare ogni giorno con gioia le piccole e grandi incertezze della vita.

Ringrazio i miei meravigliosi genitori, la mamma Mirta, che purtroppo non è più con noi, ed il babbo Giuseppe, per tutto l'impegno e la fatica per crescermi ed educarmi.

Grazie anche ai miei suoceri, Roberto e Maura, per l'affetto e la pazienza mostrata verso di me in questi anni.

Ringrazio tutti i miei amici più veri, in particolare le coppie del gruppo famiglie e il gruppetto di Scuola di Comunità che hanno condiviso con me e mia moglie questi anni; penso con affetto alla mia figlioccia, la piccola Gemma.

Nell'ambito accademico un doveroso grazie va al Prof. Dino Giuli per gli innumerevoli incoraggiamenti, l'aiuto morale e materiale che mi ha fornito nei momenti di fragilità, e soprattutto per aver creduto in me fino alla fine, andando oltre il ruolo che ricopre.

Ringrazio il Prof. Franco Pirri, per la stima e la fiducia dimostratemi e per tutti i confronti costruttivi che hanno arricchito il mio bagaglio di conoscenze. Lo ringrazio inoltre per l'amicizia paterna di cui mi ha onorato, attraverso e grazie alla quale ho potuto superare molte delle mie insicurezze.

Ringrazio il Dr. Ing. Maria Chiara Pettenati, la cara Maria Chiara, per la disponibilità che ha sempre avuto nei miei confronti e per i suoi preziosi suggerimenti professionali e personali.

Ringrazio il Dr. Ing. Federica Paganelli, per la disponibilità mostrata e per i consigli offerti nella parte finale del mio percorso di ricerca.

Ringrazio l'Ing. Davide Chini, per la grande disponibilità avuta all'inizio del mio percorso di ricerca, nel passarmi il "testimone".

Ringrazio tutti i colleghi del laboratorio per i piacevoli momenti passati insieme, in particolare Lucia, Stefano e Lorenzo per il sostegno e l'aiuto personale e professionale sempre offerti.

Ringrazio il caro Luca Capannesi, per il supporto tecnico, l'efficienza, la competenza e lo zelo con cui amministra i sistemi del "Laboratorio Radar e Sistemi Telematici" del "Dipartimento di Elettronica e Telecomunicazioni".

Ringrazio gli ex-studenti, adesso ingegneri e validi professionisti, che ho seguito come correlatore di tesi e tutti gli studenti dei corsi di “Telematica” e “Sistemi Telematici” che con i loro elaborati hanno contribuito agli sviluppi del progetto.

Grazie infine Chi ha permesso che potessi affrontare questo percorso, con tutti i momenti lieti e quelli difficili, e che attraverso i volti delle persone che hanno costellato questa mia esperienza mi ha detto ciò che Thorin Scudodiquercia disse a Bilbo Baggins prima di morire: “*In te c'è più di quanto tu creda, figlio delle miti terre d'Occidente*”.

Firenze, 31 dicembre 2012

Riccardo Billero

"In te c'è più di quanto tu creda,
figlio delle miti terre d'Occidente"

John Ronald Reuel Tolkien
Lo Hobbit

A Cristina

Indice

Introduzione **xxi**

I Analisi delle tecnologie **1**

1 L'evoluzione del Web **3**
1.1 Service Oriented Architecture 4
1.1.1 Web services SOAP 5
1.2 REpresentational State Transfer 6
1.3 Resource Oriented Architecture 7
1.4 Read-Write Web of Data 8
1.5 Gli standard del Semantic Web 13
1.6 Web services ed annotazione semantica 16
1.6.1 SAWDSL (Semantic Annotation for WSDL) 16
1.6.2 SAREST e MicroWSMO 22

2 Open Data e dati geografici **23**
2.1 Open Data 24
2.1.1 Iniziative internazionali 27
2.1.2 Open Data in Italia 28
Dataset pubblicati in Italia 28
2.2 Sistemi di riferimento geografico 29
2.2.1 Identificare univocamente un punto della Terra 30
2.2.2 Datum ed ellissoidi di approssimazione del geoide . . . 31
2.3 Rappresentazione di informazioni geografiche 32
2.3.1 Keyhole Markup Language 33
Rappresentazione di un documento KML elementare . 34
2.3.2 ShapeFile 38
2.3.3 GeoJSON 38
2.4 Open Data del Comune di Firenze 43

II InterDataNet **45**

3 InterDataNet Information Model **49**
3.1 IDN Information Model: concetti generici 50
3.2 Requisiti dell'Information Model 54
3.3 I nodi informativi 56
3.4 Relazioni strutturali tra nodi e documenti 58
3.5 Strutturare con criteri di responsabilità 59
3.6 Identificazione di nodi e documenti 61
3.7 Storico dei documenti 64
3.7.1 La propagazione delle modifiche 68
3.7.2 I parametri di versione 69

4 InterDataNet Service Architecture **75**
4.1 IDN-SA: vista d'insieme 79
4.1.1 Gli IDN-Node 79
4.1.2 Identificazione degli IDN-Node 80
4.1.3 Attraversamento dei livelli 83
4.1.4 Imbustamento di dati e metadati 84
4.2 IDN naming system 86
4.3 Storage Interface 87
4.3.1 Integrazione di sistemi legacy 89
4.4 Replica Management 90
4.4.1 Localization Service 91
4.5 Information History 91
4.5.1 History Name System 92
4.6 Virtual Resource 92
4.6.1 Resource Aggregation Service 93
4.6.2 Logical Domain Name Service 94

III IDN Template **97**

5 IDN Template Information Model **99**
5.1 La necessità di individuare regole per i dati 100
5.1.1 Struttura di un documento IDN 100
5.1.2 Formato dei dati 101
5.1.3 Sintassi dei dati 103
5.1.4 Semantica dei dati 106
5.2 Individuare un modello dei dati 106
5.2.1 Struttura dell'Information Model 110
5.2.2 Formato e sintassi dei dati di livello applicativo 112
5.2.3 Semantica dei dati di livello applicativo 114
5.3 Analogie con la programmazione ad oggetti 114

6 Riuso di IDN Template **117**
6.1 Il riuso del codice 117
6.1.1 Il design pattern Adapter 119
6.2 Il riuso dei dati 121
6.3 Il riuso dell'IDN Template 123
6.4 Basi teoriche di riferimento 126
6.5 Riuso multiplo di nodi modello 127

7 IDN Template Data Model **131**
7.1 Rappresentazione dell'IDN Template 131
7.1.1 Abilitare l'IDN Template nei documenti IDN 136
7.2 Caso d'uso: Open Data di Firenze 141
7.3 Validazione di un documento IDN 152

IV Caso di studio: Open Data Firenze **155**

8 Riuso degli Open Data di Firenze **157**
8.1 IDN Compliant Applications 157
8.2 ETL 159
8.3 ETL per gli Open Data di Firenze 160
8.4 Riuso degli Open Data in Easy Florence 165

Conclusioni **171**

V Appendici **175**

A InterDataNet Data Model **177**

Bibliografia **183**

Elenco delle figure

1.1 Esempi pratici di *loose coupling*. 5
1.2 La nuvola del Linked Open Data. 12
1.3 Gli strati che compongono il Semantic Web. 14

3.1 Esempio di documento IDN. 59
3.2 Esempio di relazioni tra documenti IDN-IM. 63
3.3 Generazione delle revisioni. 65
3.4 Merge di due nodi. 67
3.5 Esempio di propagazione delle modifiche. 68
3.6 Esempio di elemento recente rispetto al nodo di partenza. . . 71
3.7 Esempi di "last" relativi al branch. 71
3.8 Espressione regolare per definire gli identificativi di versione. 72
3.9 Convenzione sui nomi dei nodi nello storico. 72

4.1 Relazione tra IDN-IM e IDN-SA. 76
4.2 Livelli dell'architettura IDN. 78
4.3 Spazi dei nomi delle risorse IDN. 79
4.4 Nomi di risorse replicate. 82
4.5 Interazione request/response applicata ad IDN. 84
4.6 Imbustamento di dati e metadati. 85
4.7 Sistema dei nomi di InterDataNet (risoluzione diretta). 87
4.8 Gerarchia delle risorse "nome" di InterDataNet. 88
4.9 Storage Interface e Legacy Storage Interface. 89
4.10 Revisioni successive di documenti UEVM. 93
4.11 Creazione controllata e non di alias. 95

5.1 Documento IDN sugli Open Data dei musei di Firenze 108
5.2 Documento IDN compatto sugli Open Data di Firenze 109
5.3 Corrispondenza tra nodo e nodo modello (1) 110
5.4 Corrispondenza tra nodo e nodo modello (2) 110
5.5 Rappresentazione di link modello. 111
5.6 IDN Template degli Open Data dei musei di Firenze 113
5.7 Diagramma delle classi degli Open Data dei musei di Firenze 116

6.1 Diagramma delle classi degli Open Data dei musei di Firenze 118
6.2 Struttura del design pattern Adapter. 120
6.3 Riuso di classi degli Open Data dei musei di Firenze 121
6.4 Rappresentazione grafica del riuso di un nodo modello. 123
6.5 Aggiunta di un link modello 124
6.6 Rimozione di un link modello 124
6.7 Aggiunta ed eliminazione di link modello 125
6.8 Specifica grafica del server autoritativo 126
6.9 Documento IDN contenente Open Data di Firenze 126
6.10 Riuso di nodi modello: casi possibili. 128
6.11 Esempio di riuso multiplo di nodi modello (1) 129
6.12 Esempio di riuso multiplo di nodi modello (2) 130
6.13 Rappresentazione di un nodo che effettua riuso ricorsivo. . . . 130

7.1 IDN Template degli Open Data dei musei di Firenze. 142
7.2 Esempio di ontologia per i musei di Firenze. 146
7.3 Struttura di IDN Template di esempio (1) 148
7.4 Struttura di IDN Template di esempio (2) 150
7.5 Struttura di IDN Template di esempio (3) 151

8.1 IDN Template generico degli Open Data di Firenze 163
8.2 IDN Template degli Open Data sui musei di Firenze 164
8.3 IDN Template degli Open Data sui bagni di Firenze 165
8.4 Schermata iniziale di Easy Florence e punto di interesse. . . . 166
8.5 Commenti e relativo inserimento in Easy Florence 166
8.6 Posizione e menù in Easy Florence 168
8.7 Utenti e profilo utente in Easy Florence 168
8.8 IDN Template dei documenti IDN di Easy Florence. 170

Elenco dei listati

1.1 Esempio di RDF/XML. 14
1.2 XML Schema relativo alla gestione di un ordine. 18
1.3 Ontologia OWL relativa alla gestione di un ordine. 19
1.4 Trasformazione XSLT da triple RDF a documento XML. . . . 20
1.5 Trasformazione XSLT da documento XML a triple RDF. . . . 21
2.1 File KML rappresentante un *Placemark*. 35
2.2 File KML rappresentante un *LineString*. 35
2.3 File KML rappresentante un *Polygon*. 36
2.4 File KML con HTML in un elemento CDATA. 37
2.5 File KML con HTML e riferimenti di entità. 37
2.6 File GeoJSON rappresentante un punto. 39
2.7 File GeoJSON rappresentante una linea. 39
2.8 File GeoJSON rappresentante un poligono. 40
2.9 File GeoJSON rappresentante una collezione. 41
2.10 File GeoJSON rappresentante un elemento *Feature*. 41
2.11 File GeoJSON rappresentante un elemento *FeatureCollection*. 42
5.1 Esempio di documento XML. 105
5.2 Esempio di XML Schema. 106
7.1 XML Schema dell'IDN Template. 131
7.2 Modifiche al VR-Documento per abilitare IDN Template. . . 136
7.3 XML Schema di documento IDN che abilita IDN Template. . 137
7.4 IDN Template dei musei degli Open Data di Firenze. 142
7.5 Ontologia OWL per i musei di Firenze. 145
7.6 IDN Template che effettua il riuso dei musei degli Open Data di Firenze, mediante tag *FilteredLinks*. 148
7.7 IDN Template che effettua il riuso dei musei degli Open Data di Firenze, mediante tag *MaintainedLinks*. 149
7.8 IDN Template con riuso multiplo. 149
7.9 IDN Template con riuso multiplo. 151
A.1 XML Schema dei documenti IDN 177

Abstract

The Web Science aims to drive the evolution of the Web providing possible answers to key aspects of this socio-technical system such as: identity, privacy, security, trust and governance. The current "Web" – intended as a social-technical system – does not provide a mechanism for collaboration that will maintain a suitable infrastructural support; the Web of Data is moving in that direction through the Linked Data and Open Data approaches.

This work is part of the InterDataNet framework (IDN) that wants to be an architectural solution in order to allow users, distributed across time and space, to collaborate around information elements that belong to a global area of distributed data with which is possible to interact through the metaphor of documents.

Specifically, this work aims to enrich the IDN infrastructure by proposing a solution aimed at an efficient and effective data reuse; such a mechanism, called *IDN Template*, makes possible the description of a model that defines a set of rules to which a structured document must comply with, both as a whole and in all its component parts.

The IDN Template provides the designer with the ability to make available to third parties details regarding both the document as a whole - through the description of the different elements composing the structure - both in terms of its individual components - through the description of format, syntax, structural constraints and semantics of data.

The reuse mechanism enabled by the IDN Template, makes easy to reuse by third parties data for which has been defined a suitable model.

As use case, a collection of geographical Open Data made available by the Municipality of Florence, Italy is evaluated; such information – duly processed – was inserted into InterDataNet infrastructure through the use of the template technique and then reprocessed together with data produced by third parties.

The results obtained have strengthened the belief that the InterDataNet infrastructure can provide an innovative solution for a web of structured and reusable data in a collaborative manner while maintaining the full compatibility with the standards suggested by Linked Data and Semantic Web.

Sommario

La Web Science si pone l'obiettivo di guidare l'evoluzione del Web fornendo possibili risposte ad aspetti cruciali di questo sistema socio-tecnico quali: identità, privacy, sicurezza, trust e governance. Il "Web" attuale – inteso come sistema socio-tecnico – non fornisce un meccanismo di collaborazione che possa avvalersi di un adeguato supporto infrastrutturale; il movimento Web of Data si sta muovendo in questa direzione, mediante gli approcci Linked Data ed Open Data.
Il presente lavoro si colloca all'interno di InterDataNet, un framework che si propone come soluzione architetturale per consentire ad utenti distribuiti nel tempo e nello spazio di collaborare intorno ad elementi informativi che appartengono ad uno spazio globale di dati distribuito con cui è possibile interagire mediante la metafora dei documenti.
In particolare, il presente lavoro si pone l'obiettivo di arricchire l'infrastruttura InterDataNet proponendo una soluzione finalizzata ad un riuso dei dati efficiente ed efficace; tale meccanismo, cui è stato assegnato il nome di IDN Template, rende possibile la descrizione di un modello che definisce un insieme di regole al quale un documento strutturato, sia nel complesso sia in ogni sua singola parte, deve risultare conforme.
L'IDN Template fornisce al progettista la possibilità di rendere disponibili a soggetti terzi, informazioni sia riguardo il documento nel complesso – attraverso la descrizione dei vari elementi che vanno a comporne la struttura – sia riguardo le sue singole parti – attraverso la descrizione del formato, della sintassi, dei vincoli strutturali e della semantica dei dati.
Il meccanismo abilitato dall'IDN Template, facilita il riuso – da parte di progettisti terzi – dei dati per i quali sia stato definito un opportuno modello.
Come caso d'uso viene analizzato un insieme di Open Data geografici, messi a disposizione dal Comune di Firenze; tali informazioni sono state opportunamente elaborate ed inserite all'interno dell'infrastruttura InterDataNet, modellate mediante il ricorso alla tecnica di modellazione descritta e successivamente rielaborate congiuntamente a dati prodotti da soggetti terzi.
I risultati conseguiti hanno rafforzato la convinzione che l'infrastruttura InterDataNet possa fornire una soluzione innovativa per un Web di dati strutturati e riusabili in modo collaborativo, mantenendo la piena compatibilità con gli standard proposti dal Linked Data e dal Semantic Web.

Introduzione

Ad oltre venti anni dalla sua nascita, avvenuta nel 1989, il Web è ormai il più grande costrutto informativo nella storia dell'umanità, e consente a tutt'oggi ad oltre due miliardi di utenti, di consultare con facilità documenti distribuiti a livello globale, sia nella vita lavorativa che in quella quotidiana.

Tra i fattori che hanno determinato il suo successo possono essere annoverati la sua interfaccia semplice ed uniforme, il collegamento ipertestuale, la semplicità del linguaggio di markup HTML e del protocollo HTTP ed il fatto di essere un"unico Web" [Aye06].

Negli ultimi anni si è fatto un gran parlare di "Web 2.0" e poi ancora di "Web 3.0", ad indicare un particolare fiorire di applicazioni che hanno ottenuto successo grazie alla imponente partecipazione di utenti, nella creazione, gestione e condivisione di contenuti informativi, in modo sempre più intelligente e orientato alla semantica.

Sebbene tale fenomeno sia ancora in evoluzione, nella comunità scientifica si parla ormai da anni di un futuro Web semantico avente l'obiettivo di introdurre nuove tecnologie rispetto al Web di oggi, con informazioni che potranno essere "comprese" anche dalle macchine.

Nel lungo cammino svolto da parte della comunità scientifica nell'ultimo decennio è emerso che tale visione comporta prima la realizzazione di una transizione intermedia che è stata chiamata "Web of Data", la cui costruzione sta avvenendo basandosi su quattro principi fondamentali, chiamati Linked Data Principles. Il Web of Data così realizzato, risulta quindi in un Web costituito da dati con un significato semantico esplicito, pubblicati on-line sotto forma di triple RDF (Resource Description Framework) ed interconnessi tra dataset diversi e distribuiti [Hea08].

Tuttavia, sebbene rivoluzioni il paradigma da Web di documenti a Web di dati, l'approccio Linked Data non modifica la condizione strutturale del Web come prevalentemente di tipo "read", ovvero con dati di sola lettura; ad oggi la capacità di "write" è limitata alle singole applicazioni che offrono soluzioni ad-hoc quali wiki, blog, social network, ecc...

Inoltre si evidenzia che il Web of Data ottenuto finora, sebbene abbia una grande diffusione in termini di pubblicazioni scientifiche, non è ancora una tecnologia di uso comune.

Il presente lavoro si basa su InterDataNet (o IDN), un modello infrastrutturale per il Read Write Web of Data [Inn08, Chi09, Cio10] che mira a realizzare uno spazio decentralizzato e scalabile dove pubblicare dati interconnessi, mediante l'indirizzabilità globale delle risorse ed una serie di servizi base per la collaborazione (controllo dell'authorship, versioning e gestione delle repliche) in modo da permettere la gestione di dati strutturati distribuiti ed eterogenei grazie ad un riuso su scala globale.
IDN si propone come una soluzione architetturale innovativa che consenta ad utenti "distribuiti" nel tempo e nello spazio di collaborare intorno ad elementi informativi che appartengono ad uno spazio globale di dati distribuito con cui è possibile interagire mediante la metafora dei documenti [Cio10], intendendo una collaborazione che possa avvalersi di un adeguato supporto infrastrutturale.
L'architettura InterDataNet può essere descritta attraverso l'utilizzo di tre viste, che risultano complementari ed integrate tra loro, ognuna delle quali pone l'attenzione su degli aspetti specifici del problema.
Esse sono:

1. IDN Information Model (IDN-IM), la vista su risorse e informazione;
2. IDN Service Architecture (IDN-SA), la vista sull'architettura;
3. IDN Compliant Application (IDN-CA), la vista sulle applicazioni.

IDN Information Model (IDN-IM) è un modello dell'informazione [PS03, CT04, HSGT94] finalizzato a rappresentare dati, che definisce le proprietà di base dell'informazione da trattare, ma volutamente non fornisce specifiche soluzioni implementative, ovvero rappresenta l'informazione indipendentemente dalla tecnologia ed in tal modo abilita l'interoperabilità fra sistemi eterogenei.
IDN Service Architecture definisce in maniera stratificata i servizi necessari al fine di soddisfare il modello informativo IDN-IM, in modo conforme ai principi REST (REpresentational State Transfer) [Fie00]; IDN-SA implementa le varie funzionalità, mediante la definizione di sottosistemi, protocolli ed interfacce per una gestione collaborativa del documento rappresentato tramite IDN-IM; inoltre espone una API (Application Programming Interface) REST che può essere utilizzata dalle applicazioni (al fine di manipolare informazioni rappresentate tramite IDN-IM) ricorrendo direttamente al protocollo HTTP; in tal modo è fornita la garanzia di scalabilità ed interoperabilità.
Infine, le IDN Compliant Application implementano le logiche di business per la risoluzione di specifici problemi di dominio e/o rappresentano lo user-agent per l'interazione con il sistema. Le applicazioni IDN operano trattando informazione conforme ai dettami fissati dall'Information Model ed utilizzano l'IDN-SA per accedere ai documenti IDN-IM ed eventualmente manipolarli.

Il presente lavoro apporta un contributo all'interno di tale architettura attraverso l'introduzione di una tecnologia per la modellazione e l'arricchimento semantico di documenti IDN-IM, che consente anche di definire opportuni *template*, ovvero modelli, (insiemi di regole) cui tali documenti devono conformarsi.
In particolare, tale meccanismo consente di:

- indicare quali nodi possono comporre un documento, specificandone anche l'opzionalità o la molteplicità;
- per ogni nodo, specificare il formato e la sintassi dei relativi dati;
- per ogni nodo, indicare la semantica dei relativi dati, attraverso il riferimento ad un concetto semantico opportunamente specificato.

Diviene quindi possibile, attraverso una opportuna *separation of concerns*, specificare i vari aspetti relativi alla gestione delle singole parti di un documento, riuscendo inoltre ad aprire la strada verso una compatibilità tra l'infrastruttura IDN e le tecnologie proposte dal Semantic Web (triple RDF ed ontologie OWL).
Inoltre la tecnica proposta fornisce anche la possibilità di effettuare il riuso parziale di documenti strutturati, nell'intenzione di aiutare una gestione migliore di dati provenienti da fonti diverse.

La presente tesi si compone di quattro parti.
Nella prima parte viene effettuata una *Analisi delle tecnologie*. Il capitolo 1 (*L'evoluzione del Web*) illustra i principali stili architetturali per la progettazione di servizi web (SOA, WS-*, REST, ROA), i paradigmi alla base del Read-Write Web of Data e del Semantic Web e le tecnologie esistenti per l'annotazione semantica di web services. Il capitolo 2 (*Open Data e dati geografici*) fornisce una panoramica sul movimento Open Data (in particolare riguardo i dati di natura georeferenziata), concentrandosi sullo stato attuale di tale realtà in Italia ed in particolare nel Comune di Firenze.
Nella seconda parte si introduce *InterDataNet*. Il capitolo 3 (*InterDataNet Information Model*) illustra nel dettaglio il modello dell'informazione (IDN-IM) attraverso la definizione dei requisiti, dei principi e della struttura delle risorse da un punto di vista astratto. Nel capitolo 4 (*InterDataNet Service Architecture*) viene descritto in maniera integrata l'insieme dei servizi che compongono l'architettura InterDataNet.
Nella terza parte viene illustrato il meccanismo che va sotto il nome di *IDN Template*. Il capitolo 5 (*IDN Template Information Model*) introduce il concetto di IDN Template, focalizzandosi sulla definizione di un modello elementare dei dati. Quest'ultimo viene ampliato all'interno del capitolo 6 (*Riuso di IDN Template*), dove viene descritta la modalità per effettuare il riuso di un modello precedentemente definito da terzi parti. Quindi, il

capitolo 7 (*IDN Template Data Model*) presenta, attraverso il ricorso a vari esempi, una proposta di modello dati in grado di mettere in pratica i concetti illustrati nei due capitoli precedenti.
La quarta parte, infine, presenta un caso d'uso effettivo del meccanismo di IDN Template. Il capitolo 8 (*Riuso degli Open Data di Firenze*), illustra dapprima il concetto di IDN Compliant Application, quindi presenta una prima applicazione che trasforma gli Open Data rilasciati dal Comune di Firenze in documenti IDN-IM, conformi ad un opportuno IDN Template. Un adeguato utilizzo del meccanismo di riuso del modello, consente quindi di riutilizzare tale IDN Template all'interno di una seconda applicazione mobile, denominata Easy Florence.

Parte I

Analisi delle tecnologie

Capitolo 1

L'evoluzione del Web

Il World Wide Web viene utilizzato oggi da oltre due miliardi di utenti con varie finalità, che vanno dalla semplice ricerca di informazioni alla lettura di e-mail e news, dal download di musica e video agli acquisti on-line.
La diffusione del Web, che è risultata probabilmente superiore alle più rosee aspettative iniziali, ne ha tuttavia evidenziato alcuni limiti, tra i quali il fatto di essere fruibile esclusivamente dagli esseri umani, mentre le macchine non riescono a sfruttare in modo autonomo l'immenso patrimonio informativo che esso offre.
La comunità scientifica ha cercato di superare tale limitazione rendendo il Web "semantico" (il cosiddetto Semantic Web) mediante l'utilizzo di nuovi standard e di tecnologie allo stato dell'arte già esistenti nel mondo dell'Intelligenza Artificiale.
Questo approccio non ha tuttavia consentito di risolvere il problema sopra evidenziato, ma ha comunque permesso di rilevare l'esistenza di lacune a livello infrastrutturale che devono essere necessariamente colmate per raggiungere i traguardi prefissati.
Il presente capitolo si pone lo scopo di presentare brevemente quali sono i principali settori di ricerca che si sono mossi nella direzione di far evolvere il Web da un sistema principalmente orientato alla pubblicazione di risorse fruibili da esseri umani ad una piattaforma per la collaborazione utilizzabile sia dagli umani che dalle macchine.
Considerando ormai noti i principi alla base del Web classico [1], all'interno del presente capitolo verrà dapprima effettuata una analisi delle tecnologie esistenti per la realizzazione dei cosiddetti *web services* [2], in particolare SOA, REST e ROA.

[1] Una adeguata descrizione di cosa sia il World Wide Web e di quali siano i suoi componenti fondamentali (come ad esempio il protocollo HTTP ed il linguaggio HTML) può essere ritrovata in letteratura [KR03].

[2] I web services sono software che possono essere invocati da altri programmi, indipendentemente dal linguaggio, dalla piattaforma utilizzata e dalla posizione all'interno della rete, grazie all'uso di standard adeguati.

Quindi sarà descritta l'evoluzione storica che ha portato alla nascita di Web of Data e Linked Data; infine saranno descritte le tecnologie che possono essere utilizzate per la descrizione della semantica dei dati, focalizzandosi in particolare su quelle relative all'annotazione dei web services.

1.1 Service Oriented Architecture

Le molteplici definizioni di *SOA* (*Service Oriented Architecture*) esistenti a tutt'oggi in letteratura [MLM+06], [Erl06], [NL04], [Nat03], [Han07], [WCL+05] concordano tutte nel ritenerlo un paradigma architetturale in grado di consentire una migliore flessibilità nella fase di progettazione di una architettura software.
SOA consente di far fronte alla necessità di flessibilità nelle moderne architetture software, al fine di mantenere soluzioni di qualità e rispettare i tempi del mercato; tale paradigma fornisce una serie di suggerimenti su come affrontare vari aspetti relativi alla gestione di progetti, processi e ruoli delle entità coinvolte, rivolgendosi in particolare ai sistemi distribuiti (ovvero afferenti a diversi proprietari) sistematizzando l'approccio all'eterogeneità della codifica dei dati e della comunicazione. Infatti i grandi sistemi usano piattaforme e linguaggi di programmazione differenti, sono distribuiti su reti che non sono controllate da un singolo responsabile e utilizzano ed organizzano capacità distribuite [MLM+06].
Mentre gli approcci precedenti all'introduzione di SOA prevedevano di risolvere i problemi di compatibilità e d'interoperabilità tra sistemi e applicazioni adottando un'unica soluzione (ed eliminando in tal modo le differenze esistenti), SOA parte dal presupposto che non è possibile rimuovere l'eterogeneità e sulla base di ciò definisce tre concetti fondamentali:

- i servizi;
- l'interoperabilità;
- il basso accoppiamento (*loose coupling*).

Un *servizio* costituisce l'interfaccia di una rappresentazione di una qualche funzionalità reale; pertanto deve essere progettato limitando che traspaiano aspetti tecnici e ciò che viene elaborato internamente. Mediante una interfaccia ben definita ed uniforme, i dettagli dipendenti dalle specifiche piattaforme risultano irrilevanti e l'eterogeneità non rappresenta una preoccupazione per gli utenti del servizio.
L'*interoperabilità* consente di connettere sistemi eterogenei facendo sì che possano funzionare insieme, ad esempio scambiandosi risorse o invocando funzioni remote; in SOA ciò può essere raggiunto mediante il disaccoppiamento, ovvero con la riduzione delle dipendenze ai minimi termini.

Il *loose coupling* contribuisce alla tolleranza ai guasti e alla flessibilità, evitando colli di bottiglia che ostacolerebbero la scalabilità dell'architettura; esso può essere ottenuto aumentando la modularità del sistema, mantenendo uniformi le interfacce, evitando la centralizzazione più di quanto non sia necessario.
In tabella 1.1 sono elencati alcuni esempi pratici che illustrano il concetto di disaccoppiamento.

	Accoppiamento stretto	**Loose coupling**
Connessioni fisiche	Punto-Punto	Tramite mediatore
Stile di comunicazione	Sincrono	Asincrono
Data model	Tipi complessi comuni	Solo tipi semplici
Tipizzazione del sistema	Forte	Debole
Pattern di interazione	Navigazione attraverso oggetti ad albero complessi	Incentrata sui dati, messaggi autocontenuti
Controllo della logica dei processi	Centrale	Distribuito
Binding	Statico	Dinamico
Piattaforma	Forti dipendenze dalla piattaforma	Indipendente dalla piattaforma
Transazioni	Commit in due fasi	Compensazione
Deployment	Simultaneo	In tempi differenti
Versionamento	Aggiornamenti espliciti	Aggiornamenti impliciti

Figura 1.1: Esempi pratici di *loose coupling*.

1.1.1 Web services SOAP

La declinazione più nota di SOA è costituita da quelli che sono generalmente indicati come web services SOAP (o anche con il termine WS-*). Tale tecnologia si compone di tre standard diversi, i quali si combinano insieme per fornire ad un web service la capacità di funzionare, descrivere se stesso ed essere individuato all'interno di una rete.
Il primo standard, *SOAP* (*Simple Object Access Protocol*), è un linguaggio XML per la rappresentazione del messaggio scambiato tra client e server e costituisce la lingua franca dei web services. Nato nel 1997 come standard industriale per creare un meccanismo di *RPC* [3] in ambiente distribuito, in modo trasparente rispetto al sistema operativo o al linguaggio di programmazione, è divenuto uno standard del World Wide Web Consortium nel 2003 [GHM+07]. Ogni messaggio SOAP si compone di una busta (*envelope*), al cui interno trovano posto una intestazione (*header*), con informazioni riguardanti mittente e destinatario, e un corpo (*body*) che costituisce il messaggio informativo vero e proprio.
Il secondo standard, *WSDL* (*Web Services Description Language*), è un linguaggio XML [4] finalizzato a descrivere esattamente quali sono i compiti del

[3] Una *RPC* (*Remote Procedure Call*) consente ad un programma di eseguire subroutine su computer remoti accessibili in rete, in modo il più possibile analogo a quello della tradizionale chiamata verso una procedura locale.

[4] Il metalinguaggio XML viene descritto all'interno del paragrafo 5.1.3.

web service, dove risiede e come invocarlo; anche esso è regolamentato dal World Wide Web Consortium [CCMW01]. A motivo delle capacità offerte da WSDL, i web services vengono considerati come elementi software auto-descriventi, consentendo ad un qualsiasi programmatore (opportunamente autorizzato) di connettersi ad un determinato web service semplicemente conoscendone il WSDL.

Il terzo standard, *UDDI* (*Universal Discovery, Description and Integration*), è un registry (ovvero una base dati ordinata e indicizzata), basato su XML e indipendente dalla piattaforma hardware, che permette alle aziende la pubblicazione dei servizi offerti su internet attraverso un registro paragonabile ad una sorta di "pagine gialle" dei web services.

1.2 REpresentational State Transfer

REST (*REpresentional State Transfer*) è uno stile per architetture software definito all'interno della tesi di dottorato di Roy T. Fielding, pubblicata nel 2000 [Fie00].

Secondo tale lavoro è possibile asserire che il REST è lo stile architetturale del Web, poiché è nato nel corso della stesura del protocollo HTTP [FGM+99] di cui lo stesso Fielding è coautore; per la precisione il REST è un modello astratto di cui il Web sarebbe una architettura concreta.

REST definisce un insieme di vincoli che applicati ad un sistema lo rendono *RESTful*; occorre comunque osservare che il Web è RESTful non soltanto perché si conforma a tali vincoli, ma anche perché è il caso specifico dal quale sono stati dedotti i principi per definirne il modello.

Il primo vincolo che una architettura RESTful deve rispettare è l'interazione tra componenti di tipo *client-server*, all'interno di uno scenario in cui i server offrono servizi e rimangono in ascolto di richieste dei client, mentre questi ultimi fruiscono di tali servizi e utilizzano un *connettore* [5] per inviare richieste.

REST impone inoltre che la comunicazione tra client e server sia *priva di stato*. Ogni richiesta da parte di un client deve contenere tutte le informazioni necessarie affinché possa essere compresa, e non può trarre vantaggio da contenuti memorizzati nel server; pertanto lo stato della sessione è mantenuto interamente nel client. L'applicazione di questo principio incrementa l'affidabilità e la scalabilità del sistema, poiché una interazione che contiene tutte le informazioni può essere più facilmente monitorata; tuttavia il sovraccarico introdotto dalla ripetizione dei dati può portare ad un peggioramento delle prestazioni.

[5] Il connettore incapsula le attività di accesso alle risorse e di trasferimento delle loro rappresentazioni.

REST consente l'utilizzo di *cache*, per migliorare l'efficienza dei sistemi, permettendo al client di usare una risposta memorizzata in precedenza ed eliminando la necessità di alcune interazioni.

REST si distingue dalle altre architetture basate sul networking attraverso il vincolo di *interfaccia uniforme* tra componenti; in tal modo le implementazioni sono disaccoppiate dalle funzionalità, il sistema è semplificato e la visibilità delle interazioni migliorata; tuttavia l'efficienza viene degradata in quanto le informazioni vengono trasferite in una forma generalizzata invece che in una forma specificatamente progettata per i bisogni applicativi.

L'interfaccia di un sistema RESTful deve consentire l'*identificazione* delle informazioni e la loro manipolazione deve avvenire tramite una *rappresentazione*. In REST una qualsiasi informazione dotata di nome viene detta risorsa, sia essa un documento, un'immagine o un oggetto non virtuale (una persona, un animale, una città). Le risorse sono manipolate attraverso una loro rappresentazione che in generale è costituita da una sequenza di byte più alcuni metadati che li descrivono.

Un'architettura REST deve essere *stratificata* consentendo di adattare i sistemi ai requisiti di scalabilità dei grandi network come Internet. La stratificazione consente di disaccoppiare i componenti di un'architettura e promuovere l'indipendenza tra stati non adiacenti; tuttavia incrementa la complessità dei sistemi, aumentando il sovraccarico informativo e la latenza nell'elaborazione dei dati.

Il REST non prescrive l'uso di particolari protocolli o standard e non specifica quali operazioni deve offrire una interfaccia, purché questa sia uniforme e sia orientata alle risorse; sotto questo aspetto può pertanto essere posto su un piano di confronto delle Service Oriented Architecture.

1.3 Resource Oriented Architecture

I web services RESTful hanno riscosso un discreto interesse da parte degli sviluppatori orientati alla realizzazione di servizi di largo consumo per il Web, e quindi non interessati alla costruzione di complessi sistemi *enterprise*.

Occorre tuttavia sottolineare che molti servizi etichettati come RESTful non rispettano in realtà tutti i vincoli di tale stile; pertanto i puristi della filosofia REST li definiscono come servizi "accidentally RESTful", "low REST" o "REST-RPC".

All'interno di questo panorama si inserisce *ROA* (*Resource Oriented Architecture*), un insieme di linee guida per realizzare architetture RESTful orientate al web [RR07]. Infatti mentre REST specifica solo dei criteri di progettazione generici, le Resource Oriented Architecture sono esplicitamente legate alle tecnologie del Web; le risorse sono indirizzate da URI HTTP *descrittivi* (ovvero deve esistere una corrispondenza intuitiva tra questi e le

risorse che indirizzano) e che possiedono uno *schema sintattico* secondo le seguenti regole base:

- utilizzare il path per codificare la gerarchia (ad es. `/parent/child`);
- se non esiste gerarchia, utilizzare la punteggiatura (ad es. `//parent/child1;child2`);
- utilizzare le query per le variabili che rappresentano l'input di un algoritmo (ad es. `/search?q=jellyfish&start=20`).

Una Resource Oriented Architecture dispone delle sole operazioni base definite dal protocollo HTTP, le principali delle quali sono:

- `GET`, per recuperare la rappresentazione di una risorsa;
- `POST`, per creare una nuova risorsa come subordinata ad una esistente;
- `PUT`, per modificare una risorsa esistente o crearne una nuova;
- `DELETE`, per cancellare una risorsa.

Il metodo `GET` è *safe*, ovvero la sua esecuzione non altera lo stato della risorsa nel server.
I metodi `GET`, `PUT` e `DELETE` sono *idempotenti*, ovvero effettuando una richiesta ripetuta ed in maniera identica con uno di essi lo stato della risorsa non cambia; le richieste dalla seconda a seguire lasceranno la risorsa esattamente nello stato in cui lo ha lasciato la prima.
Le due proprietà citate sono di fondamentale importanza poiché consentono ai client di fare richieste HTTP affidabili in reti che per loro natura non lo sono; l'operazione `POST` non è né safe né idempotente.
L'approccio ROA si propone come una alternativa più semplice rispetto ai web services basati su SOAP, offrendo comunque funzionalità che coprono tutti gli aspetti necessari ai sistemi più complessi. Negli ultimi anni sta avvenendo una convergenza delle due tecnologie [Vin07], ad esempio le nuove specifiche WSDL supportano la definizione delle interfacce su HTTP in stile ROA [Chi07].

1.4 Read-Write Web of Data

Sin dalla sua nascita ad opera di Tim Berners Lee (1989) [BL89], il Web è stato oggetto di una evoluzione continua con fasi di sviluppo che oggi vengono denominate Web 1.0 e Web 2.0.
Tuttavia, nonostante si sia verificata una modifica dell'interazione tra l'utente e il Web tale da giustificare la classificazione citata, non è mai cambiato il paradigma alla base dell'organizzazione dell'informazione, il cosiddetto

ipertesto; esso può essere considerato come un insieme di documenti collegati tra loro tramite "ancore" all'interno dei documenti stessi, i cosiddetti collegamenti ipertestuali.

Pertanto si può dire che il Web è costituito, in ultima analisi, da una grande quantità di documenti collegati fra loro, i quali risultano essere il mezzo principale per convogliare informazioni su di esso. Tuttavia, tali documenti non sono strutturati ovvero permettono la lettura e l'estrapolazione delle informazioni solo da parte di un utente umano e non da parte di un software, poiché non sono dotati di un qualsiasi tipo di struttura predefinita.

Quest'ultima possibilità risulta acquisire un interesse crescente, come dimostrato ad esempio dalla recente diffusione delle Web API, le quali permettono alle applicazioni di interagire con i dati contenuti in sorgenti diverse tramite il Web, come avviene ad esempio per le API di Google Maps [Goo12b].

Tuttavia si è ancora lontani dal realizzare un Web che possa essere gestito in modo automatico dalle applicazioni, in analogia a quanto viene fatto dagli esseri umani per i documenti; le Web API ad esempio sono semplicemente soluzioni realizzate ad-hoc per lo specifico servizio che si intende integrare.

Si può concludere pertanto che allo stato attuale è ancora lontana dal concretizzarsi all'interno del panorama Web la possibilità di liberare i grandi potenziali informativi presenti al suo interno.

In tale direzione, lo stesso Tim Berners-Lee, introdusse nel 2001 il termine "Semantic Web" per indicare la sua visione di quello che avrebbe dovuto essere il Web del futuro, ovvero un'estensione del Web attuale in cui "verrà data una struttura al contenuto significativo delle pagine" [BLHL01].

Il primo periodo di studi sul Semantic Web (dal 2001 al 2007 circa) fu caratterizzato dalla fondamentale importanza delle competenze sviluppate in molti anni nell'ambito di ricerca relativo all'Intelligenza Artificiale, concentrandosi sul problema dell'inferenza, della rappresentazione della conoscenza mediante ontologie e della codifica delle stesse tramite il linguaggio RDF [6] [KC04]. Nonostante l'intenso lavoro a livello mondiale, i risultati ottenuti non condussero alla effettiva realizzazione di un Semantic Web a livello globale, ma ad una serie di "semantic web locali" che costituivano delle soluzioni idonee per contesti particolari all'interno di un ambito chiuso.

Secondo lo stesso Berners-Lee tale approccio si rivelò insoddisfacente poiché fu trascurato l'aspetto Web del problema, per concentrarsi sul solo lato Semantic [Mac09].

Tali difficoltà condussero alla nascita di una serie di riflessioni in seno alla comunità scientifica, che portarono dapprima ad una critica sull'approccio adottato [SHBL06] ed in seguito ad un cambio della prospettiva con cui affrontare il problema.

[6] Il linguaggio *RDF* (*Resource Description Framework*) è illustrato all'interno del paragrafo 1.5.

Questo nuovo filone di ricerca, attivo dal 2007 sino ad oggi, ha raggiunto risultati concreti più consistenti rispetto al lavoro svolto in precedenza, soprattutto grazie al fatto di avere riportato l'attenzione sullo spazio globale di dati fittamente interconnessi, integrati a partire dal basso. In tal modo diviene possibile la convergenza su alcuni concetti condivisi (a partire dall'eterogenea miriade di entità di cui ogni organizzazione dispone per la gestione del proprio dominio di interesse) tramite l'uso di opportuni collegamenti (i cosiddetti *owl:sameAs* [BM12]).
Questo nuovo filone di ricerca si è inoltre focalizzato sull'importanza di un cambio di paradigma nei confronti delle informazioni sul Web, attraverso il passaggio dall'attuale "Web of Documents" al cosiddetto "Web of Data". Lo stesso inventore del Web, nel 2006 riconobbe la necessità di indicare alcuni capisaldi per la realizzazione di un effettivo Semantic Web globale, che si concretizzarono nei "Linked Data Principles" [BL06], ovvero delle regole per definire la struttura dei dati da pubblicare sul Web come *Linked Data*.
Tali principi sono:

1. usare gli URI per assegnare nomi alle cose;
2. usare URI HTTP di modo che le persone possano cercare questi nomi;
3. quando qualcuno visita un URI, fornire informazioni utili mediante gli standard;
4. includere link ad altri URI, di modo che sia possibile scoprire altro.

Il primo principio afferma la necessità di utilizzare URI [7] come nomi per identificare le "cose"; questo ultimo termine non è erroneo, ma vuole esprimere la volontà di utilizzare i principi Linked Data come regole di pubblicazione sia per le risorse informative (ovvero le informazioni "pure" come un articolo scientifico, o una pagina web) sia per le cosiddette risorse non informative (ossia tutto ciò che non è informazione di per sé, come un'auto, una città) [8].
Il secondo principio fornisce l'indicazione di utilizzare URI HTTP, tenendo conto dell'esistenza di una infrastruttura funzionante per la gestione di tali identificatori. Per quanto riguarda le risorse non informative, il Technical Architecture Group del W3C suggerisce l'uso del codice di stato HTTP 303 `See Other`, corrispondente alla dereferenziazione verso l'URI di una risorsa informativa contenente una rappresentazione della risorsa non informativa in questione.

[7] Un *URI* (*Uniform Resource Identifier*) [BLFM05] è una stringa che identifica univocamente una risorsa generica come un documento, un'immagine, ecc... Gli URI possono rendere disponibili risorse per vari protocolli di rete, tra i quali è incluso anche HTTP.

[8] Risulta normale rappresentare risorse non informative con una o più risorse informative, come ad esempio una pagina web che parla di una città [Lew07].

Il terzo principio (detto "follow your nose") indica la necessità di fornire informazioni utili per la dereferenziazione di un URI in particolare mediante l'adozione degli standard RDF [KC04] e SPARQL [9] [PS08].
Il quarto principio, infine, evidenzia la necessità di realizzare collegamenti fra gli URI in modo da poter navigare da una risorsa ad un'altra, poiché si desidera realizzare dei Linked Data, ovvero costruire una ragnatela mondiale di dati che possa essere navigata per mezzo di apposite applicazioni, così come avviene oggi con la navigazione delle pagine Web attraverso i collegamenti ipertestuali.
Dai suddetti principi è nato il "Linking Open Data (LOD) project" [BCH07], che ha portato alla realizzazione di uno spazio globale di dati interconnessi, la cosiddetta nuvola LOD, illustrata in figura 1.2, costituita da milioni di triple RDF.
Tale successo ha provocato un grande entusiasmo in seguito al quale lo stesso Berners Lee affermò che il "Web of Data è il Semantic Web fatto bene", ovvero che si era finalmente giunti ad una base infrastrutturale su cui sarebbe stato possibile implementare le applicazioni ideate per il Semantic Web nel corso della prima fase della ricerca.
Tuttavia, nonostante l'evidenza di tali risultati, il Web of Data è rimasto tema dell'ambito della ricerca scientifica senza riuscire ad entrare nell'uso quotidiano da parte dell'uomo comune, come dimostra l'assoluta carenza di applicazioni diffuse su larga scala che sfruttino tale tecnologia. Questo insuccesso è dovuto soprattutto al fatto che il Web of Data attuale è stato realizzato utilizzando le tecnologie del Web corrente, dove i dati possono essere dereferenziati, ma non è possibile manipolarli in modo uniforme. Il processo di pubblicazione è infatti esterno alla struttura stessa del Web, e sebbene il tipico utente Web 2.0 abbia la sensazione di scrivere utilizzando il Web stesso, questo è in realtà una caratteristica legata a contesti locali e chiusi, i cosiddetti *walled-garden*.
La necessità di manipolare ed elaborare i dati risulta essere quindi un aspetto molto importante per il Web of Data e gli sforzi della ricerca scientifica si stanno concentrando in questa direzione, mirando a realizzare un vero Read-Write Web of Data, nella cui struttura il processo di scrittura sia inserito direttamente; le soluzioni attualmente proposte soffrono tuttavia di mancanza di uniformità [HKO+09, BL09].
La ricerca scientifica internazionale ha rivolto la propria attenzione ai problemi tecnici legati alla realizzazione di applicazioni che manipolano i dati, trascurando tuttavia gli aspetti legati all'utilizzo degli stessi. In particolare non sono stati tenuti in considerazione alcuni importanti campi applicativi, quali le applicazioni enterprise o l'e-Government [10]; entrambi i settori ne-

[9] Il linguaggio *SPARQL* è illustrato all'interno del paragrafo 1.5.

[10] L'e-Government è uno dei settori su cui si concentra attualmente l'attenzione dei governi di alcuni paesi anglosassoni, come dimostrato dal movimento Open Data, presentato al capitolo 2.

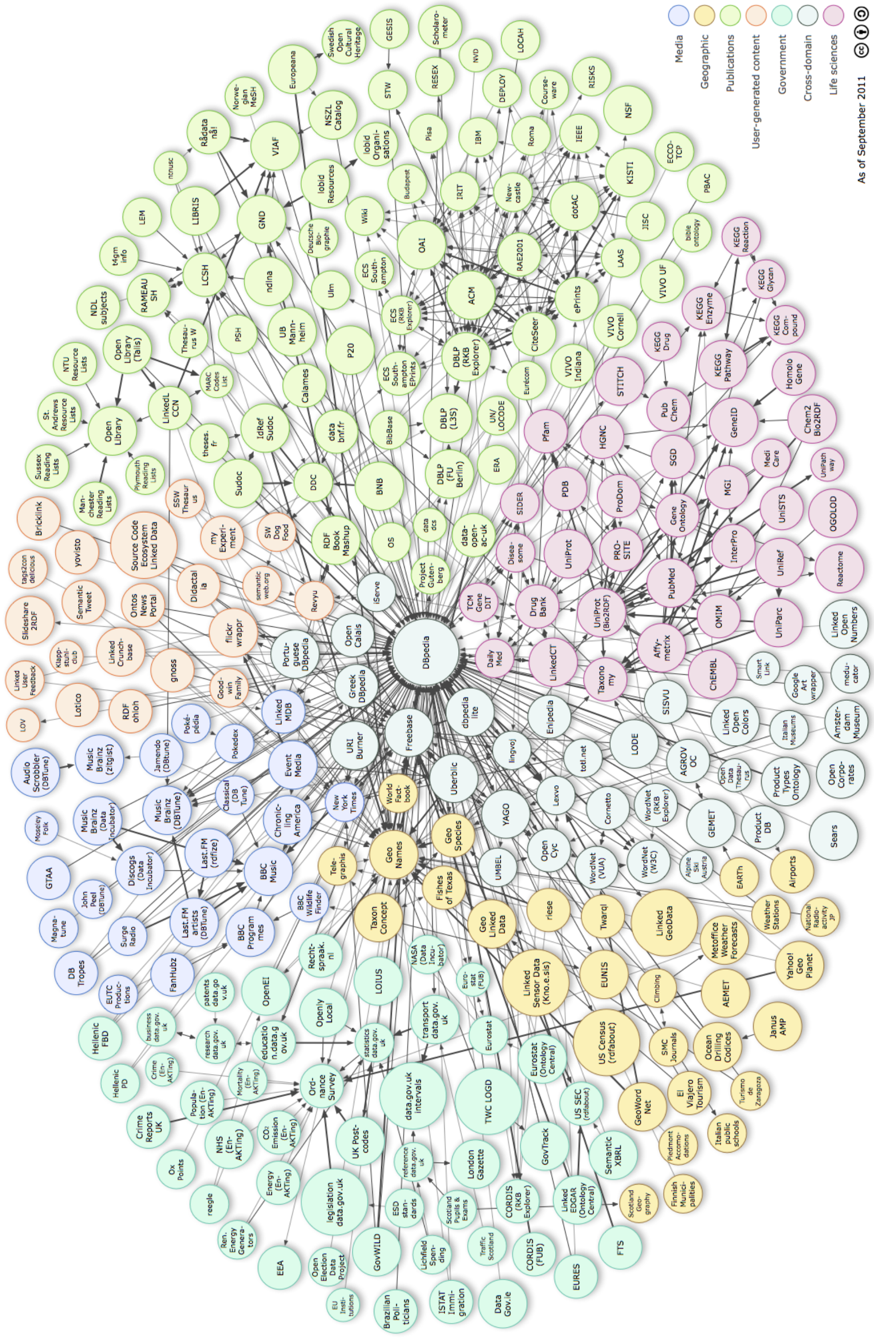

Figura 1.2: La nuvola del Linked Open Data.

cessitano di alcune garanzie che sono oggi assenti in un Web of Data il quale non consente di sapere come i dati pubblicati saranno integrati e riusati.
Esistono inoltre altri aspetti critici, come il fatto che durante la costruzione del Web of Data molti collegamenti di identità non sono stati realizzati con la dovuta accuratezza, e questo potrebbe implicare difficoltà o problemi da parte di eventuali reasoners [11] [HHM+10]; inoltre non sono ancora state definite le interazioni con un tale spazio di dati, ad eccezione di quelle collegate alla ricerca.
Il presente lavoro assume una impostazione critica nei confronti dei principi Linked Data, ritenendo possibile una alternativa che porti ad un vero Read-Write Web of Data. In particolare viene ritenuta di grande importanza la necessità di posizionare al livello infrastrutturale (anziché a quello applicativo) le proprietà che abilitino funzionalità collaborative verso il livello superiore, al fine di consentire il riuso dei dati e la collaborazione intorno ad essi, di modo che il Web of Data possa uscire dalla sola ricerca scientifica per entrare nel mondo comune.

1.5 Gli standard del Semantic Web

Al fine di raggiungere gli obiettivi proposti dal Semantic Web (introdotto all'interno del paragrafo precedente) è stata definita la cosiddetta *pila semantica*, illustrata in figura 1.3. Essa si compone di una serie di strati software (basati su tecnologie già esistenti quali URI ed XML) alcuni dei quali sono già stati realizzati, mentre altri lo saranno in futuro; tra gli standard appartenenti alla prima categoria saranno descritti nel seguito RDF, RDF Schema, OWL e SPARQL.
Il *Resource Description Framework* (*RDF*) è il linguaggio proposto dal World Wide Web Consortium per descrivere le "risorse"; tale termine indica "qualunque cosa che abbia una identità" [LS99] e, come già illustrato all'interno del paragrafo 1.4, generalmente si distingue tra risorse informative (ovvero le informazioni "pure" come un articolo scientifico, o una pagina web) e risorse non informative (ossia tutto ciò che non è informazione di per sé, come un'auto, una città).
Il modello dati di RDF permette di formulare delle affermazioni (*statements*) a proposito delle risorse sotto forma di espressioni, dette *triple*, del tipo soggetto-predicato-oggetto: il soggetto indica la risorsa, mentre il predicato ne denota un aspetto ed esprime una relazione esistente tra il soggetto e l'oggetto. Al fine di evitare ambiguità di riferimento, sia il soggetto che il predicato sono espressi mediante un URI, mentre l'oggetto può essere espresso da un URI o semplicemente da un tipo di dati primitivo.
Un modello RDF può essere rappresentato mediante un grafo orientato, dove i nodi indicano le risorse (o i tipi primitivi) e gli archi le proprietà; tale grafo

[11] Il concetto di *reasoners* è illustrato all'interno del paragrafo 1.5.

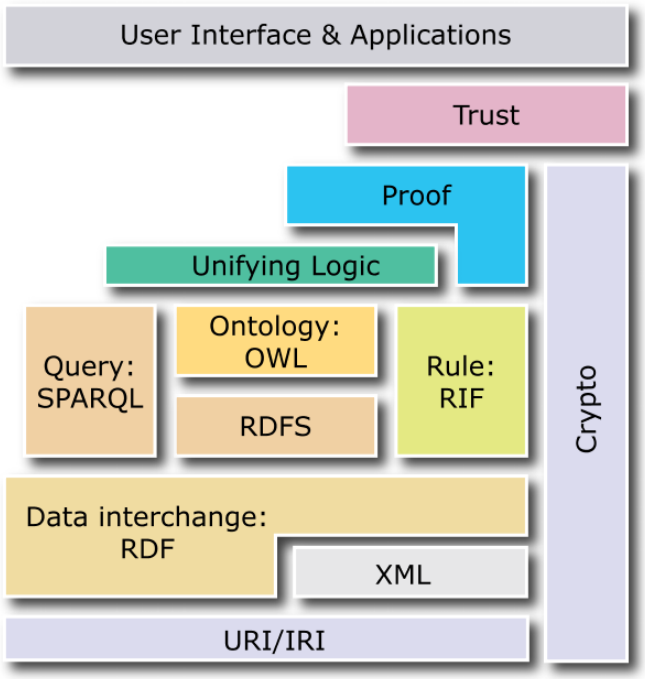

Figura 1.3: Gli strati che compongono il Semantic Web.

può essere serializzato attraverso il ricorso all'*RDF/XML*, un opportuno linguaggio XML definito dal World Wide Web Consortium [BM04a], oppure mediante N3, una serializzazione non XML progettata in modo da essere scritta (e molto spesso letta) con maggiore facilità e basata su una notazione in stile tabulare.
Il listato 1.1 riporta l'esempio di due triple espresse in notazione RDF/XML; la prima afferma che la risorsa costituita dalla pagina di Wikipedia indicata dall'attributo *rdf:about* (soggetto) ha come titolo (predicato) "Tim Berners Lee" (oggetto), mentre la seconda sostiene che la medesima risorsa è espressa in lingua inglese.

Listato 1.1: Esempio di RDF/XML.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc= "http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Tim_Berners_Lee"
    >
    <dc:title>Tim Berners Lee</dc:description>
    <dc:language>en</dc:language>
  </rdf:Description>
</rdf:RDF>
```

Il data model RDF consente di descrivere le relazioni esistenti tra le risorse, in termini di proprietà, ma non fornisce alcun meccanismo per dichiarare quali esse siano, né le loro relazioni con le risorse. A tale compito il World Wide Web Consortium ha proposto *RDF Schema* (o *RDFS*) [BGM04] attraverso il quale risulta possibile creare vocabolari per i documenti RDF. Tra la varie caratteristiche che RDF Schema offre, occorre sottolineare la possibilità di definire classi di risorse, le quali possono anche essere integrate per comporre una opportuna gerarchia, mediante l'uso del costrutto *rdfs:subClassOf*; inoltre per ogni proprietà possono essere imposti dei vincoli quali il dominio e il codominio, attraverso l'uso di *rdfs:domain* e *rdfs:range* rispettivamente.

Il passaggio successivo per l'implementazione del Semantic Web ha riguardato la definizione di un linguaggio per la rappresentazione di ontologie. Tale termine (derivato da una delle branche fondamentali della filosofia, riguardante lo studio dell'essere in quanto tale e delle sue categorie [12] fondamentali) è stato mutuato in informatica ad indicare la concettualizzazione formale, condivisa ed esplicita riguardo un dominio di interesse, da parte di un attore. Un software può utilizzare una ontologia per vari scopi, come il ragionamento induttivo o la classificazione.

Una ontologia informatica deve essere espressa attraverso un opportuno linguaggio formale [13]; a tal fine, per quanto riguarda il Semantic Web, il World Wide Web Consortium propone lo standard *OWL* (*Web Ontology Language*) [BM12], il cui scopo consiste nel descrivere delle basi di conoscenza su cui sia in seguito possibile effettuare delle deduzioni attraverso appositi software di *reasoning*.

OWL è espresso mediante una sintassi XML ed è stato sviluppato come passo successivo di RDF e RDF Schema, cui aggiunge nuovi costrutti; questi consentono di descrivere le relazioni tra classi, la cardinalità delle risorse, le caratteristiche delle proprietà (quali simmetria, transitività, ecc...) ed il concetto di l'equivalenza tra risorse, ovvero la possibilità di affermare (mediante il costrutto *owl:sameAs*) che due o più URI rappresentano in realtà lo stesso elemento, consentendo in tal modo di unire risorse distribuite su dominî semantici diversi.

All'interno del presente paragrafo non viene presentato un esempio di ontologia espressa mediante OWL; tuttavia il lettore può fare riferimento al listato 1.3 a pagina 19, oppure al caso illustrato in figura 7.2 a pagina 146 e riportato nel listato 7.5 a pagina 145.

Le basi di conoscenza formalizzate attraverso l'uso di RDF possono essere interrogate attraverso l'uso del linguaggio *SPARQL* (*Simple Protocol And RDF Query Language*), una raccomandazione del World Wide Web Con-

[12] In filosofia, una categoria è l'attribuzione di un predicato ad un soggetto.

[13] Un linguaggio formale è un insieme di stringhe di lunghezza finita costruite sopra un alfabeto finito (ovvero un insieme finito di simboli).

sortium [PS08] con sintassi stile SQL, che permette di scrivere query non ambigue; queste possono essere distribuite ad endpoints SPARQL [14] multipli, e i risultati possono essere riuniti, in modo da eseguire una *query federata*. Per quanto riguarda invece l'interrogazione di conoscenze formalizzate con OWL, a tutt'oggi non è stato ancora definito uno standard, ed i singoli reasoners implementano linguaggi di query proprietari.

1.6 Web services ed annotazione semantica

I web services WS-* o REST, illustrati rispettivamente nei paragrafi 1.1 e 1.2, non consentono per loro natura una descrizione semantica dei propri componenti. Al fine di venire incontro alle esigenze del Semantic Web e degli standard da esso utilizzati (descritti al paragrafo precedente), sono stati definite alcune tecniche per l'annotazione semantica dei web services.

Nel seguito sono analizzate tra esse dapprima SAWSDL, che si rivolge ai web services WS-*, ed in seguito SAREST e MicroWSMO, che si rivolgono al mondo REST.

1.6.1 SAWDSL (Semantic Annotation for WSDL)

SAWSDL (*Semantic Annotation for WSDL*) [Wor07b] definisce alcuni attributi di estensione che possono essere utilizzati per annotare i documenti WSDL o gli XML Schema, facendo riferimento ad opportuni modelli semantici.

In particolare questa tecnologia fornisce dei meccanismi standard attraverso i quali i concetti dei modelli semantici, che sono tipicamente definiti all'esterno di documenti WSDL o XML Schema, possono essere riferiti da questi ultimi attraverso il ricorso ad alcune annotazioni [LF07].

Il namespace di SAWSDL [Wor07a] consente di aggiungere come attributi ad un qualsiasi elemento presente all'interno di un documento WSDL o di un XML Schema le tre annotazioni seguenti:

- *modelReference*; consente di specificare una associazione tra un componente di un documento WSDL o XML Schema ed un concetto presente in un modello semantico. In particolare, può essere utilizzato per annotare definizioni di tipi, elementi o attributi in un XML Schema, così come per la dichiarazione di interfacce, operazioni ed errori in un documento WSDL.

[14] Un *endpoint* SPARQL è un servizio che accetta query in tale linguaggio e ne restituisce il risultato.

L'attributo *modelReference* può assumere come valore uno o più URI [15] (separati da spazi), ognuno dei quali identifica un determinato concetto presente nel modello semantico in oggetto [16]; in tal modo ogni URI consente di fornire informazioni semantiche riguardo al particolare componente annotato all'interno del documento WSDL o XML Schema.

- *loweringSchemaMapping*; può essere aggiunto a dichiarazioni di elementi o definizioni di tipi in un XML Schema.

  Tale attributo può assumere come valore uno o più URI verso opportune definizioni di regole di mappatura le quali possono essere applicate nella fase in cui una selezione di dati presenti in un modello semantico viene trasformata (*to lower*, abbassare) in un documento XML. In generale, ma non necessariamente, i dati forniti in input a tale trasformazione sono stati selezionati grazie all'uso di una query SPARQL applicata sul modello semantico, mentre le regole di mappatura sono espresse come trasformazione XSLT [17].

- *liftingSchemaMapping*; può essere aggiunto a dichiarazioni di elementi o definizioni di tipi in un XML Schema.

  Tale attributo può assumere come valore uno o più URI verso opportune definizioni di regole di mappatura le quali possono essere applicate nella fase in cui un documento XML (ovviamente conforme all'XML Schema considerato) viene trasformato (*to lift*, sollevare) in un modello semantico. In generale, ma non necessariamente, le regole di mappatura sono espresse come trasformazione XSLT.

Al fine di chiarire i concetti sin qui illustrati, viene presentato un esempio tratto dal sito del World Wide Web Consortium, in particolare dalla pagina relativa a SAWSDL [LF07] riguardante un sistema per la gestione di ordini; nei due listati 1.2 e 1.3 sono rappresentati rispettivamente un XML Schema ed una ontologia OWL relativi al problema citato.
È possibile osservare che all'interno di tale XML Schema sono presenti due elementi, in particolare *OrderRequest* e *OrderResponse* i quali contengono entrambi al loro interno un attributo *modelReference* il cui URI consente di fare riferimento ad un particolare concetto semantico presente nell'ontologia OWL citata.

[15] Se un componente è annotato con un attributo *modelReference* che include più di un URI, si ipotizza che ognuno di essi faccia riferimento a tale componente; tuttavia non viene definita alcuna relazione logica tra i concetti semantici riferiti.

[16] Gli URI utilizzati all'interno dell'attributo *modelReference* possono identificare concetti espressi in linguaggi semantici diversi.

[17] L'*XSLT* (*eXtensible Stylesheet Language Transformations*) è un linguaggio per la trasformazione di un documento XML in un altro documento [Sax07].

Il primo elemento (*OrderRequest*) è dotato anche dell'attributo *loweringSchemaMapping*, il quale consente di fare riferimento al file rappresentato nel listato 1.4. Quest'ultimo si compone di due sezioni, la prima contenente una query SPARQL utilizzabile per selezionare un insieme di dati presenti nel modello semantico rappresentato dal listato 1.3 e la seconda contenente una trasformazione XSLT mediante la quale risulta possibile creare un documento XML che utilizzi i dati estratti dalla suddetta query SPARQL. Per quanto riguarda invece il secondo elemento (*OrderResponse*), esso è dotato dell'attributo *liftingSchemaMapping*, il quale consente di fare riferimento alla trasformazione XSLT rappresentata nel listato 1.5, utilizzabile per creare un modello semantico a partire da un documento XML conforme all'XML Schema considerato nell'esempio.

Listato 1.2: XML Schema relativo alla gestione di un ordine.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/
  order#"
  elementFormDefault="qualified">
  <xs:element name="OrderRequest"
    sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/
      purchaseorder#OrderRequest"
    sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/
      mapping/RDFOnt2Request.xslt">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="customerNo" type="xs:integer" />
        <xs:element name="orderItem" type="item" minOccurs="1"
          maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="item">
    <xs:all>
      <xs:element name="UPC" type="xs:string" />
    </xs:all>
    <xs:attribute name="quantity" type="xs:integer" />
  </xs:complexType>
  <xs:element name="OrderResponse" type="confirmation" />
  <xs:simpleType name="confirmation"
    sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/
      purchaseorder#OrderConfirmation"
    sawsdl:liftingSchemaMapping="http://www.w3.org/ws/sawsdl/spec/mapping/
      Response2Ont.xslt">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Confirmed" />
      <xs:enumeration value="Pending" />
      <xs:enumeration value="Rejected" />
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Listato 1.3: Ontologia OWL relativa alla gestione di un ordine.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xml:base="http://www.w3.org/2002/ws/sawsdl/spec/ontology/
  purchaseorder#">
  <owl:Ontology />
  <owl:Class rdf:ID="OrderRequest" />
  <owl:ObjectProperty rdf:ID="hasLineItems">
    <rdfs:domain rdf:resource="#OrderRequest" />
    <rdfs:range rdf:resource="#LineItem" />
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:ID="hasCustomer">
    <rdfs:domain rdf:resource="#OrderRequest" />
    <rdfs:range rdf:resource="#Customer" />
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty" /
      >
  </owl:FunctionalProperty>
  <owl:Class rdf:ID="LineItem" />
  <owl:ObjectProperty rdf:ID="hasQuantity">
    <rdfs:domain rdf:resource="#LineItem" />
    <rdfs:range rdf:resource="#Quantity" />
  </owl:ObjectProperty>
  <owl:FunctionalProperty rdf:ID="hasProduct">
    <rdfs:domain rdf:resource="#LineItem" />
    <rdfs:range rdf:resource="#Product" />
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty" /
      >
  </owl:FunctionalProperty>
  <owl:Class rdf:ID="Customer" />
  <owl:FunctionalProperty rdf:ID="hasCustomerID">
    <rdfs:domain rdf:resource="#Customer" />
    <rdfs:range rdf:resource="#CustomerID" />
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty" /
      >
    <rdfs:subPropertyOf rdf:resource="#hasIdentifier" />
  </owl:FunctionalProperty>
  <owl:Class rdf:ID="CustomerID">
    <rdfs:subClassOf rdf:resource="#Identifier" />
  </owl:Class>
  <owl:Class rdf:ID="Product" />
  <owl:FunctionalProperty rdf:ID="hasProductCode">
    <rdfs:domain rdf:resource="#Product" />
    <rdfs:range rdf:resource="#ProductCode" />
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty" /
      >
    <rdfs:subPropertyOf rdf:resource="#hasIdentifier" />
  </owl:FunctionalProperty>
  <owl:Class rdf:ID="UPCCode">
    <rdfs:subClassOf rdf:resource="#ProductCode" />
  </owl:Class>
  <owl:Class rdf:ID="ProductCode">
    <rdfs:subClassOf rdf:resource="#Identifier" />
  </owl:Class>
  <owl:Class rdf:ID="Identifier" />
```

```
  <owl:FunctionalProperty rdf:ID="hasLexicalRespresentation">
    <rdfs:domain rdf:resource="#Identifier" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
       />
  </owl:FunctionalProperty>
  <owl:Class rdf:ID="Quantity" />
  <owl:DatatypeProperty rdf:ID="hasAmount">
    <rdfs:domain rdf:resource="#Quantity" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float" />
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:ID="hasUnit">
    <rdfs:domain rdf:resource="#Quantity" />
    <rdfs:range rdf:resource="#Unit" />
  </owl:ObjectProperty>
  <owl:Class rdf:ID="Unit" />
  <owl:Class rdf:ID="ItemUnavailable" />
  <owl:Class rdf:ID="OrderConfirmation" />
  <owl:FunctionalProperty rdf:ID="hasStatus">
    <rdfs:domain rdf:resource="#OrderConfirmation" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"
       />
  </owl:FunctionalProperty>
  <owl:ObjectProperty rdf:ID="hasIdentifier">
    <rdfs:domain rdf:resource="http://www.w3.org/2002/07/owl#Thing" />
    <rdfs:range rdf:resource="#Identifier" />
  </owl:ObjectProperty>
  <owl:Class rdf:ID="RequestPurchaseOrder" />
</rdf:RDF>
```

Listato 1.4: Trasformazione XSLT da triple RDF a documento XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<lowering>

  <sparqlQuery>
    PREFIX po: <http://www.w3.org/2002/ws/sawsdl/spec/ontology/
      purchaseorder#>
    SELECT ?qty ?UPC ?CustomerNo
    WHERE {
      ?order po:hasCustomer ?customer .
        ?customer po:hasCustomerID ?id .
          ?id po:hasLexicalRespresentation ?CustomerNo .
      ?order po:hasLineItems ?item .
        ?item po:hasQuantity ?qtyClass .
          ?qtyClass po:hasAmount ?qty .
        ?item po:hasProduct ?product .
          ?product po:hasProductCode ?code .
            ?code po:hasLexicalRespresentation ?UPC }
  </sparqlQuery>
```

```
  <xsl:transform version="2.0"
    xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:sp="http://www.w3.org/2005/sparql-results#">
    <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="
      yes" />
    <xsl:template match="/sp:sparql">
      <po:OrderRequest>
        <po:customerNo>
          <xsl:value-of
            select="sp:results/sp:result[position()=1]/sp:binding[@name='
              CustomerNo']/sp:literal" />
        </po:customerNo>
        <xsl:apply-templates select="sp:results/sp:result" />
      </po:OrderRequest>
    </xsl:template>
    <xsl:template match="sp:result">
      <po:orderItem>
        <xsl:attribute name="quantity">
          <xsl:value-of select="sp:binding[@name='qty']/sp:literal" />
        </xsl:attribute>
        <po:UPC>
          <xsl:value-of select="sp:binding[@name='UPC']/sp:literal" />
        </po:UPC>
      </po:orderItem>
    </xsl:template>
  </xsl:transform>
</lowering>
```

Listato 1.5: Trasformazione XSLT da documento XML a triple RDF.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:transform version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
  xmlns:POOntology="http://www.w3.org/2002/ws/sawsdl/spec/ontology/
    purchaseorder#">
  <xsl:output method="xml" version="1.0" encoding="iso-8859-1" indent="yes"
     />
  <xsl:template match="/">
    <rdf:RDF>
      <POOntology:OrderConfirmation>
        <hasStatus rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
          <xsl:value-of select="po:OrderResponse" />
        </hasStatus>
      </POOntology:OrderConfirmation>
    </rdf:RDF>
  </xsl:template>
</xsl:transform>
```

1.6.2 SAREST e MicroWSMO

SAREST (*Semantic Annotation of Web Resources*) [GRS10] è un posh-format [Mic11] per aggiungere metadati a documenti HMTL contenenti la descrizione di API REST, al fine di migliorare la ricerca, facilitare l'intermediazione di dati e l'integrazione dei servizi. I metadati sono aggiunti attraverso l'opportuno uso degli attributi *class* e *title* delle specifiche HTML [JRLH99]; in particolare il valore dell'attributo *class* indica il nome del metadato, mentre l'attributo *title* ne indica il valore.
MicroWSMO (*Micro Web Service Modeling Ontology*) [VK08] persegue un analogo fine, ma attraverso un opportuno uso degli attributi *rel* e *href* del tag <a> delle specifiche HTML [JRLH99]; in particolare il valore dell'attributo *rel* indica il nome del metadato, mentre l'attributo *href* ne indica il valore. I metadati che possono essere utilizzati sono *model*, *lifting* e *lowering* ed assumono un significato analogo a quello dei corrispondenti elementi presenti all'interno di SAWSDL; pertanto non sono qui descritti nel dettaglio.

Capitolo 2

Open Data e dati geografici

Un particolare ambito compreso all'interno del mondo Linked Data, illustrato al capitolo 1, è quello che va sotto il nome di Open Data, ovvero dati "aperti", cioè liberamente accessibili a tutti, con restrizioni di copyright assenti o limitate e riutilizzabili [Spa10].
Gli Open Data sono rilasciati da svariate realtà, tra le quali è possibile annoverare le Pubbliche amministrazioni; queste possono rendere fruibili dati che sono generalmente suddivisibili in più categorie, tra le quali si annoverano ad esempio i dati georeferenziati, ovvero informazioni derivanti da dati geografici, ed i dati alfanumerici, ovvero dati costituiti da un insieme di stringhe o numeri e che, in genere, si adattano perfettamente ad una rappresentazione in forma tabulare.
Molte istituzioni di tutto il mondo (e quindi anche italiane) facendo seguito alla pratica dell'Open Government [1], hanno rilasciato nel corso degli ultimi anni vari dataset di informazioni di cui disponevano.
In particolare, all'interno della realtà italiana, a causa dell'elevato numero di dataset rilasciati, risulta di particolare interesse analizzare a titolo di esempio il Comune di Firenze, amministrazione che ha rilasciato dati geografici nei formati KML e Shapefile (per alcuni dataset l'amministrazione ha optato anche per il formato GeoJSON) e dati alfanumerici nel formato CSV.
All'interno del presente capitolo sarà dapprima descritta la filosofia Open Data, elencando le principali iniziative a livello internazionale ed italiano. Quindi la trattazione si focalizzerà nell'illustrare cosa è un sistema di riferimento geografico, facendo particolare attenzione al modello di riferimento comunemente utilizzato, il WGS84, e a quelli utilizzati in Italia ed in Europa nello scorso secolo.
Subito dopo saranno introdotte le varie modalità attraverso le quali è possibile rappresentare le informazioni geografiche in un calcolatore elettronico;

[1] La filosofia definita Open Government prevede che tutte le attività dei governi e delle amministrazioni di uno stato siano essere aperte e disponibili ai cittadini, al fine di favorire azioni efficaci e garantire un controllo pubblico sull'operato di tali organismi.

particolare attenzione sarà posta ai formati usati dal Comune di Firenze e prima citati, ovvero il KML, lo Shapefile e il GeoJSON.
Lo sguardo sarà focalizzato infine sull'esperienza di pubblicazione di Open Data da parte del Comune di Firenze, attraverso la descrizione dei dataset rilasciati di maggiore interesse.

2.1 Open Data

Il termine Open Data fa riferimento ad una filosofia, ma al tempo stesso ad una pratica, la quale implica che varie tipologie di dati siano liberamente accessibili a tutti, ovvero senza la presenza di restrizioni quali copyright, brevetti o altre forme di controllo che ne limitino la riproduzione. Pertanto, gli Open Data si basano sull'idea che taluni dati siano disponibili gratuitamente, al fine di essere utilizzati e ripubblicati liberamente. Tali dati possono essere paragonati a quanto avviene nel mondo software con il fenomeno dell'Open Source.
In particolare molti Open Data sono legati al cosiddetto "Open Government", una teoria secondo la quale le Pubbliche Amministrazioni dovrebbero essere aperte ai cittadini, sia in termini di trasparenza sia di partecipazione diretta al processo decisionale, anche facendo ricorso alle nuove tecnologie disponibili. Solitamente gli Open Data fanno riferimento a informazioni rappresentate in forma di database e relative alla tematiche più disparate, quali ad esempio: cartografia, dati anagrafici, dati governativi, genetica, composti chimici, formule matematiche e scientifiche, dati medici, ecc. Sebbene la pratica e l'ideologia che caratterizzano i dati aperti siano da anni ben consolidate, al momento non si riscontra un accordo generale e condiviso su di una definizione puntuale del termine, contrariamente a quanto già avviene con il software libero [Fre12], l'accesso aperto [Sub04] o l'open source [Ope06] dove diverse dichiarazioni formali sono state comunemente accettate e condivise a livello internazionale. Il progetto Open Definition di Open Knowledge Foundation sostiene che "un contenuto o un dato si definisce aperto se chiunque è in grado di utilizzarlo, riutilizzarlo e ridistribuirlo, soggetto, al massimo, alla richiesta di attribuzione e condivisione allo stesso modo" [Ope09c]. La definizione, molto sintetica, viene poi meglio esplicata attraverso il documento "Open Knowledge" [Ope09d] (i cui contenuti sono molto simili a quelli della OSI definition [Ope06]) all'interno del quale sono elencati i seguenti 11 punti che mettono chiarezza sulle modalità di distribuzione e di accesso a tali informazioni:

1. **Accesso**: l'opera deve essere disponibile nella sua interezza ed a un costo di riproduzione ragionevole, preferibilmente tramite il download gratuito via Internet. L'opera deve inoltre essere disponibile in un formato comodo e modificabile.

2. **Ridistribuzione**: la licenza non deve imporre alcuna limitazione alla vendita o all'offerta gratuita dell'opera singolarmente considerata o come parte di un pacchetto composto da opere provenienti da fonti diverse. La licenza non deve richiedere alcuna "royalty" o altra forma di pagamento per tale vendita o distribuzione.

3. **Riutilizzo**: la licenza deve consentire la realizzazione di modifiche e di opere derivate e deve consentire la loro distribuzione agli stessi termini dell'opera originaria.

4. **Assenza di restrizioni tecnologiche**: l'opera deve essere fornita in un formato che non ponga ostacoli tecnologici allo svolgimento delle attività sopraelencate. Ciò può essere conseguito mediante la messa a disposizione dell'opera in un formato aperto, vale a dire un formato le cui specifiche siano pubblicamente e liberamente disponibili e che non imponga nessuna restrizione economica o di altro tipo al suo utilizzo.

5. **Attribuzione**: la licenza può richiedere di citare i vari contributori e creatori dell'opera come condizione per la ridistribuzione ed il riutilizzo di quest'ultima. Se imposta, questa condizione non deve essere onerosa. Per esempio, se viene richiesta la citazione, un elenco di coloro che devono essere citati deve accompagnare l'opera.

6. **Integrità**: la licenza può richiedere, come condizione perché l'opera venga distribuita in forma modificata, che l'opera derivata abbia un nome o un numero di versione diverso dall'opera originaria.

7. **Nessuna discriminazione di persone o gruppi**: la licenza non deve discriminare alcuna persona o gruppo di persone.

8. **Nessuna discriminazione nei settori d'attività**: la licenza non deve impedire a nessuno di utilizzare l'opera in un determinato settore d'attività. Per esempio, la licenza non può impedire che l'opera sia utilizzata da un'azienda, o che venga utilizzata ai fini di ricerca genetica.

9. **Distribuzione della licenza**: i diritti relativi all'opera devono valere per tutte le persone a cui il programma viene ridistribuito senza che sia per loro necessario accettare o sottostare ad alcuna licenza aggiuntiva.

10. **La licenza non deve essere specifica per un pacchetto**: i diritti relativi all'opera non devono dipendere dal fatto che l'opera sia parte di un particolare pacchetto. Se l'opera viene estratta da quel pacchetto e usata o distribuita in conformità con i termini della licenza dell'opera, tutte le persone a cui il lavoro viene ridistribuito devono avere gli stessi diritti concessi in congiunzione con il pacchetto originario.

11. **La licenza non deve limitare la distribuzione di altre opere**: la licenza non deve imporre restrizioni su altre opere distribuite insieme all'opera licenziata. Per esempio, la licenza non deve insistere sul fatto che tutte le altre opere distribuite sullo stesso supporto siano aperte.

Tra i principali problemi che impediscono alla pratica dei dati aperti una larga diffusione possiamo citare a titolo di esempio il valore commerciale che gli stessi dati, visti sia in forma puntuale che aggregata, possono avere. Infatti i dati sono solitamente controllati da organizzazioni, siano esse pubbliche o private, le quali spesso sono reticenti di fronte alla possibilità di diffondere il proprio patrimonio informativo, ed effettuano pertanto delle forme di controllo sui dati, ad esempio attraverso limitazioni all'accesso, alle licenze con cui vengono rilasciati, ai diritti d'autore, ai brevetti e ai diritti di riutilizzo.
I sostenitori del movimento Open Data ritengono che tali restrizioni siano un limite al bene della comunità e che pertanto i dati dovrebbero essere resi disponibili senza alcuna restrizione o forma di pagamento, e soprattutto che siano riutilizzabili senza necessità di ulteriore autorizzazione; a sostengo di tale tesi apportano numerose argomentazioni:

- vi sono dati che appartengono al genere umano, quali i genomi, i dati sugli organismi, i dati ambientali e meteorologici, ecc. . . ;
- vi sono dati prodotti dalla pubblica amministrazione, che, poiché sono stati finanziati da denaro pubblico, devono essere restituiti ai contribuenti, e alla comunità in generale, sotto forma di dati aperti;
- le restrizioni sui dati e sul loro riutilizzo limitano lo sviluppo della comunità;
- i dati sono necessari per agevolare l'esecuzione di comuni attività umane (ad esempio i dati cartografici, le istituzioni pubbliche, ecc.);
- in campo scientifico il tasso di scoperta è accelerato da un migliore accesso ai dati: è essenziale che i dati scientifici siano resi aperti per fare in modo che la scienza sia più efficace e la società ottenga il massimo beneficio dalla ricerca scientifica.

In estrema sintesi si possono individuare alcuni aspetti che caratterizzano un insieme di dati come "aperto":

- i dati aperti devono essere indicizzati dai motori di ricerca;
- i dati aperti devono essere disponibili in un formato aperto, standardizzato e leggibile da un'applicazione informatica per facilitare la loro consultazione ed incentivare il loro riutilizzo anche in modo creativo;

- i dati aperti devono essere rilasciati attraverso licenze libere che non impediscano la diffusione e il riutilizzo da parte di tutti i soggetti interessati.

È possibile indicare un vasto numero di aree nelle quali i dati pubblici aperti stanno creando vantaggi; tra queste sono da evidenziare:

- trasparenza e controllo democratico;
- partecipazione;
- miglioramento o creazione di prodotti e servizi privati;
- innovazione;
- miglioramento dell'efficienza dei servizi pubblici;
- miglioramento dell'efficacia dei servizi pubblici;
- misurazione dell'impatto delle politiche pubbliche;
- estrazione di nuova conoscenza dalla combinazione di diverse fonti di dati e dall'identificazione di regolarità che emergono dall'analisi di grandi masse di dati.

Nell'ambito della trasparenza, esistono progetti come il sito britannico "Where does my money go" [Ope10] che permette di identificare in che modo i soldi delle tasse dei cittadini vengono utilizzati dal governo.

2.1.1 Iniziative internazionali

La principale spinta che ha contribuito all'affermarsi del movimento Open Data in ambito governativo è stata fornita dall'amministrazione americana del presidente Obama attraverso la promulgazione della Direttiva sull'Open Government nel dicembre 2009 [Exe09], secondo la quale "fin dove possibile e sottostando alle sole restrizioni valide, le agenzie devono pubblicare le informazioni on line utilizzando un formato aperto (open) che possa cioè essere recuperato, soggetto ad azioni di download, indicizzato e ricercato attraverso le applicazioni di ricerca web più comunemente utilizzate. Per formato open si intende un formato indipendente rispetto alla piattaforma, leggibile dall'elaboratore e reso disponibile al pubblico senza che sia impedito il riuso dell'informazione veicolata" [Exe09]. Tale direttiva è stata seguita dall'apertura di un opportuno sito pubblico [Gov09b], creato con l'obiettivo di raccogliere in un unico portale tutte le informazioni rese disponibili dagli enti statunitensi in formato aperto. Nei mesi successivi anche il Regno Unito ha aperto il proprio sito per gli Open Data [Gov09a], fortemente voluto e sponsorizzato da Tim Berners-Lee, l' "inventore" del World Wide Web. Successivamente la pratica degli Open Data e dei Data Store governativi si

è estesa in Australia [Aus11], in Canada [Gov11a], in Norvegia [Dir10] e in Francia [Gou11].

2.1.2 Open Data in Italia

In Italia, il portale del Governo italiano relativo agli Open Data [Gov11b] è stato reso disponibile on line nell'ottobre del 2011, dopo che alcune istituzioni avevano già provveduto a realizzare dei propri portali, come ad esempio la regione Piemonte [Pie10], nel maggio 2010 o la regione Emilia-Romagna [ER11] nel 2011. La versione beta della Italian Open Data License, rilasciata nell'ottobre 2010 dal Formez (Centro servizi, assistenza, studi e formazione per l'ammodernamento delle P.A.) era caratterizzata dalla restrizione per uso commerciale. Tuttavia, in seguito alle varie critiche ricevute, è stata presentata in maggio 2011 la versione definitiva 1.0 della licenza Italian Open Data License [FOR11], compatibile con i modelli di licenza Creative Commons 2.5 [Cre07a] e Open Data Commons [Ope09a] e dotata delle seguenti caratteristiche:

- è una licenza Share-Alike (ovvero è necessario mantenere i lavori derivati sotto la stessa licenza o una delle licenze compatibili);
- è compatibile (a senso unico) sia con Open Data Commons Open Database License (ODbL) [Ope09b] che con Creative Commons Attribution-ShareAlike (CC BY-SA) 3.0 [Cre07b]
- è pensata per la Pubblica Amministrazione.

Successivamente, nel marzo 2012 è stata rilasciata la versione 2.0 della Italian Open Data License [FOR12], priva di clausole del tipo "condividi-allo-stesso-modo" e con la sola richiesta di attribuzione della fonte per il riutilizzo dei dati. Esistono anche dei siti che si occupano di offrire un indice dei dataset italiani disponibili online; tra questi citiamo il servizio offerto da Open Knowledge Foundation Italia e Centro NEXA su Internet & Società del Politecnico di Torino [Ope11b], e i servizi analoghi gestiti dalla comunità del sito Spaghetti Open [Spa10] o dall'associazione LinkedOpenData.it [Lin12], l'Associazione italiana per l'Open Government [Ass11] ed il progetto OpenGeoData Italia [Ope12].

Dataset pubblicati in Italia

Il sito del Governo italiano relativo agli Open Data ha pubblicato nel mese di dicembre 2012 gli ultimi dati relativi al monitoraggio dello stato dell'Open Data in Italia. Per quanto riguarda il numero assoluto di dataset liberati dalle pubbliche amministrazioni il trend di crescita è positivo, rispetto alla prima rilevazione effettuata a marzo 2012. Se il trend generale è sempre stato positivo per tutte le rilevazioni effettuate, il tasso di crescita ha fatto

registrare un andamento variabile nel tempo, infatti, si passa da un incremento di 517 dataset del primo trimestre di rilevazioni (da marzo a maggio 2012) ad un incremento di 180 dataset rilevato nel trimestre da giugno ad agosto 2012; a novembre 2012 il numero totale di dataset disponibili era oltre 4000 unità.
Un aspetto interessante che emerge dai dati del monitoraggio è quello relativo al numero di dataset per livello di riusabilità. Si registra infatti una crescita quantitativa costante di tutte le tipologie di dataset ad eccezione di quelli resi disponibili in formato leggibile da applicazioni proprietarie; infatti alcune iniziative nella loro fase iniziale sperimentale hanno messo a disposizione degli utenti dati in formato proprietario, per rendere successivamente disponibili le stesse informazioni attraverso formati più evoluti e soprattutto "non proprietari". Oltre il 70% dei dati rilasciati dalle amministrazioni sono disponibili in formato machine-readable e non proprietario.
Da notare che quasi i due terzi dei dataset rilevati vengono distribuiti con licenze di tipo by (IODL 2.0 e CC BY), risultando ormai assodato che le pubbliche amministrazioni che decidono di intraprendere un percorso di apertura dei propri dati lo facciano con il preciso intento di favorire al meglio il riuso del proprio patrimonio informativo, senza porre limiti legali al riutilizzo se non la citazione della fonte di provenienza dei dati.
Una condizione riproposta costantemente ad ogni rilevazione è la chiara differenza tra il Nord e il Sud del nostro Paese in relazione alla distribuzione geografica delle piattaforme di diffusione dei dati aperti; escludendo le amministrazioni nazionali e alcune episodiche esperienze di amministrazioni locali del Sud, la maggior parte delle iniziative sono state intraprese da amministrazioni del Centro e del Nord Italia. L'Open Data potrebbe rappresentare una importante opportunità di crescita e rilancio, con investimenti anche ridotti, soprattutto per le regioni del Sud. La tendenza attuale evidenzia la scarsa attenzione, in alcune aree territoriali del Paese, verso benefici e fattori di sviluppo potenzialmente raggiungibili attraverso la messa a disposizione di tutti e il riutilizzo dei dati del settore pubblico.
Più nel dettaglio, le principali realtà che hanno contribuito alla pubblicazione di Open Data in Italia sono indicate all'interno della tabella 2.1 con indicato anche il numero di dataset rilasciati al dicembre 2012.

2.2 Sistemi di riferimento geografico

Un sistema di coordinate geografiche consente di identificare univocamente ogni punto della superficie terrestre attraverso un insieme di numeri; nella pratica è solitamente utilizzata la coppia formata da latitudine e longitudine, cui può essere eventualmente associata come terzo elemento l' altitudine.
Nel seguito saranno dapprima illustrati i concetti di latitudine e longitudine, congiuntamente ai concetti di parallelo e meridiano (funzionali alla tratta-

Istituzione	dataset
ISTAT	600
Regione Lombardia	446
Comune di Firenze	347
Regione Piemonte	335
Comune di Roma	291
Provincia di Lodi	255
Comune di Torino	247
CNR	240
Regione Veneto	187
Comune di Bologna	163
Provincia autonoma di Trento	160
INPS	146
Regione Liguria	118
Comune di Milano	107
Comune di Vicenza	79
Ministero della Salute	66
Comune di Venezia	56
Camera dei deputati	49

Tabella 2.1: Principali amministrazioni italiane che hanno rilasciato dataset Open Data, con relativo numero aggiornato al novembre 2012 [Gov12].

zione), quindi saranno descritti datum ed ellissoide di riferimento, sistemi utilizzati per misurare le coordinate geografiche di un punto sulla superficie della Terra.

2.2.1 Identificare univocamente un punto della Terra

In geografia la superficie della Terra viene solitamente suddivisa in un reticolato di paralleli e meridiani.
Un **parallelo** è una linea immaginaria perpendicolare all'asse terrestre (ovvero l'asse di rotazione della Terra); deve il proprio nome al fatto di essere geometricamente parallela all'**equatore**, la circonferenza massima della superficie della Terra, che viene pertanto detto anche parallelo di riferimento. I paralleli formano sulla Terra delle circonferenze immaginarie che risultano perpendicolari all'asse terrestre, e che sono più piccole mano a mano che si avvicinano ai due **poli** terrestri (ovvero ai punti sulla superficie terrestre posti in coincidenza dell'asse di rotazione e che sono generalmente indicati come Polo Nord e Polo Sud).
Un **meridiano** è invece un arco immaginario che congiunge i due poli terrestri; si ottiene suddividendo in due parti una qualsiasi circonferenza massima - ottenuta dall'intersezione tra la superficie della Terra e un ipotetico piano passante per l' asse di rotazione – in due archi diametralmente opposti (detti

rispettivamente *meridiano* e *antimeridiano*) e aventi come estremi il Polo Nord ed il Polo Sud. Tutti i meridiani hanno uguale lunghezza e misurano 20.003,93 km. Generalmente vengono considerati 360 meridiani principali, di cui 180 sono meridiani effettivi e i restanti 180 antimeridiani. Il meridiano di riferimento (o meridiano 0) è quello passante per l'Osservatorio di Greenwich, alla periferia di Londra, Gran Bretagna, e viene pertanto detto anche Meridiano di Greenwich.
La **latitudine** di un punto è la coordinata geografica che ne indica la distanza angolare dall'equatore (ovvero l'angolo che la verticale di un punto sulla superficie della Terra forma con il piano equatoriale).Tale valore viene espresso utilizzando i gradi sessagesimali e può assumere valori compresi nell'intervallo tra 0 a 90° N (ovvero a nord dell'equatore, in quello che viene comunemente detto *emisfero boreale*) e da 0 a 90° S (ovvero a sud dell'equatore, in quello che viene comunemente detto *emisfero australe*). L'indicazione dell'emisfero nord (N) o sud (S) può essere sostituita utilizzando valori positivi per le latitudini dell'emisfero nord e valori negativi per quelle dell'emisfero sud.
La **longitudine** di un punto è la coordinata geografica che ne indica la distanza angolare in senso est o ovest dal meridiano considerato fondamentale [2]. Il valore della longitudine viene misurato in gradi sessagesimali su un piano perpendicolare all'asse terrestre e può assumere valori nell'intervallo da 0 a 180° E e da 0 a 180° O. L'indicazione dell'emisfero est (E) o ovest (O) può essere sostituita utilizzando valori positivi per le latitudini dell'emisfero est e valori negativi per quelle dell'emisfero ovest.
L'**altitudine** (o quota) è la distanza espressa in unità di lunghezza (generalmente metri), misurata lungo la verticale del punto considerato sulla superficie terrestre dal livello del mare.

2.2.2 Datum ed ellissoidi di approssimazione del geoide

Il pianeta Terra ha una forma irregolare (cui viene assegnato il nome *geoide*) che solitamente è assimilata a quella di un ellissoide, in modo che quest'ultimo approssimi bene la superficie del pianeta (soprattutto per quanto riguarda le quote); tuttavia vista l'estrema variabilità della superficie terrestre, sono state proposte diverse forme ellissoidiche. Solitamente gli ellissoidi vengono orientati localmente per una superficie terrestre riconducibile a quella di una regione, una nazione o un continente; pertanto quando si utilizzano le coordinate geografiche risulta importante menzionare anche il sistema geodetico di riferimento in base al quale vengono effettuate le misure, denominato **datum**. Esso viene definito dall'ellissoide di riferimento (modello della figura della terra) e dal centro di emanazione [3]. Per effettuare

[2] Come già illustrato, generalmente viene considerato come meridiano fondamentale quello passante per l'Osservatorio di Greenwich, Gran Bretagna.
[3] Il centro di emanazione è il punto dove ellissoide e geoide sono tangenti.

il passaggio da un datum ad un altro si possono applicare diverse tecniche di trasformazione, che tuttavia comportano sempre una certa approssimazione. Il datum utilizzato risulta quindi essere un'informazione fondamentale all'interno delle applicazioni GIS [4] in quanto è l'informazione attraverso la quale diviene possibile localizzare la cartografia utilizzata e sovrapporla correttamente con altra cartografia proveniente da diversi soggetti.
Nel corso degli ultimi tre secoli sono stati suggeriti diversi datum; in particolare in Europa, dopo il termine della seconda guerra mondiale, fu introdotto l'**ED50** (acronimo di European Datum 1950), basato sul geoide di Hayford (1909) e con centro di emanazione presso Potsdam, Germania. Tale datum fu definito per la connessione delle reti geodetiche europee, dopo che nel corso delle battaglie della seconda guerra mondiale erano emerse incongruenze tra i sistemi di riferimento utilizzati tra Germania e Francia. Tale sistema fu utilizzato in gran parte dell'Europa Occidentale, ad eccezione di Gran Bretagna, Irlanda, Svezia e Svizzera, che continuarono ad utilizzare un proprio datum. L'ED50 fu aggiornato alla fine degli anni '70 dello scorso secolo, dapprima nell'ED77 e poi nell'ED79.
In Italia fu utilizzato per molto tempo anche il datum **Roma 40**, riferito ai dati astronomici disponibili nel 1940, anch'esso basato sul geoide di Hayford, ma con centro di emanazione a Roma Monte Mario, e che utilizzava come meridiano fondamentale quello passante per il medesimo Monte Mario.
Attualmente viene comunemente utilizzato il datum denominato **WGS84** (acronimo di World Geodetic System 1984) e l'omonimo ellissoide correlato, che utilizza come fondamentale il meridiano di Greenwich e non ha bisogno di un centro di emanazione, a causa della coincidenza tra il centro del geoide e il centro dell'ellissoide utilizzato. Questo standard è impiegato anche dal GPS (Global Positioning System), la rete di satellite utilizzata dai comuni navigatori per automobile. Nel 1996 il WGS84 è stato lievemente aggiornato e corretto in quello che è noto come Earth Gravitational Model 1996 o EGM96.

2.3 Rappresentazione di informazioni geografiche

In ambito informatico, la rappresentazione di informazioni geografiche può avvenire utilizzando una delle due principali tipologie di dato esistenti:

- i dati vettoriali, costituiti da elementi semplici quali punti, linee e poligoni, codificati e memorizzati sulla base delle loro coordinate. Per individuare un punto all'interno di un sistema informativo geografico è sufficiente specificare le sue coordinate (x, y); invece per individuare

[4] Un GIS (acronimo di Geographic Information System) è un sistema informativo computerizzato che permette di acquisire, memorizzare, analizzare, visualizzare e rappresentare le informazioni derivanti da dati geografici.

una linea o un poligono occorre indicare la coordinate (x_i, y_i) di ogni nodo i;

- i dati raster, che consentono di rappresentare il mondo reale attraverso una matrice di celle, generalmente di forma quadrata o rettangolare, dette pixel, a ciascuno dei quali sono associate le informazione relative a ciò che esso rappresenta sul territorio.

L'esistenza delle due diverse tipologie di dati è dovuta al fatto che esse si prestano ad usi diversi; mentre la cartografia vettoriale risulta adatta alla rappresentazione di dati che variano in modo discreto (quali l'ubicazione delle fermate delle linee autobus o la rappresentazione di percorsi), la cartografia raster è più adatta alla rappresentazione di dati con variabilità continua (come ad esempio un modello digitale di elevazione).
Per quanto riguarda i dati vettoriali, è possibile elencare un ampio numero di formati di rappresentazione, tra i quali:

- il KML (o Keyhole Markup Language), illustrato nel paragrafo 2.3.1;
- il GML (o Geography Markup Language), la grammatica XML definita dall'Open Geospatial Consortium per descrivere elementi geografici e divenuto standard ISO 19136 [ISO07] nell'agosto 2007;
- il GeoJSON, illustrato nel paragrafo 2.3.3;
- gli Shapefile, utilizzati all'interno dei principali software GIS; tale formato è illustrato nel paragrafo 2.3.2

Nel corso della presente trattazione saranno illustrati in particolare i formati KML, Shapefile e GeoJSON, dato che questi sono i principali standard utilizzati dal Comune di Firenze per la rappresentazione dei propri Open Data georeferenziati (oggetto di elaborazione, così come descritto al capitolo 8).

2.3.1 Keyhole Markup Language

Il Keyhole Markup Language (generalmente abbreviato con KML) è un linguaggio basato su XML creato al fine di esprimere l'annotazione e la visualizzazione di dati geospaziali all'interno del programma Google Earth (in versione tridimensionale) o della web application Google Maps (in versione bidimensionale) [Goo12c].
KML fu sviluppato per essere utilizzato all'interno di Google Earth, il cui nome originale era Keyhole Earth Viewer, software sviluppato da Keyhole Inc., che fu acquistata da Google nel 2004.
KML è divenuto uno standard internazionale dell'Open Geospatial Consortium nel 2008 [Ope08c, Ope08a] e condivide parte della sua grammatica strutturale con il GML.

Ogni file KML specifica un insieme di elementi (segnalibri geografici, immagini, poligoni, modelli 3D, descrizioni ed etichette testuali, ecc...) da visualizzare in Google Earth e Google Maps (sia nella versione web che in quella mobile). Ogni luogo deve possedere obbligatoriamente una latitudine e una longitudine; inoltre possono essere specificati altri dati al fine di rendere la visualizzazione più specifica, come l'inclinazione, inquadratura e la quota del punto di vista che insieme definiscono una vista (in inglese *camera view*).
Spesso i file KML sono distribuiti come KMZ, ovvero file compressi *.zip* cui l'estensione è stata modificata in *.kmz*; in particolare deve essere utilizzata una compressione legacy (ZIP 2.0) al fine di risultare compatibile con tutti i programmi di visualizzazione dei dati geometrici. Solitamente all'interno di un file KMZ si trova un solo file KML (chiamato generalmente, ma non necessariamente *doc.kml*) ed eventualmente opportune directory con i vari strati (o overlay), immagini o icone cui viene fatto riferimento all'interno del file KML [Goo12e].
I documenti KML sono conformi ad un opportuno XML Schema [Ope08b].
Il MIME Type associato con il formato KML è

application/vnd.google-earth.kml+xml,

mentre il MIME Type associato con il formato KMZ è

application/vnd.google-earth.kmz [Goo12d].

Il KML usa un sistema di riferimento con coordinate geografiche tridimensionali, esprimendo nell'ordine longitudine, latitudine e altitudine, con valori negativi per ovest, sud e sotto il livello del mare (nel caso in cui l'altitudine sia disponibile). In particolare, la longitudine e la latitude sono definite secondo il datum WGS84 illustrato nel paragrafo 2.2.2, mentre l'altitudine è misurata dal datum EGM96 (anche esso descritto all'interno del paragrafo 2.2.2).

Rappresentazione di un documento KML elementare

Generalmente all'interno di un documento KML sono presenti uno o più **Placemark**, ovvero posizioni sulla superficie del pianeta Terra.
La struttura di un elemento *Placemark* è generalmente così composta:

- un tag <name>, utilizzato come etichetta per il *Placemark*;
- un tag <description> che viene visualizzato all'interno di un apposito fumetto all'interno di Google Earth o Google Maps;
- uno o più elementi geometrici, quali <Point>, <LineString> , <Polygon> o <MultiGeometry>, descritti nel seguito.

Point L'elemento <Point> rappresenta un particolare punto sulla superficie della Terra.
Il listato 2.1 illustra un semplice documento contenente un solo *Placemark*, rappresentato da un *Point*.

Listato 2.1: File KML rappresentante un *Placemark*.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Placemark>
      <name>Duomo di Firenze</name>
      <description>Il Duomo di Santa Maria del Fiore in Firenze
        un luogo di arte e di fede meraviglioso</description>
      <Point>
        <coordinates>11.2553,43.7731,0</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

LineString L'elemento <LineString> rappresenta un particolare percorso sulla superficie della Terra.
Il listato 2.2 illustra un semplice documento contenente un solo Placemark, rappresentato da un *LineString*.

Listato 2.2: File KML rappresentante un *LineString*.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Placemark id="ciclabile-morgagni">
      <name>Pista ciclabile Viale Morgagni</name>
      <description>
          La pista ciclabile di Firenze che collega
        il quartiere di Rifredi con l'Ospedale di Careggi
      </description>
      <LineString>
        <coordinates>
          11.24583854,43.8030306 11.24582352,43.80301508
          11.24564693,43.80257903 11.24547063,43.80215058
          11.24538258,43.80190401 11.24529971,43.80169064
          11.24512299,43.80128599 11.24491629,43.80085243
          11.24469446,43.80047054 11.24457276,43.8002722
          11.2443912,43.7999837 11.24435774,43.79993204
          11.24321264,43.79842952 11.24312224,43.79834674
        </coordinates>
      </LineString>
    </Placemark>
  </Document>
</kml>
```

Polygon L'elemento <Polygon> rappresenta una particolare area sulla superficie della Terra (eventualmente forata).
Il bordo esterno del poligono viene rappresentato attraverso l'utilizzo del tag <outerBoundaryIs>, mentre ciascun foro del poligono può essere incluso all'interno del tag <innerBoundaryIs>.
Il listato 2.3 illustra un semplice documento contenente un solo *Placemark*, rappresentato da un *Polygon*, dotato di un foro.

Listato 2.3: File KML rappresentante un *Polygon*.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Placemark id="piazza-san-giovanni">
      <name>Piazza di San Giovanni</name>
      <description>La Piazza di San Giovanni, nel centro storico
        di Firenze, circonda l'omonimo Battistero.
      </description>
      <Polygon>
        <outerBoundaryIs>
          <LinearRing>
            <coordinates>
              11.25450565,43.77352202 11.25448868,43.77352362
              11.2544774,43.77347783 11.25446108,43.77339749
              11.25441253,43.77303777 11.2543942,43.77290389
              11.25455562,43.77271271 11.25466014,43.77279267
              11.25533351,43.77281422 11.25530866,43.77291073
              11.25532196,43.77334997 11.25525997,43.77346239
              11.25505706,43.77348548 11.25482178,43.77350984
              11.25459561,43.77352205 11.25450565,43.77352202
            </coordinates>
          </LinearRing>
        </outerBoundaryIs>
        <innerBoundaryIs>
          <LinearRing>
            <coordinates>
              11.25478891,43.77309577 11.25478904,43.7732074
              11.25484378,43.77320632 11.25484263,43.77322984
              11.25495316,43.77330991 11.25512558,43.77330945
              11.25525206,43.77322029 11.25524934,43.77309403
              11.2551244,43.77300836 11.25494996,43.77300887
              11.25484364,43.77309469 11.25478891,43.77309577
            </coordinates>
          </LinearRing>
        </innerBoundaryIs>
      </Polygon>
    </Placemark>
  </Document>
</kml>
```

MultiGeometry L'elemento <MultiGeometry> rappresenta un insieme contenente oggetti appartenenti alle categorie descritte sinora.

Descrizione in HTML All'interno del tag <description>, oltre al semplice testo grezzo, è possibile inserire codice HTML in due diverse modalità:

- formattato in un elemento CDATA;
- scrivendo le parentesi angolari come riferimento di entità (ovvero il simbolo > deve essere scritto come `>` e il simbolo < come `<`)

Nel listato 2.4 alcune informazioni HTML sono state inserite all'interno di un elemento CDATA, mentre nel listato 2.5 è rappresentato lo stesso documento, utilizzando il codice HTML con i riferimenti di entità.

Listato 2.4: File KML con HTML in un elemento CDATA.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Placemark>
      <name>Duomo di Firenze</name>
      <description>
        <![CDATA[
          <h1>Duomo</h1>
          <p>Il <b>Duomo di Santa Maria del Fiore</b> in Firenze,
          un luogo di arte e di fede meraviglioso</p>
        ]]>
      </description>
      <Point>
        <coordinates>11.2553,43.7731,0</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

Listato 2.5: File KML con HTML e riferimenti di entità.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Document>
    <Placemark>
      <name>Duomo di Firenze</name>
      <description>
        &gt;h1&lt;Duomo&gt;/h1&lt;
        &gt;p&lt;Il &gt;b&lt;Duomo di Santa Maria
        del Fiore&gt;/b&lt; in Firenze, un luogo
        di arte e di fede meraviglioso&gt;/p&lt;
      </description>
      <Point>
        <coordinates>11.2553,43.7731,0</coordinates>
      </Point>
    </Placemark>
  </Document>
</kml>
```

2.3.2 ShapeFile

Lo Shapefile ESRI [ESR98] è un formato di dati vettoriale per sistemi informativi geografici divenuto ormai uno standard de facto e come tale utilizzato da una grande varietà di sistemi GIS; è regolato da ESRI come una specifica (quasi) aperta.
Uno Shapefile descrive geometrie spaziali, ovvero punti, polilinee [5] e poligoni; questi possono ad esempio rappresentare rispettivamente pozzi, fiumi e laghi. Ogni elemento può essere caratterizzato da degli attributi che lo descrivono, quali ad esempio il nome o la temperatura.
Più correttamente il termine "Shapefile" indica un insieme di file che condividono lo stesso nome. Ogni file è conforme alla convenzione per i nomi MS-DOS 8.3 [6] per essere compatibile con la maggior parte delle applicazioni. Per ogni "Shapefile" sono obbligatori tre files:

- il file *.shp* che conserva le geometrie vere e proprie;
- il file *.shx* che conserva l'indice posizionale delle geometrie per permettere di cercare una forma rapidamente;
- il file *.dbf* che conserva il database degli attributi di ciascuna forma.

Tra i vari files opzionali di uno "Shapefile" occorre citare il file *.prj* che conserva l'informazione sul sistema di coordinate, espresso in Well-Known Text [7].

2.3.3 GeoJSON

GeoJSON è un formato aperto per la codifica e la memorizzazione di strutture dati geografiche [BDD+08] leggibile dagli essere umani (o *human-readable*). Tale nome è dovuto al fatto di essere basato sul formato di interscambio dati JSON (JavaScript Object Notation) [Cro06], ovvero ogni struttura dati GeoJSON è al tempo stesso un oggetto JSON e risulta pertanto possibile utilizzare gli strumenti JSON disponibili per processare tali informazioni. Una struttura dati GeoJSON consiste in una collezione di coppie nome/-valore, noti anche come *membri*; per ogni membro il nome è sempre una

[5] Con il termine *polilinea* si intende una curva, specificata da una sequenza di punti detti vertici, tali che la curva consiste in una serie di segmenti di linea che connettono i vertici consecutivi.

[6] La convenzione per i nomi MS-DOS 8.3 specifica che i nomi dei file debbano avere un numero massimo di 8 caratteri di prefisso, seguito da un punto, e massimo 3 caratteri di suffisso o *estensione*.

[7] Il Well-Known Text (spesso abbreviato in WKT) è un linguaggio a markup per rappresentare oggetti di geometria vettoriale su una mappa. Il formato fu definito in origine dall'Open Geospatial Consortium (OGC) e descritto nelle specifiche Simple Feature Access [Ope11a, ISO04]; attualmente trova definizione nello standard ISO/IEC 13249-3:2011 [ISO11, ME01].

stringa, mentre il valore può essere una stringa, un numero, un oggetto, un array o una parola chiave tra: *true*, *false* e *null* [8].
All'interno di ogni oggetto GeoJSON ogni posizione è rappresentata da un array di numeri composto da due o tre elementi, espressi nell'ordine x, y, z a rappresentare longitudine, latitudine ed altitudine in un sistema di riferimento geografico.
Attraverso l'utilizzo del membro *crs* è anche possibile specificare un datum; se tale proprietà non è definita, GeoJSON userà per default il geoide WGS84.
Un oggetto GeoJSON può rappresentare:

- una geometria;
- una *feature*;
- una collezione di *feature*;

ognuno dei quali è analizzato nel seguito.
Una **geometria** può appartenere ad una delle sette seguenti tipologie:

- **Point**: rappresenta un particolare punto su una superficie (ad esempio quella del pianeta Terra); il membro *coordinate* è costituito da una singola posizione. Il listato 2.6 illustra un documento GeoJSON contenente un elemento *Point*.

Listato 2.6: File GeoJSON rappresentante un punto.

```
{
  "type": "Point",
  "coordinates": [11.2553,43.7731]
}
```

- **LineString**: rappresenta un particolare percorso su una superficie (ad esempio quella del pianeta Terra); il membro *coordinate* è costituito da un array di due o più posizioni. Il listato 2.7 illustra un documento GeoJSON contenente un elemento *LineString*.

Listato 2.7: File GeoJSON rappresentante una linea.

```
{
  "type": "LineString",
  "coordinates":
    [[11.24583854,43.8030306],[11.24582352,43.80301508],
    [11.24564693,43.80257903],[11.24547063,43.80215058],
    [11.24538258,43.80190401],[11.24529971,43.80169064],
    [11.24512299,43.80128599],[11.24491629,43.80085243],
    [11.24469446,43.80047054],[11.24457276,43.8002722],
    [11.2443912,43.7999837],[11.24435774,43.79993204],
    [11.24321264,43.79842952],[11.24312224,43.79834674]]
}
```

[8] JSON e la terminologia qui utilizzata sono descritti all'interno della IETF RFC 4627 [Cro06].

- **Polygon**: rappresenta una particolare area su una superficie (ad esempio quella del pianeta Terra), eventualmente forata; il membro *coordinate* è costituito da un array di **LinearRing**, ovvero di *LineString* con 4 o più posizioni, la prima e l'ultima delle quali sono equivalenti. I poligoni forati sono rappresentati da più di un LinearRing, intendendo il primo come anello esterno ed gli altri come anelli interni (o fori). Il listato 2.8 illustra un documento GeoJSON contenente un elemento *Polygon* in cui è presente anche un foro.

Listato 2.8: File GeoJSON rappresentante un poligono.

```
{
  "type": "Polygon",
  "coordinates": [
    [[11.25450565,43.77352202],[11.25448868,43.77352362],
      [11.2544774,43.77347783],[11.25446108,43.77339749],
      [11.25441253,43.77303777],[11.2543942,43.77290389],
      [11.25455562,43.77271271],[11.25466014,43.77279267],
      [11.25533351,43.77281422],[11.25530866,43.77291073],
      [11.25532196,43.77334997],[11.25525997,43.77346239],
      [11.25505706,43.77348548],[11.25482178,43.77350984],
      [11.25459561,43.77352205],[11.25450565,43.77352202]],
    [[11.25478891,43.77309577],[11.25478904,43.7732074],
      [11.25484378,43.77320632],[11.25484263,43.77322984],
      [11.25495316,43.77330991],[11.25512558,43.77330945],
      [11.25525206,43.77322029],[11.25524934,43.77309403],
      [11.2551244,43.77300836],[11.25494996,43.77300887],
      [11.25484364,43.77309469],[11.25478891,43.77309577]]
  ]
}
```

- **MultiPoint**: rappresenta un insieme di punti su una superficie (ad esempio quella del pianeta Terra); il membro *coordinate* è costituito da un array di coordinate *Point*.

- **MultiLineString**: rappresenta un insieme di percorsi su una superficie (ad esempio quella del pianeta Terra); il membro *coordinate* è costituito da un array di coordinate *LineString*.

- **MultiPolygon**: rappresenta un insieme di aree su una superficie (ad esempio quella del pianeta Terra), eventualmente forate; il membro *coordinate* è costituito da un array di coordinate *Polygon*.

- **GeometryCollection**: rappresenta una collezione di oggetti geometrici delle tipologie finora citate; come tale deve possedere un membro di nome *geometries*, il cui valore è un array di oggetti geometrici. Il listato 2.9 illustra un documento GeoJSON contenente un elemento *GeometryCollection*.

Listato 2.9: File GeoJSON rappresentante una collezione.

```
{
  "type": "GeometryCollection",
  "geometries": [
  {
    "type": "Point",
    "coordinates": [11.2553,43.7731]
  },
  {
    "type": "LineString",
    "coordinates":
      [[11.24583854,43.8030306],[11.24582352,43.80301508],
      [11.24564693,43.80257903],[11.24547063,43.80215058],
      [11.24538258,43.80190401],[11.24529971,43.80169064],
      [11.24512299,43.80128599],[11.24491629,43.80085243],
      [11.24469446,43.80047054],[11.24457276,43.8002722],
      [11.2443912,43.7999837],[11.24435774,43.79993204],
      [11.24321264,43.79842952],[11.24312224,43.79834674]]
  }
  ]
}
```

Una **Feature** è un oggetto GeoJSON composto da:

- un membro *geometry*, avente come valore o un oggetto geometrico (ovvero una geometria come prima definita) o il valore *null*;
- un membro *properties*, avente come valore un qualsiasi oggetto JSON o il valore *null*;
- opzionalmente un membro *id*, con l'identificativo della feature.

Il listato 2.10 illustra un documento GeoJSON contenente un elemento *Feature*.

Listato 2.10: File GeoJSON rappresentante un elemento *Feature*.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [11.2553,43.7731]
  },
  "properties": {
    "name": "Duomo di Firenze",
    "description": "Il Duomo di Santa Maria del Fiore in Firenze,
       un luogo di arte e di fede meraviglioso"
  }
}
```

Una collezione di *Feature* (o **FeatureCollection**) rappresenta un elenco di *features*; come tale è un oggetto GeoJSON composto da un membro *features*, avente come valore un array di *features*.

Il listato 2.11 illustra un documento GeoJSON contenente un elemento *FeatureCollection*.

Listato 2.11: File GeoJSON rappresentante un elemento *FeatureCollection*.

```
{
  "type": "FeatureCollection",
  "features": [
  {
    "type": "Feature",
    "geometry": {
      "type": "Point",
      "coordinates": [11.2553,43.7731]
    },
    "properties": {
    "name": "Duomo di Firenze",
      "description": "Il Duomo di Santa Maria del Fiore in Firenze,
        un luogo di arte e di fede meraviglioso"
      }
    },
  {
    "type": "Feature",
    "geometry": {
      "type": "Polygon",
      "coordinates": [
      [[11.25450565,43.77352202],[11.25448868,43.77352362],
        [11.2544774,43.77347783],[11.25446108,43.77339749],
        [11.25441253,43.77303777],[11.2543942,43.77290389],
        [11.25455562,43.77271271],[11.25466014,43.77279267],
        [11.25533351,43.77281422],[11.25530866,43.77291073],
        [11.25532196,43.77334997],[11.25525997,43.77346239],
        [11.25505706,43.77348548],[11.25482178,43.77350984],
        [11.25459561,43.77352205],[11.25450565,43.77352202]],
      [[11.25478891,43.77309577],[11.25478904,43.7732074],
        [11.25484378,43.77320632],[11.25484263,43.77322984],
        [11.25495316,43.77330991],[11.25512558,43.77330945],
        [11.25525206,43.77322029],[11.25524934,43.77309403],
        [11.2551244,43.77300836],[11.25494996,43.77300887],
        [11.25484364,43.77309469],[11.25478891,43.77309577]]
      ]
    },
    "properties": {
    "name": "Piazza di San Giovanni",
    "description": "La Piazza di San Giovanni, nel centro storico
      di Firenze, circonda l'omonimo Battistero."
    }
    }
  ]
}
```

2.4 Open Data del Comune di Firenze

All'interno del paragrafo 2.1.2 è stato illustrato come varie realtà italiane abbiano reso pubblici i dati a loro disposizione, secondo i suggerimenti del movimento Open Data; in particolare, tra le varie istituzioni risulta interessante analizzare il caso del Comune di Firenze [Com12a].
La Città di Firenze, capoluogo dell'omonima provincia e della regione Toscana, vanta una popolazione di 378.236 abitanti [9] [Uff12], configurandosi quindi come ottavo comune italiano per popolazione ed il primo della propria regione. All'interno della superficie del comune, ed in particolar modo nell'area del centro storico un tempo compreso tra le mura arnolfiane (oggi sostituite in larga parte dai viali di circonvallazione), riconosciuto come patrimonio dell'umanità dall'UNESCO, trova sede un numero altissimo di opere d'arte, che vanno dai monumenti religiosi – difficile elencarli tutti, tra i maggiori e più importanti possono essere annoverati la Cattedrale di Santa Maria del Fiore, il Battistero di San Giovanni, il Campanile di Giotto, la Basilica di Santa Croce – e civili – quali Palazzo Vecchio, Piazza della Signoria, il Ponte Vecchio, il Palazzo Pitti, il Palazzo del Bargello – alle opere raccolte all'interno di importanti musei come la Galleria degli Uffizi.
Una tale quantità di opere d'interesse artistico/culturale ben si presta a finalità di interesse turistico, e come tale alla possibilità di realizzare applicazioni e/o servizi che utilizzino informazioni ad esso relative.
L'amministrazione comunale della città ha da tempo avviato un percorso di condivisione dei propri progetti con la cittadinanza, mediante la metafora dei 100 luoghi [Com11], ed ha inoltre portato avanti da alcuni anni un processo di razionalizzazione e bonifica del proprio patrimonio informativo, che va ben oltre le informazioni artistico culturali appena citate, spingendosi in tutti i campi di interesse della vita cittadina. Seguendo la metodologia tipica dell'Open Government, ha quindi pubblicato on-line [Com12b] un elenco di dataset che sono resi disponibili in formati aperti ed utilizzabili dalle comunità legate al mondo degli Open Data.
In particolare, a dicembre 2012 il Comune di Firenze aveva pubblicato circa 350 dataset; la tabella 2.2 elenca alcuni tra i più interessanti dataset pubblicati, mantenendo la suddivisione per categorie utilizzata dal Comune di Firenze.
I vari dataset sono stati pubblicati in uno o più dei seguenti formati:

- CSV, per quanto riguarda i dati alfanumerici o tabellari;
- KML o KMZ, per quanto riguarda i dati georeferenziati;
- Shapefile, per quanto riguarda i dati georeferenziati;
- RDF, per i soli dataset musei, sinistri, toponomastica e viario.

[9] Il dato relativo alla popolazione è aggiornato all'ottobre 2012.

Categoria	**Principali dataset**
ambiente	agenti inquinanti rilevati, alberi pubblici, aree per la sgambatura dei cani, aree verdi, fontane pubbliche, impianti fotovoltaici
amministrazione	dati di bilancio a partire dal 2003, informazioni relative ai dipendenti e alle ordinanze pubbliche, sedi e sezioni elettorali
attività economiche	esercizi di vicinato, medie e grandi strutture di vendita, variazioni annue del numero di attività commerciali
ciclabilità	percorsi delle piste ciclabili, punti per il noleggio di biciclette, rastrelliere
cultura e turismo	biblioteche, consolati, musei, uffici di informazioni turistica, teatri, numero di biglietti venduti nei musei per anno, bagni pubblici
dati sul territorio	associazioni di pubblica assistenza, disposizione dei numeri civici delle abitazioni, numero dei transiti annui per ogni casello autostradale, lapidi, incroci cui è stato storicamente assegnato un nome (canti), informazioni relative alla popolazione residente e al lavoro, stradario di tutte le vie del comune
istruzione	iscrizioni presso asili nido e centri estivi
mobilità e sicurezza	confini delle aree pedonali o a traffico limitato, stazioni ferroviarie, colonnine per la ricarica dei veicoli elettrici, sinistri per via, stalli di sosta a pagamento
opere pubbliche	lavori in corso e lavori terminati
reti	luoghi in cui è disponibile un servizio gratuito di connessione ad internet in modalità senza fili
sanità e sociale	centri anziani, centri assistenziali per disabili fisici o psichici, cimiteri, farmacie, ospedali, presidi ASL
sport	aree sportive, percorsi jogging, società sportive
urbanistica	quartieri, centri abitati

Tabella 2.2: Principali dataset Open Data pubblicati dal Comune di Firenze.

Parte II

InterDataNet

InterDataNet

Il presente lavoro è situato all'interno di InterDataNet (o più brevemente IDN), una infrastruttura per la collaborazione ed il riuso dei dati [Inn08, Chi09, Cio10].

Secondo il lavoro di Ciofi, InterDataNet può essere definita come un framework che consente ad utenti "distribuiti" nel tempo e nello spazio di collaborare intorno ad elementi informativi che appartengono ad uno spazio globale di dati distribuito con cui è possibile interagire mediante la metafora dei documenti [Cio10].

Un'altra definizione è quella proposta da Innocenti, secondo il quale, InterDataNet può essere visto come una architettura per l'integrazione delle risorse e delle informazioni con l'obiettivo di sviluppare una infrastruttura per la cooperazione applicativa tra organizzazioni eterogenee e distribuite in grado di abilitare la collaborazione tra utenti [Inn08].

InterDataNet si muove nella direzione del Web, acquisendone i principi fondanti per estenderne le funzionalità al fine di far evolvere il Web stesso in modo da renderlo non solo l'ambiente principe per la pubblicazione di informazione fruibile da umani, ma una piattaforma sopra la quale poter collaborare in modo efficace ed efficiente.

Come si vedrà nel seguito, risulta vantaggioso ai fini della progettazione, introdurre livelli di servizi i quali siano in grado di astrarre le operazioni di memorizzazione, elaborazione e comunicazione dell'informazione, che avvengono ai livelli inferiori; tale astrazione consente di abbattere e perfino eliminare la complessità dei processi a livello applicativo e lascia maggiore spazio per la realizzazione di applicazioni a valore aggiunto.

Seguendo quanto suggerito da altri lavori [Val00, EM04, WN01], all'interno della definizione dell'infrastruttura InterDataNet sono state adottate viste multiple del problema generale, con il fine di separare i comportamenti e le funzionalità e di formulare vincoli, requisiti e soluzioni.

Pertanto l'intero sistema IDN è descritto (logicamente e fisicamente) attraverso l'utilizzo di tre viste, che risultano complementari ed integrate tra loro. Ogni vista pone l'attenzione su degli aspetti specifici del problema, partendo dalla rappresentazione astratta del contesto, delle informazioni e delle esigenze, per arrivare verso un livello più concreto relativo all'in-

troduzione dell'architettura fino alla descrizione degli apparati necessari all'implementazione.
Le tre viste considerate, che vengono analizzate nel seguito, sono:

1. vista sulle risorse e sull'informazione;
2. vista sull'architettura dei servizi;
3. vista sulle applicazioni.

La prima vista prende il nome di **IDN Information Model (IDN-IM)**; come sarà illustrato all'interno del capitolo 3, si tratta di un modello dell'informazione [PS03, CT04, HSGT94] finalizzato a rappresentare dati, recependo i principi di responsabilità, paternità, storicizzazione e ciclo di vita dell'informazione.
Tale vista definisce le proprietà di base dell'informazione che dovrà essere trattata, ma volutamente non fornisce specifiche soluzioni implementative, ovvero rappresenta l'informazione indipendentemente dalla tecnologia ed in tal modo abilita l'interoperabilità fra sistemi eterogenei.
La seconda vista prende il nome di **IDN Service Architecture (IDN-SA)**; come sarà illustrato all'interno del capitolo 4, essa definisce in maniera stratificata i servizi necessari al fine di soddisfare il modello informativo IDN-IM, in modo conforme ai principi REST [Fie00], illustrati al capitolo 1.
IDN-SA implementa le varie funzionalità, mediante la definizione di sottosistemi, protocolli ed interfacce per una gestione collaborativa del documento rappresentato tramite IDN-IM; inoltre espone una IDN-API (Application Programming Interface) REST che può essere utilizzata dalle applicazioni (al fine di manipolare informazioni rappresentate tramite IDN-IM) ricorrendo direttamente al protocollo HTTP; in tal modo è fornita la garanzia di scalabilità ed interoperabilità. L'approccio REST è utilizzato anche per le interazioni che si sviluppano fra le componenti interne di IDN-SA stessa; questo avviene sicuramente fra componenti appartenenti a domini diversi in cui è necessario interagire tramite interfacce standard, ma al fine di garantire la massima interoperabilità è auspicabile che ciò avvenga in ogni circostanza.
La terza vista è costituita infine dalle applicazioni IDN o **IDN Compliant Application** le quali implementano le logiche di business per la risoluzione di specifici problemi di dominio e/o rappresentano lo user-agent per l'interazione con il sistema. Le applicazioni IDN operano trattando informazione conforme ai dettami fissati dall'Information Model ed utilizzano l'IDN-SA per accedere ai documenti IDN-IM ed eventualmente manipolarli. Il paragrafo 8.1 contiene una breve presentazione di tale vista.

Capitolo 3

InterDataNet Information Model

InterDataNet Information Model (IDN-IM) si pone l'obiettivo di mettere in discussione il concetto tradizionale di documento generalmente considerato come monolitico e non strutturato, il quale limita le possibilità di soddisfare i requisiti di qualità del dato elettronico e la reale distribuzione dell'informazione [Inn08].

L'utilizzo di questo modello informativo permette di realizzare uno spazio globale di dati distribuiti su quali è possibile interagire mediante il ricorso alle cosiddette operazioni CRUD [1] e dove è abilita una serie di proprietà collaborative a livello infrastrutturale.

Anche il movimento Linked Data persegue un simile obiettivo ma ritiene necessario utilizzare un diverso modello informativo, ovvero RDF.

IDN-IM e RDF sono per certi aspetti simili ma presentano delle notevoli differenze. Per entrambi i modelli ogni informazione viene *identificata* in modo univoco tramite un URI ma, mentre RDF permette di gestire sia *risorse informative* che *risorse non informative*, IDN-IM si propone unicamente di trattare risorse informative, la cui essenza può quindi essere completamente veicolata tramite una rappresentazione digitale. Diversamente da RDF, la granularità dell'informazione in IDN-IM non è stabilita a priori ed è possibile deciderla proprio in base allo specifico contesto di utilizzo. IDN-IM è stato progettato non tanto per rispondere ad esigenze di ricerca come avviene nel caso di RDF bensì per realizzare operazioni di lettura/scrittura su uno spazio globale di dati distribuito. IDN-IM consente inoltre di definire informazioni utili per la gestione dei dati del livello applicativo, abilitando la collaborazione.

[1] Con il termine CRUD si intendono le quattro operazioni base di un sistema atto alla gestione della persistenza di dati: Create, Read, Update, Delete.

3.1 IDN Information Model: concetti generici

L'entità gestita dall'ambiente IDN è un documento strutturato che risulta composto da contenuti, relazioni (implicite o esplicite tra i contenuti) e metadati, in conformità ai seguenti principi fondanti l'architettura stessa [Inn08, Chi09]:

- *Principio 1: Contenitore-Contenuto.* I documenti sono contenitori mentre i dati ed i metadati costituiscono il loro contenuto; questi sono importanti al pari della struttura che li contiene.
- *Principio 2: Identificatori Globali.* Ad ogni dato sono assegnati degli identificatori gestiti da un sistema globale per l'identificazione attraverso il quale sia possibile effettuare il recupero delle informazioni. Sia i contenitori che i contenuti (ovvero documenti e dati/metadati) possono avere uno o più nomi con validità locale o globale. Al momento della creazione un documento o un dato ha un nome locale, che diventa globale una volta che l'autore decide di assegnarne la visibilità ad un certo insieme di utenti (ovvero singolo utente, gruppi specifici, etc.).
- *Principio 3: Cooperazione.* Sia i documenti che i dati sono arricchiti di un insieme di metadati/attributi che abilitano la cooperazione. In sintesi si asserisce che il modello dell'informazione costituito dall'insieme "dati-metadati-struttura" deve essere abilitante alla cooperazione.

Si può affermare che un documento è la testimonianza di una evidenza fisica o intellettuale, memorizzata e strutturata in una qualsiasi forma materiale, capace di essere compresa dall'uomo, trattabile dalla macchina e comunicabile [Buc97, RMS97]. Questa definizione si adatta perfettamente al concetto di unità informativa, su cui basare la collaborazione. Conseguenza naturale è stata quella di prevedere nell'Information Model il concetto di documento, visto come insieme strutturato di informazioni più elementari eventualmente documenti a loro volta. L'importanza del concetto di documento è immediata conseguenza del fatto che, molto spesso, quando si considerano più informazioni distinte come un'unica entità (raggruppate all'interno del documento) il contenuto informativo della somma è maggiore della somma delle singole informazioni: il documento è di per sé informazione. L'informazione aggiuntiva è data dalle correlazioni presenti fra le varie informazioni e permette di caratterizzare il documento come un'entità strutturata [2].
All'interno di InterDataNet il documento viene modellato come un aggregato di informazioni elementari; la struttura e il contenuto sono completamente separati dalla presentazione. Ciascun elemento informativo viene mantenuto

[2] Nel caso in cui l'aspetto relativo alle correlazioni sia trascurabile si parla di informazione non strutturata che comunque può essere vista come un caso particolare di quella strutturata.

isolato rispetto agli altri secondo un principio gerarchico che si basa sull'associazione ad un *responsabile*, ovvero colui il quale ha la consapevolezza di dover rispondere degli effetti che possono scaturire a seguito della divulgazione dell'informazione ([Inn04, Pao06, Inn08]); egli può essere una persona fisica oppure giuridica.
Un esempio utile a comprendere tale concetto è quello di un documento che rappresenta una patente di guida [Inn08, Chi09]. Per ognuno dei contenuti informativi presenti, è possibile determinare un'organizzazione che detiene l'informazione: i dati anagrafici del titolare sono gestiti dal Comune di nascita, i dati della residenza dal Comune di residenza, la categoria di guida dal Ministero dei Trasporti, i punti dal Ministero degli Interni, i dati relativi alla scadenza dalla Motorizzazione Civile, e così via. Tale procedimento tende quindi a strutturare rigidamente il documento, seguendo le regole organizzative e procedurali che hanno contribuito alla sua creazione o rilascio.
Più un documento è finemente suddiviso nei suoi elementi, più ogni sua parte, se opportunamente indicizzata e classificata, risulta riutilizzabile per la costruzione e la produzione di nuova informazione. Pertanto, ai fini di agevolare il riuso è opportuno che l'informazione possa essere trattata con un livello di *granularità* arbitrario.
Non è necessario che la sorgente di informazione sia unica (ovvero prodotta da una sola organizzazione) e unitaria (ovvero memorizzata su un solo sistema); disporre della possibilità di modellare le informazioni con un livello di granularità arbitrario risulta essere tanto più vantaggioso quanto più è possibile distinguere le organizzazioni (e sotto-organizzazioni) che trattano le singole informazioni, costituendo in modo naturale un sistema distribuito dal punto di vista dello stesso documento.
La dinamicità e la flessibilità sono ottenute grazie all'elevato grado di autonomia delle singole sorgenti che, in modo collaborativo, partecipano alla determinazione di documenti capaci di evolvere nel tempo, lasciando che ogni parte degli stessi sia soggetta a modifiche indipendenti. In realtà il fruitore dell'informazione può trovarsi di fronte a documenti che percepisce come immutabili (un atto di nascita, ma anche un articolo di giornale ormai pubblicato) così come ad entità intrinsecamente dinamiche (come lo stato del conto corrente bancario o lo stato di famiglia) che spesso vengono quantizzate con opportune tecniche di campionamento (nel tempo) al fine di renderle immutabili e quindi più facilmente trattabili (l'estratto conto, lo stato di famiglia ad una certa data). In ogni caso è esistito un processo, eventualmente distribuito, attivo nell'intervallo $[t_1, t_2]$ (con $t_1 \leq t_2$) che ha portato alla generazione dell'informazione al tempo t_x (immutabile o meno) con $t_x \in [t_1, t_2]$.
Il documento IDN è quindi una entità attiva, in quanto può evolvere e cambiare comportamento sulla base dello stato assunto e delle relazioni espresse all'interno per effetto della gestione storica (o *versioning*).

Applicando le considerazioni precedenti all'esempio del documento patente, se i punti vengono azzerati come effetto di una serie di contravvenzioni, il documento perde di validità e possono venire automaticamente notificati degli avvisi ai soggetti interessati. Limitandosi al solo punto di vista tecnico e formale (e tralasciando quello giuridico) si determina la creazione automatica di una nuova patente, in base al principio che "se cambia una parte, allora cambia l'intero aggregato delle parti".
All'interno di IDN-IM, i documenti sono ottenuti dalla suddivisione dei contenuti informativi, fino ad ottenere quelle che sono chiamate *informazioni atomiche* (la definizione sarà fornita all'interno del paragrafo 3.3). Il procedimento di suddivisione termina nel momento in cui si incorre in un'informazione che non sia ulteriormente (e ragionevolmente) frazionabile, quale ad esempio il nome o il cognome di una persona.
Per essere fruibile questo procedimento necessita ovviamente di un compromesso tra la complessità, l'effettiva necessità di riuso dei contenuti informativi e l'adeguatezza della granularità della rappresentazione: ciò viene ottenuto attraverso il concetto di *informazione primitiva* (la cui definizione sarà fornita nel paragrafo 3.3).
Il presente lavoro si propone come un approccio per la creazione di informazioni che siano sufficientemente granulari da possedere le seguenti caratteristiche:

- *modularità*: le informazioni devono essere aggregabili in maniera versatile con altre;
- *indirizzabilità*: le informazioni devono essere identificabili sia logicamente che per indice;
- *riusabilità*: le informazioni devono essere autonome in diverse situazioni;
- *interoperabilità*: le informazioni devono essere indipendenti dal contesto.

Come regola fondamentale per la strutturazione viene utilizzato il principio di responsabilità sulle parti informative che costituiscono l'informazione: in tal modo si apre la strada alla possibilità di elaborare in autonomia le singole parti di cui è costituito il documento.
Tale metodologia assume un valore estremamente determinante quando viene coinvolto il trattamento di informazioni sensibili ed in generale da un punto di vista giuridico. Su larga scala consente inoltre di realizzare in maniera automatizzata un insieme di mattoni informativi elementari che possono essere riutilizzati dalla comunità per costruire nuova informazione, così come proposto dal capitolo 6 del presente lavoro.

Il principio di responsabilità può anche essere inteso come conseguenza naturale dei processi organizzativi e metodologici individuati all'interno delle organizzazioni.
Un'importante osservazione è legata alla assegnazione selettiva dei diritti di accesso per le singole informazioni quale condizione necessaria per poter parlare di responsabile: per esercitare il suo ruolo occorre che il responsabile abbia le garanzie sulle modalità di gestione ed accesso all'informazione.
All'interno dei documenti con i quali si interagisce possono essere individuate le seguenti categorie di informazioni:

- *contenuti*: elementi esplicitamente fruibili dall'uomo e che sono direttamente individuabili all'interno del documento. Rientrano in questa categoria i testi, le immagini, le date, i nomi, i contenuti multimediali, eccetera;

- *relazioni*: legami fra contenuti che possono essere di tipo esplicito o implicito. Un esempio di legame esplicito può essere il riferimento ad una pagina o paragrafo di un libro, ad un articolo presente in una legge, eccetera. In modo duale un legame implicito può essere quello esistente fra il titolo di un capitolo di un libro e il contenuto del capitolo stesso;

- *presentazione*: informazioni necessarie per mostrare correttamente il documento all'utente. Rientra a pieno titolo in questa categoria l'aspetto grafico, ovvero la scelta dei caratteri, dei colori, dell'impaginazione, eccetera. Questa categoria di informazioni fornisce un valore aggiunto che in particolari casi è determinante per permettere la fruibilità delle informazioni contenute nel documento;

- *informazioni aggiuntive*: ulteriori informazioni associate al documento e/o che possono essere date per scontate. Ad esempio l'autore di un brano musicale oppure il fatto che i contenuti presenti nella prima pagina di un quotidiano rappresentano un sunto delle notizie più importanti del giornale.

Queste categorie di informazioni sono state introdotte parlando di documento in senso astratto e/o nel senso convenzionale del termine. Nel momento in cui si intenda codificare un documento per renderlo trattabile dalle macchine occorre formalizzare al meglio questi concetti al fine di poterli gestire efficacemente ed efficientemente da un punto di vista informatico.
I documenti dotati di queste caratteristiche e codificati opportunamente vengono chiamati documenti strutturati. I vantaggi riscontrati trattando documenti strutturati sono molteplici in quanto gli aspetti strutturali, oltre a fornire un contenuto informativo addizionale rispetto al caso non strutturato, permettono di garantire anche un maggior controllo dei contenuti. Inoltre tramite il riferimento ad altre informazioni è possibile realizzare vari

documenti che di fatto determinano una rete di concetti. Le macchine, in questo modo, diventano sistemi in grado di elaborare il contenuto informativo dei documenti e ciò permette di sfruttarne agevolmente le potenzialità in termini di velocità, memorizzazione e comunicazione per supportare efficientemente l'uomo, il cui lavoro risulta velocizzato e facilitato nell'immissione, nella ricerca e nella consultazione delle informazioni.

3.2 Requisiti dell'Information Model

Con il termine *Information Model* si definisce una rappresentazione universale delle entità e delle loro proprietà, operazioni e relazioni, il cui scopo principale è quello di modellare gli oggetti ad un livello concettuale, indipendentemente da qualsiasi specifico repository, applicazione, protocollo o piattaforma. Esso non riguarda i dettagli, ma mira a catturare le astrazioni e i requisiti fondamentali delle entità da modellare.
Un Information Model deve celare tutti i dettagli implementativi e di comunicazione, al fine di rendere la progettazione la più chiara possibile. Inoltre un Information Model ha l'importante compito di modellare le relazioni tra le entità attraverso un linguaggio naturale o un linguaggio formale oppure con un linguaggio strutturato semi formale [PS03].
Un *Data Model*, invece, si occupa di gestire gli oggetti ad un livello di astrazione più basso, e comprende dettagli specifici dell'implementazione e del protocollo, ad esempio regole che spieghino come mappare gli oggetti gestiti su costrutti protocollari di livello più basso [PS03].
Il principale vantaggio nell'uso di un Information Model è la possibilità di generare, a partire da esso, più Data Model, intesi come i modelli concretamente realizzati; questa capacità abilita la scalabilità e l'adattabilità del modello in contesti diversi e con tecnologie diverse. Inoltre risulta possibile scegliere, per la realizzazione, tra una vasta gamma di standard e tecnologie esistenti, se rispondenti alle proprie esigenze.
D'altra parte, questa prospettiva rappresenta anche una forte restrizione, in quanto la sottile linea che divide il modello astratto dai relativi modelli concreti non è sempre di facile identificazione; in molti casi è veramente difficile stabilire se le astrazioni appartengono all'Information Model o al Data Model.
IDN-IM è dunque una rappresentazione omogenea [3], astratta [4] e consistente [5] dell'informazione, della quale cattura i requisiti, i principi e le proprietà desiderabili in termini assoluti, oltre a rappresentare un modello di

[3] Il termine omogenea indica una rappresentazione uniforme, riconducibile ai medesimi principi indipendentemente dal contenuto informativo che esprime.

[4] Il termine astratta indica una rappresentazione del tutto svincolata da specifiche implementazioni.

[5] Una rappresentazione è consistente se applicando i principi e le operazioni previste dal modello, il risultato è sempre prevedibile e coerente con il modello stesso.

documento universale, indipendente dal particolare contesto e da tecnologie specifiche.
Sulla base dell'analisi estesa dei requisiti discussa nel lavoro di Innocenti [Inn08] è possibile elencare le caratteristiche che il documento definito da IDN-IM deve soddisfare: alcune di esse risultano vincolanti per gli obiettivi del sistema, mentre altre sono desiderabili al fine di garantire la massima applicabilità ed estensibilità del modello.
Una prima caratteristica prevede che l'informazione sia *strutturata*: questo sottintende la necessità di poterla riusare ed elaborare con il minimo dispendio di risorse per la sua interpretazione. Ogni informazione deve essere strutturabile, indipendentemente dalla sua forma fisica, anche se non si è certi che la struttura sia univocamente determinabile a fronte di scelte interpretative diverse.
Inoltre, l'informazione deve poter essere *aggregabile*, infatti l'unione di più informazioni ha come risultato un'informazione nuova, diversa da ciascuna delle sue componenti. Viceversa, la scomposizione dell'informazione in entità elementari più semplici consente di massimizzare il riuso, la trasportabilità e l'autoconsistenza di ciascun aggregato. Un ulteriore vantaggio è rappresentato dalla possibilità di definire funzioni di accesso e di modifica in modo mirato.
L'informazione viene considerata come il risultato di produzioni iterative, in cui ciascun passo vede un *responsabile* (spesso, ma non necessariamente, coincidente con l'*autore*) il quale produce informazioni a partire da altre già esistenti in forma più semplice. La definizione stessa di informazione, garantisce che esista sempre almeno un attore che la crea e/o la controlla.
L'informazione deve essere *indipendente dalla locazione fisica*; spesso, infatti, la stessa informazione viene memorizzata in apparati fisici distinti ed eterogenei, che però non devono influire sui contenuti informativi. Sebbene questa affermazione possa apparire banale, agli effetti pratici la detenzione fisica dei dati da parte di una organizzazione risulta spesso essere ingessante per il sistema organizzativo ed informativo (anche internamente), il quale può risentire delle costrizioni imposte da normative o da pratiche adottate da un nucleo ristretto di affiliati.
L'informazione già esistente nelle organizzazioni è *distribuita* per definizione, per cui questo requisito risulta soddisfatto a priori; tuttavia occorre affrontare il problema di come realizzarne l'acquisizione, l'arricchimento e la manutenzione in modo distribuito e consistente, guardando alla distribuzione già esistente come a un vantaggio da sfruttare, e non come a uno svantaggio da azzerare.
Una caratteristica che non è obbligatoria ma fortemente desiderabile, in quanto sostiene i requisiti di affidabilità e scalabilità, è che l'informazione sia *replicata*. La replicazione introduce un certo numero di vantaggi, come l'aumento della disponibilità dell'informazione, garantendone il recupero a fronte di guasti, e il dimensionamento adattativo nel tempo del sistema in

funzione della richiesta. Come sarà chiarito nella descrizione di IDN-SA nel capitolo 4, il modello permette la realizzazione di sistemi per la replicazione dell'informazione, in grado di operare in modo del tutto trasparente per l'utente.
È necessario che l'informazione sia *tracciabile*, di conseguenza il sistema deve essere dotato di uno storico e di un meccanismo di versionamento (*versioning*). La sola presenza dell'informazione risulta sicuramente riduttiva e inespressiva nello svolgimento delle attività collaborative all'interno delle organizzazioni, le quali richiedono che sia possibile tracciare la storia dell'informazione in quanto legata al workflow e all'authoring concorrente e per la determinazione a posteriori delle responsabilità.
È indispensabile che l'informazione sia *indirizzabile* in modo efficace e flessibile sia sul piano telematico che come modello di supporto alle attività collaborative. La prassi di assegnare alias alla stessa informazione viene coadiuvata e formalizzata attraverso un opportuno sistema di nomi, mentre la loro gestione è delegata all'architettura stessa.
Infine, bisogna tenere presente che un'informazione è tanto più utile quanto più è *orientata alla collaborazione*, ossia è stata creata con lo scopo di essere condivisa e sfruttata da più utenti, facilitandone l'utilizzo su larga scala.

3.3 I nodi informativi

La struttura del documento in IDN-IM è rappresentata da un *grafo diretto aciclico* (*Directed Acyclic Graph*, in breve DAG) [Die06]. Il vincolo principale di un DAG è quello di non permettere cicli, ossia sequenze di nodi attraversando le quali a partire da un nodo qualsiasi si ritorna allo stesso nodo. Si noti che una struttura ad albero è un caso specifico del DAG, in cui ciascun nodo ha al più un genitore: le strutture a DAG sono perciò capaci di modellare tutte le risorse strutturate ad albero [Die06]. Tale proprietà rappresenta un grande vantaggio nella fase di passaggio dal modello informativo al modello dati.
In base all'assunzione precedente, le relazioni tra i nodi di un documento IDN sono gerarchiche, cioè esiste una relazione di tipo padre-figlio tale che il progenitore è sempre distinguibile dal suo discendente; questo non sarebbe stato possibile nel caso della presenza di cicli.
La metafora principale che è stata seguita nella progettazione del modello è quella *contenitore/contenuto*, ovvero ogni elemento informativo può essere visto come il contenitore di elementi informativi più elementari, i contenuti, i quali sono ottenuti a loro volta come composizione di elementi sempre più granulari; il procedimento va avanti fino al raggiungimento degli "atomi" non ulteriormente suddivisibili. Si definisce pertanto l'**informazione atomica (Atomic Information Unit, AIU)**, l'entità informativa più elementare; essa è indivisibile ed è costituita da una tripla `<nome,tipo,valore>`. Tali

informazioni sono quindi non ulteriormente suddivisibili se utilizzate con lo scopo di creare la "materia" informativa, ma potenzialmente scomponibili per altri scopi, quali la memorizzazione su supporto digitale in cui la granularità è sempre quella del bit. Un esempio di "atomo informativo" è senz'altro il cognome di una persona all'interno di un atto giuridico che sarà contenuto nella sezione dedicata ai dati anagrafici, che non è difficile ipotizzare come parte costituente di una qualche altra sezione dell'atto stesso.
I valori memorizzati nell'informazione atomica sono soggetti a verifica sintattica e di ammissibilità (eventualmente sulla base del tipo). Si consideri ad esempio l'informazione atomica `<CAP-FI,CAP,50100>` rappresentante il Codice di Avviamento Postale italiano per la città di Firenze; in tale AIU, il nome è un'etichetta che la identifica nel contesto in cui è collocata e pertanto, per il generico utente che inserisce o definisce il dato, può avere un certo peso semantico.
Il tipo esprime il concetto che il valore è un CAP e pertanto è un elemento dell'elenco dei CAP stabiliti dalle Poste Italiane; così il valore "FI-50100" è sintatticamente scorretto e "01234" è sintatticamente corretto, ma inammissibile.
Ogni informazione atomica, memorizzata nello spazio dei documenti definiti da IDN-IM, deve essere contenuta in uno ed un solo nodo del documento, il quale viene detto **informazione primitiva (Primitive Information Unit, PIU)** e costituisce quindi il building block dei documenti IDN-IM.
Ogni informazione primitiva è indirizzabile tramite nomi che ne riflettono la struttura, ovvero è identificata da almeno un *nome canonico* (che ne permette direttamente l'identificazione) e da zero, uno o più *alias* (che ne permettono l'identificazione indirettamente in quanto fanno riferimento ad altri nomi, alias o canonici).
Ogni nodo del grafo che non ha figli è detto *foglia* e contiene almeno un'informazione atomica; ogni nodo del grafo che non è una foglia per definizione aggrega almeno un'informazione primitiva e può contenere una o più informazioni atomiche.
Le informazioni primitive modellano gli aspetti strutturali sfruttando il principio di aggregazione fra nodi. Ogni nodo contenente un'informazione primitiva può essere genitore di una serie di nodi contenenti altre informazioni primitive, con l'unico vincolo che nella struttura non siano presenti cicli.
Ogni informazione primitiva è quindi genitore di altre informazioni primitive e contiene al proprio interno un certo numero di informazioni atomiche. Da questo caso generale possono essere definiti documenti, come casi particolari, introducendo ulteriori restrizioni.
Ad esempio un vincolo che può essere introdotto riguarda la mutua esclusione fra l'aggregazione di altre PIU e il contenimento di AIU: in questi termini ogni informazione primitiva aggrega altre informazioni primitive senza contenere informazioni atomiche oppure contiene informazioni atomiche senza aggregare informazioni primitive. Documenti strutturati secondo questa

modalità hanno le informazioni atomiche esclusivamente nelle foglie; inoltre tutte le foglie contengono necessariamente informazioni atomiche e pertanto esiste una perfetta corrispondenza fra le foglie e le informazioni atomiche e fra gli altri nodi e le informazioni primitive.
L'associazione al responsabile avviene a livello di informazione primitiva: egli deve garantire la correttezza della struttura (legame logico tra le parti) e, per le informazioni atomiche incapsulate, la correttezza dei dati associati alle coppie `<nome,tipo,valore>`.
Per esempio si faccia l'ipotesi che la patente di guida (semplificata) contenga i seguenti elementi:

- anagrafica;
- residenza;
- categoria guida;
- punti.

In questo esempio la Motorizzazione Civile deve garantire la correttezza delle triple `<punti,punti_pat,valore1>` e `<cat,categoria_pat,valore2>` e la corretta aggregazione delle informazioni di anagrafica e residenza. Invece l'anagrafe del comune di nascita deve garantire la correttezza delle informazioni atomiche relative ai dati anagrafici, mentre il comune di residenza deve garantire la correttezza dell'indirizzo.

3.4 Relazioni strutturali tra nodi e documenti

All'interno di IDN-IM sono stati definiti tre principali tipi di collegamento, i quali sono descritti nel seguito:

- *Collegamenti di aggregazione.* Tali link esprimono le relazioni genitoriali tra i nodi del grafo all'interno dello stesso documento. Un nodo A, genitore di un nodo B presenterà un link di aggregazione verso di esso; tale collegamento può essere rappresentato graficamente con una freccia uscente dal nodo A e diretta al nodo B, come illustrato in figura 3.1. In tale caso si dice che il nodo A *aggrega* il nodo B, ovvero che il nodo B *è aggregato* dal nodo A.

- *Collegamenti di segnalazione* (degli aggiornamenti). Tali link abilitano il meccanismo di propagazione degli aggiornamenti al fine di gestire il versionamento, descritto nel paragrafo 3.7. Se è definito un collegamento di segnalazione da B a A questo significa che se B viene aggiornata, il sistema informatico autoritativo (cioè responsabile) su A verrà notificato relativamente all'aggiornamento di B.

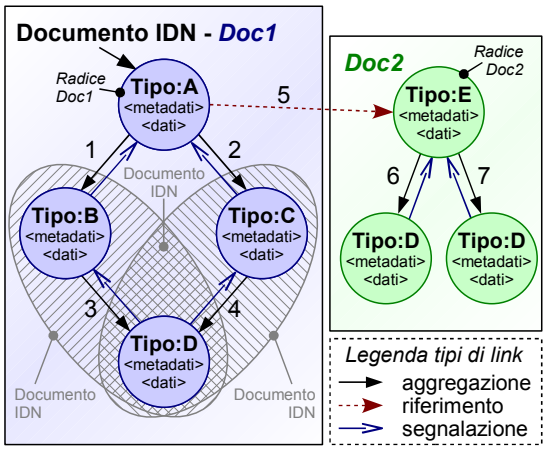

Figura 3.1: Esempio di documento IDN.

- *Collegamenti di riferimento.* Tali link esprimono relazioni tra documenti diversi (e quindi tra grafi diversi) e/o elementi esterni al modello. I collegamenti di riferimento vengono memorizzati nelle PIU e non hanno influenze di alcun tipo sul comportamento del modello, ovvero i link di riferimento vengono definiti e gestiti dall'utilizzatore del modello IDN-IM e quest'ultimo si limita solo a memorizzarli. Nell'esempio illustrato in figura 3.1, il documento `Doc1` possiede un link di riferimento verso il documento `Doc2`.

Si definisce il **documento IDN** come un insieme di nodi, rappresentanti le PIU, uno dei quali è detto *radice* [6], messi in relazione tra di loro attraverso collegamenti di aggregazione [PPCP07]. Applicando la definizione in modo ricorsivo, ciascun documento può essere considerato come un'aggregazione di documenti.
Pertanto il documento IDN è strutturato ed è il risultato dall'aggregazione di informazioni elementari, le quali a loro volta contengono le informazioni atomiche.

3.5 Strutturare con criteri di responsabilità

Come già preannunciato nel paragrafo 3.1, viene proposto un approccio per la creazione di informazioni sufficientemente granulari tali da possedere le seguenti qualità:

- modulari, ovvero aggregabili in maniera versatile con altre;

[6] Il nodo radice di un documento IDN ha la caratteristica di non essere aggregato da alcun nodo del documento considerato.

- indirizzabili, ovvero identificabili sia logicamente che per indice;
- riusabili, ovvero per la loro autonomia in diverse situazioni;
- interoperabili, ovvero indipendenti dal contesto.

Si consideri una organizzazione ed in particolare le informazioni in essa create dalla collaborazione di più persone: da un punto di vista organizzativo ciascuna persona avrà contribuito agendo su una parte e lasciando alle altre il compito di integrarla o di validare l'unione. L'organigramma aziendale, i ruoli ed il ciclo di vita dell'informazione sono gli elementi che stabiliscono in maniera non ambigua chi è responsabile di cosa e ognuno deve essere abilitato ad agire sulle parti di sua competenza.

Tenendo presenti le esigenze della collaborazione, si propone di applicare come regola strutturante il principio di responsabilità sulle parti informative che costituiscono l'informazione; tale principio apre la strada alla possibilità di elaborare in autonomia singole parti di cui è costituito il documento.

Come già illustrato nel paragrafo 3.2, il *responsabile* è colui il quale ha la consapevolezza di dover rispondere degli effetti che possono scaturire a seguito della divulgazione dell'informazione; egli può essere una persona fisica oppure giuridica la quale crea o modifica l'informazione [PPI+09]. Si osservi che questa metodologia assume un valore estremamente determinante quando viene coinvolto il trattamento di informazioni sensibili ed in generale da un punto di vista giuridico.

La *responsabilità* risulta quindi essere un criterio per definire e strutturare l'informazione a prescindere dal suo contenuto, basandosi sulla gerarchia delle organizzazioni ed in tal modo il problema della strutturazione viene stato spostato su un piano organizzativo. In contesti specifici, come ad esempio nella Pubblica Amministrazione, per ogni nodo deve essere definita un'entità responsabile (la quale risulta comunque facilmente individuabile grazie alle normative).

Occorre osservare che strutturare con il principio di responsabilità non presuppone la reingegnerizzazione dei processi interni all'organizzazione, ma se essi fossero già ben ingegnerizzati allora l'applicazione del principio citato troverebbe maggiore beneficio.

Strutturare utilizzando il principio di responsabilità consente di realizzare in maniera automatizzata un insieme di mattoni informativi elementari che possono essere riutilizzati dalla comunità per costruire nuova informazione.

Nel settore della Pubblica Amministrazione o comunque in tutti quei settori in cui esiste un elevato numero di elementi informativi ricorrenti e trasversali è possibile attuare sia una forte politica del riuso sia la creazione di documenti in maniera dinamica.

3.6 Identificazione di nodi e documenti

Affinché un elemento informativo sia correttamente condiviso, divulgato e riusato, deve possedere due proprietà fondamentali: essere individuabile ed accessibile. Spesso nei sistemi telematici le risorse sono individuate attraverso il loro indirizzamento e l'indirizzo stesso viene utilizzato anche per l'accesso fisico (come nel caso degli URL [7]).
All'interno di IDN-IM l'identificazione dei nodi è una proprietà indipendente dai dettagli dell'accesso fisico, dato che quest'ultimo non rappresenta un contesto di interesse per un Information Model. In altre parole è importante disporre di identificatori che vadano ad astrarre dalla localizzazione fisica delle risorse e siano quindi indipendenti dalla stessa.
Tale affermazione è conseguenza della percezione che l'essere umano ha del concetto di "documento": ad esempio con "categoria della patente di guida di Mario Rossi" si intende un'informazione ben precisa ed indipendente dal luogo e dal formato in cui viene conservata e acceduta. In pratica la categoria è la stessa informazione sia che questa venga memorizzata negli archivi elettronici della Motorizzazione Civile sia che si trovi stampata sul documento in formato cartaceo per il titolare della patente. Questo esempio evidenzia come l'*identificazione delle risorse* sia un'operazione concettualmente diversa dall'*accesso* e pertanto, in generale, è opportuno che le due operazioni siano distinte per rispettare la separation-of-concern. In tal caso, una volta effettuata l'identificazione in una prima fase, è possibile procedere, in una fase successiva, all'accesso.
Le risorse [8] all'interno di IDN sono identificate mediante il ricorso agli **LRI (Logical Resource Identifier)**; sebbene siano stati definiti come sottoclasse degli HTTP-URI, essi garantiscono la completa separazione fra indirizzamento ed accesso. Grazie al principio di responsabilità l'authority dell'URI [9] permette di individuare non tanto il server che detiene l'informazione a cui l'URI fa riferimento bensì il server "responsabile" su quella informazione. Come risulterà più chiaro nel seguito (dopo la descrizione di IDN-SA) l'informazione verrà infatti ricostruita dinamicamente attivando una serie di elaborazioni che, partendo dal server responsabile, interesseranno altri server, fino al reperimento effettivo di tutti i dati necessari, che risiederanno su sistemi remoti, per la ricomposizione dell'informazione di interesse. Il responsabile non detiene quindi l'informazione, ma è colui il quale deve orchestrare tutti i processi necessari per ricostruire l'informazio-

[7] Gli *Uniform Resource Locator* (o URL) [Hof05a, Hof05b] sono gli indirizzi utilizzati nel web al fine di identificare ed accedere alle risorse. Sono un caso particolare di URI (*Uniform Resource Identifier* [BLFM05]).

[8] In questo contesto, con il termine risorsa si indica un documento IDN, un nodo di un documento, un dato/metadato in esso contenuto; in generale una qualunque entità definita dal modello stesso.

[9] L'authority è la componente compresa fra ://" ed il primo "/" successivo, vedi la RFC3986 [BLFM05].

ne; inoltre come responsabile stabilisce le condizioni sotto le quali rendere fruibili l'informazione, negando ad esempio l'accesso ai soggetti che non ne hanno diritto.
Gli LRI che fanno riferimento alla medesima informazione sono considerati equivalenti, nel senso che permettono di operare sull'informazione allo stesso modo. La necessità di assegnare più nomi logici alla medesima informazione deriva dal fatto che essa è calata all'interno di un contesto di massimo riuso, ovvero la stessa informazione riusata in contesti diversi assumerà nomi diversi: questo è un dato di fatto che deriva dal comune modo di ragionare dell'essere umano. Infatti la "patente di Mario Rossi" è il nome che chiunque potrebbe assegnare alla patente di guida di Mario Rossi; per Mario Rossi invece, il nome sarebbe "la mia patente", mentre per la Motorizzazione Civile potrebbe essere qualcosa del tipo ". . . /FI/2012/NUM_XYZ".
IDN-IM prevede l'utilizzo di due diverse sotto-classi di LRI: i *canonici* e gli *alias*. Ogni informazione è identificata per convenzione tramite uno o più nomi canonici e zero, uno o più alias; quando il nodo è creato gli viene assegnato un LRI canonico invariabile nel tempo che dipende dal contesto nel quale esso viene creato. Gli LRI canonici possono essere assimilati agli hard-link dei filesystem Unix in quanto rappresentano una "maniglia" per l'accesso diretto all'informazione a cui fanno riferimento.
Gli utenti possono arbitrariamente assegnare ulteriori LRI ai vari nodi, secondo le loro necessità: questi sono gli alias, ovvero LRI che fanno riferimento ad altri LRI (i quali possono essere a loro volta canonici o alias), in analogia ai link simbolici presenti nei filesystem Unix.
Infine, poiché come illustrato nel paragrafo 3.4, i nodi fanno parte di documenti strutturati, possono essere assegnati ad essi ulteriori LRI determinati dalla posizione assunta nella struttura; ognuno di essi assumerà la seguente forma [PPI+09]:

```
node_parent_name + / + link_name
```

dove:

- `node_parent_name` è il nome del nodo genitore;
- `+` è l'operatore di concatenazione tra stringhe;
- `link_name` è il nome del collegamento che mette in relazione il genitore con il nodo da indirizzare.

Si consideri ad esempio la figura 3.2, nella quale è mostrato un documento IDN di nome `Doc1`, che contiene quattro nodi e quattro collegamenti di aggregazione di nome `1`, `2`, `3`, e `4`.
Il nodo `Doc1URI/1/3` ha questo nome in quanto figlio del nodo `Doc1URI/1` attraverso il collegamento di nome `3`. Per lo stesso motivo, ha anche il nome

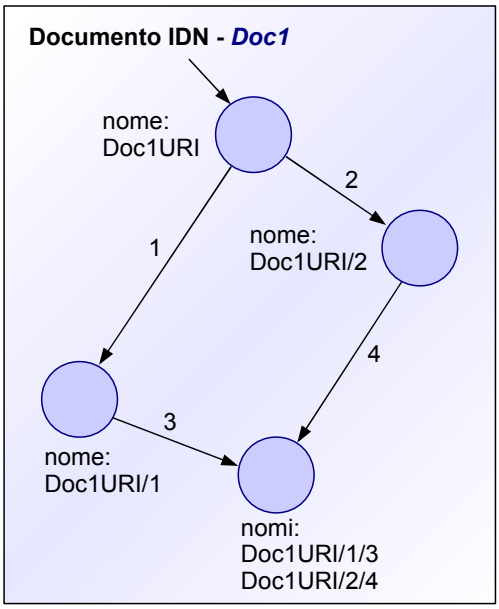

Figura 3.2: Esempio di relazioni tra documenti IDN-IM.

`Doc1URI/2/4` perché è raggiungibile anche dal nodo `Doc1URI/2` attraverso il collegamento di nome `4`.
Se si considera il nodo `Doc1URI` come radice locale, ai suoi successori sono automaticamente assegnati i seguenti nomi relativi: `./1`, `./2`, `./1/3` e `./2/4`.
Una ulteriore convenzione utilizzata per gli LRI (ancora analoga a quanto accade nei filesystem) è la seguente: gli LRI che terminano con "`/`" sono considerati come identificatori di documenti, mentre gli altri (ovvero gli LRI che non terminano con "`/`") come identificatori di nodi.
La convenzione è inoltre tale che se "`...xyz/unLRI/`" rappresenta il documento "`X`" allora il nome LRI "`...xyz/unLRI`" (senza slash finale) rappresenta il nodo radice di "`X`".
È interessante notare infine che gli alias offrono sempre la dereferenziabilità diretta, ovvero se "`A`" è un alias di "`B`" è sempre possibile raggiungere "`B`" partendo da "`A`".
Invece, per quanto riguarda la risoluzione inversa è possibile trovarsi di fronte a due casi:

- *invertibilità globale*; in questo caso a livello globale è possibile risalire ad "`A`" partendo da "`B`". Questo tipo di invertibilità offre un legame bidirezionale fra i due soggetti interessati dalla relazione instaurata dall'alias. In questo modo il legame è reso più forte, ma può essere limitata la scalabilità e la semplicità di realizzazione dell'alias, opera-

zione che richiede la collaborazione dell'organizzazione a cui fa capo l'elemento referenziato;

- *invertibilità locale*; in questo caso è possibile risalire ad "`A`" partendo da "`B`" limitatamente al contesto dell'organizzazione a cui fa capo "`A`". In altre parole all'interno di ogni organizzazione è possibile effettuare la risoluzione inversa di tutti gli alias invertibili globalmente e di tutti gli alias definiti internamente all'organizzazione stessa.

3.7 Storico dei documenti

Il mantenimento delle versioni (o *revisioni*) dell'informazione e l'automazione della generazione delle stesse garantiscono la massima tracciabilità, permettendo di ricostruire l'evoluzione dell'informazione nel tempo a fronte delle modifiche.
Quando si parla di "modifica" di un'informazione:

- nel caso di *informazione atomica* si intende il cambiamento di uno dei tre elementi `<nome,tipo,valore>`;

- nel caso di *informazione primitiva* si intende il cambiamento di una o più delle relazioni di aggregazione che escono dall'informazione in esame (quindi della struttura dei documenti che la contengono) oppure la variazione delle informazioni atomiche contenute.

IDN-IM è stato dotato dei principi del modello UEVM (*Unified Extensional Versioning Model*) [ABCM99], il quale organizza le versioni di documenti strutturati come DAG riuscendo a gestire, con tecniche automatizzate, sia le revisioni dei contenuti sia quelle degli aspetti strutturali. All'interno del presente lavoro, sono descritti gli aspetti principali riguardo il versioning; ulteriori dettagli sono disponibili nei lavori di Chini [Chi05] e Innocenti [Inn08].
Se necessario, ogni documento può quindi disporre di caratteristiche atte a memorizzarne l'evoluzione temporale; questa, che viene definita anche con il termine *storico* è una caratteristica globale del documento, tuttavia viene mantenuta memorizzando l'evoluzione delle singole informazioni che lo costituiscono, ovvero viene gestita a livello di nodo.
Le evoluzioni temporali delle singole informazioni non sono indipendenti: una modifica effettuata ad un'informazione che si trova ad un livello più basso della gerarchia del DAG si ripercuote, grazie ai link di segnalazione, su tutti i suoi predecessori. In tal modo lo storico della radice "comprende", seppur indirettamente, l'evoluzione temporale di tutto il documento.
All'interno di IDN-IM è previsto un meccanismo di navigazione nello storico che, partendo dalla radice ed attraversando i vari nodi del documento, permette di ricomporlo come richiesto. È utile evidenziare come il meccanismo,

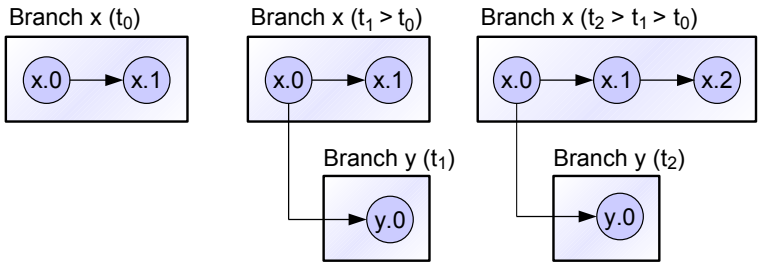

Figura 3.3: Generazione delle revisioni.

basandosi su un modello estensionale, permetta, a differenza di altri modelli di versioning, di ripercorrere lo storico del documento nel modo più naturale possibile per l'utente: infatti ogni versione viene ricostruita correttamente sia per quel che riguarda i dati contenuti sia per quanto riguarda gli aspetti strutturali.
Per descrivere i meccanismi di gestione dello storico è utile inizialmente fare riferimento ad un documento costituito da un unico nodo e, successivamente, estendere il concetto al caso generale.
Se si ipotizza che ogni nodo, una volta creato, sia una grandezza immutabile nel tempo, l'operazione di modifica dà vita ad un nuovo nodo: si definisce allora *versione* un'informazione primitiva ottenuta modificando il contenuto informativo di un nodo esistente. Anche la nascita di una nuova informazione rientra in questa definizione considerando che tale operazione può essere vista come la modifica dell'informazione nulla.
Viene definito uno stato *UPDATE*, associato ad ogni informazione, per determinare se è necessario generare nuove versioni oppure effettuare sovrascritture; esso può assumere uno di due valori: *changing* o *frozen*.
Il primo caso modella documenti e/o informazioni non versionati e pertanto non sarà oggetto di ulteriore trattazione.
Per quanto riguarda invece il caso *frozen*, IDN-IM consente:

- di effettuare la modifica dell'ultimo nodo all'interno dello storico. Questa è un'operazione trasparente all'utente, il quale si limita a richiedere un "aggiornamento" dell'ultima versione disponibile;
- di effettuare modifiche a nodi diversi dall'ultimo, mediante la creazione di una diramazione o branch (come illustrato in figura 3.3); questa operazione avviene sempre su esplicita richiesta.

Per quanto riguarda il secondo caso, si consideri ad esempio il ramo x al tempo t_0 in figura 3.3; la richiesta di nascita di una diramazione a partire dalla prima versione $x.0$ al tempo $t_1 > t_0$ comporta la creazione del ramo y contenente la nuova versione $y.0$. Al tempo $t_2 > t_1$ la modifica di $x.1$ viene applicata nello stesso ramo con la revisione $x.2$.

Da un punto di vista concettuale effettuare un merge significa integrare le modifiche presenti su un branch all'interno di un altro; nella pratica questa operazione può essere effettuata secondo modalità diverse che variano con il contesto.
Per quanto riguarda le informazioni atomiche, all'interno di IDN-IM sono state implementate delle modalità di fusione elementari a partire dalle quali sarà possibile definire metodologie di merge più complesse in base alle specifiche esigenze del dominio applicativo di interesse.
È possibile definire un'operazione di confronto fra due informazioni atomiche (che dipende dal tipo di informazione atomica trattata e quindi, in ultima analisi, dal contesto) che ne determina l'uguaglianza o la differenza. Nel primo caso il merge è banale e il risultato è l'informazione stessa; nel secondo caso l'operazione di confronto può permettere di stabilire l'entità della differenza e si possono presentare le seguenti situazioni:

- le informazioni sono diverse, ma confrontabili nei contenuti; in via automatica o sotto la supervisione dell'utente è possibile generare una terza informazione atomica sulla base delle altre due;

- le informazioni non sono confrontabili e pertanto non è definibile/possibile l'operazione di merge.

Per quanto riguarda invece le informazioni primitive, poiché costituiscono il punto di accesso all'informazione complessiva che si sviluppa da esse fino alle foglie, il concetto di uguaglianza fra esse non è esprimibile esclusivamente sulla base del confronto del contenuto informativo dei nodi che le rappresentano, ma in generale occorre andare a considerare e confrontare ricorsivamente i nodi che tali informazioni primitive aggregano (in altre parole l'intero documento in esse radicato).
Riepilogando le considerazioni precedenti, in riferimento alla figura 3.4, è possibile fornire la seguente definizione: effettuare la fusione (o merge) di due elementi A e B appartenenti a branch diversi dello stesso storico, significa generare un terzo elemento C ottenuto da essi a seguito dell'esecuzione di un determinato algoritmo; il nuovo elemento C figura nello storico come successore sia di A che di B e, per convenzione, appartiene al ramo di A [10]. La definizione e l'esecuzione dell'algoritmo di generazione dipendono dal dominio applicativo ovvero dal tipo di informazione rappresentata attraverso il documento IDN: l'infrastruttura si limita ad offrire le funzionalità di base per realizzare le operazioni, abilitando la possibilità di creare merge arbitrariamente complessi.

[10] In questi termini, il merge è un'operazione definita fra due branch che fonde il secondo sul primo. L'operatore di "merge" non è quindi commutativo: fondere A e B su C è diverso da fondere B e A su C. Nel primo caso C appartiene al ramo di A mentre nel secondo al ramo di B.

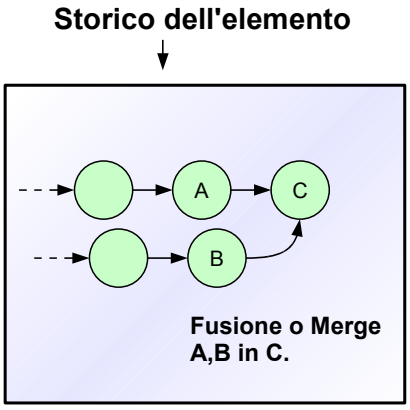

Figura 3.4: Merge di due nodi.

Poiché risulta possibile modificare in modo trasparente solamente l'ultimo elemento di ogni ramo e la creazione di un branch avviene sempre su esplicita richiesta, è possibile affermare che IDN-IM permette l'authoring collaborativo senza la necessità di prevedere meccanismi di *locking*. Infatti risulta sempre possibile individuare e gestire i conflitti, ovvero i tentativi di modificare due o più volte la medesima informazione in istanti diversi; ciò può verificarsi qualora due utenti, l'uno indipendentemente dall'altro, accedano alla stessa informazione nella versione $v_{(x)}$ (che si ipotizza essere l'ultima disponibile nel momento dell'accesso) e, dopo il tempo necessario ad elaborare la modifica, tentino l'operazione di salvataggio. Il primo di essi riuscirà nell'intento ed il sistema genererà la versione $v_{(x+1)}$, mentre il secondo verrà informato che la versione $v_{(x)}$ è diventata immutabile a causa del fatto che non è più l'ultima revisione disponibile dell'informazione. A questo punto egli può decidere di attuare una delle seguenti strategie:

- effettuare un merge della propria modifica con quella apportata dal primo utente. In questo caso si hanno due opzioni possibili: l'abbandono dell'intenzione di apportare la modifica, ad esempio perché già realizzata dall'altro utente, oppure la produzione della versione $v_{(x+2)}$ ottenuta dalla fusione di $v_{(x)}$ più le modifiche locali e $v_{(x+1)}$[11];

- creare un nuovo branch sul quale apportare le proprie modifiche (eventualmente da fondere nel ramo di partenza in un momento futuro).

[11] Come nel caso del merge di due rami, anche in questo caso $v_{(x+2)}$ può essere uguale a $v_{(x)}$ (il secondo utente decide di "annullare" le modifiche del primo utente) oppure diverso sia da $v_{(x)}$ che da $v_{(x+1)}$. Il caso $v_{(x+2)} = v_{(x+1)}$ non è significativo in quanto coincide con l'opzione in cui il secondo utente rinunci a creare $v_{(x+2)}$ lasciando $v_{(x+1)}$ come ultima versione dell'informazione.

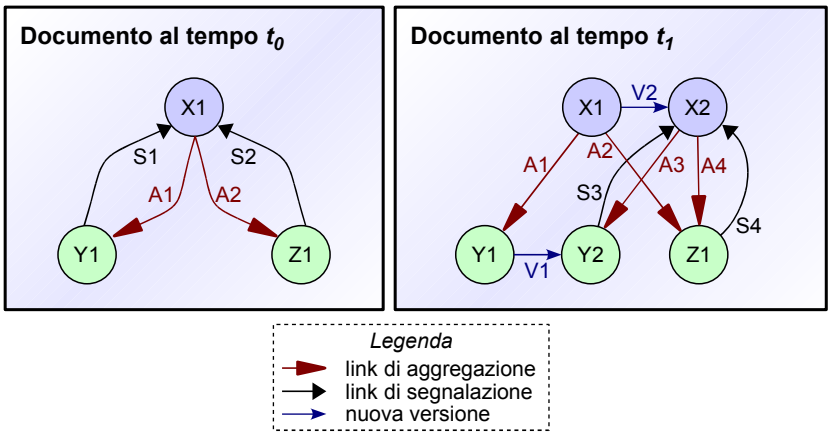

Figura 3.5: Esempio di propagazione delle modifiche.

3.7.1 La propagazione delle modifiche

Per poter implementare il modello di versioning UEVM è necessario prevedere un meccanismo che permetta di propagare le modifiche dei figli verso i genitori; pertanto IDN-IM prevede i link di segnalazione (già introdotti nel paragrafo 3.4) che permettono di risalire nella gerarchia dei nodi presenti nel dominio del grafo IDN-IM.
Per quanto riguarda la relazione tra i link di segnalazione e quelli di aggregazione, occorre evidenziare il fatto che questi ultimi sono suddivisi in:

- link *propaganti*: sono quelli per i quali, in corrispondenza dell'ultima versione di ogni branch di informazioni primitive, esiste il corrispondente link di segnalazione nel senso opposto.

- link *non propaganti*: sono quelli per i quali non si hanno i link di segnalazione nel senso opposto; pertanto le modifiche apportate ai nodi a valle non si ripercuotono su quelli a monte.

Per chiarire le modalità con cui avviene la propagazione si faccia riferimento alla figura 3.5 nella quale il documento al tempo t_0 è costituito da una radice X1 che aggrega due figli Y1 e Z1 tramite opportuni link di aggregazione, rispettivamente A1 e A2. Dato che, per ipotesi, tali link di aggregazione sono propaganti, esiste un link di segnalazione in senso inverso (S1 e S2) in corrispondenza di ognuno di essi.
Il documento al tempo t_1 mostra lo stato successivo ad una modifica (in conformità al meccanismo previsto in IDN-IM) relativa a Y e propagata alla radice X. In questo caso i link di segnalazione S1 e S2 sono stati rimossi e sono stati inseriti S3 e S4 in corrispondenza dei link di aggregazione A3 e A4 presenti nell'ultima revisione della radice X.

3.7.2 I parametri di versione

All'interno dello storico di IDN sono stati introdotti anche i *parametri di versione* i quali, in combinazione con il nome LRI di base, offrono la possibilità di navigare nello storico di informazioni primitive e documenti IDN.
Nella definizione è stata usata la seguente convenzione: il nome del parametro è diviso, tramite il simbolo "_", in due parti:

- la prima parte è composta da una singola lettera finalizzata ad identificare l'ambito in cui opera. I valori ammessi sono:
  - *R* per *revision*;
  - *H* per *history*;
  - *B* per *branch*;
  - *T* per *time*;
- la seconda parte è una stringa che definisce una relazione all'interno del contesto in cui il parametro è definito.

I parametri, riportati di seguito, sono accompagnati da una breve descrizione, che può meglio essere compresa facendo riferimento alla figura 3.6, nella quale il nodo corrente è quello creato al tempo t=7:

- *R_NEXT*: *Revision, Next.* Si riferisce alla revisione successiva al nodo corrente. Si osservi che, a seguito di una richiesta con tale parametro di versione, possono presentarsi i seguenti casi:
  - non esiste la revisione successiva: la risposta è quindi negativa. Questo è il caso dell'ultimo elemento presente nel ramo;
  - esiste la revisione successiva: la risposta fornisce il nodo che la contiene;
  - ortogonalmente il nodo può essere radice di un branch. In questo caso la risposta contiene, se esiste, il nodo relativo alla revisione successiva e l'indirizzo LRI più opportuno del primo elemento del branch. In questo modo viene soddisfatta la richiesta relativa alla revisione successiva ed i livelli superiori vengono messi a conoscenza dell'esistenza del branch con la possibilità di indirizzarlo.

  In relazione alla figura 3.6, l'*R_NEXT* del nodo creato al tempo t=7 è quello creato al tempo t=8;
- *R_PREV*: *Revision, Previous.* Si riferisce alla revisione precedente al nodo corrente. In modo duale a *R_NEXT* possono presentarsi i seguenti casi:

  - non esiste la revisione precedente: la risposta è quindi negativa. Ciò si verifica solo per la radice dello storico;
  - esiste la revisione precedente: la risposta fornisce il nodo che la contiene.
  - il nodo può essere stato creato come nuova revisione o in seguito ad un'operazione di merge. In quest'ultimo caso, similmente a quanto è stato definito per le diramazioni, nella risposta sono presenti il nodo che contiene la revisione precedente (che si trova sullo stesso ramo del nodo di partenza) e l'indirizzo LRI più opportuno dell'elemento precedente presente nell'altro branch.

  In relazione alla figura 3.6, l'*R_PREV* del nodo creato al tempo t=7 è quello creato al tempo t=6;

- *H_ROOT*: *History, Root.* Questo parametro di versione si riferisce alla radice dello storico ovvero al primo elemento che è stato creato. In relazione alla figura 3.6, l'*H_ROOT* del nodo creato al tempo t=7 è quello creato al tempo t=1;

- *B_ROOT*: *Branch, Root.* Si riferisce alla radice del branch ovvero all'elemento che è stato preso come riferimento per creare tale branch. Si osservi che per quanto riguarda i nodi appartenenti al ramo principale *B_ROOT* coincide con *H_ROOT*. In relazione alla figura 3.6, il *B_ROOT* del nodo creato al tempo t=7 è quello creato al tempo t=3;

- *T_ABSLAST*: *Time, Absolute Last.* Indica l'elemento dello storico più recente (da un punto di vista temporale). In relazione alla figura 3.6, il *T_ABSLAST* del nodo creato al tempo t=7 è quello creato al tempo t=16;

- *T_RELATLAST*: *Time, Relative Last.* Indica l'elemento più recente presente nello storico, relativamente considerato; equivale all'elemento *T_ABSLAST* dello storico ipotetico ottenuto a partire da detto nodo. In relazione alla figura 3.6, il *T_RELATLAST* del nodo creato al tempo t=7 è quello creato al tempo t=12;

- *B_LAST*: *Branch, Last.* Indica l'ultima revisione del branch cui il nodo considerato appartiene. In relazione alla figura 3.6, il *B_LAST* del nodo creato al tempo t=7 è quello creato al tempo t=10.

  Nel caso del merge dello storico per i nodi sul ramo principale il *B_LAST* si trova al termine di detto ramo, mentre per i nodi sul branch, il *B_LAST* è l'ultimo nodo prima del merge; ad esempio in figura 3.7, il *B_LAST* relativo ai nodi A, C, E ed F è F, mentre quello relativo ai nodi B e D è D. Il parametro *B_LAST* è da considerarsi il caso di default in IDN-IM, ovvero in assenza di un parametro di versione,

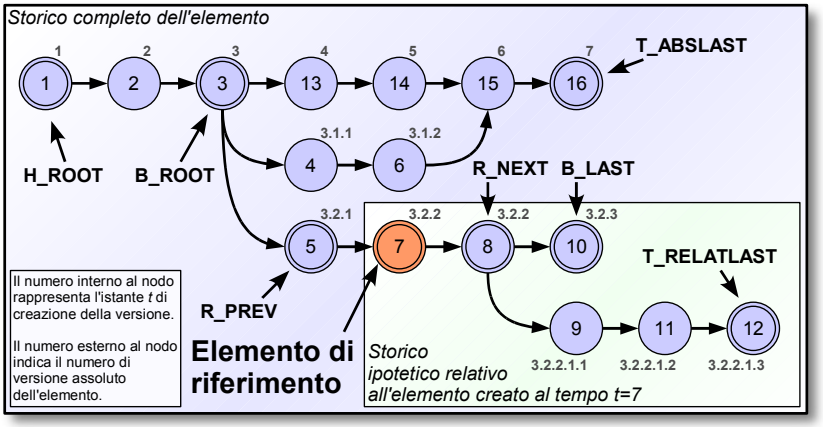

Figura 3.6: Esempio di elemento recente rispetto al nodo di partenza.

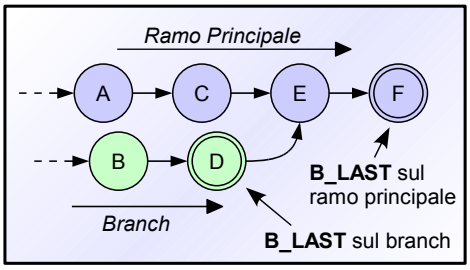

Figura 3.7: Esempi di "last" relativi al branch.

relativo o assoluto, il nome esprime l'ultimo elemento del branch. Tale meccanismo permette di esporre una vista simile a quella offerta dalla semantica Unix – la quale offre, per ogni istante temporale, l'accesso all'ultima versione disponibile dell'informazione – pur in presenza della semantica a file immutabili – ogni versione dell'informazione è immutabile ed apportare una modifica significa creare una nuova versione – introdotta, con il versioning, a questo livello. Lo scopo ultimo di questa scelta è quello di fare in modo che i nomi utilizzati dagli utenti, a meno che non contengano esplicitamente un parametro di versione, indichino la release corrente dell'informazione.

I parametri di versione relativi appena introdotti sono un meccanismo per navigare nello storico esprimendo una versione rispetto ad un'altra; il paradigma dei file immutabili viene raggiunto introducendo i *parametri di versione assoluti*. Un LRI con un parametro di versione assoluto indica in modo univoco e non ambiguo (oltre che persistente) una ben precisa versione

```
([1-9][0-9]+\.[1-9][0-9]*\.)*[1-9][0-9]*
```

Figura 3.8: Espressione regolare per definire gli identificativi di versione.

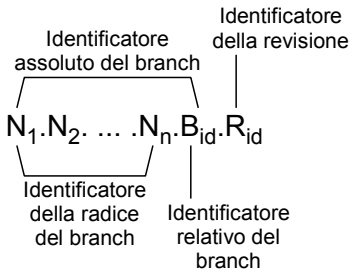

Figura 3.9: Convenzione sui nomi dei nodi nello storico.

dell'informazione, indipendentemente dalla storia futura che l'informazione avrà, o ha avuto, dopo la creazione della versione in questione.
La sintassi dei parametri di versione assoluti è definita tramite l'espressione regolare [Goy09] riportata in figura 3.8.
Tale espressione regolare permette di rappresentare la categoria delle stringhe contenenti un numero arbitrario positivo di elementi costituiti da cifre terminanti con un punto e una sequenza di cifre. Le sequenze di cifre rappresentano numeri interi e non possono iniziare con "*0*". Il numero di caratteri "." è pari, vale a dire che la quantità di sequenze di cifre presenti è sempre dispari.
Ad esempio, appartengono a questa classe le stringhe: "*21*", "*8.9.12*", eccetera, mentre non appartengono a questa classe le stringhe: "*7.*", "*.12*", "*01.3*", "*Ver=1.4*", "*8.9.12.4*", eccetera. In particolare l'ultima indicata non è valida in quanto presenta una quantità pari (non dispari) di sequenze di cifre.
All'interno della figura 3.6 sono stati riportati (in piccolo, esternamente ai nodi) i numeri di versione assoluti assegnati alle varie revisioni dell'informazione mostrata nell'esempio.
La struttura dell'identificativo è riportata in figura 3.9; tale rappresentazione permette di individuare univocamente le versioni all'interno dello storico e il branch di appartenenza. Per convenzione le prime $N-1$ cifre rappresentano l'identificatore assoluto del branch all'interno dello storico. Esso si ottiene aggiungendo al nome del nodo dal quale il branch è stato creato una cifra sequenziale, definita identificatore relativo del branch. In questo modo il

primo branch del nodo "*X.Y.Z*" è "*X.Y.Z.1*", il secondo branch è "*X.Y.Z.2*", eccetera.
Il nome di ogni nodo presente all'interno di un branch è ottenuto posponendo l'identificatore della revisione all'identificatore relativo del branch. Pertanto la prima revisione nel primo branch citato è "*X.Y.Z.1.1*", la seconda è '*X.Y.Z.1.2*"; più in generale, la n-esima revisione nel m-esimo branch è "*X.Y.Z.m.n*".

Capitolo 4

InterDataNet Service Architecture

Nel precedente capitolo è stato presentato il modello dell'informazione IDN-IM mettendo in evidenza la rappresentazione delle risorse informative da un punto di vista teorico ed astratto. In questo capitolo sarà discussa l'architettura a servizi di IDN, la quale consente di rendere operativo IDN-IM; questa vista del sistema prende il nome di InterDataNet Service Architecture (o IDN-SA).

IDN-SA si pone l'obiettivo di descrivere l'insieme di servizi ritenuti necessari a gestire operativamente l'*Information Model* definito e quindi fondare un'architettura di riferimento in grado di offrire una soluzione a livello infrastrutturale al problema della condivisione collaborativa, efficace ed efficiente, dell'informazione.

Le relazioni esistenti tra IDN-IM e IDN-SA sono schematicamente espresse in figura 4.1.

IDN-SA si propone come un "repository avanzato" per le applicazioni che trattano le specifiche problematiche di dominio utilizzando documenti IDN-IM per rappresentare l'informazione (es. fatture, moduli, testi, multimedia, ecc...). Esiste un unico punto di accesso logico, l'API REST attraverso la quale sono esposte le informazioni.

Sebbene IDN-SA possa, in questo senso, essere considerato come un database utilizzato dalle applicazioni per lo storage dei dati, occorre rilevare una differenza sostanziale rispetto a tale tecnologia di immagazzinamento dati: mentre un database ha un unico punto di accesso sia logico che fisico, IDN-SA è un sistema realmente distribuito. Non esistono infatti "istanze" separate come accade nei database, le quali possono eventualmente essere inter-relazionate con tecniche quali i db-link, clustering, ecc ...; esiste invece un'unica ragnatela di informazioni (e server che le gestiscono) a livello globale. Pertanto IDN-SA si presenta come una soluzione per la realizzazione del Read-Write Web of data che, sebbene in presenza di un sistema

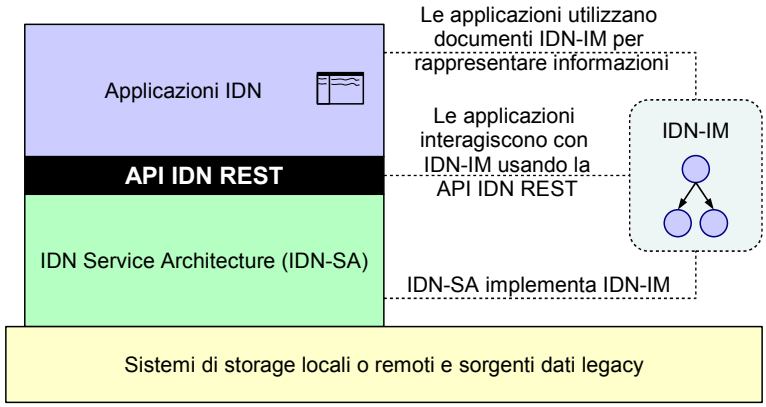

Figura 4.1: Relazione tra IDN-IM e IDN-SA.

decentralizzato, offre un unico punto di accesso logico, ovvero l'interfaccia uniforme per la manipolazione dei documenti.
Internamente ad IDN-SA i servizi sono organizzati ed ordinati in livelli e, grazie alla stratificazione, le applicazioni interagiscono esclusivamente con l'interfaccia REST del livello più alto. Così come Internet risolve i problemi della comunicazione per le reti interconnesse delegando a livelli diversi la cura di aspetti specifici [Alm92, KS06], IDN-SA propone l'uso di livelli differenti per abilitare la collaborazione tra persone, imprese ed organizzazioni a livello globale. Ogni livello affronta una problematica o caratteristica propria dell'informazione (documento) ponendola su un piano condiviso e collaborativo.
È opportuno ricordare che i principi fondanti la stratificazione (e quindi anche IDN-SA) sono i seguenti:

- *separation of concern*: ogni livello viene definito trattando separatamente i soli problemi che lo riguardano, tralasciando problematiche inessenziali o che verranno trattate da altri livelli;

- *information hiding*: ogni livello espone all'esterno solo l'informazione indispensabile alla comunicazione con i livelli adiacenti, mantenendo interna ogni altra necessaria informazione;

- *good enough*: la progettazione deve essere sufficiente a risolvere il problema, senza pretendere il miglior risultato possibile in tutte le circostanze.

Ogni livello è tenuto a rispondere in maniera corretta alle chiamate che gli competono e che verranno generate dai livelli ad esso adiacenti; la logica

interna, con cui le funzioni di competenza verranno elaborate, non è visibile dall'esterno.
La stratificazione come stile architetturale è ormai un dato di fatto ed è vincente non solo nelle reti di telecomunicazione (ISO/OSI), ma anche nella progettazione software [FRF+03, BMR+96, AZ05].
I vantaggi inoltre si presentano anche a lungo termine attraverso una migliore divulgazione e comprensione, così come nella riduzione dei costi di sviluppo a parità di qualità con altre metodologie [ZEWH95], quando si cerca una soluzione scalabile ed implementabile per parti successive migliorabili in modo incrementale, promuovendo il riuso, la crescita e l'interoperabilità.
La stratificazione si rileva vantaggiosa anche nei costi di sviluppo e di aggiornamento, poiché l'indipendenza della logica implementativa dei vari strati può essere migliorata e modificata senza ripercuotersi sugli strati adiacenti, a patto che le interfacce non subiscano modifiche.
La stratificazione è vantaggiosa su larga scala non solo per l'erogazione dei servizi, ma anche per la gestione ed il controllo scalabile delle risorse; è possibile ricorrere ad una elevata astrazione delle entità in modo da attuare una chiara separazione tra le componenti del sistema, mediante una strategia *divide et impera* [GKT02] . Ovviamente la stratificazione non riduce la complessità del problema, ma aiuta a ridurre la complessità della progettazione dei singoli elementi costituenti il sistema.
Infine la stratificazione apporta considerevoli vantaggi per quanto riguarda la coreografia poiché, imponendo dei rigidi vincoli operativi, consente di ridurre il numero delle possibili interazioni fra i vari servizi che costituiscono l'architettura, limitandone l'esplosione della complessità.
In IDN-SA la stratificazione è applicata in modo da lasciare maggiore libertà implementativa ad entrambi gli estremi della pila: a livello più basso deve essere resa flessibile l'integrazione con tecnologie esistenti (tra cui quelle legacy), mentre a livello più elevato occorre consentire lo sviluppo delle applicazioni di dominio.
Sebbene IDN-SA ponga l'attenzione sui servizi e su come essi interagiscono è importante chiarire che essa elabora risorse, rispettando i principi fondanti il REST/ROA, oltre a recepire i principi generali di SOA [Jos07].
Come illustrato in figura 4.2, IDN-SA si compone di quattro livelli, ognuno dei quali caratterizza il comportamento delle risorse:

- il livello più alto, di nome *Virtual Resource* (o *VR*), gestisce l'aggregazione e l'indirizzamento logico ed unitario;

- subito sotto, l'*Information History* (o *IH*) si occupa della storicizzazione;

- quindi, il *Replica Management* (o *RM*) gestisce la replicazione e l'indirizzamento uniforme;

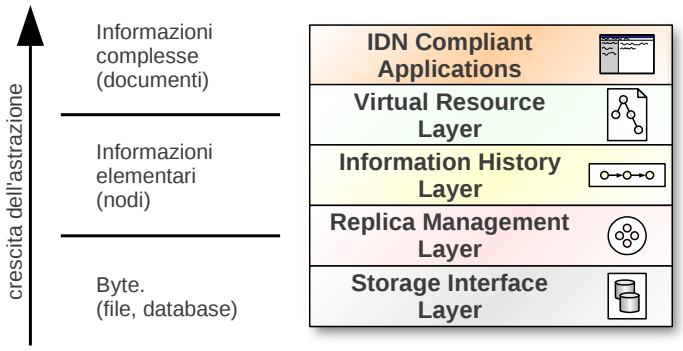

Figura 4.2: Livelli dell'architettura IDN.

- infine, il livello di *Storage Interface* (o *SI*) cura la memorizzazione fisica e l'accesso.

I livelli più alti percepiscono l'informazione come entità complessa sotto forma di documento, costituito da una serie di informazioni elementari legate tra loro da relazioni strutturali. Spostando l'attenzione verso i livelli intermedi gli aspetti strutturali dell'informazione vengono meno e l'informazione viene vista come un insieme di tante unità elementari indipendenti. Infine per i livelli più bassi l'informazione è semplicemente una particolare codifica delle entità definite ai livelli superiori, in altre parole "sequenze di byte" (per esempio file o record di database).

A sinistra in figura 4.3 è riportata la pila dei servizi di cui IDN-SA si compone, con indicati agli estremi le tecnologie ed i contesti che sebbene esulino dalla specifica trattazione dell'architettura, rappresentano comunque dei vincoli.

In IDN-SA il sistema di indirizzamento assume un ruolo strutturalmente portante e permeante: infatti ogni livello specializza il modo di nominare, individuare ed accedere alle risorse e pertanto risulta necessario introdurre delle regole e delle metodologie operative al fine di definire operazioni di *risoluzione* per permettere il passaggio dalla classe di nomi del livello superiore a quella del livello inferiore.

Attraverso l'applicazione della separation of concerns sono "estratti" opportuni sottoservizi dedicati ai nomi, ciascuno dotato di un risolutore che, analogamente a quanto accade nella pila Internet, partendo da uno qualunque dei livelli permette di raggiungere il SAP (Service Access Point) del livello inferiore, abilitando quindi l'attraversamento del livelli.

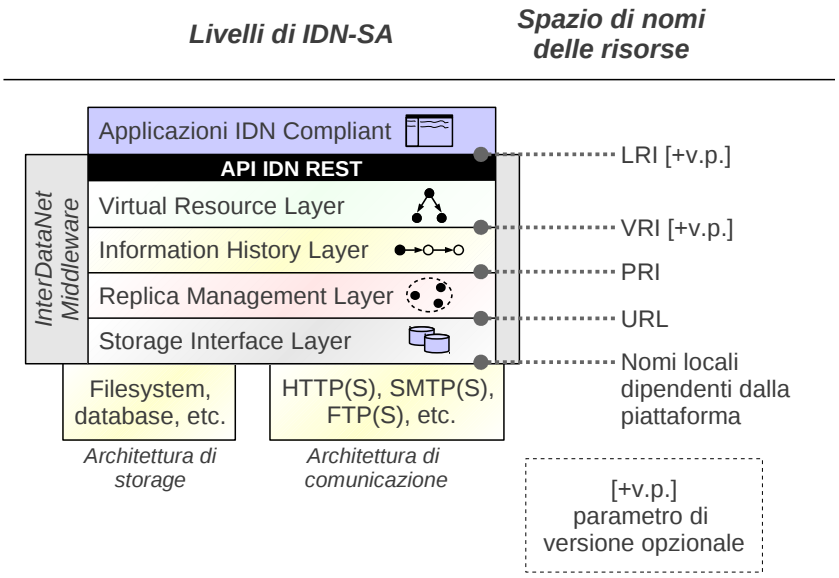

Figura 4.3: Spazi dei nomi delle risorse IDN.

4.1 IDN-SA: vista d'insieme

In questo paragrafo verranno riportati alcuni vincoli e considerazioni che derivano dall'aver utilizzato la stratificazione come paradigma architetturale. Saranno illustrate le modalità di interazione che si sviluppano fra i livelli a seguito di una richiesta dell'applicazione e come l'informazione viene identificata, trasformata e rappresentata durante lo svolgimento dell'interazione.

4.1.1 Gli IDN-Node

InterDataNet è un sistema orientato alle risorse e come tale la realizzazione delle interfacce è guidata dai principi REST/ROA.
IDN-SA espone alle applicazioni un'interfaccia uniforme; gli stessi principi e concetti che sono applicati in quest'ultima sono utilizzati anche per le interfacce esistenti fra le componenti interne ai livelli (i cosiddetti sottolivelli, introdotti nel paragrafo precedente) e quelle fra livelli adiacenti.
A tal fine è stato definito l'*IDN-Node*, una struttura informativa che si attiene alla metafora contenitore/contenuto, utilizzata come soggetto principale dell'interazione fra i livelli IDN, le cui caratteristiche principali sono:

- ha un nome di tipo HTTP-URI che lo identifica a livello globale;
- è il contenitore di una serie di triple `<nome,tipo,valore>` dotate delle seguenti caratteristiche:

  - un nome che le identifica univocamente all'interno dell'IDN-Node in cui sono contenute;
  - un valore che ne esprime il contenuto informativo vero e proprio;
  - un tipo (espresso tramite un HTTP-URI) che ne determina le caratteristiche, in particolare il formato del valore;

- avendo scelto di utilizzare l'approccio REST, ogni IDN-Node può essere manipolato sfruttando direttamente richieste HTTP:
  - `GET` per richiedere un IDN-Node;
  - `PUT` per creare un nuovo IDN-Node o per aggiornarne uno esistente;
  - `DELETE` per eliminare un IDN-Node.

Come si può intuire esiste una stretta correlazione fra gli IDN-Node e le informazioni primitive del modello IDN-IM e fra i contenuti appena citati e le informazioni atomiche. In modo particolare i *VR-Node*, che rappresentano la specializzazione degli IDN-Node trattata al livello Virtual Resource, sono un possibile Data Model dell'IDN-IM. I livelli inferiori dell'architettura specializzano gli IDN-Node (in *IH-Node*, *RM-Node* e *SI-Node* rispettivamente) per la rappresentazione dell'informazione al loro livello, ma in questo caso non esiste una diretta correlazione fra tali specializzazioni e gli elementi dell'IDN-IM.
Riassumendo ogni livello dell'architettura mostrata in figura 4.3 espone le proprie funzionalità al livello superiore utilizzando un'interfaccia uniforme REST rappresentando le risorse che tratta tramite una specializzazione degli IDN-Node.

4.1.2 Identificazione degli IDN-Node

Affinché possa essere correttamente condiviso, divulgato e riusato, un elemento informativo necessita di possedere due proprietà fondamentali: deve essere individuabile ed accessibile. Nei sistemi telematici una risorsa viene individuata mediante il suo indirizzo e spesso questo diviene anche il metodo utilizzato per l'accesso fisico (come nel caso degli URL).
Il livello più basso di IDN-SA, ovvero Storage Interface è dedicato alla memorizzazione fisica delle informazioni e utilizza per l'indirizzamento degli SI-Node, proprio gli URL (intesi come HTTP-URI che forniscono identificazione ed accesso fisico tramite HTTP).
All'interno di IDN l'identificazione dei nodi è una proprietà che deve essere mantenuta indipendente dai dettagli dell'accesso fisico, poiché quest'ultimo non rappresenta un contesto di interesse per un Information Model. In altre parole è importante disporre di identificatori che vadano ad astrarre dalla localizzazione fisica delle risorse e siano quindi indipendenti dalla stessa.

Gli obiettivi del livello superiore, Replica Management, consistono nel fornire una visione delocalizzata dell'informazione e, compatibilmente con i requisiti di affidabilità e prestazioni richiesti, replicarla in locazioni fisiche (ovvero istanze di Storage Interface) diverse. La delocalizzazione è uno dei temi trattati dall'IETF nella definizione degli *URN* (*Uniform Resource Name*) [SM94, Moa97, DvGIF02, Dan97]; questi ultimi sono URI che differiscono dagli URL per il fatto di identificare una risorsa indipendentemente dalla sua locazione fisica in modo non ambiguo ed univoco e non contenere informazioni variabili nel tempo [1]. All'interno di IDN si definiscono *Persistent Resource Identifier* (*PRI*) gli URN, espressi comunque utilizzando HTTP-URI, che identificano gli RM-Node; tale nome evidenzia la principale proprietà che li caratterizza, ovvero la persistenza. In basso in figura 4.4 è possibile notare come n repliche della stessa informazione (codificate in n SI-Node distinti) siano esposte dall'interfaccia di SI tramite URL e che tutte facciano capo ad un unico PRI attraverso il quale Replica Management espone l'informazione al livello superiore (in un unico RM-Node). Replica Management ha il compito di mantenere consistenti le repliche e le relative associazioni PRI $\rightarrow$ URL mascherando completamente la delocalizzazione e la replicazione al livello superiore, il quale manipola esclusivamente RM-Node tramite il relativo PRI.

Il livello superiore a Replica Management, ovvero Information History, ha il compito di versionare l'informazione, rendendo parte dello stesso storico RM-Node distinti. Passando quindi dallo spazio dell'informazione non versionata (RM-Node) a quella versionata (IH-Node) si è resa necessaria l'introduzione di un'opportuna classe di nomi finalizzata ad indirizzare l'informazione esposta da questo livello: i *VRI* (*Versioned Resource Identifiers*). Ogni VRI identifica un elemento all'interno di uno storico del quale è possibile indicare altri elementi mediante l'uso di opportuni parametri di versione; i VRI espongono per ogni istante temporale l'accesso all'ultima versione disponibile. L'approccio utilizzato per raggiungere questo risultato è il medesimo implementato in Subversion [CSFP04], ovvero il VRI privo del parametro di versione indica l'ultima release (nel branch in cui si trova l'elemento) della risorsa informativa; similmente in Subversion l'URI che individua una risorsa nel repository, privo del parametro di versione, fa riferimento alla release HEAD, ovvero l'ultima, della risorsa stessa.

In figura 4.4 VRI1 e VRI2+vp1 (dove vp sta per *version parameter*) fanno capo allo stesso PRI, ovvero PRI1. Riprendendo l'esempio riportato nel capitolo 3 in figura 3.6 (a pagina 71), VRI1 potrebbe essere il VRI che, essendo privo del parametro di versione, indica l'ultimo elemento del ramo principale ovvero il nodo il cui numero di versione assoluto (indicato esternamente al nodo stesso) è 7. Il nome VRI2 potrebbe essere relativo ad uno qualunque

[1] Un buon esempio di URN è il codice ISBN [LPD98, HW01], il quale identifica un libro, ma nessuna delle sue copie.

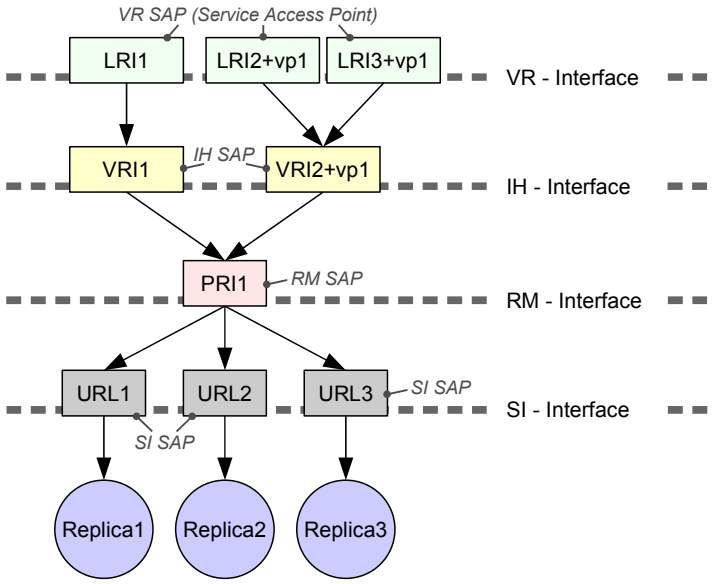

Figura 4.4: Nomi di risorse replicate.

dei nodi presenti negli altri rami, ma ipotizzando che vp1 valga T_ABSLAST la coppia VRI2+vp1 indica anch'essa l'elemento il cui numero di versione assoluto è 7 e quindi, dato che ogni nodo è legato da una relazione biunivoca al proprio PRI, entrambi i nomi VRI1 e VRI2+vp1 fanno capo a PRI1.
Poiché tutti gli identificatori citati (URL, PRI e VRI) sono ad uso e consumo della macchina, non sono necessarie restrizioni per renderli facilmente utilizzabili e trascrivibili dall'uomo, anche se ciò è comunque auspicabile. Dato che gli utenti hanno la necessità di assegnare nomi facilmente intellegibili e condivisibili con altri per indicare concetti e contenuti, seguendo quanto suggerito nella RFC 3401 [Mea02], sono stati definiti i *Logical Resource Identifier* (*LRI*), i quali possono essere considerati come un caso particolare di *Human Friendly Name* (*HFN*), ovvero nomi di alto livello progettati a misura d'uomo [SM94]; anche gli LRI vengono rappresentati tramite HTTP-URI.
Gli LRI rappresentano la classe dei nomi introdotta per identificare le informazioni offerte dal Virtual Resource, il quale ha il compito principale di aggregare le risorse ed offrirle all'applicazione sotto forma di documento IDN.
Gli LRI sono nomi flessibili, descrittivi e relativi ad uno specifico contesto; inoltre possono risultare facilmente comprensibili e interpretabili dagli utenti purché definiti ed assegnati in modo opportuno, ad esempio sulla base del paradigma di contenimento e catalogazione "folder/file" [2].

[2]Il path degli URI segue già la convenzione citata: è quindi sufficiente assegnare i nomi in modo oculato.

Gli LRI permettono la definizione di alias, ovvero di nomi diversi che indicano la stessa risorsa, allo stesso modo con cui vengono usati gli alias negli indirizzi di posta elettronica o i link simbolici nei file system Unix. Come vantaggio di ciò gli utenti possono assegnare in libertà nomi di convenienza alle risorse e se decidono di modificare il proprio LRI l'operazione risulterà trasparente agli utenti che utilizzano altri LRI per indicare la stessa risorsa. Ad esempio in figura 4.4 LRI2 e LRI3 fanno capo allo stesso VRI, ovvero VRI2; questo significa che i due LRI sono l'uno alias dell'altro. Si noti inoltre che il parametro di versione è necessariamente lo stesso in quanto questo elemento attraversa trasparentemente il livello Virtual Resource venendo gestito dal livello inferiore (Information History). Questo rende evidente come i parametri di versione introdotti nel contesto dei VRI siano gli stessi utilizzati con gli LRI e quindi nel modello IDN-IM (come illustrato nel paragrafo 3.7.2).

4.1.3 Attraversamento dei livelli

L'interazione fra i quattro livelli di IDN-SA avviene tramite il paradigma client/server, ovvero ogni livello è client di quello inferiore (al quale richiede servizi) e server di quello superiore (al quale fornisce servizi). Quindi l'interazione avviene, per ogni coppia di livelli, mediante un approccio request/response, utilizzando HTTP [FGM+99]. Il paradigma utilizzato è quindi di tipo *pull*: in seguito ad un'azione dell'utente, la pila viene attraversata da una sequenza di request che si sviluppa dall'alto verso il basso e da una sequenza corrispondente di response che la percorre in senso opposto. Ogni livello si attiva per generare la response a seguito di ogni request proveniente dal livello superiore, eventualmente effettuando delle richieste ai livelli inferiori (come evidenziato in figura 4.5) per richiedere l'espletamento dei servizi necessari per il completamento dell'operazione.
Le operazioni di request e response ruotano attorno al concetto di IDN-Node: infatti ogni livello richiede lo svolgimento di una determinata azione (`GET`, `PUT`, `DELETE`) sulla specializzazione dell'IDN-Node esposta dal livello sottostante (VR-Node, IH-Node, RM-Node e SI-Node) utilizzando il nome HTTP-URI dell'elemento ed attende la response contenente l'esito della richiesta.
A titolo esemplificativo si consideri una `GET` di un documento (ovvero una operazione scaturita dal livello applicativo): il flusso di operazioni di request e response può essere così descritto:

1. Virtual Resource si attiverà per comporre il documento effettuando tante richieste ad Information History quanti sono i nodi che lo costituiscono;

2. Information History richiederà uno o più (dipendentemente dalle modalità di navigazione dello storico che esulano dalla presente descrizione) RM-Node al livello Replica Management;

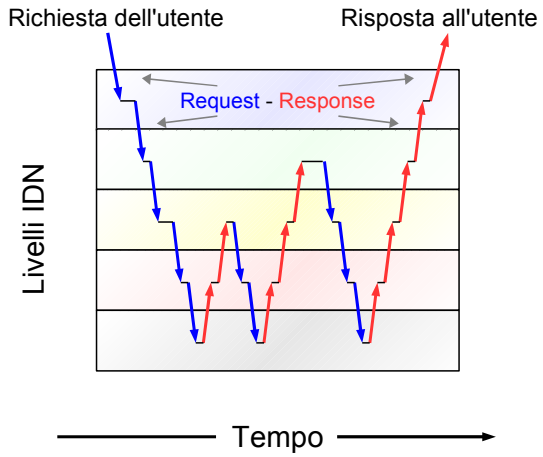

Figura 4.5: Interazione request/response applicata ad IDN.

3. Replica Management, per ogni RM-Node richiesto, accederà all'istanza di Storage Interface autoritativa sulla replica prescelta.

4. infine a partire dal basso si hanno le response che permettono di far risalire l'informazione nella pila IDN trasformandola, livello per livello, fino a farle assumere il formato voluto (il documento IDN-IM).

All'interno del paragrafo successivo saranno descritti i meccanismi che attuano la trasformazione citata all'ultimo passo.

4.1.4 Imbustamento di dati e metadati

Come illustrato al paragrafo precedente, le informazioni vengono elaborate e trattate a seguito della comunicazione fra i livelli. Da un punto di vista generale si può affermare che ogni livello di IDN-SA elabora le informazioni scomponendole, con la granularità necessaria, in dati elementari. Inoltre ogni livello associa alle stesse un insieme di metadati necessari (o opzionali) al fine di trattare le informazioni secondo quanto previsto da IDN-IM. Sia i dati sia i metadati sono rappresentati, livello per livello, tramite l'IDN-Node della apposita tipologia, applicando il principio di *information hiding*.
Più nel dettaglio, come rappresentato in figura 4.6:

- le applicazioni IDN definiscono a proprio uso e consumo le rappresentazioni delle informazioni che ritengono opportune, interagendo con il Virtual Resource. Quest'ultimo offre alle applicazioni le funzionalità necessarie per la gestione dei VR-Node, utilizzando un'interfaccia REST. I VR-Node offrono un'area per la memorizzazione dei dati e dei metadati (indicata come VR-USER in figura 4.6); inoltre contengono metadati aggiuntivi necessari alla gestione del documento IDN-IM,

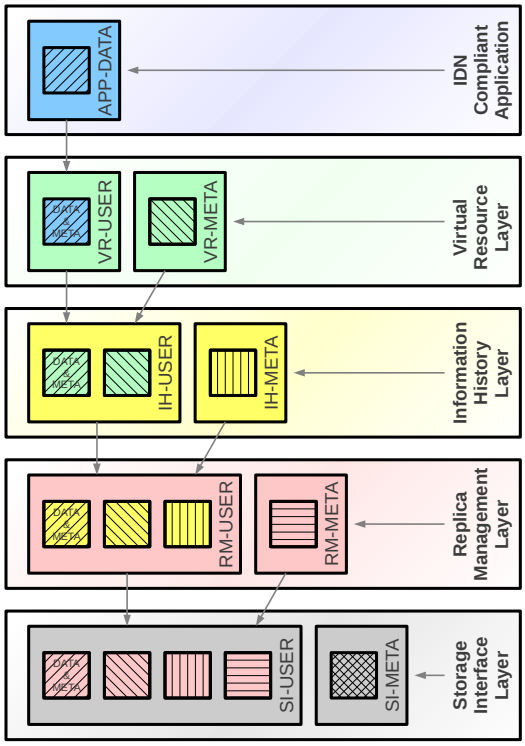

Figura 4.6: Imbustamento di dati e metadati.

come ad esempio i link di relazione fra i nodi, l'autore, la data di creazione, ecc... (si tratta della sezione VR-META in figura 4.6). Al fine di delegare la gestione dei propri dati ad InterDataNet, l'applicazione crea un nuovo VR-Node, imbustando i propri dati e metadati nell'area VR-USER, e ne richiede la gestione al Virtual Resource;

- Virtual Resource trasforma il VR-Node (aggiungendo, rimuovendo e/o modificando tutti quei metadati di sua competenza, indicati come VR-META) in uno o più IH-Node e ne delega la gestione al livello inferiore. Tali nodi mantengono in IH-META tutti i metadati di competenza del livello Information History ed in IH-USER i dati ed i metadati del livello superiore, incapsulati;

- Information History elabora gli IH-Node negli opportuni RM-Node, per la memorizzazione persistente; a tal fine crea un nuovo RM-Node incapsulando in RM-USER tutti i dati e metadati di sua competenza e in RM-META i metadati necessari a Replica Management per espletare le proprie mansioni;

- Replica Management partendo dal RM-Node crea un SI-Node per ogni replica, rispettando i principi esposti precedentemente;

- infine Storage Interface offre il servizio di memorizzazione fisica degli SI-Node.

Se invece si analizza la situazione da un punto di vista bottom-up, si osserva che ogni livello sfrutta il livello inferiore (al fine di servire le richieste che riceve dal superiore) richiedendo ad esso il nodo (o i nodi) di interesse da cui estrarre, in modo complementare all'incapsulamento, i propri dati e i propri metadati che utilizza per comporre la risposta relativa alla richiesta di servizio che ha ricevuto.

4.2 IDN naming system

All'interno del paragrafo precedente è stato descritto il meccanismo di incapsulamento e trasformazione degli IDN-Node durante l'attraversamento dei livelli; in particolare nel paragrafo 4.1.2 è stato messo in evidenza come ogni classe di IDN-Node (e quindi indirettamente ogni livello dell'architettura) abbia una corrispondente classe di nomi di tipo HTTP-URI per la relativa identificazione; i nomi utilizzati non sono semplici etichette testuali che identificano gli IDN-Node, ma attraverso di essi è possibile interrogare, utilizzando HTTP, un ben determinato host (indicato nell'hostname dell'URI) per richiedere una rappresentazione della risorsa, coerentemente ai principi REST/ROA.
In questo paragrafo verrà descritto *come* avviene il passaggio dalla classe di nomi del livello superiore alla classe che rappresenta il SAP (Service Access Point) del livello inferiore ovvero, facendo riferimento alla figura 4.4, sarà chiarito come vengono gestite le relazioni espresse tramite le frecce che legano i nomi LRI con i VRI, i VRI con i PRI ed infine i PRI con gli URL. Tale operazione viene effettuata utilizzando un opportuno sistema dei nomi, simile al DNS [3]. Più precisamente, per la separation of concerns, si hanno tre sottosistemi diversi che condividono i principi di funzionamento:

- *Logical Domain Name System* (LDNS), sottoservizio di Virtual Resource, per la risoluzione diretta ed inversa da LRI in VRI;
- *History Name System* (HNS), sottoservizio di Information History, per la risoluzione diretta ed inversa da VRI in PRI;
- *Localization Service* (LS), sottoservizio di Replica Management, per la risoluzione diretta ed inversa da PRI in URL.

La figura 4.7 illustra le classi di nomi ed i relativi sistemi di risoluzione localizzati internamente ai livelli di afferenza.

[3] Il *DNS* (*Domain Name Sistem*) è il sistema che offre (principalmente) il servizio di risoluzione da hostname ad indirizzo IP; è realizzato tramite un database distribuito, costituito dai server DNS.

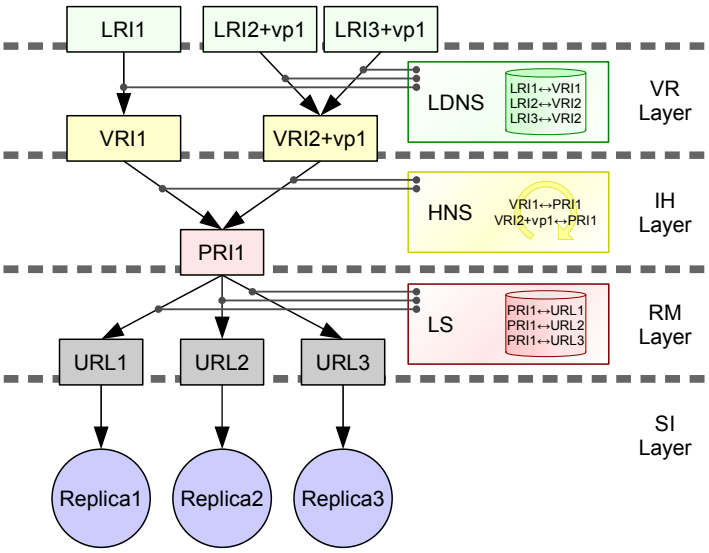

Figura 4.7: Sistema dei nomi di InterDataNet (risoluzione diretta).

L'interfaccia del sistema dei nomi stata progettata utilizzando per tutti e tre i sottosistemi gli stessi principi della definizione delle interfacce dei livelli; infatti, sebbene ogni sottosistema sia interno al layer di appartenenza, esso può essere raggiunto ed interrogato tramite la generica interfaccia REST basata sul concetto di IDN-Node. In particolare, come illustrato in figura 4.8 sono state create alcune specializzazioni dell'IDN-Node al fine di rappresentare le varie "*risorse nome*"; esse sono gli LRI-Node, i VRI-Node, i PRI-Node e gli URL-Node. Questo significa che, dato un nome, esiste una risorsa di tipo IDN-Node che ne offre una rappresentazione; in linea generale quindi ogni nome indica la risorsa informativa a cui fa riferimento ed ha associata (almeno) un'altra risorsa informativa che lo descrive.

4.3 Storage Interface

Il livello più basso di IDN-SA, lo Storage Interface, si occupa di definire l'interfaccia verso i sottosistemi di storage esistenti, consentendo di memorizzare e gestire risorse sulle piattaforme sottostanti (anche eterogenee), esponendo verso il livello superiore un'interfaccia uniforme, indipendentemente dalle particolarità tecnologiche mascherate.

Le risorse possono essere memorizzate su file system, database locali o remoti oppure generate dinamicamente a partire da una qualunque sorgente; dal punto di vista dell'architettura, l'accesso fisico ad esse viene effettuato a questo livello.

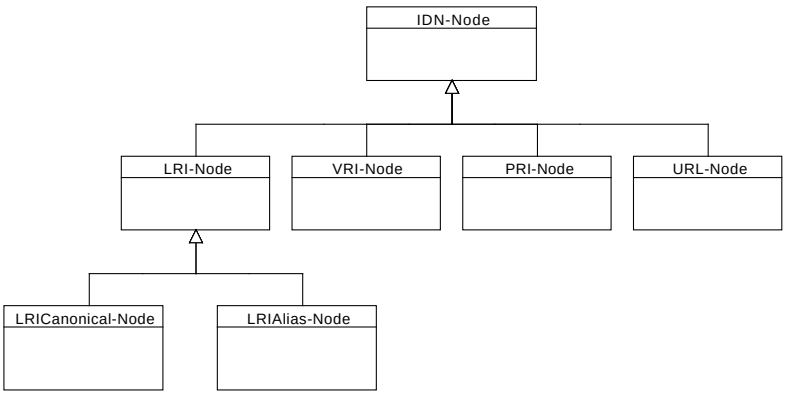

Figura 4.8: Gerarchia delle risorse "nome" di InterDataNet.

L'interfaccia esposta dallo Storage Interface prevede indirizzamento tramite URL, la stessa tipologia di indirizzi che viene utilizzata con la finalità di identificare ed accedere ad entità di livello applicativo in Internet; pertanto ogni risorsa (detta SI-Node) disporrà di un proprio URL che la identifica e ne permette l'accesso.
Lo scopo principale di questo livello è quello di memorizzare repliche di risorse (logiche) definite al livello superiore Replica Management. Tuttavia SI può essere utilizzato anche al di fuori di InterDataNet come servizio di storage e/o di accesso con interfaccia uniforme verso sorgenti dati legacy, dato che l'accoppiamento tra i livelli è unidirezionale, ovvero solamente RM è vincolato ad usare l'interfaccia di SI, mentre quest'ultimo è del tutto indipendente dal primo. Nel seguito verrà dato comunque per scontato che Storage Interface sia parte dello stack IDN.
L'interfaccia uniforme esposta da SI utilizza le operazioni base CRUD (Create, Read, Update e Delete tipiche dei database) implementate in stile REST:

- *creazione*: operazione che vede la nascita di una nuova risorsa, eventualmente vuota, grazie all'allocazione di spazio (fisico o logico) necessario per la memorizzazione, fornendo al livello superiore l'*handle* (l'URL) per accedervi. La creazione, in linea con i principi REST, viene effettuata tramite `PUT`, utilizzando il nome stabilito dal client;

- *modifica*: operazione che consente di aggiornare il contenuto di una risorsa già esistente. In ottica REST questo avviene ancora tramite la `PUT`. Mentre in questo caso la primitiva viene invocata su un nome di una risorsa esistente, nel caso della creazione il nome non esiste ancora;

- *lettura*: operazione che permette di ottenere il contenuto della risorsa, implementata con una `GET`;

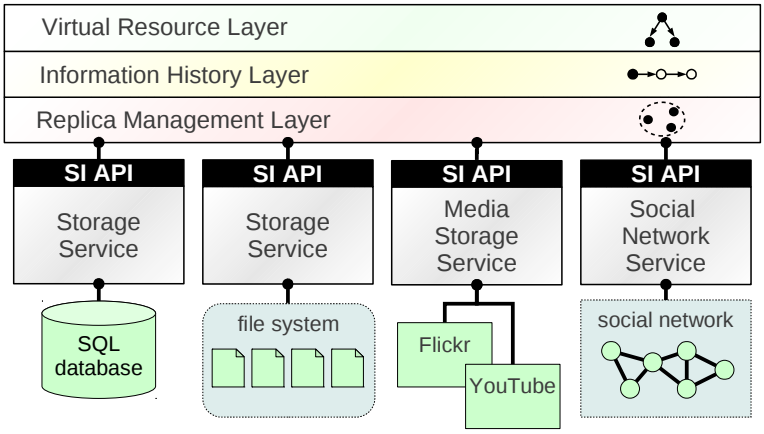

Figura 4.9: Storage Interface e Legacy Storage Interface.

- *cancellazione*: operazione che va ad eliminare una risorsa già esistente, "liberando" handle e spazio; è implementata con la `DELETE`.

Focalizzando l'attenzione sui dati e sui metadati, è possibile notare che il livello offre un'arbitraria granularità di memorizzazione ed indirizzamento.

Per ulteriori dettagli sul livello Storage Interface si rimanda alla bibliografia esistente [Inn08].

4.3.1 Integrazione di sistemi legacy

Come introdotto precedentemente, uno degli scopi di SI è quello di esporre risorse ai livelli superiori tramite un'interfaccia uniforme.

Realizzare un'applicazione che espone tale API, significa implementare un *servizio di storage* per IDN. Le proprietà definite dalle specifiche dell'interfaccia fanno sì che su tale applicazione si possa montare lo stack dei livelli superiori permettendo quindi di andare ad utilizzare una moltitudine eterogenea di soluzioni per lo storage fisico. In questo senso il layer inferiore (così come quello applicativo) è stato vincolato in modo quanto più lasco possibile, andando ad imporre esclusivamente la sua interfaccia di accesso.

In figura 4.9 si può vedere una esemplificazione nella quale gli strati superiori di IDN-SA pongono le loro basi su una varietà di differenti servizi di storage, implementati con tecnologie diverse.

Diviene così possibile realizzare istanze specializzate di Storage Interface di modo che le organizzazioni interessate possano integrare i loro sistemi all'interno di IDN-SA, rimanendo contemporaneamente aperti all'integrazione di nuove soluzioni per lo storage.

4.4 Replica Management

Il livello Replica Management si occupa della gestione delle repliche delle informazioni, ovvero della creazione di copie (ognuna delle quali è detta replica) che vengono memorizzate in differenti apparati e mantenute equivalenti nel tempo (a meno della latenza). Le singole repliche delle informazioni sono singolarmente indirizzate ed accedute tramite URL utilizzando il servizio esposto dal livello Storage Interface, mentre come insieme sono identificate attraverso un PRI.
Replica Management implementa tutti i meccanismi necessari per esporre un'interfaccia simile a quella del livello inferiore Storage Interface, definita però su RM-Node e non su SI-Node [4].
La presenza di un servizio per la gestione delle repliche diverse dell'informazione permette di:

- aumentare la disponibilità delle risorse;
- aumentare la velocità di risposta del sistema;
- aumentare la tolleranza ai guasti [Bar07].

Una sequenza tipica di accesso si svolge nel modo seguente:

1. il livello superiore (ovvero IH) invia ad RM una richiesta di accesso ad un determinato PRI;
2. RM interroga il proprio sottosistema dei nomi (LS) ed ottiene da esso la lista degli URL che sono associati alla risorsa;
3. RM seleziona un URL tra quelli disponibili;
4. RM richiede la risorsa all'istanza di servizio di storage SI corrispondente;
5. RM trasforma l'SI-Node ricevuto al fine di estrarre le informazioni di interesse e confezionare l'RM-Node da inviare al livello superiore.

Una sequenza tipica di aggiornamento si svolge nel modo seguente:

1. il livello superiore (ovvero IH) invia una richiesta di aggiornamento di un RM-Node;
2. RM interroga il proprio sottosistema dei nomi (LS) ed ottiene da esso la lista degli URL che sono associati alla risorsa;

[4] Dal punto di vista dell'utilizzatore le differenze fra l'interfaccia di Replica Management e quella dello Storage Interface sono minime, cambiano però le proprietà degli elementi informativi manipolati.

3. RM trasforma l'RM-Node ricevuto in tanti SI-Node al fine di incapsulare le informazioni di interesse in ognuna delle repliche;

4. RM aggiorna le repliche [5].

Per ulteriori dettagli sulla replicazione e sul livello Replica Management si rinvia all'apposita bibliografia [Bar07, Inn08].

4.4.1 Localization Service

La figura 4.7 mostra come Localization Service sia un servizio interno a Replica Management; esso infatti permette al livello RM di memorizzare e mantenere nel tempo le associazioni "PRI" → "URL1, URL2, ..., URLn", ovvero è lo strumento che, a partire da un nome che non ha riferimenti legati alla locazione fisica delle risorse (il PRI), permette di individuare le repliche. Localization Service è quindi un database locale ad ogni istanza di Replica Management e come tale memorizza le sole associazioni sui cui PRI è autoritativa (ovvero i PRI il cui hostname coincide con quello del server che ospita l'istanza di RM).

4.5 Information History

Lo strato di Information History (IH) si occupa della gestione delle versioni; per svolgere tale compito si comporta come un filtro, intercettando le operazioni che modificano le risorse, controllandone il numero di versione e tenendo traccia di revisioni, diramazioni e merge, come nelle più diffuse tecniche di versioning; in tal modo rende possibile richiedere una risorsa come istanza storica, in uno dei suoi stati assunti dalla sua creazione fino al tempo attuale.
All'interno dell'IH i dati sono collegati strutturalmente mediante i *link di versioning*; essi non sono direttamente visibili nell'istanza di documento IDN-IM e possono essere considerati a tutti gli effetti metadati interni a Information History.
Ogni risorsa di questo livello è dotata di un identificatore costituito da una parte persistente, chiamata Versioned Resource Identifier (VRI), e da un parametro di versione (PARVER) opzionalmente giustapposto al VRI.
Generalmente, nel corso di un'operazione di lettura, il livello VR, grazie alla risoluzione di LDNS, richiede al livello IH una risorsa individuata da un parametro di versione (indicato anche con PARVER). Un apposito algoritmo, detto *Version Traversing Algorithm* (o *VTA*) consente di navigare nello storico scorrendo i link di versioning in maniera iterativa fino a recuperare l'elemento individuato dal parametro di versione; tale algoritmo termina nel

[5] L'operazione di aggiornamento delle repliche può avvenire in modo sincrono oppure asincrono.

momento in cui recupera il PRI della versione richiesta. Si può affermare a tutti gli effetti che VTA esegue una trasformazione da VRI+PARVER in un PRI utilizzando nel processo il sottosistema dei nomi per il livello IH, ovvero HNS. In tal modo risulta possibile accedere all'RM-Node di interesse, una volta ottenuto il quale viene effettuata la trasformazione necessaria per confezionare l'IH-Node da fornire al livello superiore.
Per ulteriori dettagli su Information History, il versioning ed i vari algoritmi interessati si rimanda all'apposita bibliografia [Chi05, Inn08].

4.5.1 History Name System

Il sottolivello HNS (History Name System) si occupa della risoluzione da un VRI (privo di parametro di versione) al PRI della radice del branch in cui si trova l'elemento di interesse.
La figura 4.7 mostra che HNS è un servizio interno a Information History; esso non è un database locale all'istanza di IH (come avviene invece per LS ed LDNS), ma opera algoritmicamente, al fine di garantire estrema flessibilità nell'assegnazione delle relazioni fra VRI e PRI. Infatti la farfalla "molti LRI" $\leftrightarrow$ "VRI/PRI" $\leftrightarrow$ "molti URL" è sufficiente per ottenere tutte le caratteristiche desiderate dai nomi ed è opportuno che esistano dei vincoli stringenti fra VRI e PRI per poter modellare senza complicazioni il comportamento evolutivo previsto dal versioning in IDN-IM.

4.6 Virtual Resource

Il Virtual Resource (VR) si occupa dell'indirizzamento delle risorse al fine di consentire la navigazione nello spazio delle informazioni; attraverso la sua interfaccia è possibile indirizzare ed individuare le risorse disponibili conformi al modello concettuale IDN-IM.
Quando viene individuata una risorsa, questa viene analizzata ed elaborata in maniera completamente trasparente al richiedente, recuperando tutte le sotto-informazioni (ovvero le PIU) di cui è composta e provvedendo alla loro aggregazione.
All'interno di VR possono essere individuati i seguenti sottoservizi:

- aggregazione di PIU (tale compito spetta al *RAS*, *Resource Aggregation Service*);
- risoluzione di nomi logici e alias (di competenza del *Logical Domain Name Service*, *LDNS*);
- controllo delle identità (gestito dall'*Identity Service*, *IS*). Per la descrizione dettagliata di tale servizio si rimanda all'opportuna bibliografia [Inn08, Chi09].

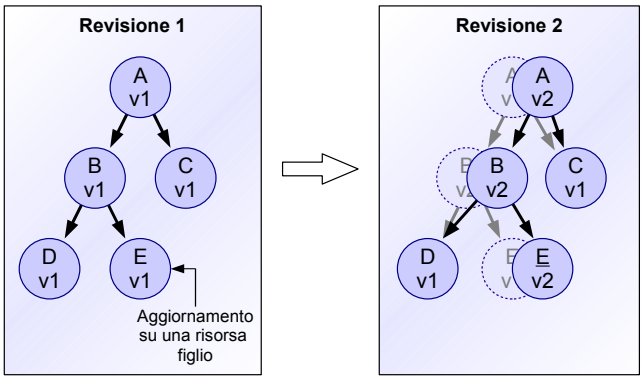

Figura 4.10: Revisioni successive di documenti UEVM.

4.6.1 Resource Aggregation Service

Quando un utente indirizza un documento al fine di visualizzarlo, dopo che IS avrà accertato che egli disponga della necessaria autorizzazione, RAS provvede a recuperare tutti i nodi (PIU) da cui è costituito il documento, visitando in profondità la struttura a DAG.
Una volta recuperate le informazioni (ovvero i singoli nodi del DAG), il RAS è in grado di costruire il documento IDN di interesse dell'utente.
Per effettuare la corretta selezione delle versioni dei nodi, esplorando in profondità il DAG, RAS si avvale dei servizi offerti dal sottostante livello Information History (IH).
Non è compito di RAS operare direttamente sulle versioni della risorsa, ma sarà comunque suo onere richiedere al livello sottostante i servizi necessari per gestirle e relazionarle correttamente nel tempo.
In maniera duale all'aggregazione, RAS si occupa di avviare la propagazione delle versioni (introdotta nel capitolo 3), percorrendo i link strutturali di segnalazione. La modifica di una risorsa prevede infatti la propagazione dell'aggiornamento ai padri del nodo modificato e applicando il ragionamento alla gerarchia degli antenati, la propagazione risale fino alla radice del documento (si veda la figura 4.10).
Tale risultato viene ottenuto mediante un meccanismo di notifica: quando un nodo viene modificato, l'istanza autoritativa di Virtual Resource su di esso invia una notifica a tutti i nodi raggiungibili seguendo i link di segnalazione. Ad esempio si ipotizzi che `A` sia il nodo padre (come la radice in figura 4.10), `A(<aggr1 link=B />)` sia il link di aggregazione di nome `aggr1` che punta al nodo figlio `B` ed infine che `B(<prop1 link=A />)` sia il link di segnalazione, di nome `prop1`, da `B` ad `A`. In questo scenario a seguito della modifica di `B`, con la conseguente nascita della versione `v2`, il Virtual Resource autoritativo su `B` (quello che ha creato B_{v2}) andrà ad eseguire l'operazione `PUT(A(<aggr1 link=B changed=true />))` andando ad aggiornare

la proprietà `changed` sul link di aggregazione in `A`. Questa operazione verrà gestita dal Virtual Resource autoritativo su `A` come notifica dell'aggiornamento dell'elemento `B`, scatenando gli algoritmi necessari alla gestione di questo evento. In modo particolare il sistema andrà a proseguire l'algoritmo distribuito di propagazione creando la nuova versione della radice ($\mathtt{A}_{v2}$).

4.6.2 Logical Domain Name Service

Una medesima risorsa può essere nominata in modo diverso e collocata in contesti diversi pur mantenendosi invariabile da un punto di vista di contenuti. All'interno di IDN questo aspetto viene gestito assegnando a ciascuna risorsa uno o più nomi canonici a cui possono essere aggiunti ulteriori alias; gli LRI corrispondenti sono gestiti dal Logical Domain Name Service (LDNS), sottoservizio di Virtual Resource.
Esso è in grado di determinare l'identificativo canonico attuando una risoluzione che può essere svolta in uno o più passi (qualora il nome LRI di partenza sia un alias) e quindi, a partire da esso, stabilire il VRI necessario per poter inoltrare la richiesta al livello inferiore.
Inoltre LDNS gestisce anche la risoluzione inversa degli alias; si ricordi che, come illustrato nel paragrafo 3.6, sono definite due categorie di alias:

- *invertibili globalmente* (ovvero bidirezionali);
- *invertibili localmente* (ovvero monodirezionali).

La prima categoria offre un maggior livello di awareness globale in quanto, instaurando una relazione bidirezionale fra le parti in gioco, permette al responsabile del nome (relativamente al quale viene creato un alias) di conoscerne tutti gli alias.
La creazione di alias bidirezionali è comunque un processo relativamente complesso che richiede un livello di trust fra le parti in gioco; perciò sono stati introdotti gli alias non invertibili globalmente (ovvero invertibili solo a livello locale) i quali, instaurando una relazione più lasca, offrono minori garanzie, ma sono più semplici da creare e gestire.
La scelta di utilizzare un tipo di alias o l'altro dipende sostanzialmente dalle proprietà richieste dalla relazione di alias che si intende instaurare fra i due LRI e nessuno dei due casi è da considerarsi migliore dell'altro. Se da un lato infatti l'invertibilità globale è una proprietà auspicabile, dall'altro lato la possibilità di creare alias in modo trasparente per il responsabile del nome riferito è una semplificazione che può apportare notevoli vantaggi; inoltre quest'ultima soluzione non introduce problemi di scalabilità a differenza della prima che deve essere maggiormente monitorata e controllata.
La creazione di alias invertibili a livello globale potrebbe portare infatti alla nascita di punti di accumulazione critici, ovvero LRI relativamente ai quali vengono creati numerosi alias. Il limite di scalabilità deriva dal fatto che

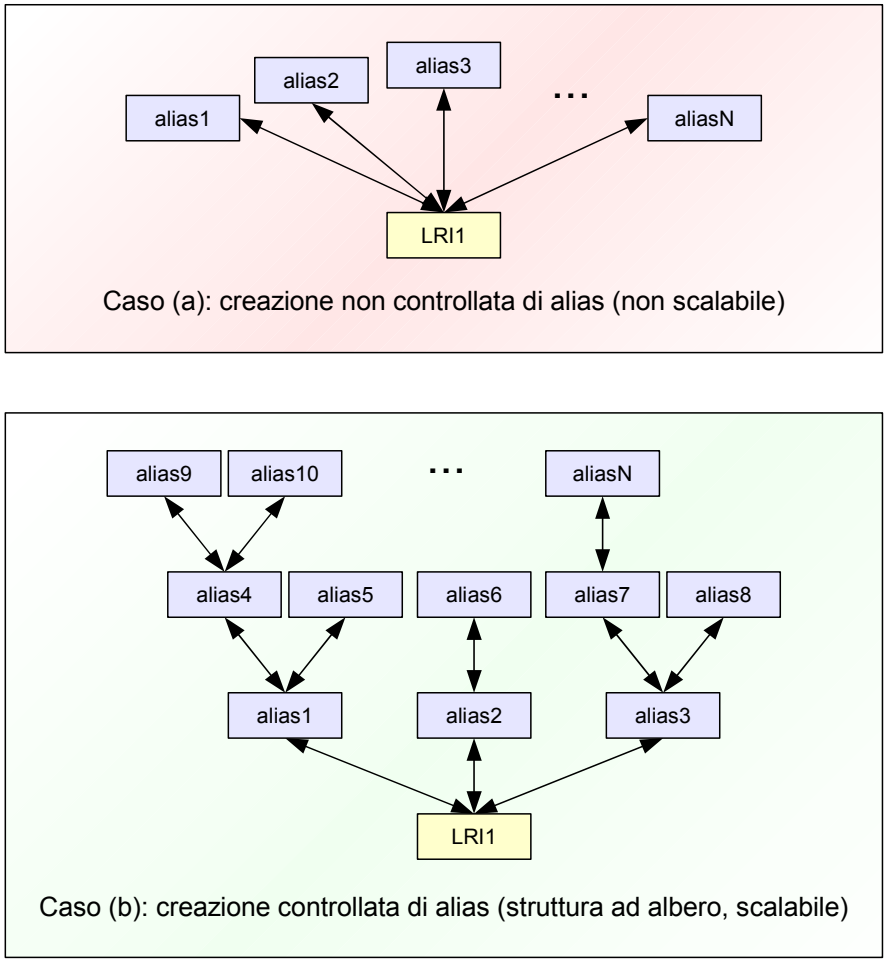

Figura 4.11: Creazione controllata e non di alias.

ogni alias invertibile deve essere noto al sistema autoritativo sull'elemento al quale fa riferimento e pertanto, da questo punto di vista, InterDataNet offrirebbe prestazioni paragonabili ad un comune sistema centralizzato.
Una soluzione scalabile a questo problema consiste nel creare gli alias in modo controllato, ovvero creare relazioni fra alias che rispettino la struttura ad albero (come nel caso (b) della figura 4.11), anziché creare tutti gli alias verso un unico LRI (come nel caso (a) in figura 4.11). La struttura ad albero offre infatti il vantaggio di permettere la distribuzione del carico di lavoro fra un numero arbitrario di host senza penalizzare la risoluzione diretta, nell'ipotesi che l'albero sia ben bilanciato e non degenerato su una lista.

Parte III

IDN Template

Capitolo 5

IDN Template Information Model

All'interno di ogni ambito della vita ed in particolar modo per quanto riguarda l'ingegneria, un medesimo problema può essere affrontato da soggetti diversi facendo ricorso a soluzioni completamente differenti.

Si pensi ad esempio, alla progettazione di un ponte, di un'automobile, di un componente hardware, di una libreria software, ecc.... Restando lo stesso il fine che due ingegneri intendono raggiungere (come ad esempio la realizzazione di un software per risolvere un determinato problema), le modalità mediante le quali l'obiettivo viene raggiunto possono differire sensibilmente e con molte variabili: ad esempio, nel caso dei programmi software può cambiare il linguaggio di programmazione utilizzato, i diagrammi UML ottenuti nella fase di ingegnerizzazione, fino alle librerie software utilizzate e così via.

Una situazione analoga a quella descritta si presenta anche all'interno dell'infrastruttura InterDataNet: risulta di fondamentale importanza il fatto che ciascun progettista di documenti IDN metta a disposizione della comunità una serie di informazioni volta a specificare, in una modalità che possa essere quanto più standard possibile, quali sono le scelte da lui effettuate durante la fase di progettazione e che hanno portato a strutturare i documenti in una certa modalità piuttosto che in altre. In particolare è necessario che il progettista dichiari quale è il *modello dei dati* che ha utilizzato, palesando quindi le interconnessioni esistenti tra i vari nodi di cui il documento IDN è costituito, nonché tutte quelle informazioni relative al contenuto di cui ogni nodo si fa carico (ovvero ai cosiddetti dati di livello applicativo) quali: il formato, la sintassi, la semantica.

All'interno del presente capitolo saranno dapprima analizzate le motivazioni delle suddette esigenze concentrandosi sugli aspetti fondamentali per la definizione di un modello dei dati per i documenti IDN, ovvero nell'ordine la sua struttura, quindi il formato e la sintassi del contenuto ed infine la relativa semantica. Successivamente verrà introdotta la soluzione proposta per un

information model [1] di un opportuno modello dei dati (il cosiddetto **IDN Template**); al fine di chiarificare tale presentazione sarà fatto ricorso ad un particolare esempio, ovvero la gestione all'interno di InterDataNet degli Open Data del Comune di Firenze relativi ai musei, della quale si parlerà più dettagliatamente all'interno del capitolo 8.
Infine, verrà tracciato un parallelo tra il dominio della programmazione orientata agli oggetti e quello dei documenti IDN, mettendo in risalto alcune analogie individuabili. Tale trattazione risulta di particolare importanza per la comprensione del capitolo 6, al cui interno sarà illustrato come è possibile effettuare il riuso di parti di un IDN Template all'interno di un altro. In tale ottica il modello costituisce quindi anche una forma di rappresentazione consultabile di documenti IDN-IM, in grado di aiutare un progettista a operare con profitto il riuso virtuoso dei dati.

5.1 La necessità di individuare regole per i dati

All'interno del capitolo 3 è stato illustrato il data model dei documenti IDN allo stato dell'arte: esso fornisce a qualunque progettista la possibilità di realizzare documenti che possono essere utilizzati da utenti distribuiti nel tempo e nello spazio per collaborare intorno ad elementi informativi.
Allo stato dell'arte non risulta immediata la possibilità di riuso di tali dati da parte di soggetti terzi: infatti un documento IDN di livello VR non dispone di un meccanismo che fornisca al suo progettista la possibilità di specificare informazioni quali il modo in cui sono stati organizzati i dati all'interno del dataset, con quale granularità, con quale rappresentazione di livello applicativo, quale sia la semantica dei dati.
All'interno del presente paragrafo saranno illustrati nel dettaglio quattro aspetti di particolare importanza, cercando di motivare la necessità di una loro descrizione da parte del progettista. Tali aspetti sono: la struttura di un documento IDN, il formato, la sintassi e la semantica dei dati di livello applicativo gestiti dai nodi.

5.1.1 Struttura di un documento IDN

Un primo aspetto che risulta di fondamentale importanza per ogni progettista nel momento in cui si approccia ai documenti IDN realizzati da soggetti terzi [2] è relativo alla conoscenza della loro struttura.

[1] Si ricordi che con il termine Information Model si intende una rappresentazione universale delle entità e delle loro proprietà, operazioni e relazioni, il cui scopo principale è quello di modellare gli oggetti ad un livello concettuale, indipendentemente da qualsiasi specifico repository, applicazione, protocollo o piattaforma. Esso non riguarda i dettagli, ma mira a catturare le astrazioni e i requisiti fondamentali delle entità da modellare. [PS03, Inn08]

[2] L'operazione di memorizzazione di documenti all'interno dell'infrastruttura InterDataNet può essere avvenuta facendo ricorso a varie modalità, come il loro inserimento

In particolare, risulta di primaria importanza che il progettista abbia la possibilità di conoscere se un determinato documento è stato realizzato in conformità ad una struttura ben determinata (ed ipoteticamente utilizzata da altri documenti) e, in caso positivo, quale essa sia; in tal modo egli conosce con certezza che il documento corrisponde ad una logica ben precisa, della quale può trovare traccia all'interno della struttura stessa.
Questa consente al programmatore di venire a conoscenza ad esempio del fatto che un determinato nodo possiede obbligatoriamente un link di aggregazione verso un altro nodo, oppure che possiede un numero indefinito di link di aggregazione verso nodi di una data tipologia; questo gli permette di riutilizzare al meglio tali dati all'interno delle proprie applicazioni, avendo una chiara percezione delle potenzialità messe a sua disposizione.

5.1.2 Formato dei dati

Ognuno dei nodi di cui è formato un documento IDN costituisce un'entità atomica, ovvero può essere considerato anche come un elemento a sé stante; pertanto può disporre di memorizzare il proprio contenuto (genericamente detto *dati di livello applicativo*) con il formato che il progettista ha ritenuto più idoneo alle finalità per le quali è utilizzato.
Si ricordi che per *formato dei dati* o, più genericamente, *formato di file* si intende la convenzione che viene usata al fine di leggere, scrivere o interpretare le informazioni ivi memorizzate.
Per identificare il formato di un file si possono utilizzare tre diverse modalità:

- tramite l'estensione, ovvero una serie di lettere (in genere tre) separata dal nome del file mediante un punto; possono quindi esistere file *.txt*, ovvero file di testo, file *.jpg*, ovvero immagini, e così via. Questa tecnica viene adottata dai sistema operativi DOS e Windows;

- tramite un "magic number", ovvero i primi due o più byte del file; ad esempio #! identifica gli script nei sistemi Unix e Unix-like, mentre 0xffd8 identifica le immagini in formato JPEG;

- tramite metadati espliciti; ad esempio il file system HFS (Macintosh) affianca ad ogni file una serie di informazioni dettagliate come il suo formato e il programma che l'ha creato. Un approccio simile viene utilizzato anche dai MIME Type (descritti nel seguito) usati per identificare il formato dei file trasferiti tramite internet.

La conoscenza del formato dei dati di livello applicativo risulta avere una primaria importanza ai fini di consentirne il riuso da parte di altri progettisti.

manuale da parte del progettista, sia mediante l'utilizzo di una particolare applicazione che consente ad utenti terzi di elaborare tali dati, sia attraverso l'utilizzo di una opportuna procedura batch realizzata *ad hoc*.

Infatti, l'infrastruttura InterDataNet, come già enunciato nel capitolo 3, non fornisce per scelta architetturale informazioni relative al contenuto dei dati di livello applicativo, visto che si limita semplicemente ad abilitare l'applicazione IDN che utilizza tali dati a disporre di un dispositivo virtuale (il cosiddetto Virtual Resource) al quale è delegato lo storage dei dati di interesse dell'applicazione.
L'architettura InterDataNet tratta infatti i dati di livello applicativo come un blob, delegando al singolo progettista la scelta del formato più opportuno, in base alle proprie esigenze ed alle scelte progettuali.
Pertanto egli potrebbe scegliere ad esempio, ma non necessariamente, di utilizzare uno tra i formati elencati nel seguente elenco:

- testo semplice;
- testo separato da virgole (CSV);
- pagine HTML;
- file XML;
- file JSON.

Come già accennato, al fine di riconoscere il formato dei file trasferiti all'interno di internet viene usato spesso il MIME (Multipurpose Internet Mail Extensions); tale standard, definito per la prima volta attraverso la RFC 1341 [BF92] [3], nacque per estendere SMTP (Simple Mail Transfer Protocol [Kle08], il protocollo utilizzato per inviare le email) al fine di risolvere alcuni limiti insiti in esso, permettendo tra le altre cose, l'invio di testo in codifiche diverse dall'ASCII, l'inserimento di allegati binari come immagini, suoni e filmati [KR03].
Nonostante questo, l'uso di tale standard è andato oltre il suo scopo iniziale ed oggi è generalmente utilizzato come modalità per descrivere il tipo di contenuto dei file distribuiti via web (e non solo).
Una delle caratteristiche chiave del MIME è il cosiddetto *MIME Type* o *Content-Type*, un meccanismo usato per specificare la natura dei dati del corpo di un'entità MIME, assegnando ai media un tipo (*type*) ed un sottotipo (*subtype*).
Il tipo viene utilizzato per dichiarare la categoria generica dei dati, mentre il sottotipo specifica il formato particolare per quel tipo di dati.
Il MIME è stato progettato per essere ampliabile, fornendo delle linee guida per la messa a punto di nuove estensioni; al fine di assicurare che l'insieme delle coppie tipo/sottotipo si sviluppi in modo ordinato, ben specificato

[3] Attualmente MIME è uno standard specificato attraverso le sei diverse RFC 2045 [FB96b], 2046 [FB96c], 2047 [Moo96], 4288 [FK05a], 4289 [FK05b] e 2049 [FB96a].

e pubblico, è istituito un processo di registrazione che usa l'Internet Assigned Numbers Authority (IANA), come registro centrale per le varie aree di ampliamento del MIME [IAN12].
Attualmente sono definiti sette tipi:

- ***application***: per i dati che devono essere elaborati da un'applicazione prima di poter essere utilizzati dall'utente. Ne fanno parte ad esempio: i files PDF (*application/pdf*), XML (*application/xml*), ZIP (*application/zip*), ecc...;
- ***audio***: utilizzato per i files audio, come MP3 (*audio/mpeg*), Ogg Vorbis (*audio/ogg*), ecc...;
- **image**: utilizzato per le immagini, come GIF *(image/gif*), JPEG (*image/jpeg*), PNG (*image/png*), ecc...;
- ***message***: consente ad un messaggio di posta elettronica di essere incapsulato all'interno di un altro, e risulta utile per l'inoltro di email;
- ***model***: utilizzato per i files modelli 3D;
- ***multipart***: utilizzato per indicare che il messaggio email si compone di più parti, le quali sono di tipo e sottotipo diversi;
- ***text***: utilizzato per indicare il testo leggibile o il codice sorgente. Uno dei sottotipi più frequenti è il *text/plain*, che indica un testo normale, senza comandi di formattazione o di stile, che deve essere mostrato a video così come è. Altri sottotipi sono utilizzati per i files CSS (*text/css*), CSV (*text/csv*), HTML (*text/html*), ecc...;
- ***video***: utilizzato per i files video, come MPEG (*video/mpeg*), QuickTime (*video/quicktime*), ecc...

È infine da evidenziare l'esistenza di un prefisso *vnd*, che può essere utilizzato per specificare sottotipi relativi a formati di files prodotti da uno specifico "vendor", quali i formati di Microsoft Excel (*application/vnd.ms-excel*), gli audio Wave (*audio/vnd.wave*), ecc...

5.1.3 Sintassi dei dati

Come evidenziato in precedenza, ognuno dei nodi di cui è formato un documento IDN, costituisce un'entità atomica, ovvero può essere considerato anche come un elemento a sé stante; pertanto può disporre di memorizzare il proprio contenuto (genericamente detto *dati di livello applicativo*) con la sintassi che il progettista ha ritenuto più idonea alle finalità per le quali è utilizzata.

La sintassi di un linguaggio formale specifica l'insieme di regole che i dati devono seguire per essere considerati conformi ad esso. Tali regole specificano come le sequenze di caratteri devono essere raggruppate a formare token e quali sono le sequenze permesse di tali token.
Tra i formati utilizzati allo stato dell'arte, si passa da alcuni, come i cosiddetti dati grezzi (o *raw*), quale il testo semplice, per i quali non è generalmente possibile individuare una sintassi, ad altri per quali è possibile individuare una sintassi "blanda" – come i file CSV, nei quali generalmente la prima riga è utilizzata per specificare i nomi delle colonne – ad altri ancora per i quali esistono sintassi ben determinate, come i cosiddetti linguaggi a markup (come HTML, XML, ecc. . .).
All'interno di quest'ultima famiglia rientra XML (acronimo di eXtensibile Markup Language), realizzato da un apposito workgroup del W3C [Wor97] con il tentativo di semplificare il preesistente SGML [4]. La caratteristica principale di XML (messa in evidenza dal suo nome) è quella di essere un linguaggio estensibile, in quanto permette di creare tag personalizzabili [5]; pertanto risulta più corretto affermare che XML è un metalinguaggio.
La definizione di un nuovo linguaggio (e della relativa sintassi) facendo ricorso all'XML può avvenire in due modi diversi:

- attraverso un Document Type Definition (o DTD), uno strumento derivato dall'SGML;
- attraverso un XML Schema; questo è l'unico linguaggio di descrizione di un XML validato dal W3C (la versione 1.0 in [FW04, TBMM04, BM04b], successivamente la versione 1.1 in [GSMT$^+$12, PGM$^+$12]) e che lo stesso consorzio consiglia di adottare al posto del DTD.

Entrambi gli strumenti citati rendono possibile definire la grammatica di un documento XML, ovvero gli elementi che sono leciti al suo interno, specificando la struttura di ognuno, ovvero cosa possono contenere, l'ordine, la quantità di elementi che possono comparire e se sono opzionali o obbligatori. Sia il DTD sia l'XML Schema possono essere utilizzati da un apposito programma, detto *parser*, per verificare la correttezza di un documento XML (operazione che prende il nome di *validazione*).
Confrontando le due tecnologie l'utilizzo di XML Schema presenta alcuni vantaggi rispetto all'utilizzo del DTD; in particolare:

- al contrario del DTD, gli XML Schema sono veri e propri documenti XML utilizzati per definire la grammatica di altri XML e memorizzati in file con estensione *.xsd* (XML Schema Definition). Diventa pertanto possibile riutilizzare gli strumenti a disposizione per tale linguaggio;

[4] Lo Standard Generalized Markup Language (SGML) è un metalinguaggio definito come standard ISO [ISO86] con lo scopo di definire linguaggi da utilizzare per la stesura di documenti in forma leggibile da computer (*machine readable*).
[5] Un linguaggio ottenuto da XML è detto *linguaggio obiettivo* o anche *applicazione*.

- un DTD non consente la definizione del tipo di dati, diversamente da un XML Schema. Risulta quindi impossibile avere un controllo accurato sul contenuto degli elementi di un documento XML o sul valore degli attributi;

- un documento XML può essere associato un solo DTD, mentre attraverso l'uso di XML Schema è possibile definire degli ambiti di validità di una grammatica (detti *namespace*), i quali consentono l'utilizzo di tag definiti in schemi diversi all'interno di un medesimo documento XML.

Il principale svantaggio dato dall'uso dell'XML Schema è invece la sua maggiore complessità rispetto al DTD.
Si consideri ad esempio il file XML presentato nel listato 5.1; si può affermare con facilità che esso è costituito da un insieme di tag diversi, organizzati in una struttura ad albero.

Listato 5.1: Esempio di documento XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<collezione>
    <libro>
        <autore>John Ronald Reuel Tolkien</autore>
        <titolo>The Lord of the Rings</titolo>
        <anno>1954</anno>
        <genere>Fantasy</genere>
    </libro>
    <libro>
        <autore>Clive Staples Lewis</autore>
        <titolo>The Chronicles of Narnia</titolo>
        <anno>1956</anno>
    </libro>
</collezione>
```

Ognuno di tali tag:

- deve trovarsi in una ben precisa posizione (ad esempio il tag <autore> deve necessariamente trovarsi all'interno del tag <libro>);

- può essere presente con molteplicità diverse (nell'esempio considerato, si può ipotizzare che il tag <titolo> sia necessariamente – e al più una sola volta – all'interno del tag <libro>, mentre è evidente che all'interno del tag <collezione> può trovare posto un numero indefinito di tag <libro>, così come è evidente che la presenza del tag <genere> all'interno del tag <libro> è opzionale);

- deve contenere un certo tipo di dati (nell'esempio considerato, il tag <anno> deve contenere un numero, mentre il tag <autore> deve contenere una stringa).

Tutte queste osservazioni possono essere opportunamente formalizzate in un insieme ben definito di regole all'interno di un XML Schema come quello presentato nel listato 5.2.

Listato 5.2: Esempio di XML Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="collezione">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="libro" maxOccurs="unbounded">
        <xs:complexType>
          <xs:all>
            <xs:element name="autore" type="xs:string"/>
            <xs:element name="titolo" type="xs:string"/>
            <xs:element name="anno" type="xs:integer"/>
            <xs:element name="genere" type="xs:string" minOccurs="0"/>
          <xs:all>
        <xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

5.1.4 Semantica dei dati

La conoscenza del formato, della sintassi e struttura dei dati fin qui descritta consente, come già detto in precedenza, ad un qualsiasi programmatore di capire quali sono le relazioni che intercorrono tra gli elementi che compongono i dati che egli sta considerando.
Tuttavia tali informazioni non sono sufficienti al fine di comprendere quale sia il *significato* che a tali dati è stato assegnato, ovvero non ne è chiara la semantica. Quest'ultima, così come descritto nel paragrafo 1.5 entra in gioco nel momento in cui si decide di elaborare i dati presenti attraverso l'uso di opportuni software al fine di individuare relazioni implicite intercorrenti tra diversi insiemi di dati.

5.2 Individuare un modello dei dati

Le esigenze dettagliate all'interno del paragrafo precedente rendono evidente come sia di massima importanza individuare una modalità per rappresentare le "regole" fondamentali di un documento IDN, ovvero:

- la sua struttura;
- il formato, la sintassi e la struttura del contenuto di ogni nodo;

- la semantica del contenuto di ogni nodo.

Si osservi che generalmente attraverso una qualsiasi applicazione saranno realizzati un numero indefinito di documenti IDN, i quali, pur essendo composti da nodi diversi tra loro (in particolare ciascun nodo si fa carico di dati diversi da quelli degli altri), sono tuttavia accomunati dallo stesso insieme di "regole". Risulta pertanto possibile individuare un "modello" (o *template*) al quale un insieme ben determinato di documenti IDN sia conforme, ovvero sotto un altro punto di vista indicare che le "regole" individuate sono valide per ciascun documento dell'insieme considerato.
Si definisce, pertanto, **IDN Template** un insieme di regole riguardanti, come già specificato:

- la struttura del documento;
- il formato e la sintassi del contenuto di ogni nodo;
- la semantica del contenuto di ogni nodo.

Ogni documento IDN che dichiara di essere conforme ad uno specifico IDN Template deve sottostare alle regole da esso specificate.
Al fine di rendere più chiaro il significato delle affermazioni che seguiranno, verrà fatto ricorso ad alcuni esempi tratti dal caso di studio relativo agli Open Data del Comune di Firenze, presentato al capitolo 8, al quale si rimanda per una descrizione dettagliata.
Si consideri quindi il documento IDN mostrato in figura 5.1, che rappresenta la sezione relativa ai musei dell'esempio citato; tale documento aggrega una serie di nodi, quali ad esempio *alinari* e *bargello*, accomunati dalla caratteristica evidente di rappresentare il nodo radice di uno specifico museo, caratterizzata dal proprio nome, come risulta anche dalla presenza di un nodo con il nome generico {*museo*}[6].
Sempre all'interno della medesima immagine è possibile osservare come ciascun nodo rappresentante un museo aggreghi i tre nodi *coordinate*, *id* e *descrizione*, e quest'ultimo a sua volta aggrega i nodi *nome*, *indirizzo*, *telefono* e *sitoWeb*.
Risulta quindi evidente come esistano delle caratteristiche che accomunano le varie parti del documento analizzato e che permettono ad esempio di ipotizzare che:

- il nodo *musei* aggrega un numero non meglio precisato di nodi {*museo*};
- tutti i nodi {*museo*} possiedono la medesima struttura, corrispondente a quella descritta poco sopra.

[6]Un nome generico racchiuso tra parentesi graffe viene indicato anche con il termine **URI template** ad indicare che non rappresenta il nome specifico di un nodo, quanto una *tipologia* di nome da assegnare ai nodi corrispondenti a tale classe [GFH+12].

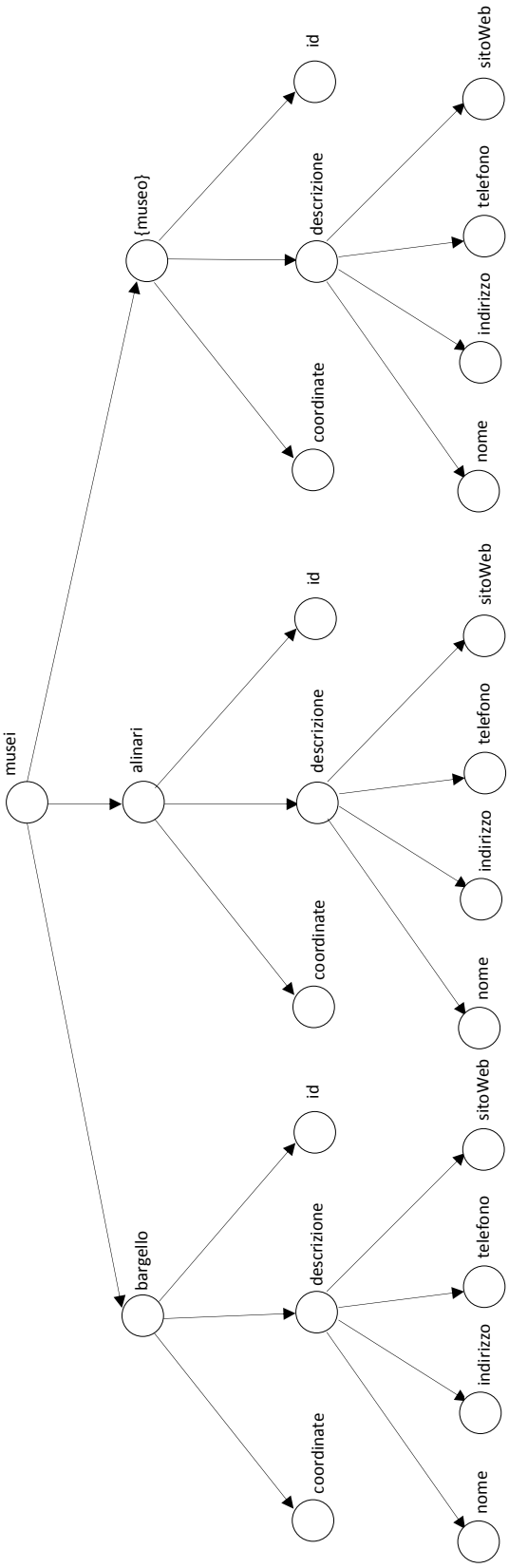

Figura 5.1: Vista parziale del documento IDN relativo agli Open Data del Comune di Firenze.

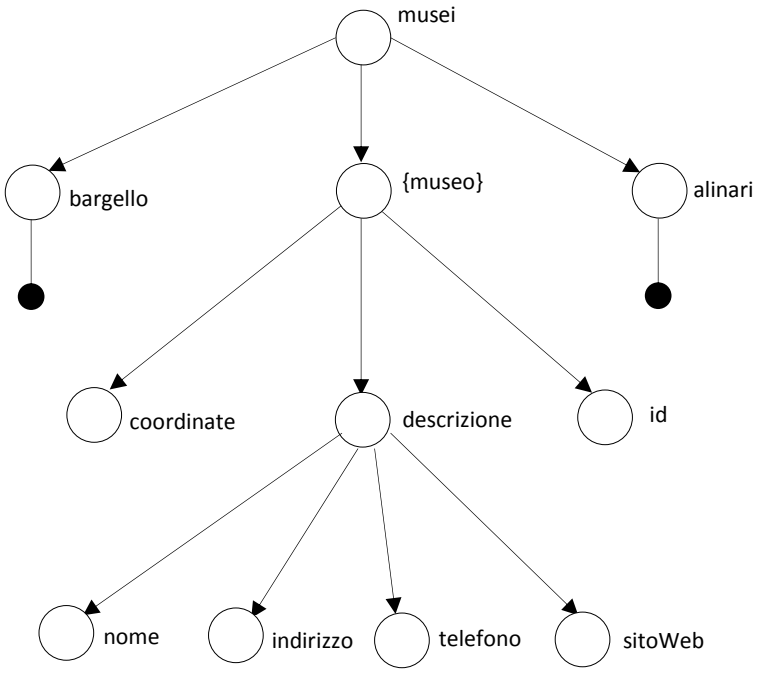

Figura 5.2: Vista parziale e compatta del documento IDN relativo agli Open Data del Comune di Firenze.

Pertanto il documento illustrato in figura 5.1 può essere equivalentemente sostituito da quello in figura 5.2, dove le strutture uscenti dai nodi *bargello* ed *alinari* rappresentano il fatto che tali nodi aggregano una struttura analoga a quella del nodo {*museo*}.

Si consideri adesso i nodi quali *alinari* e *bargello*; dato che, per quanto precedentemente affermato, essi possono essere compresi nella categoria rappresentata dal nodo {*museo*}, è possibile applicare ad essi lo stesso insieme di "regole", ovvero si può affermare che tali nodi corrispondono ad un medesimo modello.

Si definisce **nodo modello** un elemento informativo in grado di indicare una serie di regole cui devono sottostare uno o più nodi di un documento IDN. Il nodo modello risulta essere un singolo elemento informativo che in unione con altri, così come verrà illustrato nel paragrafo 5.2.1, costituisce un IDN Template.

In analogia alla rappresentazione grafica comunemente utilizzata di disegnare il singolo nodo di un documento IDN mediante un cerchio, è possibile rappresentare graficamente ogni nodo modello attraverso l'uso di un quadrato. La conformità di uno specifico nodo ad un determinato nodo modello sarà indicata attraverso una freccia tratteggiata dal primo (appartenente al dominio dei documenti IDN) al secondo (appartenente al dominio dei

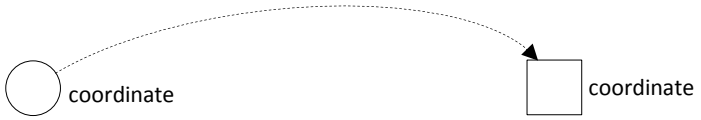

Figura 5.3: Corrispondenza tra il nodo *coordinate* ed il nodo modello *coordinate*.

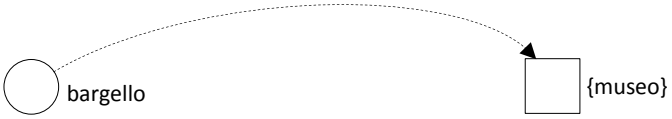

Figura 5.4: Corrispondenza tra il nodo *bargello* ed il nodo modello {*museo*}.

modelli IDN).
A fianco del nodo può essere presente una etichetta che ne contiene il nome; invece a fianco del nodo modello può essere presente:

- o l'etichetta che ne specifica il nome obbligatorio, come *coordinate*;
- o una etichetta racchiusa tra parentesi graffe che ne rappresenta una "tipologia" di nome, come {*museo*}.

In figura 5.3 il nodo *coordinate* ha una corrispondenza con un nodo modello *coordinate*, mentre in figura 5.4 il nodo *bargello* ha una corrispondenza con il nodo modello {*museo*}.
Si osservi che la relazione tra il mondo dei documenti e quello dei modelli non è iniettiva, ovvero infiniti nodi possono essere in relazione con lo stesso nodo modello. Tuttavia un nodo è in relazione con uno ed un solo nodo modello.
Individuato quindi il legame tra un nodo ed il suo modello, nel seguito viene analizzato come quest'ultimo è in grado di contenere informazioni utili ai fini di indicare:

- la struttura dell'IDN Template;
- il formato e la sintassi del contenuto dei nodi conformi;
- la semantica del contenuto dei nodi conformi.

5.2.1 Struttura dell'Information Model

All'interno dell'infrastruttura IDN, i link di aggregazione svolgono il ruolo fondamentale di esprimere le relazioni genitore-figlio esistenti tra i nodi di un documento e permettono di costruire delle strutture complesse a partire da risorse elementari [Cio10].

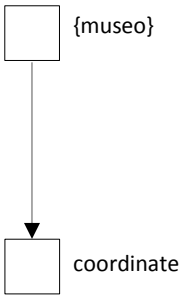

Figura 5.5: Rappresentazione di link modello.

Come emerge dall'esempio descritto precedentemente, esistono link di aggregazione che è plausibile (o sicuro) attendersi tra i diversi nodi che costituiscono un documento IDN; un qualunque nodo {*museo*} dell'esempio considerato possiede tre link di aggregazione verso i nodi *coordinate*, *id* e *descrizione* (si riveda la figura 5.2).

Ognuno dei quattro nodi appena citati possiede un proprio nodo modello di riferimento cui è conforme; dalla presenza dei link di aggregazione suddetti, si può inferire l'esistenza di opportuni collegamenti tra i nodi modello che svolgono il ruolo di paradigma per i link di aggregazione.

Si definisce **link modello** un elemento informativo in grado di svolgere il ruolo di paradigma per un link di aggregazione esistente tra due nodi di un documento IDN.

Il link modello viene rappresentato graficamente attraverso una apposita freccia uscente dal nodo modello genitore e diretto al nodo modello figlio, come in figura 5.5 dove viene evidenziata la sola relazione esistente tra il nodo modello {*museo*} ed il nodo modello *coordinate*.

Ritornando alla figura 5.2 si osservi che i link di aggregazione uscenti da un nodo {*museo*} sono in numero diverso rispetto a quelli uscenti dal nodo *musei*: infatti mentre un nodo {*museo*} aggrega un solo nodo di ciascun tipo (*descrizione*, *id* e *coordinate*), il nodo *musei* possiede un numero multiplo di link di aggregazione verso nodi {*museo*}.

Tale situazione viene espressa facendo ricorso al concetto di molteplicità di seguito illustrato.

Si definisce **molteplicità** di un link modello il numero minimo e massimo di link di aggregazione che possono esistere nel dominio dei documenti IDN tra un qualsiasi nodo conforme al nodo modello genitore e nodi conformi al nodo modello figlio.

I casi possibili di molteplicità sono indicati all'interno della tabella 5.1. Verrà utilizzata una notazione grafica mutuata dall'UML [Obj], indicando la molteplicità del link modello a fianco della relativa freccia.

Card	Descrizione	Notazione
0/1	Il link è opzionale, e al massimo può esservene uno solo	0,1
0/inf.	Il link è opzionale e non vi è un limite massimo al numero di link	0...*
1	Il link è obbligatorio, e al massimo può esservene uno solo	1
1/inf.	Il link è obbligatorio e non vi è un limite massimo al numero di link	1...*
n	Il link è obbligatorio e devono esservene n (ne' più ne' meno)	n
n/inf.	Il link è obbligatorio e devono esservene almeno n, senza limite massimo al numero di link	n...*

Tabella 5.1: Molteplicità dei link modello.

La figura 5.6 illustra il modo con cui è possibile rappresentare la struttura dell'IDN Template relativa al documento di esempio considerato.
Per chiarire meglio il concetto di molteplicità, si consideri l'esempio illustrato in figura 5.2 e lo si confronti con il relativo modello illustrato in figura 5.6. L'esistenza di un link modello tra il nodo modello {*museo*} ed il nodo modello *descrizione* con molteplicità 1, si traduce, nel dominio dei documenti IDN nella presenza di uno ed un solo link di aggregazione dal nodo *alinari* conforme al nodo modello {*museo*} verso il nodo *descrizione* conforme all'omonimo nodo modello. Invece, l'esistenza di un link modello tra il nodo modello *musei* ed il nodo modello {*museo*} con molteplicità 1/inf, può tradursi, nel dominio dei documenti IDN in un numero n qualsiasi di link dal nodo *musei* (conforme all'omonimo nodo modello) verso n nodi conformi al nodo modello {*museo*} (quali *alinari* e *bargello*).

5.2.2 Formato e sintassi dei dati di livello applicativo

Riferendosi sempre all'esempio considerato, è lecito attendersi che tutti i dati di livello applicativo presenti nel nodo *coordinate* dei vari musei esistenti siano tra di loro accomunati dal formato e dalla sintassi dei dati. Nell'esempio considerato, tali nodi si fanno carico di informazioni che sono memorizzate facendo ricorso al formato JSON, secondo la specifica sintassi GeoJSON, illustrata nel paragrafo 2.3.3. In maniera analoga è lecito aspettarsi che tutti i dati di livello applicativo dei nodi *id* relativi ai vari musei contengano informazioni che (tra loro) hanno il medesimo formato di dati e la medesima sintassi.
Tenuto conto di tale affermazione, ad ogni nodo modello viene assegnata la funzione di indicare il formato e la sintassi dei dati di livello applicativo presenti nei nodi che sono conformi ad esso.

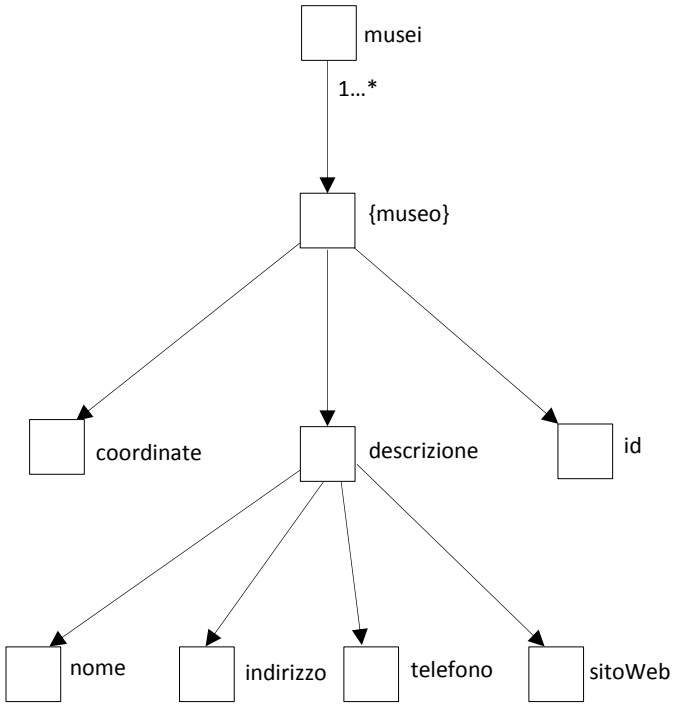

Figura 5.6: IDN Template degli Open Data del Comune di Firenze.

In base a quanto illustrato in precedenza al paragrafo 5.1.2 il MIME Type risulta essere il meccanismo che meglio si adegua ad indicare il formato dei dati. Quindi i contenuti rappresentati come testo semplice saranno indicati mediante il MIME Type *text/plain*, quelli espressi in formato XML attraverso il MIME Type *application/xml*, e così via...

Per quanto riguarda invece la sintassi del contenuto, il nodo modello non specifica l'utilizzo di una determinata tecnologia al posto di un'altra, ma fornisce la possibilità di inserire un link verso una qualsiasi risorsa, all'interno della quale il progettista (o chi per lui) dovrà provvedere ad inserire la specifica della sintassi dei dati. Attraverso il ricorso a tale soluzione ogni progettista viene lasciato libero di scegliere la tecnologia da lui ritenuta più idonea; allo stesso tempo l'architettura si apre alla compatibilità in avanti, non mettendo limiti al possibile utilizzo, in futuro, di nuove tecnologie ad oggi inesistenti.

Come già illustrato nel paragrafo 5.1.3 esistono dati per i quali non è possibile individuare regole sintattiche ed altri per i quali, invece, ciò può essere realizzato, come ad esempio i files XML; in quest'ultimo caso il progettista può ad esempio inserire all'interno del nodo modello un link verso una risorsa XML Schema dove viene memorizzata la sintassi dei dati e i vincoli strutturali da lui ritenuti opportuni.

5.2.3 Semantica dei dati di livello applicativo

Analizzando nuovamente l'esempio considerato risulta abbastanza plausibile attendersi che le informazioni presenti all'interno del nodo *coordinate* siano diverse da quelle presenti nel nodo *id*. In particolare, per quanto riguarda il primo caso, ci si attende che i dati di livello applicativo di cui esso si fa carico corrispondano a delle informazioni relative alle coordinate geografiche, mentre quelli relativi al secondo corrispondano in un certo qual modo ad una stringa che ne rappresenti l'identificativo.
Tenuto conto di tale affermazione, ad ogni nodo modello viene assegnata la funzione di indicare la semantica dei dati di livello applicativo presenti nei nodi che sono conformi ad esso.
Anche in questo caso, come in quello della sintassi dei dati, viene fornita la possibilità di inserire un link verso una qualsiasi risorsa all'interno della quale il progettista (o chi per lui) dovrà provvedere ad inserire la specifica della semantica dei dati. Come già evidenziato nel caso della sintassi, attraverso questa soluzione ogni progettista viene lasciato libero di scegliere la tecnologia da lui ritenuta più idonea; allo stesso tempo l'architettura si apre alla compatibilità in avanti, non mettendo limiti al possibile utilizzo, in futuro, di nuove tecnologie ad oggi inesistenti.
Allo stato dell'arte una soluzione opportuna potrebbe essere, ad esempio, quella di inserire un link verso una risorsa all'interno della quale viene specificata una ontologia OWL.
Si noti che questa soluzione si presta benissimo a specificare la semantica dei dati di qualsiasi formato, come ad esempio testi semplici, immagini, audio, ecc... Tuttavia, per quanto riguarda dati con una sintassi strutturata, come XML, per quanto descritto al paragrafo 1.6, è possibile individuare tecniche per l'annotazione semantica dei vari tag: questo potrebbe essere, ad esempio, effettuato arricchendo opportunamente gli XML Schema utilizzati per definire la sintassi con le annotazioni specificate dalla tecnologia SAWSDL, descritta nel paragrafo 1.6.1. In tal modo il progettista è in grado di specificare la semantica dei dati in modo ancora più dettagliato.

5.3 Analogie con la programmazione ad oggetti

Nel paragrafo precedente è stata illustrata la modalità attraverso la quale è possibile definire un modello che rappresenti le caratteristiche base di un documento IDN, ovvero che consenta di stabilire informazioni riguardo alla sua struttura, al contenuto dei singoli nodi e alla relativa semantica.
Analizzando attentamente quanto ivi descritto risulterà forse evidente ad un lettore attento la possibilità di tendere un parallelo tra l'IDN Template ed il mondo della programmazione orientata agli oggetti (o OOP, Object Oriented Programming).

Quest'ultimo è un paradigma di programmazione che permette di definire degli oggetti software in grado di interagire gli uni con gli altri. All'interno di un'opportuna area di codice sorgente, detta *classe*, sono dichiarate la struttura dati e le procedure che operano su di essa. Pertanto le classi realizzano un modello astratto, che diviene "reale" al tempo di esecuzione mediante *oggetti* software; in generale si dice che un oggetto è *istanza* di una classe.
Ritornando al parallelo, i nodi modello possono essere considerati come degli analoghi alle classi, mentre i nodi che compongono un documento IDN sono comparabili agli oggetti, ovvero rappresentano gli oggetti che possono essere ottenuti "istanziando" i nodi modello.
Dato che le relazioni fornite dai link di aggregazione, in ottemperanza al principio contenitore-contenuto di InterDataNet descritto nel capitolo II, indicano un legame tra contenuto e contenitore, è possibile ritenerle come analoghe all'aggregazione di oggetti presente all'interno della OOP.
Si consideri ad esempio l'IDN Template definito al paragrafo precedente per rappresentare gli Open Data del Comune di Firenze, presentato in figura 5.6 e le due viste particolari del relativo documento IDN, rappresentato nelle figure 5.1 e 5.2.
All'interno della figura 5.7 viene rappresentato il diagramma UML delle classi [7] relativo ad una soluzione in programmazione orientata agli oggetti per l'accesso agli Open Data del Comune di Firenze.
Il parallelo sin qui tracciato, utile al fine di una comprensione più approfondita dell'IDN Template, sarà ampliato all'interno del capitolo 6 per illustrare il meccanismo di riuso del modello.

[7] L'UML (Unified Modeling Language) [Obj] è un linguaggio di modellazione, basato sul paradigma ad oggetti, definito nel 1996 [JBR96], con l'intento di svolgere la funzione di "lingua franca" per la progettazione e la programmazione ad oggetti attraverso un'insieme di diagrammi diversi. In particolare, il *diagramma delle classi* consente di esprimere le entità considerate (le classi) e le relazioni esistenti tra esse.

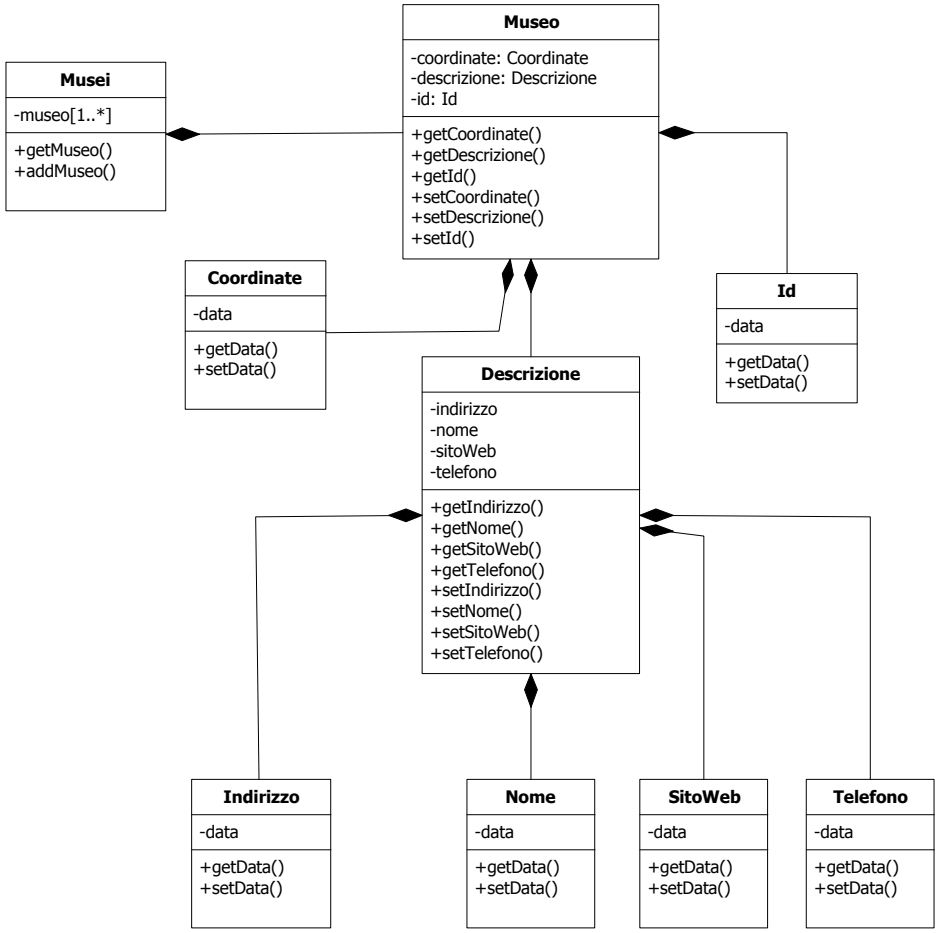

Figura 5.7: Diagramma delle classi relativo agli Open Data del Comune di Firenze.

Capitolo 6

Riuso di IDN Template

All'interno del capitolo precedente è stato illustrato IDN Template, il meccanismo attraverso il quale è possibile definire opportuni modelli per i documenti IDN.
Tenendo conto del parallelo tracciato all'interno del paragrafo 5.3 tra il dominio della programmazione orientata agli oggetti e quello degli IDN Template, in questo capitolo verrà illustrato come sia possibile estendere l'IDN Template abilitando in tal modo il riuso dei dati.
A tal scopo, verrà dapprima illustrato un esempio di riuso del codice, quindi la trattazione sarà riportata sul riuso dei dati e sui vantaggi ad esso connessi; quindi sarà descritto il meccanismo di riuso dell'IDN Template, concentrandosi in particolare sull'esempio dei musei estratti dagli Open Data del Comune di Firenze, illustrato dal diagramma UML presentato in figura 5.7 (e che per comodità è riportato in figura 6.1).
Infine la trattazione verterà sulla modalità mediante la quale è possibile effettuare un riuso multiplo dei nodi all'interno dell'IDN Template.

6.1 Il riuso del codice

Il fenomeno del riuso, ovvero di fare nuovamente uso di qualcosa di già adoperato, per lo stesso scopo o per uno diverso, è antico quanto l'uomo e riguarda tutti i campi dell'umano agire: dalla cucina all'architettura fino alla tecnologia dell'informazione.
In particolare in informatica, il riuso del codice consiste nella prassi di utilizzare parti di codice precedentemente già scritte ogniqualvolta ciò risulti necessario, evitando il costo dovuto alla riscrittura dello stesso. Nel caso in cui si consideri un particolare programma, tale azione può essere portata a buon termine attraverso la scrittura di funzioni che possono essere invocate all'interno dello stesso programma; nel caso in cui si considerino programmi diversi, questo può avvenire attraverso la creazione di opportune

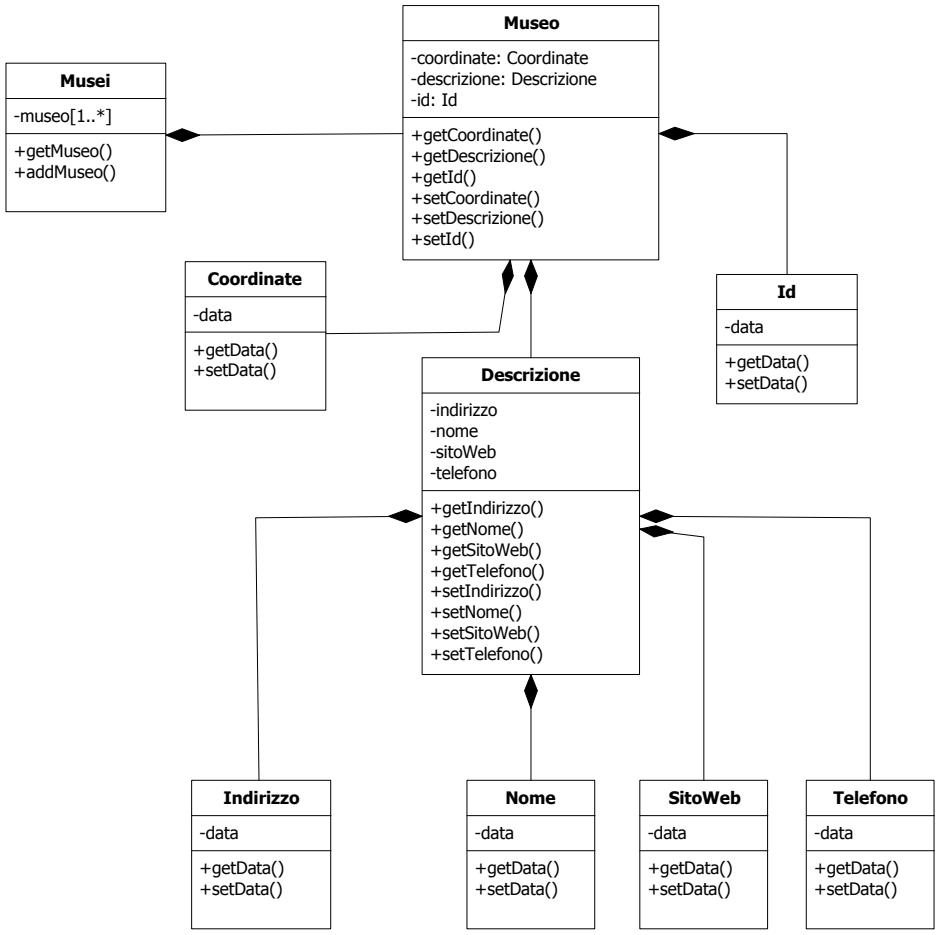

Figura 6.1: Diagramma delle classi relativo agli Open Data del Comune di Firenze.

librerie che possono essere invocate all'interno dei programmi oppure delle cosiddette API (Application Programming Interface).

Nel seguito del presente paragrafo, sarà presentato un caso di esempio che servirà a rendere più chiara la modalità attraverso la quale l'IDN Template abilita il riuso dei dati.

Si consideri adesso il caso di un programmatore che sia intenzionato a creare un'applicazione all'interno della quale sia riutilizzato il codice derivato al diagramma UML illustrato in figura 6.1; questo è lo stesso diagramma che era stato ottenuto nell'esempio descritto nel paragrafo 5.3.

Si supponga, ad esempio, che egli non abbia a disposizione il sorgente di tale codice, ma che sia riuscito ad ottenerne in qualche modo — fornitogli direttamente dal programmatore originale, scaricato dal sito web dello stesso, ecc. . . – una versione eseguibile, ma non modificabile; ad esempio nel caso in

cui fosse stato utilizzato Java come linguaggio di programmazione, questo potrebbe essere un pacchetto JAR, dal quale è possibile ottenere la *signature* dei vari metodi, ma al quale non è possibile effettuare modifiche. Oppure, si ipotizzi ancora, che sia riuscito ad ottenere, sempre attraverso una delle modalità precedentemente descritte, il codice sorgente relativo all'esempio considerato, ma tuttavia non voglia per esigenze pratiche provvedere ad apportarvi modifiche (ad esempio perché ritiene che questa sia un'operazione eccessivamente onerosa).
Pertanto, egli vuole utilizzare tale codice come una scatola nera [1]; allo stesso tempo gradirebbe che esso avesse una struttura lievemente diversa, ad esempio che il museo godesse della caratteristica di possedere un direttore, del quale specificare il nome. Inoltre, poiché ritiene che l'informazione *id* non sia significativa per le finalità di suo interesse, desidererebbe che essa non fosse presente.
Di fronte a tale esigenza, il programmatore dispone di due possibilità:

- riscrivere completamente ex novo il programma (ma questo contrasta con le esigenze del riuso);
- adottare un'opportuna strategia di programmazione che gli consenta di adattare il codice.

Nel seguito sarà illustrata una modalità utile al fine di perseguire quest'ultima strada.

6.1.1 Il design pattern Adapter

All'interno della letteratura scientifica è stata ormai illustrata da anni una soluzione al problema posto poco sopra; in particolare essa consiste nell'utilizzo del design pattern Adapter.
Nell'Ingegneria del Software, il termine design pattern indica una soluzione progettuale generale ad un problema ricorrente: non si tratta di una libreria o di un componente software riutilizzabile, ma di un modello da applicare per la soluzione di un problema che può presentarsi in svariate situazioni durante la progettazione e lo sviluppo del software.
Il termine design pattern fu introdotto inizialmente nell'architettura [AIS+77] e fu usato più tardi nell'Ingegneria del Software grazie al lavoro di Gamma et al. [GHJV95], generalmente noti come la "Banda dei quattro" (*Gang of four*). In tale libro gli autori identificarono 23 design pattern, tra i quali è annoverato anche l'Adapter.
Lo scopo di tale design pattern è quello di convertire l'interfaccia di una classe in un'altra interfaccia richiesta dal client, consentendo a classi diverse

[1] Nella Teoria dei Sistemi il termine *scatola nera* (o *blackbox*) indica un sistema del quale non è noto a priori ne' ciò che contiene, ne' come si comporta, ma che è descrivibile solo per la reazione (output) ad una determinata sollecitazione (input).

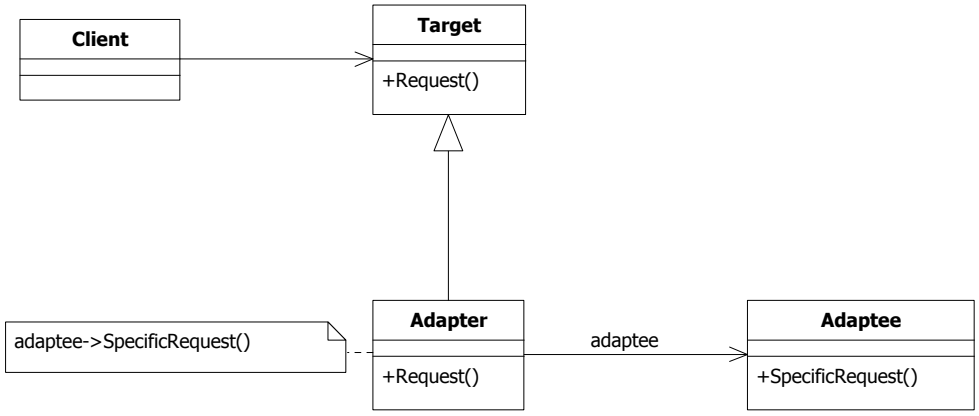

Figura 6.2: Struttura del design pattern Adapter.

di operare insieme quando ciò non sarebbe altrimenti possibile a causa di interfacce incompatibili.
Capita spesso infatti che una classe di supporto non possa essere riusata all'interno di una applicazione semplicemente perché la sua interfaccia non è compatibile con l'interfaccia specifica richiesta da un'applicazione stessa.
Il pattern, la cui struttura è illustrata nella figura 6.2, si compone dei seguenti quattro partecipanti:

- **Target**: definisce l'interfaccia specifica del dominio utilizzata dal client;
- **Client**: collabora con oggetti compatibili con l'interfaccia *Target*;
- **Adaptee**: individua un'interfaccia esistente che deve essere adattata;
- **Adapter**: adatta l'interfaccia di Adaptee all'interfaccia di *Target*.

L'utilizzo del pattern Adapter richiede di prendere in considerazione il fatto che la complessità degli adattatori varia in base alla quantità di lavoro che deve essere svolta per adattare *Adaptee* all'interfaccia *Target*. Esiste uno spettro di possibilità che va dalla semplice conversione di interfaccia, per esempio il cambio dei nomi delle operazioni, fino a fornire un insieme completamente diverso di operazioni. Pertanto la quantità di lavoro svolto da un *Adapter* è strettamente legata alle similarità esistenti tra l'interfaccia *Target* e quella di *Adaptee*.
Ritornando al problema di esempio precedentemente preso in considerazione, il programmatore può decidere di applicare il pattern Adapter e a partire dall'originale diagramma delle classi, presentato in figura 6.1, ottenere il diagramma delle classi riportato in figura 6.3.

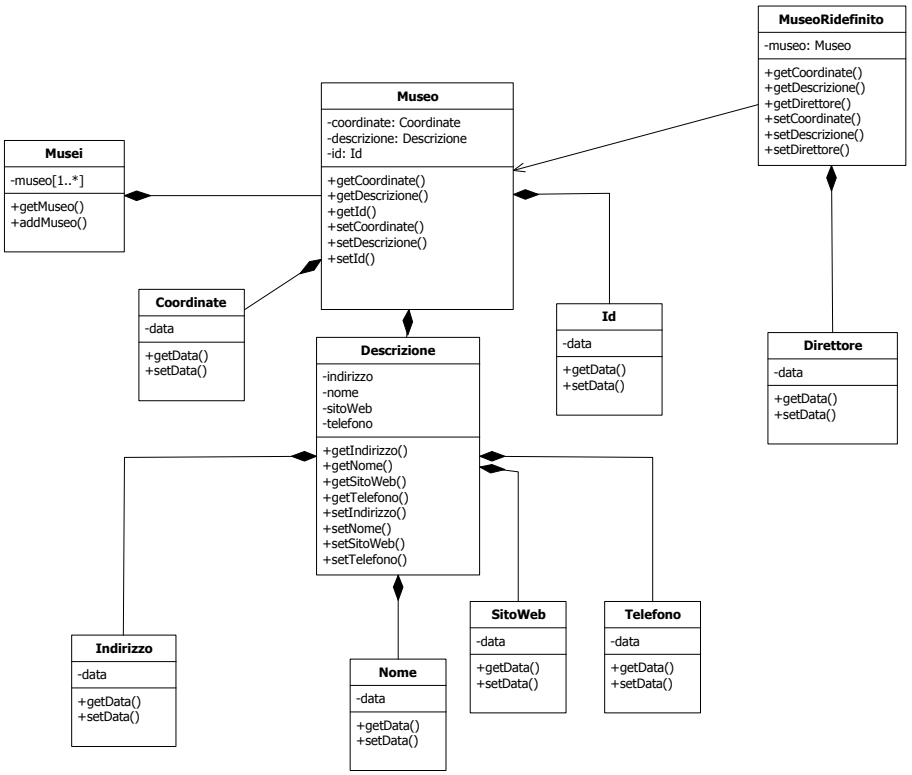

Figura 6.3: Diagramma delle classi relativo al riuso degli Open Data del Comune di Firenze.

6.2 Il riuso dei dati

La pubblicazione da parte delle Pubbliche Amministrazioni degli Open Data, che sono oggetto del capitolo 2, ha senza dubbio un impatto interessante sulla società, che va da aspetti che riguardano la trasparenza, la partecipazione, il controllo democratico e la valutazione di impatto delle politiche pubbliche di spesa fino al loro riuso in nuovi servizi ed software, creando nuova conoscenza derivante dalla combinazione di diverse fonti di dati. In particolare, il presente lavoro mira a focalizzarsi su questo ultimo aspetto.
Tra le categorie di dati che si prestano ad un riuso efficace possono essere compresi:

- bilanci delle pubbliche amministrazioni;
- dati ambientali;
- dati sanitari;
- dati sui trasporti pubblici;

- dati catastali;
- dati su attività economiche ed imprese;
- dati su sicurezza e criminalità;
- dati territoriali su attività e beni culturali.

L'informazione pubblica risulta inoltre avere un valore economico, ed in particolare si può considerare il dato come la materia prima che può essere utilizzata per nuove attività. Lo studio condotto nel 2006 dal MEPSIR (Measuring European Public Sector Information Resources) su incarico della Commissione Europea stimava un valore di mercato medio per gli Open Data europei pari a 27 miliardi di euro, ovvero lo 0.25% del PIL dell'intera Unione Europea [MEP06]. Uno studio del 2011 ritiene invece che l'impronta economica dei dati sia maggiore e stima che i guadagni derivanti dall'apertura dei dati pubblici possano essere 40 miliardi di euro annui [Gra11]; sempre il medesimo studio sostiene che, tenendo conto del fatto che gli Open Data possono essere utilizzati anche indirettamente in altre applicazioni avendo effetti economici diretti ed indiretti, l'impatto dell'uso dei dati pubblici sull'economia europea potrebbe raggiungere i 140 miliardi di euro annui.
Risulta pertanto di grande interesse abilitare meccanismi che consentano un riuso efficace ed efficiente dei dati, mirando alla creazione di un "circolo virtuoso" degli Open Data, in cui:

- le Pubbliche Amministrazioni generano i dati e le relative basi informative;
- gli sviluppatori e le imprese riutilizzano tali dati, ad esempio mediante la realizzazione di applicazioni web o mobile;
- i cittadini, le imprese e le Pubbliche Amministrazioni utilizzano i servizi a valore aggiunto prodotti da sviluppatori ed altre imprese.

In tal modo risultano evidenti le affermazioni del Commissario per l'agenda digitale della Commissione Europea, Neelie Kroes, secondo cui "gli imprenditori del web assemblano e vendono contenuti, applicazioni e pubblicità basati su dati. Con questi sforzi essi rendono la nostra vita più comoda e mantengono le autorità responsabili" [Eur11].
Tuttavia occorre notare come attualmente esistano vari ostacoli per un riuso efficace ed efficiente degli Open Data; prima di tutto occorre considerare gli aspetti giuridici, ovvero il fatto che non sempre i dati sono pubblicati mediante una licenza che ne permetta il riuso; inoltre vi sono aspetti politici e amministrativi, ovvero vi è una scarsa consapevolezza e conoscenza di quali siano i costi ed i benefici organizzativi dovuti agli Open Data. Infine vi sono ostacoli tecnologici, come ad esempio la necessità di una ontologia

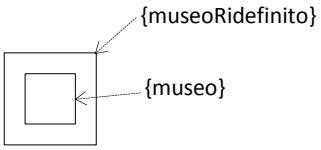

Figura 6.4: Rappresentazione grafica del riuso di un nodo modello.

semantica comune e di un meccanismo che faciliti l'interscambio e l'interoperabilità; il presente lavoro si muove nella direzione di risolvere (o quanto meno semplificare) tale problema.

6.3 Il riuso dell'IDN Template

Si consideri adesso il caso illustrato al paragrafo 6.1 riguardo all'utilizzo del pattern Adapter da parte di un programmatore; tenendo conto anche del parallelo che è possibile tracciare tra gli IDN Template e la programmazione orientata agli oggetti, così come descritto all'interno del paragrafo 5.3, è possibile introdurre un analogo del pattern Adapter, all'interno dell'IDN Template, al fine di abilitare il riuso dei dati.
Si ipotizzi pertanto un progettista che voglia creare un proprio nodo modello, di nome {*museoRidefinito*}, il quale effettua il riuso del nodo modello {*museo*} definito da un altro progettista.
La rappresentazione grafica riportata in figura 6.4 illustra la modalità attraverso la quale è possibile indicare ciò. Si ipotizzi adesso che tale progettista abbia una volontà analoga a quella illustrata nell'esempio descritto al paragrafo 6.1 relativo al dominio della programmazione orientata agli oggetti, ovvero che voglia:

- inserire il link modello verso un nodo modello *direttore*, da lui definito;
- eliminare il link modello verso il nodo *id*.

Per quanto riguarda il primo punto, ovvero l'inserimento di un link modello verso un nodo modello *direttore*, da lui definito, la figura 6.5 illustra come sia possibile rappresentare graficamente tale operazione. Il significato della rappresentazione grafica è il seguente: esiste un nodo, {*museoRidefinito*}, il quale effettua il riuso del nodo {*museo*}; i link modello di quest'ultimo, verso un insieme di nodi considerato come un "carico" non specificato, sono rappresentati da una freccia uscente dal margine del nodo modello {*museo*} e che attraversano il bordo del nodo modello {*museoRidefinito*}. Il link modello aggiunto dal progettista verso il nodo modello *direttore* è rappresentato da una freccia uscente dal margine del nodo modello {*museoRidefinito*}. Questo artificio grafico permette di rappresentare in maniera sufficientemente sem-

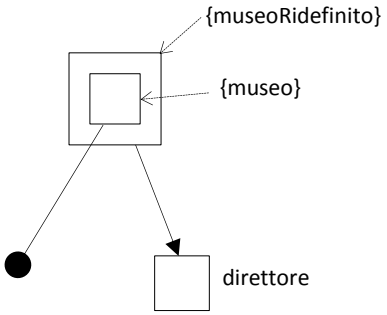

Figura 6.5: Aggiunta di un link modello ad un nodo modello riusato.

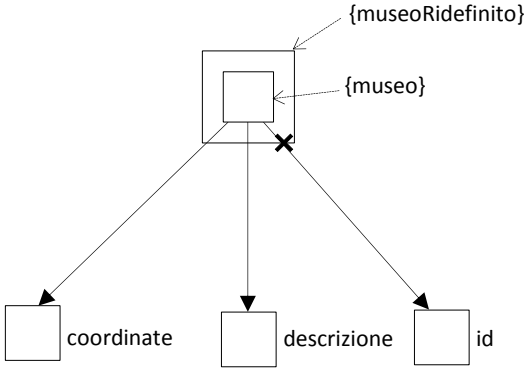

Figura 6.6: Rimozione di un link modello da un nodo modello riusato.

plice le aggiunte di link modello effettuate da un secondo programmatore su nodi definiti da altri.

Per quanto concerne invece il secondo dei punti definiti sopra, ovvero l'eliminazione del link modello verso il nodo modello *id*, la figura 6.6 illustra come sia possibile rappresentare graficamente tale operazione. Il significato della rappresentazione grafica è il seguente: esiste un nodo modello {*museoRidefinito*}, il quale effettua il riuso del nodo {*museo*}. Tra i link modello di quest'ultimo, due (quelli verso i nodi modello *coordinate* e *descrizione*) partono dal margine di tale nodo e attraversano il bordo del nodo {*museoRidefinito*}; il link verso il nodo *id* parte anch'esso dal margine del nodo {*museo*}, ma giunto al margine del nodo {*museoRidefinito*} incontra il segno X; in tal modo si vuole indicare che, mentre il nodo {*museo*} presenta i link verso i nodi *coordinate*, *descrizione* e *id*, il nodo {*museoRidefinito*}, che effettua il riuso del nodo {*museo*}, mantiene solamente i due link verso i nodi *coordinate* e *descrizione*, mentre non considera il link modello verso il nodo *id*.

In figura 6.7 è illustrata la rappresentazione grafica utilizzabile nel caso in cui siano effettuate entrambe le operazioni precedentemente considerate,

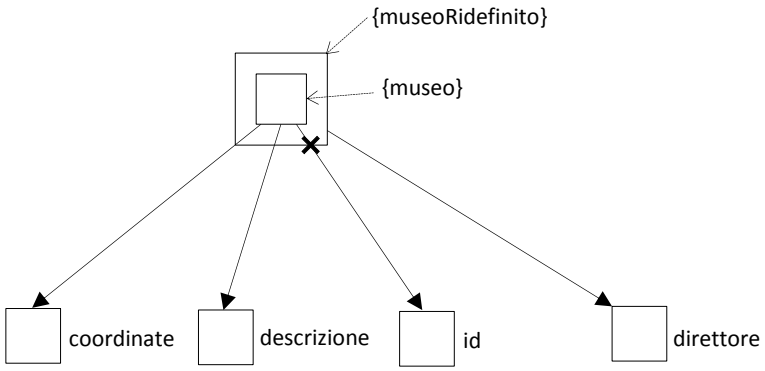

Figura 6.7: Riuso di un nodo modello con aggiunta ed eliminazione di link modello.

ovvero:

- inserire il link modello verso un nodo modello *direttore*, definito dal programmatore;
- eliminare il link modello verso il nodo modello *id*.

Infine, occorre considerare che il nodo {*museo*} ed il nodo {*museoRidefinito*} sono di competenza di due diverse web authorities, ovvero:

- il nodo modello {*museo*} ed i relativi aggregati *coordinate*, *descrizione* e *id* sono di competenza del progettista originale;
- i nodo modello {*museoRidefinito*} e *direttore* sono di competenza del progettista considerato nel presente esempio.

Un artificio grafico utilizzabile per risolvere questa ambiguità è quello di disegnare i vari quadrati che rappresentano i nodi modello utilizzando tratteggi di sfondo diversi in base alla web authority competente; ad esempio in figura 6.8 i nodi modello di competenza del progettista originale sono rappresentati come quadrati privi di tratteggio, mentre quelli di competenza del progettista considerato sono rappresentati con un quadrato dall'interno tratteggiato.
La figura 6.9 illustra un documento IDN la cui struttura risulta conforme [2] a quella del modello presentato in figura 6.8

[2]Il confronto tra modello e documento si limita in questo caso alla sola struttura. Per affermare infatti che il documento è conforme al modello considerato, occorre tenere in considerazione anche altre caratteristiche, quali il formato, la struttura e la semantica dei dati, come illustrato all'interno del capitolo 5.

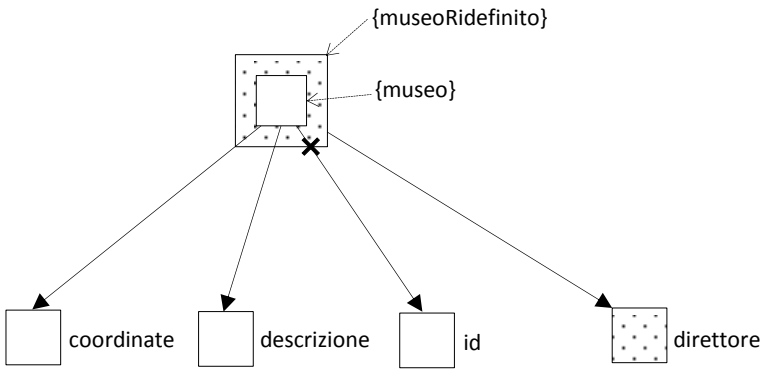

Figura 6.8: Riuso di un nodo modello con aggiunta ed eliminazione di link modello, e specifica del server autoritativo.

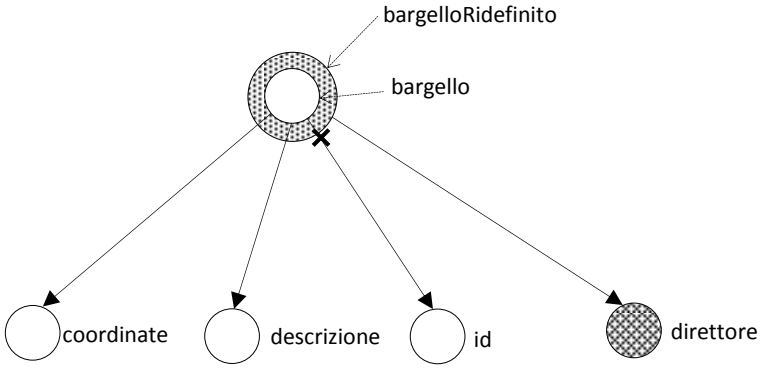

Figura 6.9: Esempio di documento IDN conforme al Template illustrato in figura 6.8.

6.4 Basi teoriche di riferimento

All'interno del presente paragrafo viene riepilogato ai fini della trattazione teorica cosa sia il meccanismo di riuso dei nodi modello e come sia possibile rappresentarlo.

Con il termine **meccanismo di riuso dei nodi modello** (o più brevemente **riuso del modello**) si intende la possibilità di cui gode un progettista di definire un proprio IDN Template facendo uso di parti di modello definite da altri. In particolare, attraverso il riuso del modello, un progettista può creare un nodo modello che fa uso di un altro nodo modello precedentemente esistente [3]. Il primo viene definito **nodo modello riutilizzante**, mentre il

[3] All'interno del paragrafo 6.5 si vedrà che è possibile avere anche più di un nodo modello riutilizzato.

secondo **nodo modello riutilizzato**. Per quanto riguarda quest'ultimo, il progettista può:

- aggiungere link modello verso nodi modello definiti da lui o da terzi. L'aggiunta di tali link viene rappresentata graficamente attraverso una freccia uscente dal bordo del nodo modello riutilizzante (e che pertanto non coinvolge il nodo modello riutilizzato);
- eliminare uno o più link di aggregazione. Essi vengono rappresentati graficamente con una freccia uscente dal bordo del nodo modello riutilizzato e che incontra una X al bordo del nodo modello riutilizzante;
- mantenere (ed utilizzare come propri) alcuni (o tutti i) link di aggregazione. Questi sono rappresentati graficamente come una freccia uscente dal bordo del nodo modello riutilizzato e che attraversa il bordo del nodo modello riutilizzante.

La figura 6.10 illustra tutte le possibili casistiche sin qui descritte.
Dato che, in genere, il riuso del modello ha luogo tra web authorities distinte [4] è buona prassi rappresentare i quadrati dei nodi modelli dei vari domini in gioco con diversi tratteggi di sfondo.
Similmente a quanto avviene con il riuso del codice, illustrato nel paragrafo 6.1, anche per quanto riguarda la pratica del meccanismo di riuso di un IDN Template esiste uno spettro di possibilità che va dalle operazioni semplici, come l'aggiunta o la rimozione di un link modello, sino ad una modifica profonda della struttura. Pertanto la convenienza dell'applicazione del meccanismo di riuso è strettamente legata alle similarità esistenti tra il nodo modello riutilizzato ed il riutilizzante. Qualora dovesse esservi una scarsa similarità il progettista potrebbe trovare più conveniente evitare il meccanismo di riuso e fare ricorso ad altre tecniche, che possono andare da un utilizzo opportuno dei link modello sino alla completa ridefinizione dei dati.

6.5 Riuso multiplo di nodi modello

All'interno dei paragrafi precedenti è stato illustrato come un progettista possa realizzare degli IDN Template che siano parzialmente composti da altri modelli, attraverso il meccanismo di riuso. In particolare, negli esempi considerati, veniva effettuato il riuso di un singolo nodo di un IDN Template; tuttavia tale meccanismo si presta benissimo anche ad effettuare il riuso di più di un nodo di un modello.

[4] Tale affermazione implica il fatto che un progettista IDN riutilizzerà nodi modello definiti da altri e sui quali ha solamente permessi di lettura, ma non di creazione, modifica o eliminazione, poiché in pratica di competenza di una diversa web authority.

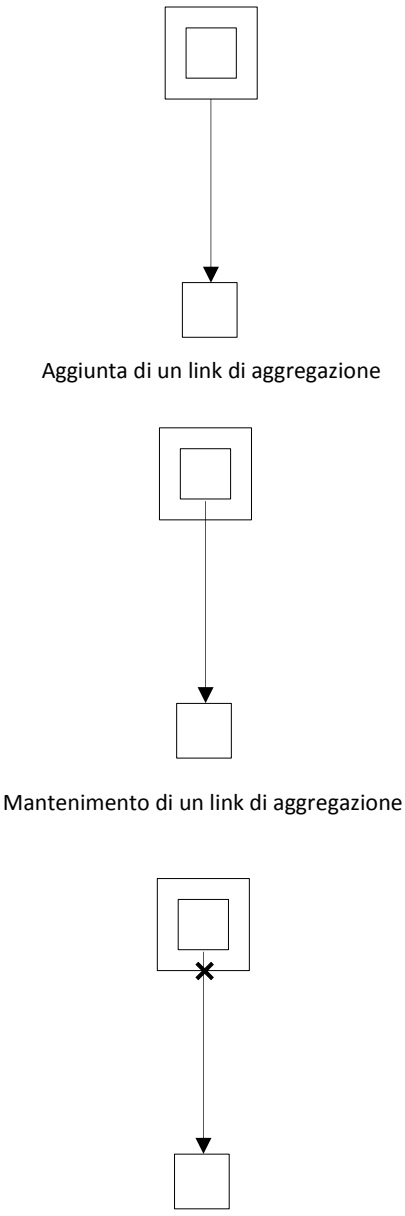

Figura 6.10: Riuso di nodi modello: casi possibili.

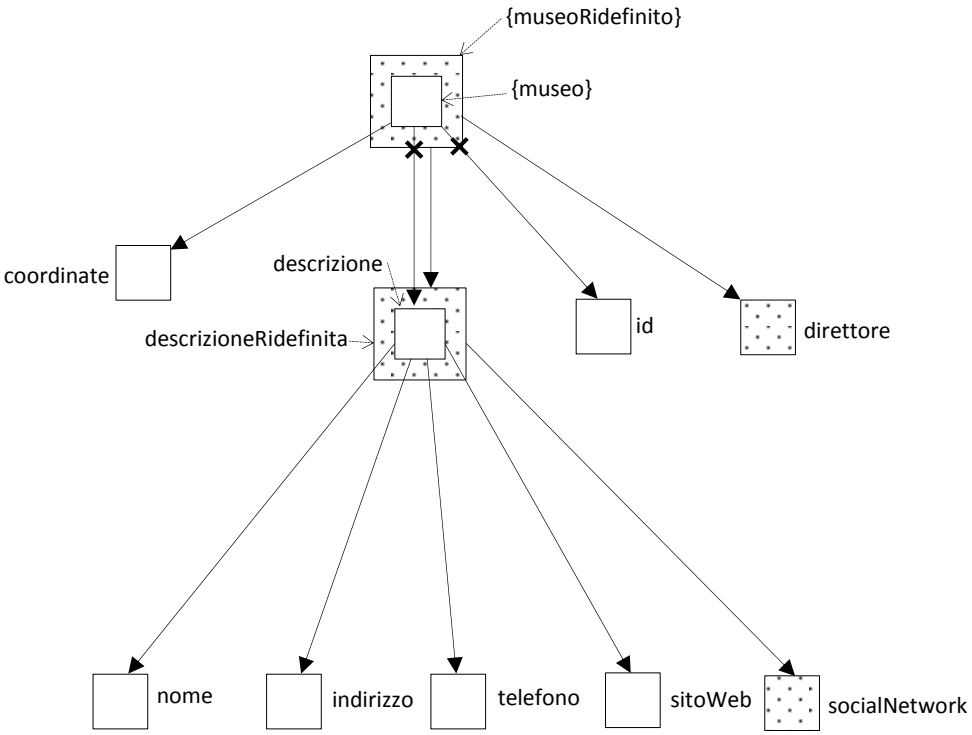

Figura 6.11: Riuso di due nodi modello dell'IDN Template relativo agli Open Data del Comune di Firenze.

Si supponga che il progettista dell'esempio precedentemente considerato voglia ora effettuare il riuso del nodo *descrizione*, e aggiungere a quest'ultimo il nodo *socialNetwork* all'interno del quale memorizzare informazioni relative alle pagine del museo considerato, eventualmente presenti su social network quali Facebook o Twitter. In tal caso il progettista può provvedere ad effettuare tale operazione mediante il riuso del nodo modello *descrizione* nella medesima modalità in cui ha svolto tale operazione con il nodo modello {*museo*}, ovvero ridefinendo un nodo modello *descrizioneRidefinita* che effettui il riuso di *descrizione*, apportando le modifiche desiderate; inoltre dovrà far sì che il nodo modello {*museoRidefinito*} aggreghi adesso il nodo modello *descrizioneRidefinita* e non *descrizione*, come avveniva precedentemente. In figura 6.11 viene illustrato il risultato di tale operazione.

Il meccanismo di riuso consente anche di modificare la gerarchia degli elementi del documento IDN; supponiamo ad esempio che il progettista voglia riutilizzare il nodo modello *descrizione*, aggregandovi un nodo modello di nome *internet*, ed intenda inoltre aggregare a quest'ultimo:

- il nodo modello *socialNetwork* da lui precedentemente definito;

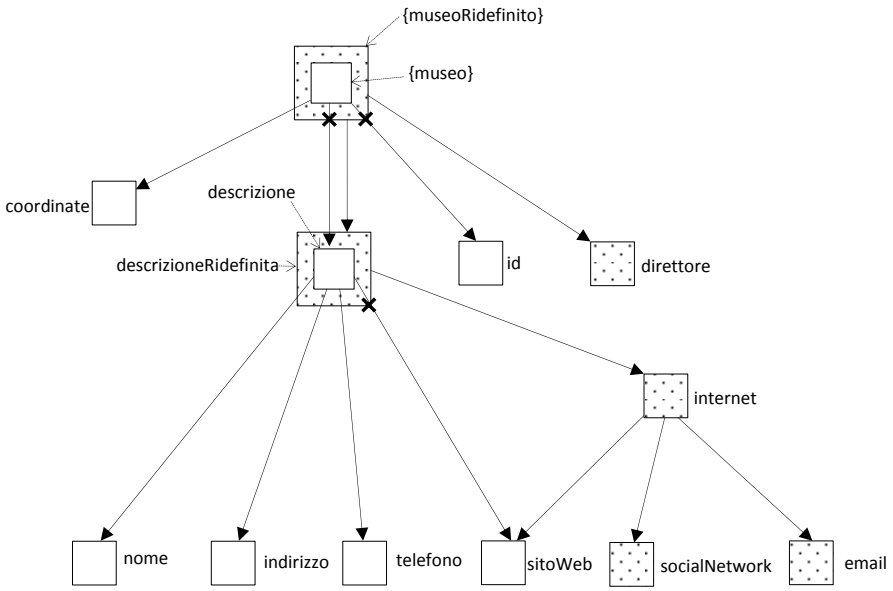

Figura 6.12: Altro esempio di riuso di due nodi modello dell'IDN Template relativo agli Open Data del Comune di Firenze.

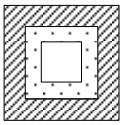

Figura 6.13: Rappresentazione di un nodo che effettua riuso ricorsivo.

- un nodo modello *email*, atto a memorizzare l'indirizzo di posta elettronica di tale museo;
- il nodo modello *sitoWeb* definito dal progettista originale come aggregato da *descrizione*; in particolare il progettista considerato si pone l'obiettivo di rimuovere tale link di aggregazione.

Tale operazione può essere effettuata comodamente ottenendo il risultato illustrato in figura 6.12.
Il meccanismo di riuso può essere applicato anche ricorsivamente, fornendo ad esempio ad un terzo progettista la possibilità di riutilizzare un modello già precedentemente riutilizzato; in figura 6.13 viene illustrata una rappresentazione grafica di tale operazione.

Capitolo 7

IDN Template Data Model

I due capitoli precedenti sono stati dedicati all'Information Model dell'*IDN Template*; in particolare il capitolo 5 ha posto l'attenzione sulla definizione di tale modalità di progettazione, mentre il capitolo 6 si è focalizzato sul meccanismo che consente di effettuare il riuso di modelli già esistenti.
Come già introdotto all'interno del paragrafo 3.2, un *Data Model* si occupa di gestire gli oggetti ad un livello di astrazione più basso rispetto ad un Information Model e comprende dettagli specifici dell'implementazione [PS03].
Risulta quindi conveniente definire almeno un Data Model che implementi il modello informativo definito per l'IDN Template; questo argomento è il tema del presente capitolo.

7.1 Rappresentazione dell'IDN Template

Seguendo quella che nel corso dell'ultimo decennio è ormai diventata una prassi comune all'interno dell'informatica, il Data Model definito dal presenta lavoro utilizza il metalinguaggio XML al fine di implementare il meccanismo di IDN Template. In base a quanto illustrato all'interno del paragrafo 5.1.3, è stato definito un opportuno XML Schema, presentato all'interno del listato 7.1, cui deve risultare conforme ogni documento che rappresenta un IDN Template.

Listato 7.1: XML Schema dell'IDN Template.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://
  www.interdatanet.org/2012/IDNTemplate"
  xmlns:idn="http://www.interdatanet.org/2012/IDNTemplate">
  <!-- IDN Template Schema version 1.0 -->

  <element name="Model" type="idn:ModelType" />
```

```
<complexType name="ModelType">
  <sequence>
    <element name="Node" type="idn:NodeType" minOccurs="1"
      maxOccurs="unbounded" />
  </sequence>
</complexType>

<complexType name="NodeType">
  <all>
    <element name="Structure" type="idn:StructureType" minOccurs="0"
      maxOccurs="1" />
    <element name="Content" type="idn:ContentType" minOccurs="0"
      maxOccurs="1" />
    <element name="Semantics" type="idn:SemanticsType" minOccurs="0"
      maxOccurs="1" />
    <element name="NodeReuse" type="idn:NodeReuseType" minOccurs="0"
      maxOccurs="1" />
  </all>
  <attribute name="uri" type="anyURI" use="required"></attribute>
  <attribute name="root" type="boolean"></attribute>
</complexType>

<complexType name="StructureType">
  <sequence>
    <element name="AggregationLinks" type="idn:AggregationLinksType"
      minOccurs="1" maxOccurs="1" />
  </sequence>
</complexType>

<complexType name="AggregationLinksType">
  <sequence>
    <element name="Link" type="idn:AggregationLinkType"
      minOccurs="1" maxOccurs="unbounded" />
  </sequence>
</complexType>

<complexType name="AggregationLinkType">
  <attribute use="required" name="uri" type="anyURI"></attribute>
  <attribute name="minOccurs" type="idn:minOccursType"></attribute>
  <attribute name="maxOccurs" type="idn:maxOccursType"></attribute>
</complexType>

<simpleType name="minOccursType">
  <restriction base="integer">
    <minInclusive value="0"/>
    <maxInclusive value="1"/>
  </restriction>
</simpleType>

<simpleType name="maxOccursType">
  <restriction base="string">
    <enumeration value="1"/>
    <enumeration value="unbounded"/>
  </restriction>
```

```
  </simpleType>

  <complexType name="ContentType">
    <sequence>
      <element name="MediaType" type="string" minOccurs="1"
        maxOccurs="1" />
      <element name="Syntax" type="anyURI" minOccurs="0" maxOccurs="1" />
    </sequence>
  </complexType>

  <complexType name="SemanticsType">
    <sequence>
      <element name="SemanticLinks" type="idn:SemanticLinksType"
        minOccurs="1" maxOccurs="1" />
    </sequence>
  </complexType>

  <complexType name="SemanticLinksType">
    <sequence>
      <element name="Link" type="idn:LinkType" minOccurs="1"
        maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <complexType name="LinkType">
    <attribute use="required" name="uri" type="anyURI"></attribute>
  </complexType>

  <complexType name="NodeReuseType">
    <choice>
      <element name="FilteredLinks" type="idn:FilteredLinksType"
        minOccurs="0" maxOccurs="1" />
      <element name="MaintainedLinks" type="idn:MaintainedLinksType"
        minOccurs="0" maxOccurs="1" />
    </choice>
    <attribute name="uri" type="anyURI" use="required"></attribute>
  </complexType>

  <complexType name="FilteredLinksType">
    <sequence>
      <element name="Link" type="idn:LinkType" minOccurs="1"
        maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <complexType name="MaintainedLinksType">
    <sequence>
      <element name="Link" type="idn:LinkType" minOccurs="1"
        maxOccurs="unbounded" />
    </sequence>
  </complexType>

</schema>
```

All'interno di tale XML Schema, è stato utilizzato il namespace di riferimento `http://www.interdatanet.org/2012/IDNTemplate`, indicato nel seguito mediante il prefisso `idn`.
La radice di ogni documento conforme a tale XML Schema è costituita dall'elemento *idn:Model*, il quale rappresenta l'intero IDN Template inteso come contenitore di un numero indefinito (ma non nullo) di nodi modello, ognuno dei quali è rappresentato dal tag *idn:Node*.
In analogia con quanto avviene all'interno di IDN-IM, dove ogni nodo di un documento dispone di un proprio identificativo detto LRI (come illustrato al paragrafo 3.6), nel contesto dell'IDN Template ogni nodo modello dispone di un opportuno URI. Tale identificativo può essere assoluto oppure relativo rispetto ad un particolare nodo modello, che gode della caratteristica di essere la radice del documento modello ed è pertanto indicato all'interno dell'XML impostandone a `true` l'attributo booleano *root*.
Ogni *idn:Node* può contenere al suo interno i seguenti tag, ciascuno al massimo con una sola occorrenza:

- *idn:Structure*;
- *idn:Content*;
- *idn:Semantics*;
- *idn:NodeReuse*.

Il tag *idn:Structure*, in conformità con quanto illustrato all'interno del paragrafo 5.2.1, fornisce informazioni riguardo i link modello, ovvero gli elementi informativi che svolgono il ruolo di paradigma per i link di aggregazione esistenti tra due nodi di un documento IDN; inoltre ne indica la molteplicità, ovvero il numero minimo e massimo di link di aggregazione che possono esistere nel dominio dei documenti IDN tra un qualsiasi nodo conforme al nodo modello considerato e nodi conformi al nodo modello aggregato.
A tal fine il tag *idn:Structure* contiene obbligatoriamente al suo interno la sezione *idn:AggregationLinks*, la quale è a sua volta costituita da una sequenza di uno o più elementi *idn:Link*, ciascuno dotato di tre attributi:

- *uri*, il quale indica l'HTTP-URI che identifica il nodo modello aggregato. La sua presenza all'interno del file XML è obbligatoria;
- *minOccurs*, che rappresenta il numero minimo di link di aggregazione che possono esistere nel dominio dei documenti IDN tra i nodi conformi al nodo modello considerato e quelli conformi al nodo modello aggregato. In analogia con quanto avviene nella definizione di qualunque XML Schema, può assumere solamente i valori `0` e `1`. La sua presenza all'interno del file XML è facoltativa;

- *maxOccurs*, che rappresenta il numero massimo di link di aggregazione che possono esistere nel dominio dei documenti IDN tra i nodi conformi al nodo modello considerato e quelli conformi al nodo modello aggregato. In analogia con quanto avviene nella definizione di qualunque XML Schema, può assumere solamente i valori `1` e `unbounded`. La sua presenza all'interno del file XML è facoltativa.

Il tag *idn:Content*, in conformità con quanto illustrato all'interno del paragrafo 5.2.2, fornisce informazioni riguardo al formato e alla sintassi (e vincoli strutturali) dei dati di livello applicativo gestiti dai nodi conformi al nodo modello considerato; include al suo interno due tag:

- *idn:MediaType*, che consente di indicare il MIME Type di detti dati, e deve essere obbligatoriamente presente;

- *idn:Syntax*, opzionale, che memorizza il link verso il file che definisce la sintassi di detti dati (ad esempio, ma non necessariamente, un file XML Schema).

Il tag *idn:Semantics*, in conformità con quanto illustrato all'interno del paragrafo 5.2.3, fornisce informazioni riguardo la semantica dei dati di livello applicativo gestiti dai nodi conformi al nodo modello considerato; si compone obbligatoriamente di una sezione *idn:SemanticLinks*, la quale è a sua volta costituita da una sequenza di uno o più elementi *idn:Link*, ognuno corredato di un attributo obbligatorio *uri* che memorizza un link verso una risorsa dove sono presenti informazioni opportune (ad esempio, ma non necessariamente, una ontologia OWL).
Il tag *idn:NodeReuse* infine, in conformità con quanto illustrato all'interno del paragrafo 6.3, abilita il meccanismo di riuso degli IDN Template; l'attributo obbligatorio *uri* indica quale è il nodo di cui viene effettuato il riuso.

All'interno dell'XML Schema considerato sono stati definiti due diversi meccanismi (uno escludente l'altro) che consentono al progettista di godere della massima libertà possibile per quanto riguarda la gestione dei link di aggregazione presenti nel nodo modello riutilizzato e che vengono eliminati o mantenuti dal nodo modello riutilizzante.
Il primo meccanismo è stato implementato mediante il ricorso al tag *FilteredLinks*, il quale racchiude al suo interno una sequenza di elementi *Link*; per ognuno, l'attributo obbligatorio *uri* indica un link modello presente nel nodo modello riutilizzato, ma eliminato dal nodo modello riutilizzante; in questo contesto si ipotizza che i nodi presenti nel nodo modello riutilizzato che non sono esplicitamente dichiarati sono mantenuti dal nodo modello riutilizzante.

Il secondo meccanismo è stato implementato mediante il ricorso al tag *MaintainedLinks*, il quale racchiude al suo interno una sequenza di elementi *Link*; per ognuno, l'attributo obbligatorio *uri* indica un link modello presente nel nodo modello riutilizzato e mantenuto dal nodo modello riutilizzante; in questo contesto si ipotizza che i nodi presenti nel nodo modello riutilizzato che non sono esplicitamente dichiarati sono eliminati dal nodo modello riutilizzante.
In tal modo il progettista può ricorrere ad una soluzione o all'altra, in base al caso particolare che ricade sotto la sua analisi. Per quanto riguarda invece la gestione di link modello non presenti nel nodo modello riutilizzato ma aggiungi all'interno del nodo modello riutilizzante, il progettista può ricorrere al tag *Structure*, descritto poco sopra.

7.1.1 Abilitare l'IDN Template nei documenti IDN

Il meccanismo di IDN Template descritto all'interno del presente lavoro, in particolare nell'implementazione suggerita dal paragrafo precedente, risulta efficace nella misura in cui viene effettivamente utilizzato all'interno dei documenti IDN.
A tal fine l'XML Schema cui questi ultimi devono risultare conformi, e che è stato illustrato all'appendice A, deve essere opportunamente modificato:

- introducendo il tag *idn:Template*, come figlio del tag *idn:VRIDNMeta*, incluso all'interno di *idn:VRNode*;
- inserendo all'interno del suddetto tag *idn:Template* un opportuno attributo obbligatorio *link*; questo ha il ruolo di indicare l'URI del nodo modello cui il VR-Node considerato deve risultare conforme.

In seguito a tale modifica la definizione dell'elemento *idn:VRNode* nell'XML Schema considerato diviene quella riportata all'interno del listato 7.2.
La definizione completa di tale XML Schema è riportata dal listato 7.3.

Listato 7.2: Modifiche al VR-Documento per abilitare IDN Template.

```
<!-- Variazione alla definizione di idn:VRIDNMeta -->
<xs:element name="Template" type="idnTemplate" minOccurs="0" maxOccurs="1"/
  >

<!-- Aggiunta all'XML Schema -->
<xs:complexType name="idnTemplate">
  <xs:attribute use="required" name="uri" type="xs:anyURI"></xs:attribute>
</xs:complexType>
```

Listato 7.3: XML Schema di documento IDN che abilita IDN Template.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="VRDoc" type="idnVrDocument"/>

  <xs:complexType name="idnVrDocument">
    <xs:sequence>
      <xs:element name="VRDocInfo" type="VrDocInfoType" minOccurs="0"
        maxOccurs="1"/>
      <xs:element name="VRNodeEnvelope" type="idnVrNodeEnvelope" minOccurs=
        "0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="VrDocInfoType">
    <xs:all>
      <xs:element name="Errors" type="ErrorsType" minOccurs="0" maxOccurs="
        1"/>
      <xs:element name="Warnings" type="WarningsType" minOccurs="0"
        maxOccurs="1"/>
    </xs:all>
  </xs:complexType>

  <xs:complexType name="ErrorsType">
    <xs:sequence>
      <xs:element name="Error" type="IdnExceptionType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="WarningsType">
    <xs:sequence>
      <xs:element name="Warning" type="IdnExceptionType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="IdnExceptionType">
    <xs:all>
      <xs:element name="Code" type="xs:string" minOccurs="1" maxOccurs="1"/
        >
      <xs:element name="Message" type="xs:string" minOccurs="1" maxOccurs="
        1"/>
      <xs:element name="TargetedLris" type="lriList" minOccurs="0"
        maxOccurs="1"/>
    </xs:all>
  </xs:complexType>

  <xs:complexType name="lriList">
    <xs:sequence>
      <xs:element name="Lri" type="xs:string" minOccurs="1" maxOccurs="
        unbounded"/>
```

```
  </xs:sequence>
</xs:complexType>

<xs:complexType name="idnVrNodeEnvelope">
  <xs:all>
    <xs:element name="VRNode" type="idnVrNode" minOccurs="1" maxOccurs="1
      " />
    <xs:element name="VRNodeInfo" type="idnVrNodeInfo" minOccurs="1"
      maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="idnVrNodeInfo">
  <xs:all>
    <xs:element name="IsRoot" type="xs:boolean" minOccurs="1" maxOccurs="
      1"/>
    <xs:element name="Etag" type="xs:string" minOccurs="0" maxOccurs="1"/
      >
    <xs:element name="HttpStatus" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="Keywords" type="xs:string" minOccurs="0" maxOccurs=
      "1"/>
  </xs:all>
</xs:complexType>

<xs:element name="VRNode" type="idnVrNode"/>

<xs:complexType name="idnVrNode">
  <xs:all>
    <xs:element name="VRApplicationData" type="vrApplicationData"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="VRApplicationMeta" type="vrApplicationMeta"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="VRIDNMeta" type="vrIdnMeta" minOccurs="0" maxOccurs
      ="1"/>
    <xs:element name="ManagementMeta" type="managementMeta" minOccurs="0"
       maxOccurs="1"/>
  </xs:all>
  <xs:attribute name="lri" type="xs:anyURI" use="required" />
</xs:complexType>

<xs:simpleType name="vrApplicationData">
  <xs:restriction base="xs:base64Binary"/>
</xs:simpleType>

<xs:complexType name="vrApplicationMeta">
  <xs:all>
    <xs:element name="IDNApplicationID" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="IDNAuthor" type="author" minOccurs="0" maxOccurs="1
      "/>
    <xs:element name="IDNModifiedOn" type="xs:string" minOccurs="0"/>
  </xs:all>
</xs:complexType>
```

```
<xs:complexType name="vrIdnMeta">
  <xs:all>
    <xs:element name="AggregationMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="AggregationLink" type="link" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="BackLinkMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="BackLink" type="link" minOccurs="0" maxOccurs=
            "unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="IncomingChangeMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="IncomingChangeLink" type="link" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="NodeLocalName" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="Template" type="xs:anyURI" minOccurs="0" maxOccurs=
      "1"/>
    <xs:element name="ReferenceMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="ReferenceLink" type="link" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>

<xs:complexType name="managementMeta">
  <xs:all>
    <xs:element name="LicensePolicy" type="licensePolicy" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="PrivacyPolicy" type="privacyPolicy" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="ProvenanceManagement" type="provenanceManagement"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="ReplicationPolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="SecurityPolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
```

```
    <xs:element name="TimePolicy" type="timePolicy" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="VersioningPolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="author">
  <xs:all>
    <xs:element name="IDNApplicationInstanceID" type="xs:string"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="IDNHostIPAddress" type="xs:string" minOccurs="0"/>
    <xs:element name="IDNHostName" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="IDNUser" type="xs:string" minOccurs="0" maxOccurs="
      1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="link">
  <xs:all>
    <xs:element name="LocalName" type="xs:string" minOccurs="0" maxOccurs
      ="1"/>
    <xs:element name="Value" type="xs:anyURI" minOccurs="0" maxOccurs="1"
      />
    <xs:element name="Meta" type="xs:string" minOccurs="0" maxOccurs="1"/
      >
  </xs:all>
  <xs:attribute name="vr-id" type="xs:int" use="required" />
</xs:complexType>

<xs:complexType name="licensePolicy">
  <xs:all>
    <xs:element name="DataLicensing" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="DataLicense" type="licenseType" minOccurs="0"
            maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="NodeLicensing" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NodeLicense" type="licenseType" minOccurs="0"
            maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>

<xs:complexType name="licenseType">
  <xs:simpleContent>
```

```
    <xs:extension base="xs:string">
      <xs:attribute name="vr-id" type="xs:int" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="privacyPolicy">
  <xs:all>
    <xs:element name="DataPrivacy" type="privacy" minOccurs="0" maxOccurs
      ="1"/>
    <xs:element name="NodePrivacy" type="privacy" minOccurs="0" maxOccurs
      ="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="privacy">
  <xs:all>
    <xs:element name="PrimaryUse" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="Retention" type="xs:string" minOccurs="0"/>
    <xs:element name="SecondaryUse" type="xs:string" minOccurs="0"/>
    <xs:element name="Sharing" type="xs:string" minOccurs="0"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="provenanceManagement">
  <xs:all>
    <xs:element name="IDNPreviousApplicationID" type="xs:string"
      minOccurs="0"/>
    <xs:element name="IDNPreviousAuthor" type="author" minOccurs="0"/>
    <xs:element name="IDNPreviousModifiedOn" type="xs:string" minOccurs="
      0"/>
    <xs:element name="ProvenancePolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="timePolicy">
  <xs:all>
    <xs:element name="AutoDestroyOn" type="xs:string" minOccurs="0"/>
    <xs:element name="ValidityNotAfter" type="xs:string" minOccurs="0"/>
    <xs:element name="ValidityNotBefore" type="xs:string" minOccurs="0"/>
  </xs:all>
</xs:complexType>

</xs:schema>
```

7.2 Caso d'uso: Open Data di Firenze

All'interno del presente paragrafo saranno illustrati vari casi di utilizzo dell'XML Schema definito, facendo riferimento all'esempio dei musei estratti

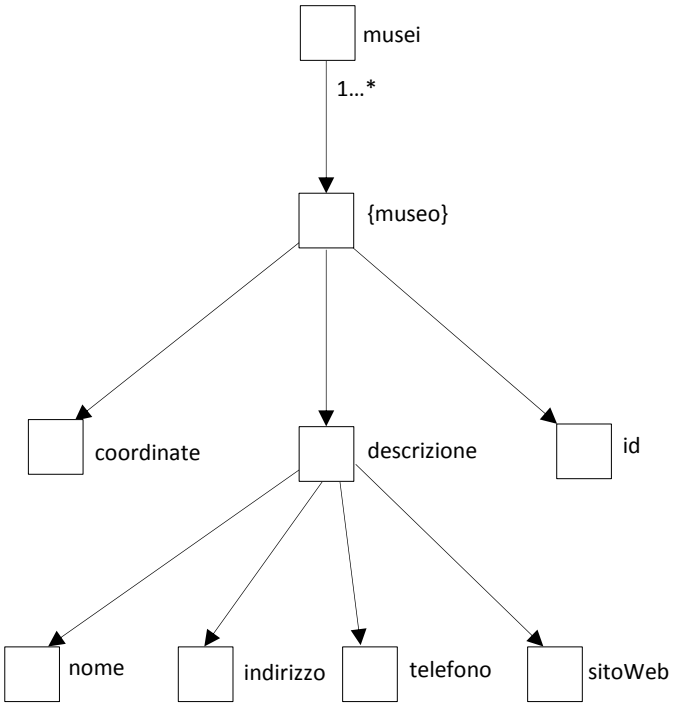

Figura 7.1: IDN Template degli Open Data del Comune di Firenze.

dagli Open Data del Comune di Firenze [1], nelle varie declinazioni considerate all'interno dei capitoli 5 e 6.
Per iniziare, si consideri l'IDN Template illustrato in figura 5.6 e riportato per comodità in figura 7.1; il listato 7.4 riporta un esempio di documento XML che implementa tale modello.

Listato 7.4: IDN Template dei musei degli Open Data di Firenze.

```
<?xml version="1.0" encoding="UTF-8"?>
<idn:Model xmlns:idn="http://www.interdatanet.org/2012/IDNTemplate"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.interdatanet.org/2012/IDNTemplate
    IDNTemplate.xsd ">

  <Node uri="http://idn.det.unifi.it/Model/musei" root="true">
    <Structure>
      <AggregationLinks>
        <Link minOccurs="1" maxOccurs="unbounded" uri="./museo" />
      </AggregationLinks>
    </Structure>
  </Node>
```

[1] Gli Open Data del Comune di Firenze, declinati in chiave IDN-IM, sono descritti dettagliatamente nel capitolo 8.

```
<Node uri="./museo">
  <Structure>
    <AggregationLinks>
      <Link uri="./museo/coordinate" />
      <Link uri="./museo/descrizione" />
      <Link uri="./museo/id" />
    </AggregationLinks>
  </Structure>
  <Semantics>
    <SemanticLinks>
      <Link uri="http://idn.det.unifi.it/Ontologies/Musei#Museo" />
    </SemanticLinks>
  </Semantics>
</Node>

<Node uri="./museo/coordinate">
  <Content>
    <MediaType>application/json</MediaType>
  </Content>
</Node>

<Node uri="./museo/descrizione">
  <Structure>
    <AggregationLinks>
      <Link uri="./museo/descrizione/nome" />
      <Link uri="./museo/descrizione/indirizzo" />
      <Link uri="./museo/descrizione/telefono" />
      <Link uri="./museo/descrizione/sitoWeb" />
    </AggregationLinks>
  </Structure>
</Node>

<Node uri="./museo/id">
  <Content>
    <MediaType>text/plain</MediaType>
  </Content>
</Node>

<Node uri="./museo/descrizione/nome">
  <Content>
    <MediaType>application/xml</MediaType>
    <Syntax>http://idn.det.unifi.it/Model/Musei/schemas/nome.xsd</Syntax>
  </Content>
</Node>

<Node uri="./museo/descrizione/indirizzo">
  <Content>
    <MediaType>application/xml</MediaType>
    <Syntax>http://idn.det.unifi.it/Model/Musei/schemas/indirizzo.xsd</
      Syntax>
  </Content>
  <Semantics>
    <SemanticLinks>
```

```
      <Link uri="http://idn.det.unifi.it/Ontologies/Musei#Indirizzo" />
    </SemanticLinks>
  </Semantics>
</Node>

<Node uri="./museo/descrizione/telefono">
  <Content>
    <MediaType>application/xml</MediaType>
    <Syntax>http://idn.det.unifi.it/Model/Musei/schemas/telefono.xsd</
      Syntax>
  </Content>
  <Semantics>
    <SemanticLinks>
      <Link uri="http://idn.det.unifi.it/Ontologies/Musei#Telefono" />
    </SemanticLinks>
  </Semantics>
</Node>

<Node uri="./museo/descrizione/sitoWeb">
  <Content>
    <MediaType>application/xml</MediaType>
    <Syntax>http://idn.det.unifi.it/Model/Musei/schemas/link.xsd</Syntax>
  </Content>
  <Semantics>
    <SemanticLinks>
      <Link uri="http://idn.det.unifi.it/Ontologies/Musei#SitoWeb" />
    </SemanticLinks>
  </Semantics>
</Node>

</idn:Model>
```

All'interno di tale file sono presenti informazioni che non sono riportate dalla figura 7.1, riguardanti il formato, la sintassi e la semantica dei dati.
Per quanto concerne il formato e la sintassi, è possibile osservare che:

- i nodi conformi al nodo modello *coordinate* gestiscono dati in formato *JSON*, per i quali non è specificata una sintassi;
- i nodi conformi al nodo modello *id* gestiscono dati in testo semplice, per i quali non è specificata una sintassi;
- i nodi conformi ai nodi modello *nome*, *indirizzo*, *telefono* e *sitoWeb* gestiscono dati in formato *XML*. Inoltre, per ognuno di essi è definito un opportuno XML Schema (che il presente lavoro non riporta) utile per la validazione di detti dati.

Per quanto riguarda invece la semantica, risulta evidente come all'interno dei vari tag *Semantics* siano presenti riferimenti all'ontologia identificata

dall'URI `http://idn.det.unifi.it/Ontologies/Musei`, illustrata in figura 7.2 e rappresentata da un opportuno file OWL, riportato all'interno del listato 7.5

Listato 7.5: Ontologia OWL per i musei di Firenze.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.interdatanet.org/2012/Ontologies/Musei#"
  xmlns:base="http://www.interdatanet.org/2012/Ontologies/Musei"
  >

    <owl:Ontology>
    <rdfs:label>Musei di Firenze</rdfs:label>
    <rdfs:comment>Ontologia per la definizione dei musei forniti dagli
      Open Data del Comune di Firenze.
    </rdfs:comment>
  </owl:Ontology>

    <owl:Class rdf:ID="LuogoInteresse">
    <rdfs:comment>Rappresenta un luogo di interesse turistico</rdfs:comment
      >
    <rdfs:label>Luogo di interesse</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="Museo">
    <rdfs:comment>Rappresenta un museo</rdfs:comment>
    <rdfs:label>Museo</rdfs:label>
    <rdfs:subClassOf rdf:resource="#LuogoInteresse" />
  </owl:Class>
  <owl:Class rdf:ID="Telefono">
    <rdfs:comment>Rappresenta un numero di telefono</rdfs:comment>
    <rdfs:label>Telefono</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="SitoWeb">
    <rdfs:comment>Rappresenta un sito web, con relativo URL</rdfs:comment>
    <rdfs:label>Sito Web</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="Indirizzo">
    <rdfs:comment>Rappresenta un indirizzo, completo di via, numero civico
      e luogo
    </rdfs:comment>
    <rdfs:label>Indirizzo</rdfs:label>
  </owl:Class>

  <owl:DataProperty rdf:ID="haCoordinate">
    <rdfs:domain rdf:resource="#LuogoInteresse" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </owl:DataProperty>
  <owl:DataProperty rdf:ID="haNome">
    <rdfs:domain rdf:resource="#LuogoInteresse" />
```

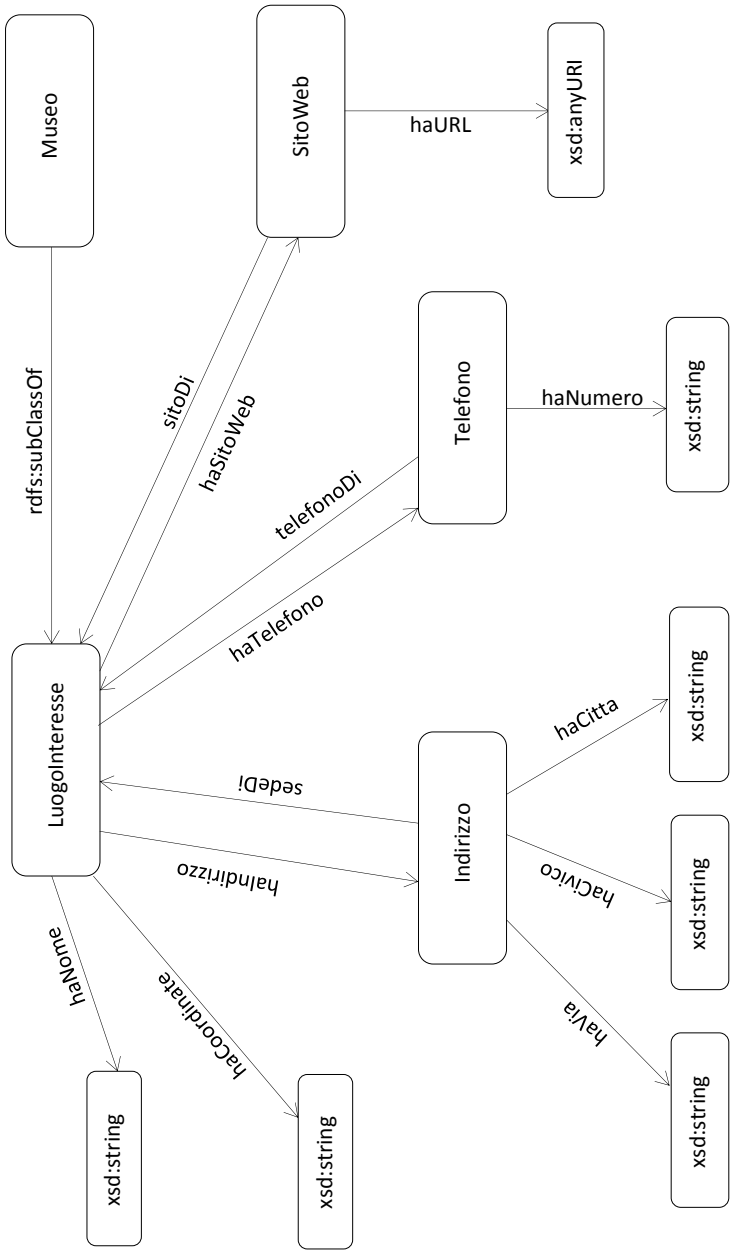

Figura 7.2: Esempio di ontologia per i musei di Firenze.

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </owl:DataProperty>
  <owl:ObjectProperty rdf:ID="haIndirizzo">
    <rdfs:domain rdf:resource="#LuogoInteresse" />
    <rdfs:range rdf:resource="#Indirizzo" />
    <owl:inverseOf rdf:resource="#sedeDi" />
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="sedeDi">
    <rdfs:domain rdf:resource="#Indirizzo" />
    <rdfs:range rdf:resource="#LuogoInteresse" />
    <owl:inverseOf rdf:resource="#haIndirizzo" />
  </owl:ObjectProperty>
  <owl:DataProperty rdf:ID="haVia">
    <rdfs:domain rdf:resource="#Indirizzo" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </owl:DataProperty>
  <owl:DataProperty rdf:ID="haCivico">
    <rdfs:domain rdf:resource="#Indirizzo" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </owl:DataProperty>
  <owl:DataProperty rdf:ID="haCitta">
    <rdfs:domain rdf:resource="#Indirizzo" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </owl:DataProperty>
  <owl:ObjectProperty rdf:ID="haSitoWeb">
    <rdfs:domain rdf:resource="#LuogoInteresse" />
    <rdfs:range rdf:resource="#SitoWeb" />
    <owl:inverseOf rdf:resource="#sitoDi" />
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="sitoDi">
    <rdfs:domain rdf:resource="#SitoWeb" />
    <rdfs:range rdf:resource="#LuogoInteresse" />
    <owl:inverseOf rdf:resource="#haSitoWeb" />
  </owl:ObjectProperty>
  <owl:DataProperty rdf:ID="haURL">
    <rdfs:domain rdf:resource="#SitoWeb" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI" />
  </owl:DataProperty>
  <owl:ObjectProperty rdf:ID="haTelefono">
    <rdfs:domain rdf:resource="#LuogoInteresse" />
    <rdfs:range rdf:resource="#Telefono" />
    <owl:inverseOf rdf:resource="#telefonoDi" />
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="telefonoDi">
    <rdfs:domain rdf:resource="#Telefono" />
    <rdfs:range rdf:resource="#LuogoInteresse" />
    <owl:inverseOf rdf:resource="#haTelefono" />
  </owl:ObjectProperty>
  <owl:DataProperty rdf:ID="haNumero">
    <rdfs:domain rdf:resource="#Telefono" />
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string" />
  </owl:DataProperty>
</rdf:RDF>
```

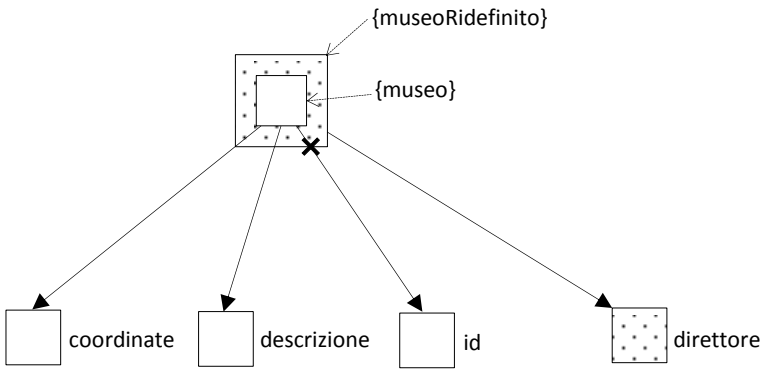

Figura 7.3: Riuso di un nodo modello con aggiunta ed eliminazione di link modello, e specifica del server autoritativo.

I listati 7.6 e 7.7 riportano due diversi esempi di documenti XML che implementano l'IDN Template illustrato in figura 6.8 e riportato per comodità in figura 7.3; il primo di tali file effettua il riuso ricorrendo al tag *FilteredLinks*, il secondo al tag *MaintainedLinks*.

Listato 7.6: IDN Template che effettua il riuso dei musei degli Open Data di Firenze, mediante tag *FilteredLinks*.

```
<?xml version="1.0" encoding="UTF-8"?>
<idn:Model xmlns:idn="http://www.interdatanet.org/2012/IDNTemplate"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.interdatanet.org/2012/IDNTemplate
    IDNTemplate.xsd ">
  <Node uri="http://www.example.com/musei/museoRidefinito" root="true">
    <NodeReuse uri="http://idn.det.unifi.it/Model/musei/museo">
      <FilteredLinks>
        <Link uri="http://idn.det.unifi.it/Model/musei/museo/id" />
      </FilteredLinks>
    </NodeReuse>
    <Structure>
      <AggregationLinks>
        <Link uri="./direttore" />
      </AggregationLinks>
    </Structure>
  </Node>
  <Node uri="./direttore">
    <Content>
      <MediaType>text/plain</MediaType>
    </Content>
  </Node>
</idn:Model>
```

Listato 7.7: IDN Template che effettua il riuso dei musei degli Open Data di Firenze, mediante tag *MaintainedLinks*.

```
<?xml version="1.0" encoding="UTF-8"?>
<idn:Model xmlns:idn="http://www.interdatanet.org/2012/IDNTemplate"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.interdatanet.org/2012/IDNTemplate
    IDNTemplate.xsd ">
  <Node uri="http://www.example.com/musei/museoRidefinito" root="true">
    <NodeReuse uri="http://idn.det.unifi.it/Model/musei/museo">
      <MaintainedLinks>
        <Link uri="http://idn.det.unifi.it/Model/musei/museo/coordinate" />
        <Link uri="http://idn.det.unifi.it/Model/musei/museo/descrizione" />
      </MaintainedLinks>
    </NodeReuse>
          <Structure>
      <AggregationLinks>
        <Link uri="./direttore" />
      </AggregationLinks>
    </Structure>
  </Node>
  <Node uri="./direttore">
    <Content>
      <MediaType>text/plain</MediaType>
    </Content>
  </Node>
</idn:Model>
```

Il listato 7.8 riporta invece un esempio di documento XML che implementa l'IDN Template illustrato in figura 6.11 e riportato per comodità in figura 7.4.

Listato 7.8: IDN Template con riuso multiplo.

```
<?xml version="1.0" encoding="UTF-8"?>
<idn:Model xmlns:idn="http://www.interdatanet.org/2012/IDNTemplate"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.interdatanet.org/2012/IDNTemplate
    IDNTemplate.xsd ">
  <Node uri="http://www.example.com/musei/museoRidefinito" root="true">
    <NodeReuse uri="http://idn.det.unifi.it/Model/musei/museo">
      <MaintainedLinks>
        <Link uri="http://idn.det.unifi.it/Model/musei/museo/coordinate" />
      </MaintainedLinks>
    </NodeReuse>
    <Structure>
      <AggregationLinks>
        <Link uri="./direttore" />
        <Link uri="./descrizioneRidefinita" />
      </AggregationLinks>
    </Structure>
  </Node>
```

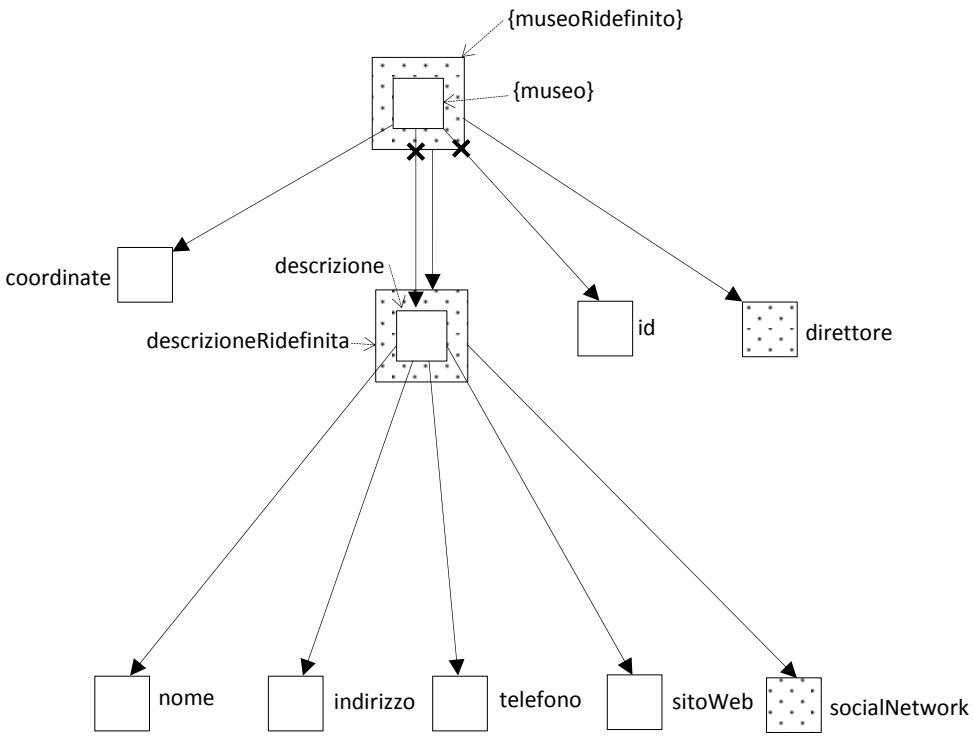

Figura 7.4: Riuso di due nodi modello dell'IDN Template relativo agli Open Data del Comune di Firenze.

```
  <Node uri="./direttore">
    <Content>
      <MediaType>text/plain</MediaType>
    </Content>
  </Node>
  <Node uri="./descrizioneRidefinita">
    <NodeReuse uri="http://idn.det.unifi.it/Model/musei/museo/descrizione"
      />
    <Structure>
      <AggregationLinks>
        <Link uri="./descrizioneRidefinita/socialNetwork" />
      </AggregationLinks>
    </Structure>
  </Node>
  <Node uri="./descrizioneRidefinita/socialNetwork">
    <Content>
      <MediaType>text/plain</MediaType>
    </Content>
  </Node>
</idn:Model>
```

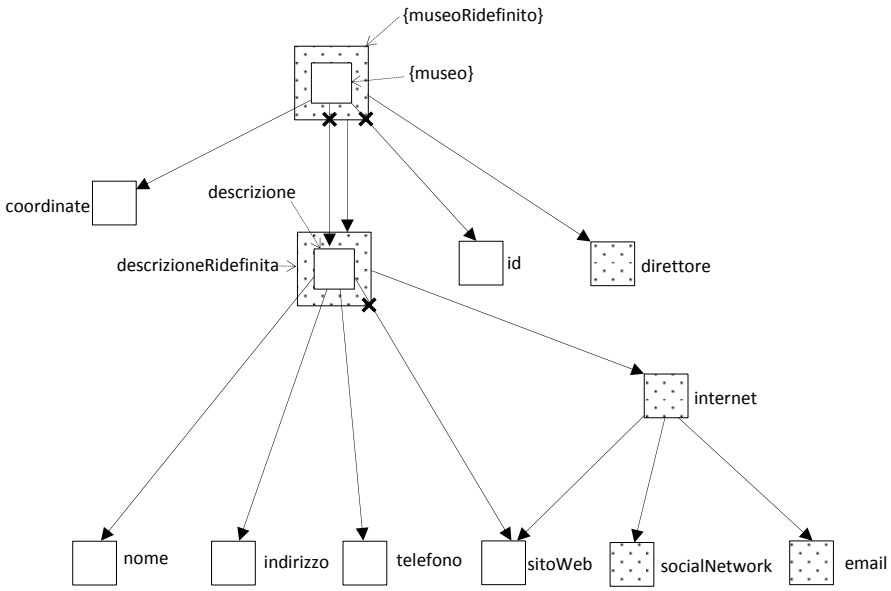

Figura 7.5: Riuso di vari nodi modello dell'IDN Template relativo agli Open Data del Comune di Firenze.

Infine, il listato 7.9 riporta un esempio di documento XML che implementa l'IDN Template illustrato in figura 6.12 e riportato per comodità in figura 7.5.

Listato 7.9: IDN Template con riuso multiplo.

```
<?xml version="1.0" encoding="UTF-8"?>
<idn:Model xmlns:idn="http://www.interdatanet.org/2012/IDNTemplate"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.interdatanet.org/2012/IDNTemplate
    IDNTemplate.xsd ">

  <Node uri="http://www.example.com/musei/museoRidefinito" root="true">
    <NodeReuse uri="http://idn.det.unifi.it/Model/musei/museo">
      <MaintainedLinks>
        <Link uri="http://idn.det.unifi.it/Model/musei/museo/coordinate" />
      </MaintainedLinks>
    </NodeReuse>
    <Structure>
      <AggregationLinks>
        <Link uri="./direttore" />
        <Link uri="./descrizioneRidefinita" />
      </AggregationLinks>
    </Structure>
  </Node>

  <Node uri="./direttore">
```

```
    <Content>
      <MediaType>text/plain</MediaType>
    </Content>
  </Node>

  <Node uri="./descrizioneRidefinita">
    <NodeReuse uri="http://idn.det.unifi.it/Model/musei/museo/descrizione">
      <FilteredLinks>
        <Link
          uri="http://idn.det.unifi.it/Model/musei/museo/descrizione/sitoWeb
            " />
      </FilteredLinks>
    </NodeReuse>
    <Structure>
      <AggregationLinks>
        <Link uri="./descrizioneRidefinita/internet" />
      </AggregationLinks>
    </Structure>
  </Node>

  <Node uri="./descrizioneRidefinita/internet">
    <Structure>
      <AggregationLinks>
        <Link uri="http://idn.det.unifi.it/Model/musei/museo/descrizione/
          sitoWeb" />
        <Link uri="./descrizioneRidefinita/internet/socialNetwork" />
        <Link uri="./descrizioneRidefinita/internet/email" />
      </AggregationLinks>
    </Structure>
  </Node>

  <Node uri="./descrizioneRidefinita/internet/socialNetwork">
    <Content>
      <MediaType>text/plain</MediaType>
    </Content>
  </Node>

  <Node uri="./descrizioneRidefinita/internet/email">
    <Content>
      <MediaType>text/plain</MediaType>
    </Content>
  </Node>

</idn:Model>
```

7.3 Validazione di un documento IDN

In analogia a quanto avviene con XML, è possibile effettuare una validazione dei documenti IDN rispetto agli IDN Template.
Tale procedura può risultare particolarmente utile nella fase di testing di una IDN Compliant Application, consentendo al programmatore di verificare

se i documenti prodotti e/o elaborati da tale software sono effettivamente conformi a quanto atteso.
In particolare occorre che siano verificate tutte le condizioni indicate di seguito:

1. **conformità di struttura**: il documento IDN deve possedere una struttura conforme a quella dichiarata dall'IDN Template.

   In particolare, per ogni nodo del DAG che costituisce il documento IDN occorre validare l'insieme dei link di aggregazione. Questo significa che ogni nodo `X` deve aggregare tutti e soli nodi che risultino ciascuno conforme (per le tre regole qui elencate) ad un nodo modello che sia aggregato da quello cui il nodo `X` in questione dichiara di essere conforme. Inoltre ogni link di aggregazione deve essere presente nel rispetto delle molteplicità previste;

2. **conformità di formato**: per ogni nodo del DAG che costituisce il documento IDN, occorre validare il formato, ovvero il MIME Type dei dati di livello applicativo da esso gestiti deve corrispondere a quello indicato dal relativo nodo modello;

3. **conformità di sintassi**: per ogni nodo del DAG che costituisce il documento IDN per cui è stata indicata la sintassi, occorre che quest'ultima sia validata, ovvero la sintassi dei dati di livello applicativo da esso gestiti deve essere conforme a quella indicata dal relativo nodo modello.

Il presente lavoro si limita a fornire la suddetta serie di regole guida cui un software validatore dei documenti deve sottostare, demandando tale procedura ad una futura implementazione.

Parte IV

Caso di studio: Open Data Firenze

Capitolo 8

Riuso degli Open Data di Firenze

Come illustrato all'interno della parte II, l'intero sistema IDN può essere descritto attraverso l'utilizzo di tre viste:

1. la vista sulle risorse e sull'informazione, IDN Information Model, illustrata al capitolo 3;

2. la vista sull'architettura dei servizi, IDN Service Architecture, illustrata al capitolo 4;

3. la vista sulle applicazioni, le cosiddette IDN Compliant Application.

All'interno del presente capitolo saranno dapprima descritte le principali nozioni di una IDN Compliant Application.
Quindi, dopo avere introdotto il concetto di ETL (Extract, Transform and Load), verrà illustrata una IDN Compliant Application che utilizza tale concetto al fine creare una "copia cache" degli Open Data pubblicati dal Comune di Firenze, sotto forma di documenti IDN-IM conformi ad un opportuno IDN Template.
Infine, sarà descritta Easy Florence, una IDN Compliant Application mobile, che effettua il riuso del precedente IDN Template e dei documenti IDN-IM rappresentanti gli Open Data rilasciati dal Comune di Firenze, ottenuti attraverso la procedura di ETL citata.

8.1 IDN Compliant Applications

Le applicazioni in grado di utilizzare documenti strutturati secondo IDN-IM (dette **IDN Compliant Applications**), costituiscono la terza vista su InterDataNet, insieme all'IDN Information Model e all'IDN Service Architecture, illustrati rispettivamente ai capitoli 3 e 4.

Mentre IDN-SA offre i dati e le API per la loro gestione, la relativa logica di manipolazione si trova all'interno delle IDN-Compliant Applications, le quali implementano un workflow distribuito, in quanto vengono coinvolte diverse entità che collaborano intorno a documenti condivisi.

Nonostante le finalità siano essenzialmente diverse risulta possibile, esclusivamente a titolo esemplificativo, ipotizzare un parallelo [Cio10] fra il mondo dei database, che è ormai consolidato da anni e quindi ben noto, ed InterDataNet.

Da un lato sia IDN-IM sia il modello relazionale dei database catturano esclusivamente i principi astratti dell'informazione, senza introdurre elementi implementativi; dall'altro lato, IDN-SA corrisponde ad una specifica implementazione, ad esempio al database PostgreSQL. Tuttavia la differenza sostanziale fra un database ed InterDataNet risiede nel fatto che mentre ad n installazioni di PostgreSQL corrispondono normalmente $m > n$ silos di dati indipendenti (i singoli database) [1], invece l'insieme di sistemi che implementano IDN-SA cooperano per la realizzazione di un'unica ragnatela interconnessa di dati su scala di livello Web [2].

Le applicazioni sono le entità che utilizzando il modello dell'informazione proposto (i database da un lato e IDN-IM dall'altro), implementano le necessarie logiche di business per affrontare e risolvere specifici problemi di dominio. Nel mondo dei database gli esempi possibili di applicazioni sono in numero pressoché infinito, ma se la visione che ha guidato gli sviluppi di InterDataNet in questi anni dovesse concretizzarsi, anche gli esempi di applicazioni IDN saranno in numero elevato. Queste ultime, date le caratteristiche del sistema, saranno orientate alla collaborazione su larga scala e non solo alla mera manipolazione di dati.

Si considerino i complessi workflow utilizzati per la produzione di un atto amministrativo quale una patente di guida o una carta di identità; essi potrebbero essere gestiti tramite decine di applicazioni IDN diverse ed ogni ente potrebbe realizzare la propria; tuttavia sarebbero in grado di manipolare non solo le patenti di guida o le carte di identità da loro emesse, ma anche quelle rilasciate da uno qualunque degli altri enti (tramite altre applicazioni). Questo è ciò che accade da anni nel Web limitatamente al contesto della pubblicazione di documenti, dove ogni ente/organizzazione è libera di selezionare il proprio stack tecnologico per la realizzazione del proprio sito web, conscio del fatto che gli utenti, con qualsiasi user-agent (entro certi limiti), potranno fruire dei contenuti pubblicati senza difficoltà.

[1] Si assume che ogni installazione di PostgreSQL esponga uno o più database. La federazione non è stata volutamente presa in considerazione per non complicare eccessivamente l'esempio.

[2] Almeno in via concettuale, niente vieta la possibilità di creare una "Intranet" InterDataNet confinata all'interno di un ben determinato perimetro anche se questo vanificherebbe, almeno in parte, i benefici apportati dal sistema.

InterDataNet intende far evolvere ulteriormente tale scenario, affinché la produzione dell'informazione avvenga "in banda" ovvero tramite gli stessi strumenti che sono utilizzati per fruire dell'informazione stessa.
Nella visione InterDataNet le applicazioni interagiscono con i servizi IDN erogati in rete, il cui obiettivo è quello di esporre le funzionalità necessarie per la gestione di informazioni rappresentate secondo un linguaggio comune, in modo tale da permettere alle stesse, e ad altre applicazioni, di utilizzare e ri-utilizzare le stesse informazioni per scopi ed in contesti diversi.

8.2 ETL

Con il termine *ETL* (*Extract, Transform and Load*) si indica un procedimento di elaborazione dati che si svolge attraverso i tre passi seguenti:

- estrazione dei dati da una fonte esterna;
- trasformazione dei dati, al fine di soddisfare le necessità operative;
- caricamento dei dati sulla destinazione finale.

La prima parte di un processo ETL, l'estrazione dei dati dai sistemi sorgenti, è generalmente la più complicata tra le fasi, poiché avrà conseguenze sui passi seguenti; molto spesso infatti i dati da elaborare sono riuniti da sorgenti diverse, ciascuna delle quali potrebbe utilizzare una differente organizzazione o formato di file (ad es. database, file di testo, file XML, ecc. . .). In generale, l'obiettivo della fase di estrazione consiste nel portare i dati verso un singolo formato che sia appropriato per il successivo processo di trasformazione. All'interno della procedura di estrazione viene effettuato anche il parsing dei dati estratti, al fine di verificare se essi rispettano la struttura attesa.
La seconda parte di un processo ETL, ovvero la trasformazione, applica una serie di regole o funzioni alle informazioni estratte dalla sorgente in modo da ottenere i dati da caricare nella destinazione finale. In alcuni casi la procedura richiesta è minima (o addirittura nulla), mentre talvolta, al fine di incontrare le necessità di business e le tecniche dello storage di destinazione, occorre eseguire uno o più processi di trasformazione, come ad esempio calcolare nuovi valori derivati da altri, ordinare i dati, eliminare i doppioni, suddividere una colonna in più colonne, trasformare valori codificati [3], ecc. . .
Infine, attraverso la fase di caricamento di un processo ETL, i dati vengono memorizzati all'interno del sistema di destinazione. Tale processo può variare profondamente in base ai requisiti dell'organizzazione: ad esempio può essere effettuata una sovrascrittura delle informazioni esistenti con altre

[3] La trasformazione può essere necessaria ad esempio quando il sistema originario utilizza un booleano assegnando 0 alle donne e 1 agli uomini (o viceversa), mentre il sistema di destinazione utilizza una stringa "M" o "F".

informazioni cumulative, nel caso in cui il caricamento dei dati sia effettuato su base giornaliera, settimanale o mensile; oppure potrebbero essere aggiunti nuovi dati in forma storicizzata, ad esempio ogni ora.

8.3 ETL per gli Open Data di Firenze

Come illustrato all'interno del paragrafo 2.4, il Comune di Firenze risulta a dicembre 2012 la terza amministrazione in Italia per numero di dataset rilasciati (circa 350, come illustrato all'interno della tabella 2.1); tali dati sono disponibili in formati liberamente riutilizzabili (quali CSV, KML e Shapefile, illustrati all'interno del paragrafo 2.3) senza alcuna restrizione.
Una analisi accurata di essi consente comunque di evidenziare alcune caratteristiche che potrebbero essere aggiunte al fine di renderli ancora più usabili di quanto siano già adesso. Innanzitutto, i dati sono talvolta presentati in maniera generica; ad esempio i file KML relativi alle attività commerciali non consentono di conoscere con precisione la tipologia di ognuna di esse (es. forni, macellerie, ecc...).
Inoltre sarebbe interessante riuscire ad individuare facilmente un singolo dato senza dovere leggere l'intero dataset; ad esempio, volendo conoscere il sito web del Museo Nazionale del Bargello, occorre dapprima aprire il file relativo ai musei, quindi individuare al suo interno la sezione relativa a tale museo, ed infine trovare l'informazione desiderata. Sarebbe invece preferibile poter indirizzare direttamente tale dato, in modo da accedervi con semplicità. Le informazioni dotate di tali caratteristica sarebbero facilmente riutilizzabili ed integrabili in applicazioni realizzate da terze parti.
A tal fine e nell'ottica di presentare alcuni casi d'uso della tecnica di modellazione mediante IDN Template, è stata realizzata una opportuna IDN Compliant Application in grado di trasformare tali dati in documenti IDN-IM.
Come già illustrato all'interno del paragrafo 2.4, il Comune di Firenze ha suddiviso i propri Open Data in varie categorie, ad ognuna delle quali appartiene un certo numero di dataset. Per quanto riguarda i file georeferenziati in formato KML [4], ogni dataset si compone di una serie di luoghi di interesse identificati mediante il tag <Placemark>. Per ognuno di essi il Comune di Firenze ha definito varie informazioni, tra le quali risultano essere più rilevanti:

- un identificativo univoco, rappresentato dall'attributo *id* all'interno del tag <Placemark>;
- le coordinate geografiche; in particolare per quanto riguarda i luoghi di interesse con geometria puntiforme, esse sono definite mediante il

[4] Una procedura analoga, che non sarà descritta all'interno del presente lavoro, è stata eseguita anche per i file CSV.

tag <Point>, mentre per quelli con geometria lineare o areale viene utilizzato il tag <MultiGeometry>;

- una descrizione, definita attraverso il tag <Description>. In particolare essa è costituita da un CDATA, al cui interno è stata inserita una lista di elementi [5], ognuno dei quali si compone di due sezioni HTML , l'una appartenente alla classe *atr-name* e l'altra a *atr-value*. Tali dati costituiscono quindi una serie di coppie nome-valore [6], dove il nome è definito dall'elemento racchiuso all'interno dello di classe *atr-name*, mentre il valore è dato dall'elemento racchiuso all'interno dello di classe *atr-value*.

Tutti gli altri elementi presenti all'interno del file KML, non sono stati invece considerati, poiché sono essenzialmente utili ai fini della visualizzazione del luogo di interesse su una mappa.
L'organizzazione del file sin qui descritta ha portato alla definizione di un IDN Template, avente come radice un nodo generico, indicato con il nome *opendata*, il cui LRI può essere considerato come indirizzo base dal quale comporre gli altri LRI di ogni risorsa descritta nel seguito (in base alle regole indicate nel paragrafo 3.6). All'interno della presente trattazione esso è indicato come:

`http://server/baseLRI` .

Per ognuna delle categorie definite dal Comune di Firenze, esso aggrega un nodo che è possibile indicare con il nome generico di {*categoria*}; pertanto il nodo *opendata* ha come figli i nodi *ambiente*, *istruzione*, *turismo*, ecc...
In base alle regole per la definizione degli LRI canonici, ognuno di tali nodi è identificato da un LRI del tipo:

`http://server/baseLRI/{categoria}` .

Ciascuno di essi aggrega un nodo per ognuno dei dataset che il Comune di Firenze ha inserito nella relativa categoria, che è possibile indicare con il nome generico di {*dataset*}; ad esempio, il nodo *turismo* aggrega i nodi *musei*, *consolati*, *bagnipubblici*, ecc..., ognuno dei quali rappresenta il corrispondente file KML.
Ognuno di tali nodi è identificato da un LRI del tipo:

`http://server/baseLRI/{categoria}/{dataset}` .

[5] Il Comune di Firenze ha scelto presumibilmente un artificio grafico per la visualizzazione di una serie di informazioni nel fumetto che viene presentato quando l'utente seleziona il luogo di interesse, inserendo all'interno di una lista (identificata dal tag HTML <ul>) una serie di elementi (ognuno identificato dal tag HTML <li>).

[6] Occorre osservare che tutti i luoghi all'interno di uno specifico dataset presentano lo stesso insieme di coppie chiave-valore.

Ciascun nodo {*dataset*} aggrega un nodo per ognuno dei luoghi di interesse definiti dal tag <Placemark> all'interno del file KML, che è possibile indicare come {*luogo*}.
Ognuno di tali nodi è identificato da un LRI del tipo:

`http://server/baseLRI/{categoria}/{dataset}/{luogo}` .

Tenendo conto delle informazioni significative presenti nel file KML, descritte precedentemente, ogni {*luogo*} aggrega:

- un nodo *id*; i dati di livello applicativo da esso gestiti contengono l'identificativo specificato dall'attributo *id* sopra illustrato;
- un nodo *coordinate*; i dati di livello applicativo da esso gestiti contengono la codifica delle coordinate geografiche in formato GeoJSON (illustrato all'interno del paragrafo 2.3.3);
- un nodo *descrizione*, il quale aggrega a sua volta un numero di nodi pari a quello delle coppie nome-valore presenti nel tag <Description> sopra illustrate. In pratica, per ogni coppia nome-valore esiste un nodo il cui LRI è stabilito dal nome ed i cui dati di livello applicativo sono impostati al valore.

La figura 8.1 illustra la struttura di un ipotetico e generico IDN Template relativo alla rappresentazione sotto forma di documento IDN-IM degli Open Data pubblicati dal Comune di Firenze. Tuttavia occorre rilevare che i vari dataset sono diversificati tra loro dall'insieme di nodi aggregati da *descrizione*, poiché le coppie chiave-valore sono diverse per ogni dataset. Inoltre la stessa natura molteplice dei dataset fa sì che essi siano diversi tra loro anche per quanto riguarda la semantica.
Pertanto, la struttura del modello presentata in figura 8.1 può essere considerata come un meta-modello da declinare opportunamente in base al dataset considerato; ad esempio la figura 8.2 illustra la sezione dell'IDN Template relativa ai musei, mentre la figura 8.3 illustra quella relativa ai bagni pubblici. In particolare, per quanto riguarda il primo esempio, è stata anche definita una apposita ontologia, illustrata al paragrafo 7.2.
È stata quindi realizzata una IDN Compliant Application che effettua un processo di ETL per gli Open Data del Comune di Firenze, al fine di ottenere dei documenti IDN-IM conformi ai relativi IDN Template sopra definiti, i quali gestiscono come dati di livello applicativo le informazioni ricavate da tali Open Data.
Tale applicazione [7] esegue i tre passi seguenti:

[7] Nella implementazione tale ETL è stato realizzato come procedura *batch* in linguaggio Java.

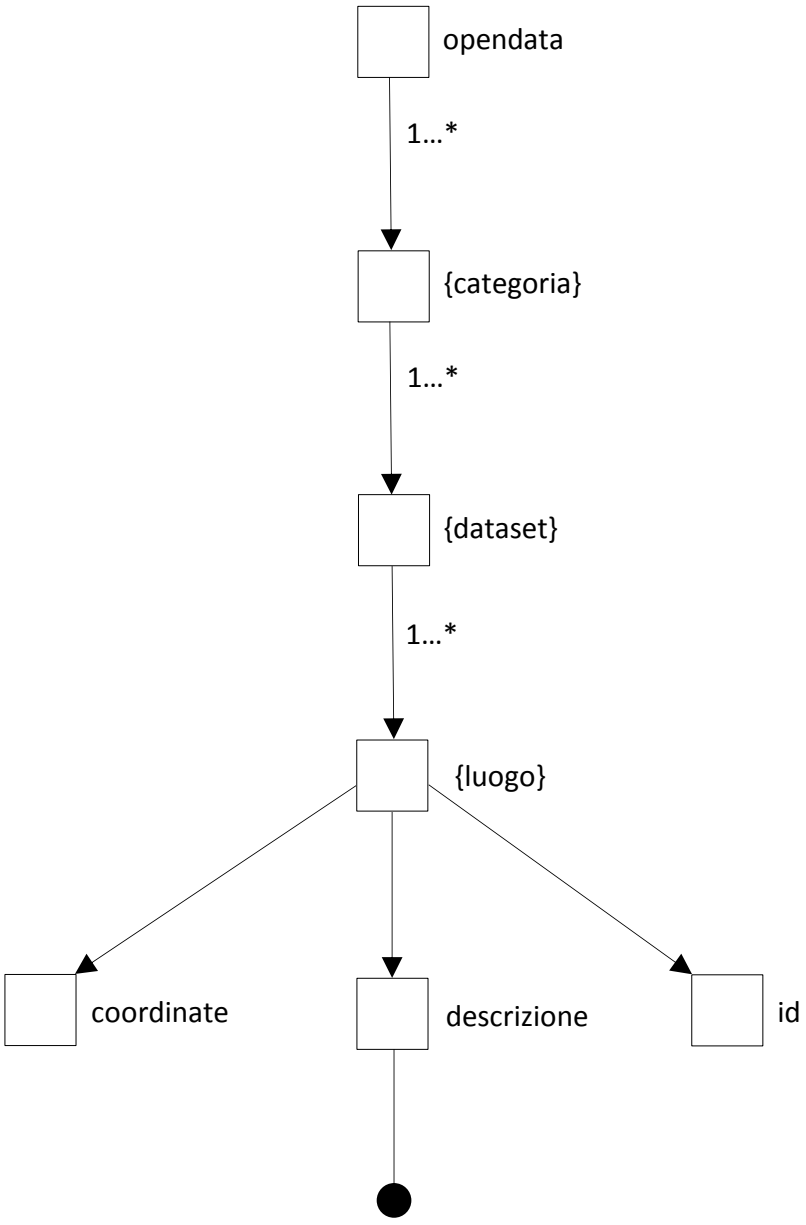

Figura 8.1: IDN Template generico per gli Open Data geografici pubblicati dal Comune di Firenze; per la natura generica dei dati, non sono specificati gli aggregati del nodo modello *descrizione*.

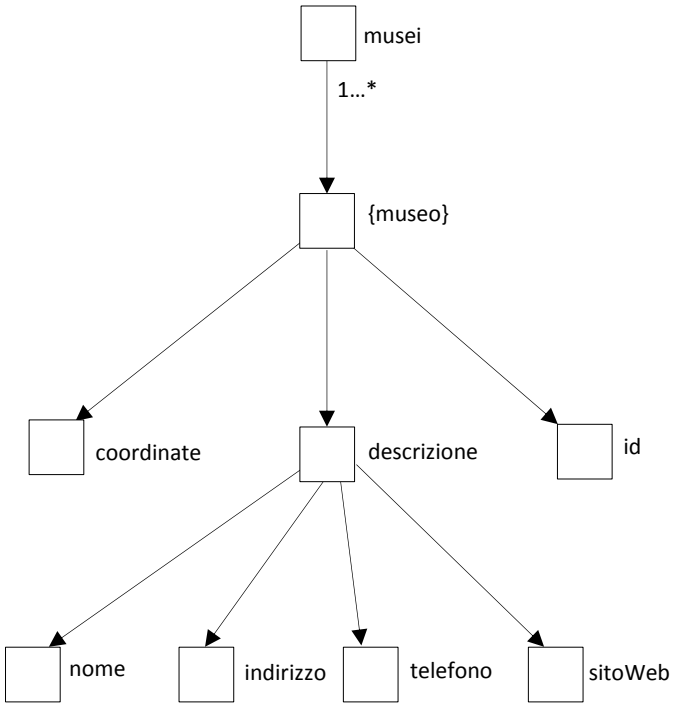

Figura 8.2: IDN Template per gli Open Data relativi ai musei pubblicati dal Comune di Firenze.

- i dati contenuti nel file KML desiderato [8] vengono estratti e caricati in memoria in una struttura analoga alla sorgente [9];
- nel processo di trasformazione, viene creata in memoria una apposita struttura equivalente a quella del documento IDN-IM desiderato; per ogni nodo, i dati di livello applicativo gestiti sono ottenuti sulla base di quelli caricati al passo precedente [10];
- infine il documento IDN-IM corrispondente alla struttura illustrata [11] viene memorizzato da un apposito Virtual Resource.

Si può quindi affermare che è stata creata in tal modo una "copia cache", sotto forma di documenti IDN-IM, degli Open Data rilasciati dal Comune di Firenze; essa può essere successivamente impiegata da IDN Compliant

[8] Il file KML in questione era stato precedentemente scaricato dal sito del Comune di Firenze relativo agli Open Data [Com12b] e memorizzato in una apposita directory.

[9] Tale operazione viene effettuata attraverso l'unmarshalling dei dati.

[10] Nella implementazione si ricorre ad un apposita libreria Java per la gestione di documenti IDN.

[11] Nella implementazione si ricorre alla medesima libreria Java per la gestione di documenti IDN, che effettua una operazione di marshalling dei dati.

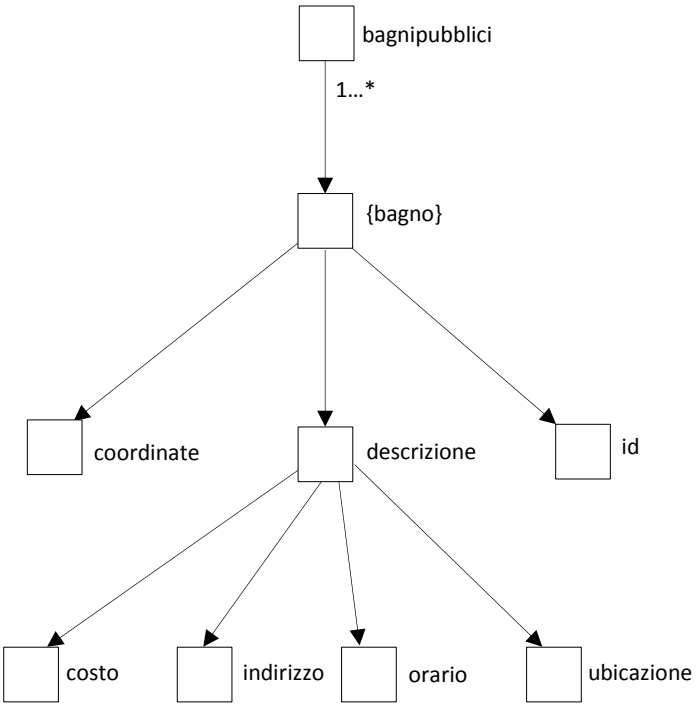

Figura 8.3: IDN Template per gli Open Data relativi ai bagni pubblici pubblicati dal Comune di Firenze.

Applications che effettuano il riuso dei tali dati, come Easy Florence, illustrata all'interno del paragrafo 8.4. Si apre così la strada alla possibilità di realizzare applicazioni che utilizzano dati provenienti anche da diverse sorgenti.

8.4 Riuso degli Open Data in Easy Florence

Easy Florence è una IDN Compliant Application per dispositivi Android [12] che si propone l'obiettivo di rendere interattiva la visita turistica nella città di Firenze. L'applicazione rende disponibile all'utente una mappa della città sulla quale sono posti in rilievo i luoghi di interesse più importanti dal punto di vista turistico. Ad ognuno di essi sono associate informazioni specifiche che possono essere visualizzate in qualsiasi momento ed in modo istantaneo con un semplice tap [13].

[12] Android è un sistema operativo open source per dispositivi mobile (smartphone e tablet) basato su kernel Linux [Goo12a]. Attualmente è il sistema operativo più utilizzato in ambito mobile [Gar12].

[13] Con il termine inglese *to tap* (traducibile con *picchiettare*) si indica il gesto attraverso il quale un utente tocca un elemento presentato da un dispositivo touchscreen; esso è analogo al clic del mouse eseguito sui tradizionali computer desktop.

Figura 8.4: Schermata iniziale di Easy Florence e punto di interesse.

Figura 8.5: I commenti relativi al Duomo di Firenze e la schermata per l'inserimento di un commento.

In Easy Florence ogni utente dispone di un profilo pubblico grazie al quale può recensire, tramite un commento testuale, i punti di interesse proposti dall'applicazione, nonché leggere i commenti degli altri utenti. Infine, il turista può osservare direttamente sulla mappa e in tempo reale, oltre alla sua posizione, anche quella degli altri turisti collegati all'applicazione, nell'ottica della condivisione di contenuti generati dagli utenti.

All'avvio di Easy Florence, l'applicazione mostra la mappa della città di Firenze, ed un messaggio di sistema avvisa che il download dei punti di interesse sta per iniziare (figura 8.4 a sinistra). Questi ultimi vengono quindi disposti sulla mappa rappresentati da un'icona che ne indica la relativa tipologia. Effettuando il tap su un particolare punto di interesse, l'applicazione ne mostra tutte le informazioni a disposizione (figura 8.4 a destra), consentendo anche la chiamata telefonica o l'accesso rapido al relativo sito web.

La schermata presenta anche due pulsanti, utilizzabili rispettivamente per visualizzare i commenti che il punto di interesse ha già ricevuto da parte degli utenti (figura 8.5 a sinistra) o per inserire il proprio commento (figura 8.5 a destra).

La posizione dell'utente sulla mappa viene segnalata mediante una icona azzurra lampeggiante; effettuando il tap su di essa viene visualizzato un messaggio di sistema in cui sono specificati la via e il numero civico corrispondente alle coordinate (figura 8.6, a sinistra).

La voce di menù `Your profile` consente di visualizzare il profilo dell'utente, ovvero le informazioni personali pubbliche e la lista dei commenti che egli ha lasciato (figura 8.7, a destra). Il tap su un commento consente di visualizzare la schermata del relativo punto di interesse.

Gli altri utenti dell'applicazione sono raffigurati sulla mappa tramite l'icona che ritrae il logo del sistema operativo Android (figura 8.6 a destra), alle coordinate dell'ultima posizione rilevata [14]; una lista degli utenti registrati è visibile selezionando il pulsante `Online users` del menù (figura 8.7 a sinistra).

Per soddisfare le caratteristiche fin qui illustrate, Easy Florence utilizza la copia cache degli Open Data messi a disposizione dal Comune di Firenze, ottenuta attraverso il processo di ETL descritto all'interno del paragrafo 8.3. In particolare l'applicazione effettua il riuso dei dati relativi alle informazioni turistiche, come musei e bagni pubblici.

La struttura dati dell'IDN Template utilizzata (limitatamente ai musei) è illustrata in figura 8.8; da essa risulta che il nodo modello *easyflorence* aggrega i due nodi modello *musei* e *utenti*. Il primo nodo di essi è finalizzato a ridefinire il nodo modello {*museo*}, descritto al paragrafo 8.3, aggiungendo il link modello verso il nodo modello *messaggi*; questo permette a sua volta

[14] Se l'utente è collegato all'applicazione e si sta muovendo, la sua icona si sposterà sulla mappa di conseguenza.

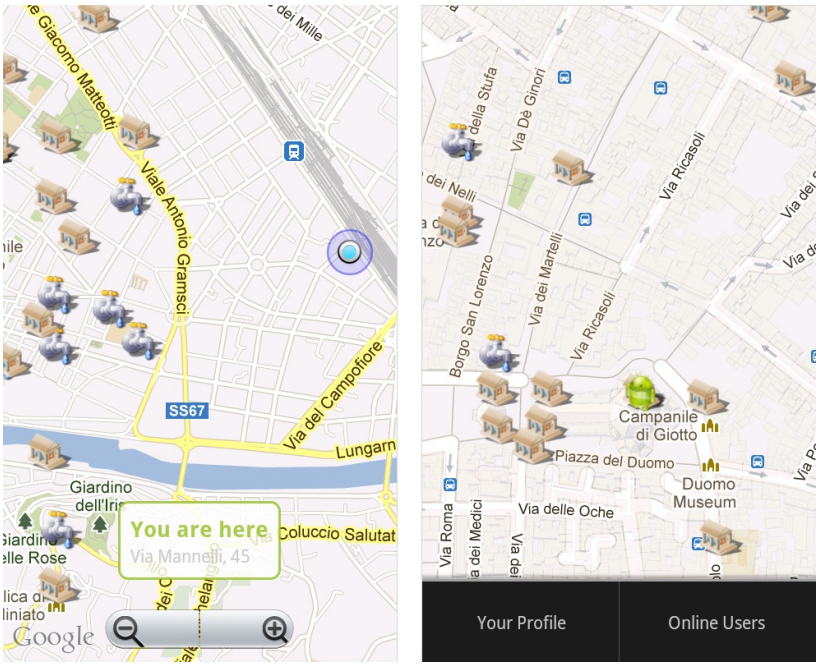

Figura 8.6: Il messaggio relativo alla propria posizione e l'apertura del menù.

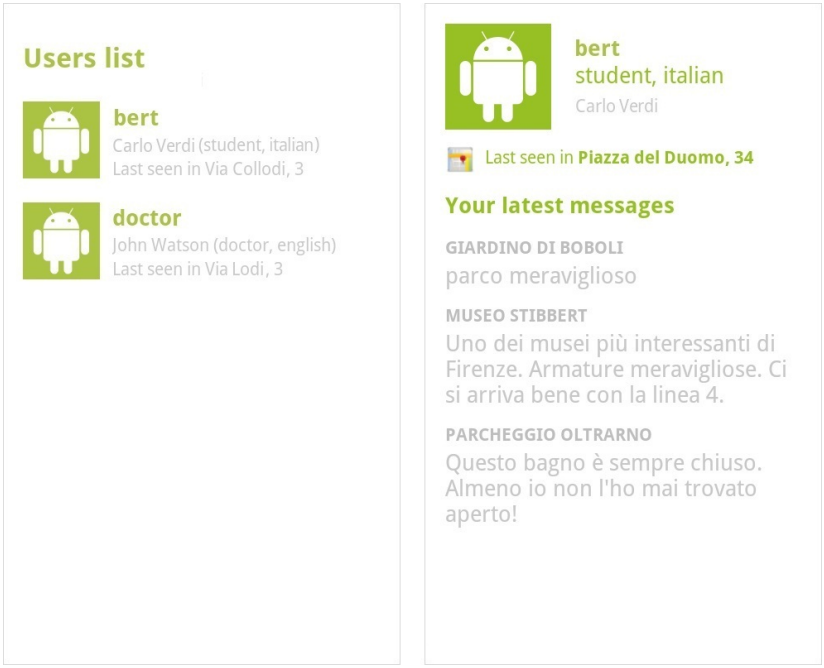

Figura 8.7: La lista degli utenti registrati ed il profilo di un utente.

di aggregare un numero indeterminato di nodi conformi al nodo modello *messaggio* e corrispondenti ai commenti lasciati dagli utenti.
Il nodo modello *utenti*, invece, aggrega un numero indeterminato di nodi modello {*utente*}, ognuno dei quali ha come figli i seguenti nodi modello:

- *info*; i nodi ad esso conformi memorizzano le informazioni relative ad un utente;
- *messaggi*; i nodi ad esso conformi aggregano un numero indeterminato di nodi contenenti i commenti inviati dall'utente;
- *posizione*; i nodi ad esso conformi memorizzano le coordinate della posizione attuale dell'utente.

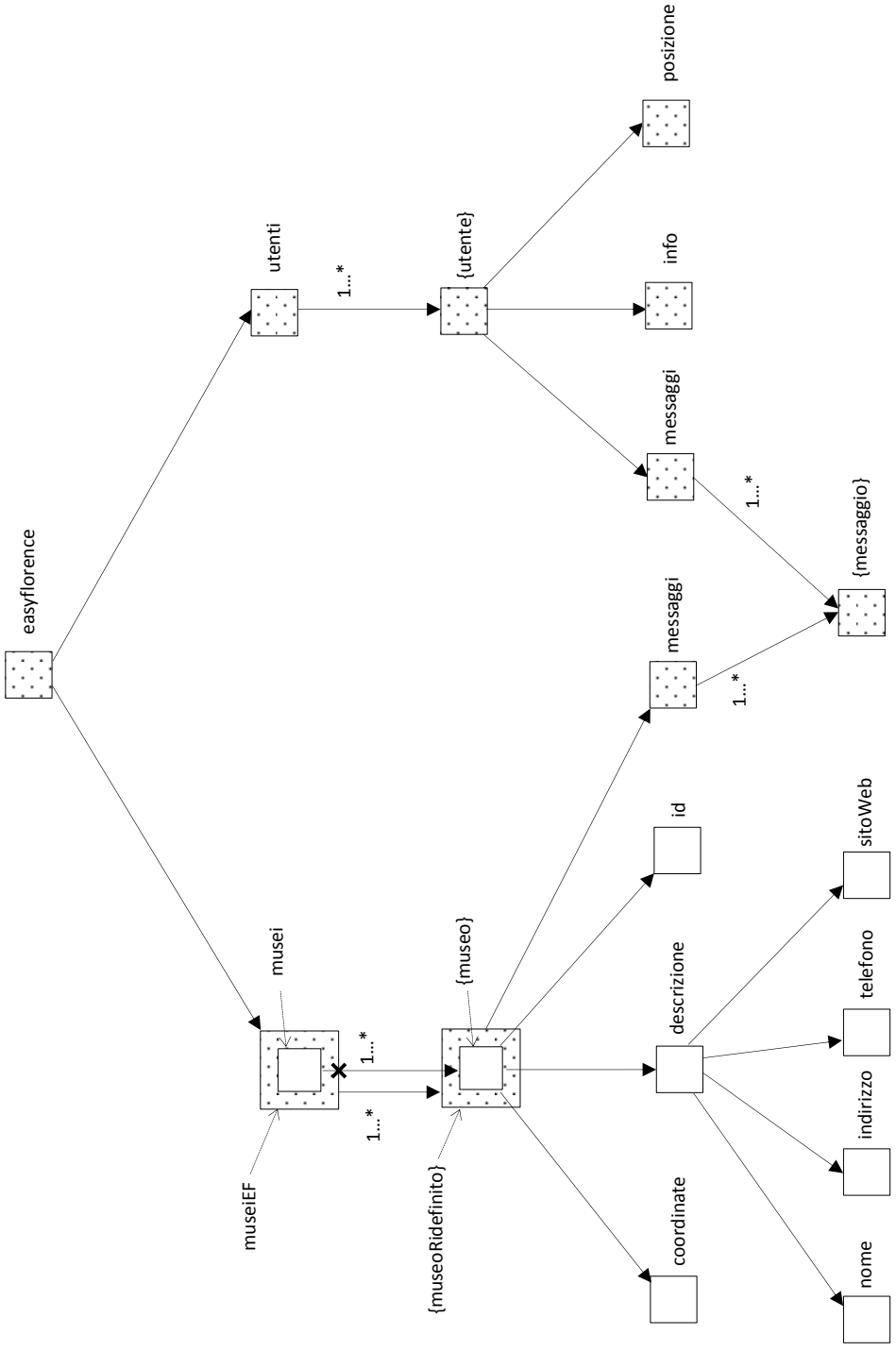

Figura 8.8: IDN Template dei documenti IDN di Easy Florence.

Conclusioni

All'interno del presente lavoro è stata introdotta una metodologia per la modellazione e l'arricchimento semantico di documenti in InterDataNet (o più brevemente IDN), un'architettura di tipo ROA (Resource Oriented Architecture) grazie alla quale diviene possibile progettare e realizzare una nuova generazione di *applicazioni collaborative su larga scala e nativamente interoperabili*, le cosiddette IDN Compliant Applications.
IDN si propone infatti come un'architettura innovativa finalizzata ad integrare ed estendere il Web, aggiungendo servizi per il supporto alla collaborazione. Inoltre IDN consente di rendere realmente "read-write" il Web, implementando le operazioni di "scrittura" mediante un approccio globalmente unificato, contrapponendosi così alle attuali soluzioni ad-hoc che gestiscono specifici problemi di dominio (come blog, wiki, social network, ecc. . .).
InterDataNet può essere considerato un "framework che consente ad utenti distribuiti nel tempo e nello spazio di collaborare intorno ad elementi informativi che appartengono ad uno spazio globale di dati con cui è possibile interagire mediante la metafora dei documenti" [Cio10]. Infatti, all'interno di IDN, l'informazione viene organizzata attraverso la metafora del documento costruendo entità informative più complesse a partire da quelle elementari, per rispondere alle necessità di una gestione collaborativa dell'informazione.
Il modello informativo di IDN (IDN-IM) è stato dettagliato in termini di requisiti, proprietà desiderabili, struttura del documento ed in maniera indipendente dallo specifico contesto applicativo e dalla tecnologia utilizzata.
IDN Service Architecture (IDN-SA) invece è l'architettura stratificata che adotta il paradigma REST (REpresentational State Transfer) e definisce le funzionalità di riferimento, i sottosistemi, i protocolli e le interfacce per la gestione collaborativa di un documento IDN-IM e fornisce i servizi a tutte le possibili IDN-Compliant Applications.
Il contributo specifico apportato dal presente lavoro consiste nella definizione di una tecnica di modellazione, denominata IDN Template, la quale fornisce ad un qualsiasi progettista di IDN Compliant Applications la possibilità di definire in maniera formale e rigorosa un *template*, ovvero un modello cui i documenti considerati devono risultare conformi.
La motivazione fondamentale che ha portato alla definizione di tale tecnica è la constatazione dell'impossibilità fattiva di inferire un template mediante

l'analisi di tutti i documenti possibili.

Tale tecnica consente innanzitutto di definire vincoli riguardanti la struttura, indicando l'insieme di nodi che possono risultare presenti in un documento IDN-IM conforme al template definito (consentendo tra l'altro di impostare alcuni nodi come opzionali o molteplici).

Inoltre diviene possibile stabilire importanti regole riguardo al formato, alla sintassi e alla semantica dei dati gestiti dai suddetti nodi, fornendo così al tempo stesso la possibilità di validarli e di conoscerne il significato.

In particolare, gli aspetti relativi alla struttura del documento, al formato e alla sintassi dei dati risultano avere una fondamentale importanza nella fase di testing di una IDN Compliant Application, al fine di verificare se i documenti IDN-IM da essa prodotti sono effettivamente conformi alle attese.

La tecnica di IDN Template proposta nel presente lavoro si pone nella direzione di una netta *separation of concerns* tra la sintassi e la semantica dei dati, ritenendo più opportuno lavorare a livelli diversi, al fine di distinguere tali aspetti.

Il lato semantico può rivelarsi di particolare importanza nel momento in cui si intenda riconciliare due (o più) dataset di origini diverse al fine di ottenere una terza informazione (da essa derivata, ma diversa), utilizzando le tecnologie definite dal Semantic Web (RDF, OWL e SPARQL) ed appositi *reasoners*. Occorre comunque tenere in considerazione che l'operazione di riconciliazione non può prescindere dall'intervento umano, il quale si rivela certamente fondamentale per la definizione delle equivalenze di concetti semantici.

Infine, all'interno del presente lavoro è stato introdotto un meccanismo per il riuso dell'IDN Template, che consente al progettista di realizzare applicazioni che effettuano l'uso di dati preesistenti, senza dovere provvedere alla duplicazione dell'informazione ed evitando quindi tutti i conseguenti problemi di incongruenza. Tale meccanismo risulta ovviamente utile quando la sua applicazione è economica; inoltre può essere preso in considerazione ed eventualmente messo in pratica quando, seppure porti con sé un costo sostenuto, fornisca tuttavia vantaggi per i quali sia ragionevole pagare tale costo, come ad esempio nel caso di un aumento di affidabilità dell'informazione. L'analisi svolta, sebbene preliminare, ha condotto a risultati per i quali risulta evidente la convenienza nei casi semplici, mentre non risulta chiara la convenienza nei casi complessi, se non attraverso una analisi *ad hoc* degli stessi.

Rispetto alle proposte del Read-Write Web, la soluzione qui presentata consente non solo il riuso di dati elementari, ma anche di quelli strutturati, rendendo così possibile riutilizzare, anche solo parzialmente, dati già precedentemente esistenti. Il presente lavoro ritiene infatti che il criterio "tutto o nessuno" applicato al riuso dei dati non si adatti adeguatamente alle reali necessità di progettazione.

Il percorso di ricerca, che ha condotto alle conclusioni sopra illustrate, è stato anche arricchito da altri risultati importanti e qui di seguito brevemente elencati:

- definizione di linee guida, strumenti, tecniche, metodologie e tecnologie con cui affrontare lo sviluppo del progetto;
- analisi e progettazione di tutti i sottosistemi, in particolare Virtual Resource, Information History e tutte le componenti del sistema dei nomi della IDN-Service Architecture;
- introduzione ed affinamento degli scenari applicativi necessari per dimostrare l'utilità del sistema in contesti reali;
- coordinamento di studenti per ricerche di tesi di laurea e realizzazione elaborati teorico-implementativi in particolare sui seguenti argomenti: IDN-Compliant Application per lo scenario riguardante il processo di Certificazione Anagrafica nella Pubblica Amministrazione, Information History, implementazione di IDN Compliant Applications di tipo mobile;
- divulgazione dei risultati, produzione di articoli scientifici e partecipazione a congressi internazionali.

La soluzione proposta dal presente lavoro è stata applicata alla definizione di applicazioni mobile per il riuso di alcuni dataset Open Data rilasciati dal Comune di Firenze, nell'ambito della propria iniziativa di Open eGovernment.
Infatti questo lavoro ha trovato origine in larga misura nella convinzione della presenza di limitazioni nei correnti approcci evolutivi dell'architettura del Web, ed in particolare per quanto riguarda il Web of Data e il Linked Data. I risultati conseguiti hanno rafforzato la convinzione che InterDataNet possa fornire una soluzione innovativa per un Web di dati strutturati e riusabili in modo più collaborativo, riuscendo tuttavia a mantenere la piena compatibilità con gli standard (RDF, OWL) proposti dal Linked Data e quindi in piena armonia con le attuali prospettive della Web Science, all'interno della quale si inserisce il presente lavoro.

Parte V

Appendici

Appendice A

InterDataNet Data Model

Il listato A.1 riporta l'XML Schema utilizzato dai documenti IDN-IM allo stato dell'arte [Cio10].

Listato A.1: XML Schema dei documenti IDN

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="VRDoc" type="idnVrDocument"/>

  <xs:complexType name="idnVrDocument">
    <xs:sequence>
      <xs:element name="VRDocInfo" type="VrDocInfoType" minOccurs="0"
        maxOccurs="1"/>
      <xs:element name="VRNodeEnvelope" type="idnVrNodeEnvelope" minOccurs=
        "0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="VrDocInfoType">
    <xs:all>
      <xs:element name="Errors" type="ErrorsType" minOccurs="0" maxOccurs="
        1"/>
      <xs:element name="Warnings" type="WarningsType" minOccurs="0"
        maxOccurs="1"/>
    </xs:all>
  </xs:complexType>

  <xs:complexType name="ErrorsType">
    <xs:sequence>
      <xs:element name="Error" type="IdnExceptionType" minOccurs="0"
        maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="WarningsType">
    <xs:sequence>
```

```
    <xs:element name="Warning" type="IdnExceptionType" minOccurs="0"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="IdnExceptionType">
  <xs:all>
    <xs:element name="Code" type="xs:string" minOccurs="1" maxOccurs="1"/
      >
    <xs:element name="Message" type="xs:string" minOccurs="1" maxOccurs="
      1"/>
    <xs:element name="TargetedLris" type="lriList" minOccurs="0"
      maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="lriList">
  <xs:sequence>
    <xs:element name="Lri" type="xs:string" minOccurs="1" maxOccurs="
      unbounded"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="idnVrNodeEnvelope">
  <xs:all>
    <xs:element name="VRNode" type="idnVrNode" minOccurs="1" maxOccurs="1
      " />
    <xs:element name="VRNodeInfo" type="idnVrNodeInfo" minOccurs="1"
      maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="idnVrNodeInfo">
  <xs:all>
    <xs:element name="IsRoot" type="xs:boolean" minOccurs="1" maxOccurs="
      1"/>
    <xs:element name="Etag" type="xs:string" minOccurs="0" maxOccurs="1"/
      >
    <xs:element name="HttpStatus" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="Keywords" type="xs:string" minOccurs="0" maxOccurs=
      "1"/>
  </xs:all>
</xs:complexType>

<xs:element name="VRNode" type="idnVrNode"/>

<xs:complexType name="idnVrNode">
  <xs:all>
    <xs:element name="VRApplicationData" type="vrApplicationData"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="VRApplicationMeta" type="vrApplicationMeta"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="VRIDNMeta" type="vrIdnMeta" minOccurs="0" maxOccurs
```

```
      ="1"/>
    <xs:element name="ManagementMeta" type="managementMeta" minOccurs="0"
       maxOccurs="1"/>
  </xs:all>
  <xs:attribute name="lri" type="xs:anyURI" use="required" />
</xs:complexType>

<xs:simpleType name="vrApplicationData">
  <xs:restriction base="xs:base64Binary"/>
</xs:simpleType>

<xs:complexType name="vrApplicationMeta">
  <xs:all>
    <xs:element name="IDNApplicationID" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="IDNAuthor" type="author" minOccurs="0" maxOccurs="1
      "/>
    <xs:element name="IDNModifiedOn" type="xs:string" minOccurs="0"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="vrIdnMeta">
  <xs:all>
    <xs:element name="AggregationMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="AggregationLink" type="link" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="BackLinkMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="BackLink" type="link" minOccurs="0" maxOccurs=
            "unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="IncomingChangeMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="IncomingChangeLink" type="link" minOccurs="0"
            maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="NodeLocalName" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="ReferenceMeta" minOccurs="0" maxOccurs="1">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="ReferenceLink" type="link" minOccurs="0"
            maxOccurs="unbounded"/>
```

```
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>

<xs:complexType name="managementMeta">
  <xs:all>
    <xs:element name="LicensePolicy" type="licensePolicy" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="PrivacyPolicy" type="privacyPolicy" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="ProvenanceManagement" type="provenanceManagement"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="ReplicationPolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="SecurityPolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="TimePolicy" type="timePolicy" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="VersioningPolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="author">
  <xs:all>
    <xs:element name="IDNApplicationInstanceID" type="xs:string"
      minOccurs="0" maxOccurs="1"/>
    <xs:element name="IDNHostIPAddress" type="xs:string" minOccurs="0"/>
    <xs:element name="IDNHostName" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="IDNUser" type="xs:string" minOccurs="0" maxOccurs="
      1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="link">
  <xs:all>
    <xs:element name="LocalName" type="xs:string" minOccurs="0" maxOccurs
      ="1"/>
    <xs:element name="Value" type="xs:anyURI" minOccurs="0" maxOccurs="1"
      />
    <xs:element name="Meta" type="xs:string" minOccurs="0" maxOccurs="1"/
      >
  </xs:all>
  <xs:attribute name="vr-id" type="xs:int" use="required" />
</xs:complexType>

<xs:complexType name="licensePolicy">
  <xs:all>
    <xs:element name="DataLicensing" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
```

```
          <xs:element name="DataLicense" type="licenseType" minOccurs="0"
            maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="NodeLicensing" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="NodeLicense" type="licenseType" minOccurs="0"
            maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:all>
</xs:complexType>

<xs:complexType name="licenseType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="vr-id" type="xs:int" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>

<xs:complexType name="privacyPolicy">
  <xs:all>
    <xs:element name="DataPrivacy" type="privacy" minOccurs="0" maxOccurs
      ="1"/>
    <xs:element name="NodePrivacy" type="privacy" minOccurs="0" maxOccurs
      ="1"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="privacy">
  <xs:all>
    <xs:element name="PrimaryUse" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
    <xs:element name="Retention" type="xs:string" minOccurs="0"/>
    <xs:element name="SecondaryUse" type="xs:string" minOccurs="0"/>
    <xs:element name="Sharing" type="xs:string" minOccurs="0"/>
  </xs:all>
</xs:complexType>

<xs:complexType name="provenanceManagement">
  <xs:all>
    <xs:element name="IDNPreviousApplicationID" type="xs:string"
      minOccurs="0"/>
    <xs:element name="IDNPreviousAuthor" type="author" minOccurs="0"/>
    <xs:element name="IDNPreviousModifiedOn" type="xs:string" minOccurs="
      0"/>
    <xs:element name="ProvenancePolicy" type="xs:string" minOccurs="0"
      maxOccurs="1"/>
  </xs:all>
</xs:complexType>
```

```
  <xs:complexType name="timePolicy">
    <xs:all>
      <xs:element name="AutoDestroyOn" type="xs:string" minOccurs="0"/>
      <xs:element name="ValidityNotAfter" type="xs:string" minOccurs="0"/>
      <xs:element name="ValidityNotBefore" type="xs:string" minOccurs="0"/>
    </xs:all>
  </xs:complexType>

</xs:schema>
```

Bibliografia

[ABCM99] Ulf Asklund, Lars Bendix, Henrik Christensen e Boris Magnusson: *The Unified Extensional Versioning Model*. Nel *System Configuration Management*, Lecture Notes in Computer Science – 1675, pagine 100–122. Springer, Berlino, Germania, 1999, ISBN 978-3-540-66484-0.

[AIS+77] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King e Shlomo Angel: *A Pattern Language: Towns, Buildings, Construction*, volume 2. Oxford University Press, New York, NY, USA, 1977, ISBN 978-0-19-501919-3.

[Alm92] Philip Almquist: *Type of Service in the Internet Protocol Suite*. RFC 1349 (Proposed Standard), luglio 1992. `http://www.ietf.org/rfc/rfc1349.txt`, Obsoleted by RFC 2474.

[Ass11] Associazione italiana per l'Open Government: *Data Gov IT*. `http://www.datagov.it`, 2011.

[Aus11] Australian Government: *DataGov Australia*. `http://data.gov.au`, 2011.

[Aye06] Danny Ayers: *The Shortest Path to the Future Web*. Internet Computing, IEEE, 10(6):76–79, 2006.

[AZ05] Paris Avgeriou e Uwe Zdun: *Architectural Patterns Revisited - A Pattern Language*. Nel *Proceedings of 10th European Conference on Pattern Languages of Programs (EuroPlop 2005)*, Irsee, Germania, 5-9 luglio 2005.

[Bar07] Maddalena Barlotti: *Progetto dell'interfaccia di Replica Management in InterDataNet per il Web of Data.* Tesi di Laurea, Facoltà di Ingegneria, Università degli Studi di Firenze, Firenze, Italia, A.A. 2006/2007.

[BCH07] Chris Bizer, Richard Cyganiak e Tom Heath: *How to Publish Linked Data on the Web*. `http://www4.wiwiss.fu-berlin.de/bizer/pub/LinkedDataTutorial/`, luglio 2007.

[BDD+08] Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub e Christopher Schmidt: *The GeoJSON Format Specification*. `http://www.geojson.org/geojson-spec.html`, giugno 2008.

[BF92] Nathaniel S. Borenstein e Ned Freed: *MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies*. RFC 1341 (Proposed Standard), giugno 1992. `http://www.ietf.org/rfc/rfc1341.txt`, Obsoleted by RFC 1521.

[BGM04] Dan Brickley, Ramanathan V. Guha e Brian McBride: *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation, W3C, febbraio 2004. `http://www.w3.org/TR/rdf-schema/`.

[BL89] Tim Berners-Lee: *Information Management: A Proposal*. `http://www.w3.org/History/1989/proposal.html`, 1989.

[BL06] Tim Berners-Lee: *Linked Data - Design Issues.* `http://www.w3.org/DesignIssues/LinkedData.html`, 2006.

[BL09] Tim Berners-Lee: *Putting Government Data online - Design Issues.* `http://www.w3.org/DesignIssues/GovData.html`, giugno 2009.

[BLFM05] Tim Berners-Lee, Roy T. Fielding e Larry Masinter: *Uniform Resource Identifier (URI): Generic Syntax.* RFC 3986 (Internet Standard), gennaio 2005. `http://www.ietf.org/rfc/rfc3986.txt`.

[BLHL01] Tim Berners-Lee, James Hendler e Ora Lassila: *The Semantic Web.* Scientific American, 284(5):28–37, 2001.

[BM04a] Dave Beckett e Brian McBride: *RDF/XML Syntax Specification (Revised).* W3C Recommendation, W3C, febbraio 2004. `http://www.w3.org/TR/REC-rdf-syntax/`.

[BM04b] Paul V. Biron e Ashok Malhotra: *XML Schema Part 2: Datatypes Second Edition.* W3C Recommendation, W3C, ottobre 2004. `http://www.w3.org/TR/xmlschema-2/`.

[BM12] Dave Beckett e Brian McBride: *OWL 2 Web Ontology Language Document Overview (Second Edition).* W3C Recommendation, W3C, dicembre 2012. `http://www.w3.org/TR/owl2-overview/`.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad e Michael Stal: *Pattern-Oriented Software Architecture: A System of Patterns*, volume 1. John Wiley & Sons, 1996, ISBN 978-0-471-95869-7.

[Buc97] Michael K. Buckland: *What Is a "Document"?* Journal of the American Society of Information Science, 48(9):804–809, 1997.

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith e Sanjiva Weerawarana: *Web Services Description Language (WSDL) 1.1.* W3C Note, W3C, marzo 2001. `http://www.w3.org/TR/wsdl`.

[Chi05] Davide Chini: *Progetto di un modello dell'informazione versionata e di un'architettura di rete finalizzati al lavoro collaborativo.* Tesi di Laurea, Facoltà di Ingegneria, Università degli Studi di Firenze, Firenze, Italia, A.A. 2004/2005.

[Chi07] Eran Chinthaka: *Enable REST with Web Services, Part 1: REST and Web Services in WSDL 2.0.* `http://www.ibm.com/developerworks/webservices/library/ws-rest1/`, maggio 2007.

[Chi09] Davide Chini: *InterDataNet: una soluzione infrastrutturale per il Read-Write Web of Data – Proof of concept dell'architettura.* Tesi di Dottorato di ricerca in Telematica e Società dell'informazione – XXII Ciclo, Università degli Studi di Firenze, Firenze, Italia, 2009.

[Cio10] Lucia Ciofi: *Future Internet: evoluzione in chiave REST per InterDataNet.* Tesi di Dottorato di ricerca in Telematica e Società dell'informazione – XXIII Ciclo, Università degli Studi di Firenze, Firenze, Italia, 2010.

[Com11] Comune di Firenze: *I 100 luoghi.* `http://www.comune.fi.it/opencms/opencms/citta_firenze/100luoghi.html`, 2011.

[Com12a] Comune di Firenze: *Città di Firenze.* `http://www.comune.fi.it`, 2012.

[Com12b] Comune di Firenze: *Open Data, I dati aperti del Comune di Firenze.* `http://opendata.comune.fi.it`, 2012.

[Cre07a] Creative Commons: *About The Licenses.* `http://creativecommons.org/licenses/`, 2007.

[Cre07b] Creative Commons: *Attribution-ShareAlike 3.0 Unported*. `http://creativecommons.org/licenses/by-sa/3.0/`, 2007.

[Cro06] Douglas Crockford: *The application/json Media Type for JavaScript Object Notation (JSON)*. RFC 4627 (Informational), luglio 2006. `http://www.ietf.org/rfc/rfc4627.txt`.

[CSFP04] Ben Collins-Sussman, Brian W. Fitzpatrick e C. Michael Pilato: *Version Control with Subversion*. O'Reilly Media, Inc., Sebastopol, CA, USA, 2004, ISBN 978-0-596-00448-4.

[CT04] John Cowan e Richard Tobin: *XML Information Set (Second Edition)*. W3C Recommendation, W3C, febbraio 2004. `http://www.w3.org/TR/xml-infoset/`.

[Dan97] Ron Daniel: *A Trivial Convention for using HTTP in URN Resolution*. RFC 2169 (Experimental), giugno 1997. `http://www.ietf.org/rfc/rfc2169.txt`.

[Die06] Reinhard Diestel: *Graph Theory*. Graduate Text in Mathematics – 173. Springer-Verlag, Berlino, Germania, 3 edizione, 2006, ISBN 978-3-540-26183-4.

[Dir10] Direktoratet for forvaltning og IKT: *Data Norge*. `http://data.norge.no`, 2010.

[DvGIF02] Leslie L. Daigle, Dirk Willem van Gulik, Renato Iannella e Patrick Faltstrom: *Uniform Resource Names (URN) Namespace Definition Mechanisms*. RFC 3406 (Best Current Practice), ottobre 2002. `http://www.ietf.org/rfc/rfc3406.txt`.

[EM04] Csaba Egyhazy e Raj Mukherji: *Interoperability architecture using RM-ODP*. Communications of the ACM, 47(2):93–97, 2004.

[ER11] Regione Emilia-Romagna: *Open Data Emilia-Romagna*. `http://www.dati.emilia-romagna.it`, 2011.

[Erl06] Thomas Erl: *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2006, ISBN 978-0-13-185858-9.

[ESR98] ESRI: *ESRI Shapefile Technical Description*. `http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`, luglio 1998.

[Eur11] European Union: *Press Conference on Open Data Strategy Brussels*. `http://europa.eu/rapid/press-release_SPEECH-11-872_en.htm`, dicembre 2011.

[Exe09] Executive Office of the President of the U.S.A.: *Open Government Directive*. `http://www.whitehouse.gov/omb/assets/memoranda_2010/m10-06.pdf`, dicembre 2009.

[FB96a] Ned Freed e Nathaniel S. Borenstein: *Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*. RFC 2049 (Draft Standard), novembre 1996. `http://www.ietf.org/rfc/rfc2049.txt`.

[FB96b] Ned Freed e Nathaniel S. Borenstein: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. RFC 2045 (Draft Standard), novembre 1996. `http://www.ietf.org/rfc/rfc2045.txt`, Updated by RFCs 2184, 2231, 5335, 6532.

[FB96c] Ned Freed e Nathaniel S. Borenstein: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. RFC 2046 (Draft Standard), novembre 1996. `http://www.ietf.org/rfc/rfc2046.txt`, Updated by RFCs 2646, 3798, 5147, 6657.

[FGM+99] Roy T. Fielding, James Gettys, Jeffrey C. Mogul, Henrik Frystyk Frystyk, Larry Masinter, Paul J. Leach e Tim Berners-Lee: *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616 (Draft Standard), giugno 1999. `http://www.ietf.org/rfc/rfc2616.txt`, Updated by RFCs 2817, 5785, 6266, 6585.

[Fie00] Roy T. Fielding: *Architectural Styles and the Design of Network-based Software Architectures*. Tesi di Dottorato di ricerca, University of California, Irvine, CA, USA, 2000.

[FK05a] Ned Freed e John C. Klensin: *Media Type Specifications and Registration Procedures*. RFC 4288 (Best Current Practice), dicembre 2005. `http://www.ietf.org/rfc/rfc4288.txt`.

[FK05b] Ned Freed e John C. Klensin: *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*. RFC 4289 (Best Current Practice), dicembre 2005. `http://www.ietf.org/rfc/rfc4289.txt`.

[FOR11] FORMEZ. Centro servizi, assistenza, studi e formazione per l'ammodernamento delle Pubbliche Amministrazioni: *Italian Open Data License v1.0*. `http://www.formez.it/iodl/`, maggio 2011.

[FOR12] FORMEZ. Centro servizi, assistenza, studi e formazione per l'ammodernamento delle Pubbliche Amministrazioni: *Italian Open Data License v2.0*. `http://www.dati.gov.it/iodl/2.0/`, marzo 2012.

[Fre12] Free Software Foundation: *Definizione di Software Libero*. `http://www.gnu.org/philosophy/free-sw.it.html`, 1996, 2012.

[FRF+03] Martin Fowler, David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee e Randy Stafford: *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2003, ISBN 978-0-321-12742-6.

[FW04] David C. Fallside e Priscilla Walmsley: *XML Schema Part 0: Primer Second Edition*. W3C Recommendation, W3C, ottobre 2004. `http://www.w3.org/TR/xmlschema-0/`.

[Gar12] Gartner Research: *Market Share: Mobile Phones by Region and Country, 3Q12*. `http://www.gartner.com/resId=2236115`, novembre 2012.

[GFH+12] Joe Gregorio, Roy T. Fielding, Marc Hadley, Mark Nottingham e David Orchard: *URI Template*. RFC 6570 (Proposed Standard), marzo 2012. `http://www.ietf.org/rfc/rfc6570.txt`.

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides: *Design Patterns – Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, USA, 1995, ISBN 978-0-201-63361-0.

[GHM+07] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean Jacques Moreau, Henrik Frystyk Nielsen, Anish Karmarkar e Yves Lafon: *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*. W3C Recommendation, W3C, aprile 2007. `http://www.w3.org/TR/soap12-part1/`.

[GKT02] Sven Graupner, Vadim Kotov e Holger Trinks: *Resource-Sharing and Service Deployment in Virtual Data Centers*. Nel *Proceedings of the 22nd International Conference onDistributed Computing Systems Workshops (ICDCSW '02)*, pagine 666–674, Vienna, Austria, 2-5 luglio 2002.

[Goo12a] Google: *Android Web Site*. `http://www.android.com`, 2012.

[Goo12b] Google Developers: *Google Maps API*. `https://developers.google.com/maps/`, 2012.

[Goo12c] Google Developers: *KML Documentation*. `https://developers.google.com/kml/documentation/`, febbraio 2012.

[Goo12d] Google Developers: *KML Tutorial*. `https://developers.google.com/kml/documentation/kml_tut`, febbraio 2012.

[Goo12e] Google Developers: *KMZ Files*. `https://developers.google.com/kml/documentation/kmzarchives`, dicembre 2012.

[Gou11] Gouvernement Francais: *Données publiques*. `http://www.data.gouv.fr`, 2011.

[Gov09a] Government of the United Kingdom: *DataGovUK*. `http://data.gov.uk`, 2009.

[Gov09b] Government of the United States of America: *DataGov*. `http://www.data.gov`, 2009.

[Gov11a] Governement of Canada – Gourvernment du Canada: *DataGov Canada*. `http://www.data.gc.ca`, 2011.

[Gov11b] Governo Italiano, Ministro per la pubblica amministrazione e la semplificazione: *I dati aperti della Pubblica Amministrazione*. `http://www.dati.gov.it`, 2011.

[Gov12] Governo Italiano. Ministero per la pubblica amministrazione e la semplificazione: *Infografica sugli OpenData italiani*. `http://www.dati.gov.it/content/infografica`, 2012.

[Goy09] Jan Goyvaerts: *Regular expressions tutorial*. `http://www.regular-expressions.info`, giugno 2009.

[Gra11] Graham Vickery Information Economics: *Review of Recent Studies on PSI Re-use and Related Market Developments*. `http://ec.europa.eu/information_society/policy/psi/docs/pdfs/report/final_version_study_psi.docx`, 2011.

[GRS10] Karthik Gomadam, Ajith Ranabahu e Amit Sheth: *SA-REST: Semantic Annotation of Web Resources*. W3C Member Submission, W3C, aprile 2010. `http://www.w3.org/Submission/SA-REST/`.

[GSMT+12] Shudi (Sandy) Gao, C. Michael Sperberg-McQueen, Henry S. Thompson, Noah Mendelsohn, David Beech e Murray Maloney: *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures*. W3C Recommendation, W3C, aprile 2012. `http://www.w3.org/TR/xmlschema11-1/`.

[Han07] Mark D. Hansen: *SOA Using Java Web Services*. Prentice Hall, 2007, ISBN 978-0-13-044968-9.

[Hea08] Tom Heath: *How Will We Interact with the Web of Data?* Internet Computing, IEEE, 12(5):88–91, 2008.

[HHM+10] Harry Halpin, Patrick J. Hayes, James P. McCusker, Deborah L. McGuinness e Henry S. Thompson: *When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data*. Nel *The Semantic Web – ISWC 2010*, Lecture Notes in Computer Science – 6496, pagine 305–320. Springer, Berlino, Germania, 2010, ISBN 978-3-642-17745-3.

[HKO+09] Michael Hausenblas, Philipp Kärger, Daniel Olmedilla, Alexandre Passant e Axel Polleres: *Preface*. Nel *Proceedings of the 1st Workshop on Trust and Privacy on the Social and Semantic Web (SPOT2009)*, pagine I–II, Heraklion, Grecia, 1 giugno 2009.

[Hof05a] Paul Hoffman: *The gopher URI Scheme*. RFC 4266 (Proposed Standard), novembre 2005. `http://www.ietf.org/rfc/rfc4266.txt`.

[Hof05b] Paul Hoffman: *The telnet URI Scheme*. RFC 4248 (Proposed Standard), ottobre 2005. `http://www.ietf.org/rfc/rfc4248.txt`.

[HSGT94] Frank Halasz, Mayer Schwartz, Kaj Grønbæk e Randall H. Trigg: *The Dexter Hypertext Reference Model*. Communications of the ACM, 37(2):30–39, 1994.

[HW01] Juha Hakala e Hartmut Walravens: *Using International Standard Book Numbers as Uniform Resource Names*. RFC 3187 (Informational), ottobre 2001. `http://www.ietf.org/rfc/rfc3187.txt`.

[IAN12] IANA: *MIME Media Types*. `http://www.iana.org/assignments/media-types/index.html`, 2012.

[Inn04] Samuele Innocenti: *Modello dell'informazione per documenti distribuiti e delocalizzati a supporto della cooperazione applicativa nelle Pubbliche Amministrazioni*. Tesi di Laurea, Facoltà di Ingegneria, Università degli Studi di Firenze, Firenze, Italia, A.A. 2003/2004.

[Inn08] Samuele Innocenti: *InterDataNet: nuove frontiere per l'integrazione e l'elaborazione dei dati – Visione e progettazione di un modello infrastrutturale per l'interdataworking*. Tesi di Dottorato di ricerca in Telematica e Società dell'informazione – XX Ciclo, Università degli Studi di Firenze, Firenze, Italia, 2008.

[ISO86] ISO: *Information processing – Text and office systems – Standard Generalized Markup Language (SGML)*. Norma ISO 8879:1986, International Organization for Standardization, Ginevra, Svizzera, 1986.

[ISO04] ISO: *Geographic information – Simple feature access – Part 1: Common architecture*. Norma ISO 19125-1:2004, International Organization for Standardization, Ginevra, Svizzera, 2004.

[ISO07] ISO: *Geographic information – Geography Markup Language (GML)*. Norma ISO 19136:2007, International Organization for Standardization, Ginevra, Svizzera, 2007.

[ISO11] ISO: *Information technology - Database languages - SQL multimedia and application packages - Part 3: Spatials*. Norma ISO/IEC 13249-3:2011, International Organization for Standardization, Ginevra, Svizzera, 2011.

[JBR96] Ivar Jacobson, Grady Booch e James Rumbaugh: *The Unified Modeling Language*. University Video Communications, 1996.

[Jos07] Nicolai M. Josuttis: *SOA in Practice: The Art of Distributed System Design*. O-Reilly Media, Inc., Sebastopol, CA, USA, 2007.

[JRLH99] Ian Jacobs, David Raggett e Arnaud Le Hors: *HTML 4.01 Specification*. W3C Recommendation, W3C, dicembre 1999. `http://www.w3.org/TR/html401`.

[KC04] Graham Klyne e Jeremy J. Carroll: *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation, W3C, febbraio 2004. `http://www.w3.org/TR/rdf-concepts/`.

[Kle08] John C. Klensin: *Simple Mail Transfer Protocol*. RFC 5321 (Draft Standard), ottobre 2008. `http://www.ietf.org/rfc/rfc5321.txt`.

[KR03] James F. Kurose e Keith W. Ross: *Computer Networking – A Top-Down Approach Featuring the Internet*. Addison-Wesley, Reading, MA, USA, 2 edizione, 2003, ISBN 978-0-201-97699-1.

[KS06] Kireeti Kompella e George Swallow: *Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures*. RFC 4379 (Proposed Standard), febbraio 2006. `http://www.ietf.org/rfc/rfc4379.txt`, Updated by RFCs 5462, 6424, 6425, 6426.

[Lew07] Rhys Lewis: *Dereferencing HTTP URIs*. `http://www.w3.org/2001/tag/doc/httpRange-14/2007-05-31/HttpRange-14`, maggio 2007.

[LF07] Holger Lausen e Joel Farrell: *Semantic Annotation for WSDL and XML Schema*. W3C Recommendation, W3C, agosto 2007. `http://www.w3.org/TR/sawsdl/`.

[Lin12] LinkedOpenData.it: *Linked Open Data Italia*. `http://www.linkedopendata.it`, 2012.

[LPD98] Clifford Lynch, Cecilia Preston e Ron Daniel: *Using Existing Bibliographic Identifiers as Uniform Resource Names*. RFC 2288 (Informational), febbraio 1998. `http://www.ietf.org/rfc/rfc2288.txt`.

[LS99] Ora Lasilla e Ralph R. Swick: *Resource Description Framework (RDF) Model and Syntax Specification*. W3C Proposed Recommendation, W3C, gennaio 1999. `http://www.w3.org/TR/PR-rdf-syntax/`.

[Mac09] Richard MacManus: *ReadWriteWeb Interview With Tim Berners-Lee, Part 1: Linked Data*. `http://www.readwriteweb.com/archives/interview_with_tim_berners-lee_part_1.php`, luglio 2009.

[ME01] Jim Melton e Andrew Eisenberg: *SQL Multimedia and Application Packages (SQL/MM)*. ACM Sigmod Record, 30(4):97–102, 2001.

[Mea02] Michael Mealling: *Dynamic Delegation Discovery System (DDDS) Part One: The Comprehensive DDDS*. RFC 3401 (Informational), ottobre 2002. `http://www.ietf.org/rfc/rfc3401.txt`.

[MEP06] MEPSIR (Measuring European Public Sector Information Resources: *Final Report of Study on Exploitation of public sector information – Benchmarking of EU framework conditions – Executive summary*. `http://ec.europa.eu/information_society/policy/psi/docs/pdfs/mepsir/executive_summary.pdf`, giugno 2006.

[Mic11] Microformats Community: *poshformats*. `http://microformats.org/wiki/poshformats`, febbraio 2011.

[MLM+06] C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F. Brown e Rebekah Metz: *Reference Model for Service Oriented Architecture 1.0 – Committee Specification 1*. `https://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf`, agosto 2006.

[Moa97] Ryan Moats: *URN Syntax*. RFC 2141 (Proposed Standard), maggio 1997. `http://www.ietf.org/rfc/rfc2141.txt`.

[Moo96] Keith Moore: *MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*. RFC 2047 (Draft Standard), novembre 1996. `http://www.ietf.org/rfc/rfc2047.txt`, Updated by RFCs 2184, 2231.

[Nat03] Yefim V. Natis: *Service-Oriented Architecture Scenario*. `http://www.gartner.com/resources/114300/114358/114358.pdf`, aprile 2003.

[NL04] Eric Newcomer e Greg Lomow: *Understanding SOA with Web Services (independent technology guides)*. Addison-Wesley Professional, 2004, ISBN 978-0-321-18086-5.

[Obj] Object Management Group: *Unified Modeling Language*. `http://www.uml.org`.

[Ope06] Open Source Initiative: *The Open Source Definition (Annotated)*. `http://opensource.org/osd-annotated`, 2006.

[Ope08a] Open Geospatial Consortium: *KML Open Standard Specification*. `http://www.opengeospatial.org/standards/kml/`, aprile 2008.

[Ope08b] Open Geospatial Consortium: *KML Open Standard XML Schema*. `http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd`, gennaio 2008.

[Ope08c] Open Geospatial Consortium: *OGC Approves KML as Open Standard*. `http://www.opengeospatial.org/pressroom/pressreleases/857`, aprile 2008.

[Ope09a] Open Data Commons: *Open Data Commons Licenses*. `http://opendatacommons.org/licenses/`, 2009.

[Ope09b] Open Data Commons: *Open Data Commons Open Database License (ODbL)*. `http://opendatacommons.org/licenses/odbl/`, 2009.

[Ope09c] Open Definition Advisory Council: *Open Definition*. `http://opendefinition.org/`, 2009.

[Ope09d] Open Definition Advisory Council: *Open Definition*. `http://opendefinition.org/okd/italiano/`, 2009.

[Ope10] Open Source Initiative: *Where Does My Money Go?* `http://wheredoesmymoneygo.org/`, 2010.

[Ope11a] Open Geospatial Consortium: *Simple Feature Access - Part 1: Common Architecture*. `http://www.opengeospatial.org/standards/sfa`, maggio 2011.

[Ope11b] Open Knowledge Foundation: *CKAN Italia*. `http://it.ckan.net`, 2011.

[Ope12] Open Geo Data Italia: *Open Geo Data Italia*. `http://www.opengeodata.it`, 2012.

[Pao06] Michela Paolucci: *Una soluzione innovativa per la modellazione e gestione distribuita di documenti giuridici in rete*. Tesi di Laurea, Facoltà di Ingegneria, Università degli Studi di Firenze, Firenze, Italia, A.A. 2005/2006.

[PGM+12] David Peterson, Shudi (Sandy) Gao, Ashok Malhotra, C. Michael Sperberg-McQueen, Henry S. Thompson e Paul V. Biron: *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. W3C Recommendation, W3C, aprile 2012. `http://www.w3.org/TR/xmlschema11-2/`.

[Pie10] Regione Piemonte: *Open Data Piemonte*. `http://www.dati.piemonte.it`, 2010.

[PPCP07] Maria Chiara Pettenati, David Parlanti, Davide Chini e Franco Pirri: *InterDataNet: An Infrastructural Approach to Data Interoperability to Enable Computer Supported Collaborative Applications*. Nel *Proceedings of the 3rd International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'07)*, pagine 130–137, Barcellona, Spagna, 28-30 novembre 2007.

[PPI+09] Franco Pirri, Maria Chiara Pettenati, Samuele Innocenti, Davide Chini e David Parlanti: *InterDataNet: an Infrastructural Approach for the Web of Data*. Nel *Proceedings of the WebSci'09: Society On-Line*, Atene, Grecia, 18-20 marzo 2009.

[PS03] Aiko Pras e Jürgen Schönwälder: *On the Difference between Information Models and Data Models*. RFC 3444 (Informational), gennaio 2003. `http://www.ietf.org/rfc/rfc3444.txt`.

[PS08] Eric Prud'hommeaux e Andy Seaborne: *SPARQL Query Language for RDF*. W3C Recommendation, W3C, gennaio 2008. `http://www.w3.org/TR/rdf-sparql-query/`.

[RMS97] Gail L. Rein, Daniel L. McCue e Judith A. Slein: *A Case for Document Management Functions on the Web*. Communications of the ACM, 40(9):81–89, 1997.

[RR07] Leonard Richardson e Sam Ruby: *RESTful Web Services*. O'Reilly Media, Sebastopol, CA, USA, 2007, ISBN 978-0-596-52926-0.

[Sax07] Michael Kay Saxonica: *XSL Transformations (XSLT) Version 2.0*. W3C Recommendation, W3C, gennaio 2007. `http://www.w3.org/TR/xslt20/`.

[SHBL06] Nigel Shadbolt, Wendy Hall e Tim Berners-Lee: *The Semantic Web Revisited*. Intelligent Systems, IEEE, 21(3):96–101, 2006.

[SM94] Karen Sollins e Larry Masinter: *Functional Requirements for Uniform Resource Names*. RFC 1737 (Informational), dicembre 1994. `http://www.ietf.org/rfc/rfc1737.txt`.

[Spa10] Spaghetti Open: *Spaghetti Opendata*. `http://www.spaghettiopendata.org/dati`, 2010.

[Sub04] Peter Suber: *Open Access Overview*. `http://www.earlham.edu/~peters/fos/overview.htm`, giugno 2004.

[TBMM04] Henry S. Thompson, David Beech, Murray Maloney e Noah Mendelsohn: *XML Schema Part 1: Structures Second Edition*. W3C Recommendation, W3C, ottobre 2004. `http://www.w3.org/TR/xmlschema-1/`.

[Uff12] Ufficio di Statistica Associato dell'area fiorentina: *Statistica dell'area fiorentina*. `http://statistica.fi.it`, 2012.

[Val00] Antonio Vallecillo: *RM-ODP: The ISO Reference Model for Open Distributed Processing*. DINTEL Edition on Software Engineering, (3):69–99, 2000.

[Vin07] Steve Vinoski: *REST Eye for the SOA Guy*. Internet Computing, IEEE, 11(1):82–84, 2007.

[VK08] Tomas Vitvar e Jacek Kopecký: *MicroWSMO Working Draft*. W3C Recommendation, W3C, febbraio 2008. `http://www.wsmo.org/TR/d38/v0.1/`.

[WCL+05] Sanjiva Weerawarana, Francisco Curbera, Frank Leymann, Tony Storey e Donald F. Ferguson: *Web Services Platform Architecture – SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall, 2005, ISBN 978-0-13-148874-8.

[WN01] Alain Wegmann e Andrey Naumenko: *Conceptual Modeling of Complex Systems Using an RM-ODP Based Ontology*. Nel *Proceedings of the 5th IEEE International Enterprise Distributed Object Computing Conference (EDOC'01)*, pagine 200–211, Seattle, WA, USA, 4-7 settembre 2001.

[Wor97] World Wide Web Consortium: *XML Core Working Group Public Page*. `http://www.w3.org/XML/Core/`, 1997.

[Wor07a] World Wide Web Consortium: *SAWSDL Namespace*. `http://www.w3.org/ns/sawsdl`, 2007.

[Wor07b] World Wide Web Consortium: *Semantic Annotation for WSDL Working Group*. `http://www.w3.org/2002/ws/sawsdl/`, settembre 2007.

[ZEWH95] Stuart H. Zweben, Stephen H. Edwards, Bruce W. Weide e Joseph E. Hollingsworth: *The Effects of Layering and Encapsulation on Software Development Cost and Quality*. IEEE Transactions on Software Engineering, 21(3):200–208, 1995.