Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Triennale in Informatica

Modelli per l'aggregazione di dati federati: il caso degli orari di apertura degli esercizi commerciali e enti pubblici

Tesi di Laurea in Tecnologie Web

Relatore:
Chiar.mo Prof.
Fabio Vitali

Presentata da:
Vincenzo Ferrari

Sessione II
Anno Accademico 2011/2012

Introduzione

Questa tesi tratta di scambio di dati aperti (*open data exchange*). In particolare, tratta di intervalli temporali con annesse informazioni di vario tipo.

Le attività commerciali, come aziende, bar, ristoranti, cinema, teatri, e via dicendo, hanno bisogno di un modo comune per poter condividere i dati inerenti ai propri orari (*timetable*). Lo scopo di questa tesi è quello di mostrare un modello efficiente, compatto e completo per poter condividere tali informazioni con l'ausilio di formati standard (*XML*).

Oggi giorno esistono diverse soluzioni atte a far fronte a questa necessità ma si tratta di soluzioni incomplete e che gestiscono tali informazioni separatamente.
Il problema sorto è quello di avere un'unica struttura che posso unificare informazioni di diverso carattere con gli orari di un'attività: un cinema ha la necessità di fornire orari di diverse proiezioni svolte in sale diverse, una tournée dovrà poter specificare le coordinate geografiche del prossimo evento e così discorrendo.

Nel primo capitolo verrà preso in considerazione il suddetto problema, approfondendo tutti gli aspetti e argomentando una serie di esempi reali: verranno poi illustrati alcuni formati standard atti a ovviare al problema dell'interscambio dei dati.
Gli esempi riportati riguardano aree e contesti differenti, come l'ambito farmaceutico, quello scolastico o quello cinematografico.

Nel secondo capitolo, invece, verranno prese in considerazioni le soluzioni già esistenti per la gestione e la condivisione di tabelle orario.
Si vedranno esempi come lo standard *RDFCalendar* [0] usato dalle più importanti applicazioni di calendario, il caso *ChefMoz* [1] un aggregatore online di ristoranti, l'applicazione *Unimaner* [2] col suo modello per gestire gli orari scolastici e infine *Google Transit* [3], la soluzione offerta dal gigante del web per l'interscambio di dati delle varie agenzie di trasporto di tutti i paesi.
Ogni esempio sarà condito con estratti di codice e di una valutazione finale per favorire la comprensione dei pro e dei contro che queste soluzioni offrono.

Dopo una panoramica nell'ambiente web, si sfocerà nella soluzione da noi proposta: un modello per l'aggregazione di dati federati.
Mostreremo come la nostra soluzione abbracci un'ampia gamma di utenza, come esercizi commerciali, enti pubblici, trasporti, fino ad arrivare al singolo individuo.
Dimostreremo come sia più compatta ed efficiente per poter condividere i dati attraverso un formato standard: per fare ciò verranno dati otto esempi, basati su attività reali, come un cinema multisala, l'orario scolastico di un istituto universitario, la programmazione di un teatro comunale, e via

dicendo.
Verranno spiegati in dettaglio gli elementi rivoluzionari della nostra proposta, come l'ereditarietà tra locazioni (*venue*), la localizzazione delle informazioni (*locale*) e il modello compatto di tabella orario (*hour – pattern*).
Infine, verrà mostrato un esempio completo di documento per rappresentare la soluzione.

L'applicazione presente nel quarto capitolo serve per validare la nostra tesi.
A tale scopo, è stata realizzata un'applicazione web in grado di creare e gestire le tabelle orario, sia quelle complesse che quelle semplici.
All'utente viene data la possibilità di visualizzare i suoi orari su un'interfaccia intuitiva e dinamica, tramite l'utilizzo delle più moderne tecnologie che il web possa offrire.
Grazie a questo strumento, l'utente può agevolmente stilare la tabella orario della propria attività e iscriverla ai registri online (*Registry*).
L'applicazione è formata principalmente dal gestore delle tabelle e dal *Registry*, un secondo modulo separato atto a mantenere la lista di tutte le attività commerciali ed enti pubblici che hanno iscritto la propria tabella orario precedentemente.

La tesi termine con le conclusioni.
In questo capitolo viene riepilogato il lavoro di ricerca, analisi e sviluppo realizzato, dopodiché viene fatta una stima con una serie di valutazioni positive e negative sul lavoro svolto.
Come ultimo passo viene dato uno sguardo al futuro sia della soluzione sia dei possibili utilizzi che se ne potranno fare.
Verrà presa in considerazione di stilare un'ontologia vera e propria con un linguaggio formale, di realizzare un aggregatore per sfruttare al meglio le capacità dei *Registry* e si tratterà anche di condire il tutto con il *Semantic Web*, facendo uso di vocabolari e ontologie esterne.

Indice generale

Capitolo 1

Il problema

Oggi giorno, se voglio conoscere gli orari di apertura di un negozio, o di una farmacia, o di una qualsiasi attività commerciale, apro il browser, cerco su *Google* il sito dell'esercizio interessato, vado alla pagina dei contatti e visualizzo una tabella orario.
Sono costretto a seguire la stessa procedura anche per un insieme di attività commerciali: infatti potrei aver bisogno di sapere tutte le programmazioni dei cinema che ci sono nell'area di Bologna questa sera.
La faccenda si complica se il dispositivo con cui accedo è *mobile*, come uno *smartphone*, perché potrei trovarmi davanti un sito non adatto per tale dispositivo o, peggio, potrei avere a che fare con un formato non supportato dal browser con cui sto navigando.
Tanto per fare un esempio, potrei trovarmi di fronte a tabelle orario stilate in diversi formati, come una pagina *HTML*, o un file *PDF*, oppure un semplice file testuale indentato in stile tabellare e, perché no, una bella immagine *JPEG*?

Ecco allora che inizia a definirsi il problema di come reperire le informazioni sugli orari di apertura e chiusura in maniera chiara, semplice, automatica.
Perché devo spendere un sacco di tempo e di energie nel cercare di recuperare alcuni dati quando potrei tranquillamente inserire alcune regole in un filtro di un programma *aggregatore* e ritrovarmi stampato a video esattamente ciò che stavo cercando?
Perché attualmente non esiste un formato aperto e compatibile che possa soddisfare questa esigenza. Almeno non fino ad ora.

Oggi giorno risulta ancora impossibile realizzare un'applicazione che prelevi i dati delle tabelle orario dei vari siti, li raggruppi, li filtri e li renda pronti per essere visualizzati con un certo *layout* dall'utente.

1.1 Casi reali

Ci sono diversi casi reali delle problematiche sopracitate: strutture proprie, formati inadatti, informazioni aggregate in maniera errata e via dicendo.
Seguiranno una serie di esempi tratti dal web che toccheranno differenti attività commerciali.

1.1.1 Federfarma

Partiamo col caso del sito di **Federfarma** Bologna [4], l'associazione provinciale delle farmacie. Alla sezione *Farmacie di Turno* [5] è possibile trovare gli orari delle farmacie di turno in un certo lasso di tempo.
In questo caso, Federfarma [4] fa da *aggregatore* di tabelle orario di tutte le farmacie della provincia e mette a disposizione i vari orari secondo una proprio struttura tramite file *PDF*.

Di seguito un esempio di tale struttura:

bologna
Febbraio 2012

dalle 8,30 del lunedì alle 8,30 del lunedì successivo, negli orari:

24 ore	8,30 - 12,30 / 15,30 - 19,30

FARMACIE APERTE SOLO IL SABATO IN AGGIUNTA AL TURNO SETTIMANALE: 8,30 - 12,30 / 15,30 - 19,30

06/02/2012 -13/02/2012

AL SACRO CUORE, via matteotti, 29 **S.LUCIA,** via battindarno, 139/c **DELLA PROVVIDENZA,** via massarenti, 254	**INTERNAZIONALE,** via indipendenza, 29 **COMUNALE REPUBBLICA,** via tomba, 29 **COMUNALE STENDHAL,** via stendhal, 5/a **S.MARIA DELLE GRAZIE,** via degli orti, 68/d-e **CASTIGLIONE,** via castiglione, 53 **LODI,** via a.costa, 47/a **DEL BORGO,** via m.e.lepido, 147	**DELLO STERLINO,** via a.murri, 16 **S.LORENZO,** via ugo bassi, 25 **BERTELLI ALLA FUNIVIA,** via porrettana, 95/f-g **DEL SOLE,** via pirandello, 22/a **BEATA VERGINE DI S.LUCA,** via m.d'azeglio, 15 **COMUNALE BARBIERI,** via f. barbieri, 121 **FOSSOLO 2 CENTRO COMMERC,** viale lincoln, 5

COMUNALE TRIUMVIRATO, S.VIOLA, BETTINI, PARCO NORD, PORTA LAME, DEI MILLE, S.SALVATORE, COMUNALE COSTA , SACCHETTI, DALLE DUE TORRI, SS.TRINITA', MORATELLO, IRNERIO, S.RITA, S.GIORGIO, COMUNALE FELSINA, COMUNALE BATTAGLIA, PONTEVECCHIO.

13/02/2012 - 20/02/2012

ANTICA FARMACIA DELLE MOLINE, via a.righi, 6/a **COMUNALE AZZURRA,** via azzurra, 52/a **DELLA BARCA,** p.za bonazzi, 9/b	**COMUNALE MARZABOTTO,** via marzabotto, 14 **GRIMALDI,** via di corticella, 184/3 **S.RUFFILLO,** via toscana, 58 **DEL PILASTRO,** via deledda, 26 **S.ISAIA,** via s.isaia, 2/a **DEL CORSO,** via s.stefano, 38 **DUE MADONNE,** via tacconi, 2/b **DELL'IMMACOLATA,** via bastia, 18 **DEL PORTO,** via marconi, 26	**DEL NAVILE,** via fioravanti, 26 **S.MAMOLO,** via s.mamolo, 25/b **SIEPELUNGA,** via borghi mamo, 6/b-c **COMUNALE EMILIA PONENTE,** via emilia ponente, 258/b **DEL PAVAGLIONE,** via archiginnasio, 2/a **VITTORIA,** via andreini, 32/m

LAVINO DI MEZZO, DE PISIS, COMUNALE DON STURZO, CARRACCI, S.CATERINA, COMUNALE FERRARESE, CROCE BIANCA, DELLA STAZIONE CENTRALE, TOSCHI, ALBERANI, FERRARI, S.SILVERIO DELLA CHIESA NUOVA, OBERDAN, AICARDI, S.ANTONIO, S.DONNINO, EMILIA.

Ogni pagina ha un orario di base e questo è applicato ad ogni farmacia definita nelle tabelle: ogni tabella riguarda un certo periodo di tempo. Inoltre, ad ogni pagina, è associata uno o più mesi, come Febbraio nell'esempio sopracitato.
Legate alle informazioni sull'orario e la durata vi sono anche informazioni ulteriori, come l'indirizzo della farmacia.

Sulla pagina web di Federfarma [4] è possibile notare che esistono più file *PDF* che differenziano le tipologie d'orario e le farmacie del territorio.

Il problema in questo caso è sia la struttura che il formato usato.
Il formato *PDF* non si presta bene per essere utilizzato dalle applicazioni che fanno aggregazione di dati, mentre la struttura è complessa e non immediata da comprendere per l'utente.
Inoltre non è possibile fare alcunché di interrogazione logica: se l'utente volesse recuperare soltanto certe farmacie o l'orario delle farmacie di un determinato lasso di tempo non potrebbe.

1.1.2 Cineflash

Cineflash [6] è un cinema multisala di Forlimpopoli.
Sul loro sito è possibile utilizzare un'applicazione *Flash* [7] per visualizzare la programmazione delle serate.
Cliccando su ogni locandina è possibile recuperare le informazioni del film e della sua proiezione.

Essendo un'applicazione realizzata in *Flash [7]* le opzioni sono limitate: l'utente potrà soltanto visualizzare le informazioni, senza alcuna possibilità d'interazione con esse.

In questo caso, non esistono **API** (*Application Programming Interface*) [8] pubbliche per poter recuperare i dati delle proiezioni e disporle quindi in modo differente, magari tramite una pagina *HTML*.
Questo rende impossibile alle applicazioni di aggregazione dati di recuperare le suddette informazioni.

Cineflash [6] è soltanto uno dei molti casi presenti sul web che utilizzano struttura e modelli chiusi e limitanti.

Flash [7] può essere usato per rappresentare il dato ma il gestore del servizio deve poter dare la possibilità di recuperare le informazioni dal proprio database in maniera alternativa, usando per l'appunto *API* [8] pubbliche.
Però, ciò che sta accadendo ormai da qualche anno, è la dismissione di questo tipo di formato per rappresentare il dato, preferendo una soluzione in *HTML5* [9] con l'uso di stili (*CSS*) [10] e di linguaggi dinamici (*Javascript*) [11]. Infine, le informazioni verrebbero ben separate dalla loro rappresentazione.

1.1.3 Informatica Unibo

La facoltà d'**Informatica** dell'*Università di Bologna* [12] mette a disposizione i suoi orari scolastici tramite tabelle *HTML* create al volo.

Orario Provvisorio Primo Anno Corso di Laurea in Informatica (codice 8009) - I Ciclo a.a. 2012/2013

N.B. L'orario pubblicato è da considerarsi provvisorio.

	Lunedì	Martedì	Mercoledì	Giovedì	Venerdì
8:30 - 9:00	Rambaldi Aula M1 - Mineralogia F	Laneve Aula M1 - Mineralogia P	Rambaldi Aula M1 - Mineralogia F	-	Rambaldi Aula M1 - Mineralogia F
9:00 - 9:30	F	P	F	-	F
9:30 - 10:00	F	P	F	-	F
10:00 - 10:30	F	P	F	-	F
10:30 - 11:00	Laneve Aula M1 - Mineralogia P	P	Ghini Aula M1 - Mineralogia AE	-	Sordoni - Mughetti Aula M1 - Mineralogia AM
11:00 - 11:30	P	P	AE	-	AM
11:30 - 12:00	P	Sordoni - Mughetti Aula M1 - Mineralogia AM	AE	-	AM
12:00 - 12:30	P	AM	AE	-	AM
12:30 - 13:00	P	AM	AE	-	AM
13:00 - 13:30	P	AM	AE	-	AM
13:30 - 14:00	-	-	-	Laneve Aula M1 - Mineralogia P	-
14:00 - 14:30	-	-	-	P	-
14:30 - 15:00	-	Dott. Luca Ferrari Aula M1 - Mineralogia Tutor OFA - AM	-	P	-
15:00 - 15:30	-	Tutor OFA - AM	-	P	-
15:30 - 16:00	-	Tutor OFA - AM	-	P	-
16:00 - 16:30	-	Tutor OFA - AM	-	P	-

Questa soluzione è migliore delle precedenti e, come già detto, questa è la strada che stanno intraprendendo molte aziende, enti pubblici e attività di vario genere.
Con lo svilupparsi di **CMS** (*Content Management System*) [13] sempre più semplici da usare e personalizzabili, è diventato ancor più facile creare pagine web di questo tipo invece di utilizzare formati più limitanti come *PDF* o *Flash* [7].

Il problema di questa soluzione è l'uso di una struttura comune per gestire le informazioni delle tabelle orario.
Un'applicazione di aggregazione non potrà adattarsi ad ogni struttura esistente, quindi nasce la

necessità di utilizzare formati aperti, standard, che diano la possibilità di definire modelli aperti e comuni per l'aggregazione di dati.

1.2 Formati aperti e standard

Per ovviare alle problematiche date dagli esempi sopracitati, vengono in nostro aiuto una serie di formati che permettono la realizzazione di modelli aperti, facili da usare, compatibili, standard e portabili su qualsiasi sistema.

1.2.1 XML

XML (*eXtensible Markup Language*) [14] è uno di questi formati.
Oggi giorno è ormai utilizzato negli ambienti più disparati, comodo, di grande comprensione per l'utilizzatore umano e semplice da interpretare anche per le applicazioni.
È facilmente personalizzabile ed è possibile utilizzarlo per creare strutture dinamiche. Segue un esempio:

```
<?xml version=”1.0” encoding=”utf-8”?>
<root>
	<record id=”1”>
		<elementA>text</elementA>
		<elementB>text</elementB>
	</record>
	<record id=”2”>
		<elementA>text</elementA>
		<elementB>text</elementB>
	</record>
	...
</root>
```

Un elemento radice (*root*) contiene tutto il corpo della struttura, formata ad altri elementi (*record*) che a loro volta sotto elementi (*elementA, elementB*): ciò che ne risulta è una struttura ad albero e proprio su tale concetto verrà sviluppata la proposta data in questa tesi.
Infine ogni elemento può contenere una serie di attributi (*id*) e un valore proprio (*text*).

Vi è una cosa importante da notare.
Stilare una struttura mirata più verso gli attributi che verso i sotto elementi ha lo stesso potere espressivo ma in alcuni casi può snellire il codice e rendere meno pesante il documento.
Di seguito una dimostrazione data dalla modifica dell'esempio precedente:

```
<?xml version=”1.0” encoding=”utf-8”?>
<root>
	<record id=”1” elementA=”text” elementB=”text”/>
	<record id=”2” elementA=”text” elementB=”text”/>
```

```
        ...
</root>
```

Il significato è sempre lo stesso: il *record* di *id 1* ha due elementi (*elementA, elementB*) aventi un certo valore (*text*).
Questo è importante da sapere siccome la nostra soluzione opterà per una sintassi orientata agli attributi piuttosto che agli elementi.

1.2.2 RDF

XML [14] è sicuramente utile per creare lo scheletro di quella struttura che diverrà di comune utilizzo tra le applicazioni di aggregazione dati.
La limitazione di tale formato sta nella semantica. Per noi umano può essere chiaro il significato di un certo elemento del documento XML [14] ma per un'applicazione non è la stessa cosa.
Ecco allora che entra in gioco il formato dati **RDF** (*Resource Description Framework*) [15].

RDF [15] è uno standard proposto dal **W3C** (*World Wide Web Consortium*) [16] per l'interoperabilità tra applicazioni che scambiano e condividono informazioni.
Questo formato viene preso in considerazione in questa tesi solo a scopo illustrativo, siccome si è ritenuto necessario concentrarsi più sulla realizzazione della struttura che della sua formalizzazione semantica.
Nonostante ciò, sarà uno dei punti affrontati nell'immediato futuro dello sviluppo di questa soluzione: infatti, tramite la *linearizzazione* XML [14] di RDF [15] è possibile specificare ulteriori informazioni all'interno della struttura proposta.

Di seguito un esempio [17]:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:foaf="http://xmlns.com/foaf/0.1/"
        xmlns:cm="http://chefmoz.org/rdf/elements/1.0/">


        <foaf:Document
rdf:about="http://www.whatsonbristol.co.uk/reviews/restaurants/casa_sudaca.html">
                <dc:title>What's On Bristol Guide - Reviews - Restaurants - Casa
Sudaca</dc:title>
                <dc:description>
                        Review: Imagine a place where the atmosphere is always warm, where the
people smile, the
                        music is upbeat, the colours are the rich oranges and pinks of a summer
sunset and the food is good
                        and nourishing. Does that appeal to you? Well then get yourself down to
Casa Sudaca at the bottom
                        of Zetland Road where you'll find a perfect little South American gem of a
restaurant.
                </dc:description>
                <foaf:topic>
```

```
                    <cm:Restaurant>
                        <dc:title>Casa Sudaca</dc:title>
                        <foaf:depiction
rdf:resource="http://www.whatsonbristol.co.uk/images/review/casa-2.jpg" />
                        <foaf:depiction
rdf:resource="http://www.whatsonbristol.co.uk/images/review/casa-1.jpg"o tr />
                    </cm:Restaurant>
                </foaf:topic>
        </foaf:Document>
</rdf:RDF>
```

L'esempio mostra l'uso di RDF [15] per rappresentare le informazioni inerente a un ristorante, con l'ausilio di ontologie quali *Dublin Core* [18], *FOAF* [19] e *ChefMoz* [1].

1.2.3 OWL

Ed è con le ontologie che chiudiamo il capitolo corrente.
Le ontologie sono una formalizzazione di una struttura proposta, esattamente come quella data in questa tesi.

Esistono diversi linguaggi atti a redigere correttamente le ontologie e uno di questi è **OWL** (*Web Ontology Language*) [20].
OWL [20] permette di facilitare l'interoperabilità di contenuti web tra le applicazioni, più di XML [14] e RDF [15], fornendo una serie di vocabolari in aggiunta alle informazioni presenti nella struttura dati.

Segue un esempio [21]:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:owl="http://www.w3.org/2002/07/owl#"
        xmlns="http://www.xfront.com/owl/ontologies/camera/#"
        xmlns:camera="http://www.xfront.com/owl/ontologies/camera/#"
        xml:base="http://www.xfront.com/owl/ontologies/camera/">
        <owl:Ontology rdf:about="">
                <rdfs:comment>
                        Camera OWL Ontology
                        Author: Roger L. Costello
                </rdfs:comment>
        </owl:Ontology>
        <owl:Class rdf:ID="Money">
                <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
        </owl:Class>
        <owl:DatatypeProperty rdf:ID="currency">
                <rdfs:domain rdf:resource="#Money"/>
                <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>
```

```
        <owl:Class rdf:ID="Range">
                <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
        </owl:Class>
        <owl:DatatypeProperty rdf:ID="min">
                <rdfs:domain rdf:resource="#Range"/>
                <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="max">
                <rdfs:domain rdf:resource="#Range"/>
                <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="units">
                <rdfs:domain rdf:resource="#Range"/>
                <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
        </owl:DatatypeProperty>
</rdf:RDF>
```

Con questo esempio si vuole definire un'ontologia di nome *Camera*.
Vengono definite due classi: *Money* e *Range*.
Entrambe sono sottoclassi di *Thing*, ossia si tratta di oggetti. *Money* viene caricata di una proprietà *Currency*, di tipo *string* (testuale), mentre *Range* viene caricata di tre proprietà: *min* e *max* di tipo *float*, ossia numeri in virgola mobile; *units* di tipo *string*.

In questo modo è possibile definire una nuova struttura, utilizzabile e comprensibile anche alle applicazioni.
Questo sarà il secondo punto affrontato nell'immediato futuro, ossia quello di stilare un'ontologia formale della soluzione proposta.

Capitolo 2

Soluzioni esistenti

Per ovviare ai problemi legati alle tabelle orario e alla necessità di dati aperti interscambiabili, esistono già differenti soluzioni.
Formati standard e modelli commerciali sono utilizzati per ogni singola casistica, come il trasporto piuttosto che la ristorazione, o come la gestione di un orario scolastico piuttosto che la programmazione di un cinema multisala.

Di seguito vengono mostrati una serie di esempi reali per la creazione e gestione di tali informazioni in formati aperti e standardizzati.

2.1 RDF Calendar

Il formato standard **RDFCalendar** [0] permette di definire informazioni inerenti a eventi temporali; ha estensione *.ics* ed è supportato da una lunga lista di programmi desktop e servizi online.

```
BEGIN:VEVENT
UID:20020630T230445Z-3895-69-1-7@jammer
DTSTART;VALUE=DATE:20020703
DTEND;VALUE=DATE:20020706
SUMMARY:Scooby Conference
LOCATION:San Francisco
END:VEVENT
```

Questo formato però incontra delle limitazioni: fu creato per trasmettere dati basati sul calendario, come eventi temporali, mentre esiste la reale necessità di dover sapere cosa fare con questi dati.

Ecco che nasce **RDFCalendar** [0], uno sforzo da parte del **W3C [16]** di applicare il *Semantic Web* a tale formato.
Di seguito l'esempio sopra trasformato in **RDF** (*Resource Description Framework*) [22]:

```
<Vevent>
        <uid>20020630T230445Z-3895-69-1-7@jammer</uid>
        <dtstart>2002-07-03</dtstart>
        <dtend>2002-07-06</date>
        <summary>Scooby Conference</summary>
        <location>San Francisco</location>
</Vevent>
```

Ogni evento viene identificato da un elemento *Vevent* e come requisito è richiesta una data di inizio (*dtstart*) e una di fine (*dtend*).
Di seguito un esempio più esteso [23]:

```
<rdf:RDF
        xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
        xmlns='http://www.w3.org/2002/12/cal/ical#'
        xmlns:i='http://www.w3.org/2002/12/cal/ical#'
        xmlns:x='http://www.w3.org/2002/12/cal/prod_apple#'>

        <Vcalendar>
                <component>
                        <Vevent>
                                <dtstart rdf:parseType='Resource'>
                                        <dateTime>2003-01-01T16:00:00</dateTime>
                                </dtstart>
                                <dtend rdf:parseType='Resource'>
                                        <dateTime>2003-01-01T23:00:00</dateTime>
                                </dtend>
                                <rrule rdf:parseType='Resource'>
                                        <byday>TU</byday>
                                        <freq>WEEKLY</freq>
                                        <interval>1</interval>
                                </rrule>
                        </Vevent>
                </component>
                <component>
                        <Vevent>
                                <dtstart rdf:parseType='Resource'>
                                        <dateTime>2003-01-01T11:30:00</dateTime>
                                </dtstart>
                                <dtend rdf:parseType='Resource'>
                                        <dateTime>2003-01-01T23:00:00</dateTime>
                                </dtend>
                                <rrule rdf:parseType='Resource'>
                                        <byday>WE,TH,FR,SA,SU</byday>
                                        <freq>WEEKLY</freq>
                                        <interval>1</interval>
                                </rrule>
```

```
                    </Vevent>
                </component>
        </Vcalendar>
</rdf:RDF>
```

2.2 Chef Moz

Chef Moz [1], una branca dell'**Open Directory Project** [24], era un sito web nato per aggregare ristoranti.
Era supportato da una comunità di volontari e faceva uso di un'ontologia gerarchica per organizzare la lista dei link dei ristoranti.
La struttura dati che utilizzava era stilata in XML/RDF. Un esempio pratico [1]:

```
<r:RDF xmlns:r="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:d="http://purl.org/dc/elements/1.1/"
    xmlns:cm="http://chefmoz.org/rdf/elements/1.0/"
    xmlns="http://chefmoz.org/rdf/elements/1.0/">
        <Restaurant cm:id="United_Kingdom/England/Bristol/Casa_Sudaca983983157">
                <Location>United_Kingdom/England/Bristol</Location>
                <d:title>Casa Sudaca</d:title>
                <Address>25 Zetland Road</Address>
                <City>Bristol</City>
                <Country>United Kingdom</Country>
                <Phone>(0117) 944 6304</Phone>
                <State>England</State>
                <Zip>BS6 7AH</Zip>
                <Neighborhood>Cotham</Neighborhood>
                <Hours>daily 6.30pm-11pm</Hours>
                <ParsedHours>18.5-23|18.5-23|18.5-23|18.5-23|18.5-23|18.5-23|18.5-
23</ParsedHours>
                <Smoking>permitted</Smoking>
                <Accessibility>partially</Accessibility>
                <AccessibilityNotes>there are small steps</AccessibilityNotes>
                <LargestParty>50</LargestParty>
                <Price>#5.01 - #15.00</Price>
                <Cuisine>Latin American</Cuisine>
                <Cuisine>Mexican</Cuisine>
                <Accepts>checks</Accepts>
                <Accepts>EnRoute</Accepts>
                <Accepts>Diners' Club</Accepts>
                <Accepts>Visa</Accepts>
                <Accepts>Japan Credit Bureau</Accepts>
                <Accepts>MasterCard/Eurocard</Accepts>
                <Accepts>Carte Blanche</Accepts>
                <Accepts>Discover</Accepts>
```

```
        </Restaurant>
</r:RDF>
```

Oltre a una serie di informazioni relative al ristorante, come l'accessibilità o i tipi di pagamenti accettati, ciò che interessa maggiormente è la gestione dell'orario di apertura. Questo è identificato da due elementi: *Hours* e *ParsedHours*.
Il primo specifica l'orario settimanale, unico per ogni giorno, mentre il secondo l'orario giorno per giorno.

Questa soluzione è limitata poiché non è possibile specificare regole precise per i giorni della settimana, come dire che domenica è chiuso, mentre lunedì è aperto da una certa ora ad un'altra. Inoltre non è possibile definire differenti fasce orario: alcuni ristoranti possono aprire per pranzo ma tenere chiuso il pomeriggio, per poi riaprire la sera.

Chef Moz [1], recuperava soltanto una serie di informazioni generiche per dare una linea guida all'utenza ma non era specifica per gli orari.
Non era infatti concepito per gestire complesse tabelle orario ed è possibile notarlo dal fatto che mancasse la possibilità d'inserire giorni precisi del calendario, come le festività.
Tale progetto è stato dismesso dopo circa 10 anni di servizio, anni in cui era diventato l'aggregatore internazionale di ristoranti più vasto del web, a causa di insufficienti risorse.

2.3 Unimaner

Unimaner [2] è un programma per la gestione dell'orario scolastico universitario per studenti.
E' interessante la struttura XML [14] che viene utilizzata per salvare le informazioni sopracitate [2]:

```
<?xml version="1.0"?>
<Unimaner fileformatversion="0.1.2">
        <lecture>
                <title>Übung zu Grundlagen der Technischen Informatik</title>
                <shortTitle>UE-GTI</shortTitle>
                <lecturer></lecturer>
                <timeSetting>
                        <eventVisitRate>0</eventVisitRate>
                        <times>
                                <type>0</type>
                                <dayOfWeek>0</dayOfWeek>
                                <startTime>12:00</startTime>
                                <endTime>14:00</endTime>
                        </times>
                        <times>
                                <type>0</type>
                                <dayOfWeek>0</dayOfWeek>
                                <startTime>08:00</startTime>
                                <endTime>10:00</endTime>
                        </times>
                        <times>
```

```
                        <type>0</type>
                        <dayOfWeek>3</dayOfWeek>
                        <startTime>14:00</startTime>
                        <endTime>16:00</endTime>
                    </times>
                    <times>
                        <type>0</type>
                        <dayOfWeek>3</dayOfWeek>
                        <startTime>16:00</startTime>
                        <endTime>18:00</endTime>
                    </times>
                    <times>
                        <type>0</type>
                        <dayOfWeek>4</dayOfWeek>
                        <startTime>12:00</startTime>
                        <endTime>14:00</endTime>
                    </times>
                </timeSetting>
                <room></room>
                <fgcolor>-16777216</fgcolor>
                <bgcolor>-3158065</bgcolor>
        </lecture>
        <lecture>
                <title>Rechnerübung zu Parallele und funktionale Programmierung
(Freiwillig)</title>
                <shortTitle>RUePFP</shortTitle>
                <lecturer></lecturer>
                <timeSetting>
                        <eventVisitRate>0</eventVisitRate>
                        <times>
                                <type>0</type>
                                <dayOfWeek>1</dayOfWeek>
                                <startTime>14:00</startTime>
                                <endTime>16:00</endTime>
                        </times>
                </timeSetting>
                <room></room>
                <fgcolor>-16777216</fgcolor>
                <bgcolor>-394099</bgcolor>
        </lecture>
</Unimaner>
```

Ogni corso è identificata dall'elemento *lecture*. Oltre a una serie di informazioni inerenti al corso, come il titolo, il docente, l'aula e via discorrendo, viene specificato l'elemento *timeSetting* per la gestione dell'orario.
Il corso può essere tenuto in ore differenti di giorni diversi; così ogni lezione del corso viene identificata col giorno della settimana, da una data di inizio e una di fine.
Il giorno è identificato attraverso un numero intero da 0 a 6 (*dayOfWeek*) mentre l'orario è basato sulle 24 ore (*startTime* & *endTime*).

La struttura è ben definita ma poco compatta; inoltre funziona soltanto per questo particolare scopo, ossia quello di definire l'orario scolastico.
Non vi è alcuna definizione dei giorni della settimana e non è possibile definire più fasce orario. Se volessimo specificare alcuni giorni del calendario, non sarebbe possibile siccome sono trattati solo i giorni settimanali in generale, senza tener conto della data effettiva.
Infine, non vi è un termine d'inizio o di scadenza del ciclo di lezioni, condizione che torna necessaria quando in un anno si hanno trimestri, quadrimestri e semestri, e gli orari cambiano: manca il versionamento della tabella orari.

In definitiva, la suddetta struttura può essere utile a uno studente universitario con limitate esigenze ma qual ora si presentasse la necessità di avere orari leggermente più complessi si dovrebbe optare per una soluzione più personalizzabile.

2.4 Google Transit

Tra i vari formati e standard c'è anche *Google* col suo **GTFS**: **General Transmit Feed Specification** [3].
Si tratta di un formato comune per le tabelle orario dei trasporti pubblici e per i dati geografici associati ad esse. Così, le aziende del trasporto pubblico possono pubblicare i loro dati, mentre gli sviluppatori possono creare applicazioni atte a consumarli.

Per realizzare un feed GTFS [3] è necessario configurare una serie di file di testo collezionati in un archivio zip. Ogni file modella un particolare aspetto delle informazioni associate al trasporto: fermate, strade, percorsi, e altri dati inerenti al programma di viaggio.
Una volta creato il feed GTFS [3] è necessario registrarlo presso Google in maniera tale che diventi visibile per le applicazioni che consumano questi dati (*aggregatori*): in questo caso Google fa da *Register* (si veda la sezione **4.2**).

I file che compongono un feed GTFS [3] sono i seguenti [25]:

- *agency.txt*
- *stops.txt*
- *routes.txt*
- *trips.txt*
- *stop_times.txt*
- *calendar.txt*
- *calendar_dates.txt*
- *fare_attributes.txt*
- *fare_rules.txt*
- *shapes.txt*
- *frequencies.txt*
- *transfers.txt*

Per la nostra tesi, prenderemo in riferimento soltanto i file: *agency.txt, stops.txt, stop_times.txt, calendar.txt e calendar_dates.txt.*
Gli altri file servono per specificare ulteriori informazioni sul trasporto, informazioni non necessari al fine di questa tesi.

Ogni file è stilato secondo il formato **CSV** (*Comma Separated Value*) [26], ossia ogni riga del file contiene una serie di informazioni separate da una virgola; visualmente si potrebbe pensare ad una gigantesca tabella le cui colonne sono delimitate dalle virgole.

Di seguito segue un esempio con relative spiegazioni [27]:

agency.txt

```
agency_id, agency_name,agency_url,agency_timezone,agency_phone,agency_lang
FunBus,The Fun Bus,http://www.thefunbus.org,America/Los_Angeles,(310) 555-0222,en
```

In questo file sono specificate le informazioni relative all'azienda: una relazione simile si può trovare nella sezione *locale* proposta nel **Capitolo 3**.

stops.txt

```
stop_id,stop_name,stop_desc,stop_lat,stop_lon,stop_url,location_type,parent_station
S1,Mission St. & Silver Ave.,The stop is located at the southwest corner of the
intersection.,37.728631,-122.431282,,
S2,Mission St. & Cortland Ave.,The stop is located 20 feet south of Mission St.,37.74103,-
122.422482,,
S3,Mission St. & 24th St.,The stop is located at the southwest corner of the intersection.,37.75223,-
122.418581,,
S4,Mission St. & 21st St.,The stop is located at the northwest corner of the intersection.,37.75713,-
122.418982,,
S5,Mission St. & 18th St.,The stop is located 25 feet west of 18th St.,37.761829,-122.419382,,
S6,Mission St. & 15th St.,The stop is located 10 feet north of Mission St.,37.766629,-122.419782,,
S7,24th St. Mission Station,37.752240,-122.418450,,,S8
S8,24th St. Mission Station,37.752240,-
122.418450,http://www.bart.gov/stations/stationguide/stationoverview_24st.asp,1,
```

In questo file invece sono specificate le informazioni relative alle fermate.
Ogni fermata ha un id, un nome, una descrizione, coordinate geografiche e un URL.

stop_times.txt

```
rip_id,arrival_time,departure_time,stop_id,stop_sequence,pickup_type,dropoff_type
AWE1,0:06:10,0:06:10,S1,1,0,0,0
AWE1,,,S2,2,0,1,3
AWE1,0:06:20,0:06:30,S3,3,0,0,0
AWE1,,,S5,4,0,0,0
```

```
AWE1,0:06:45,0:06:45,S6,5,0,0,0
AWD1,0:06:10,0:06:10,S1,1,0,0,0
AWD1,,,S2,2,0,0,0
AWD1,0:06:20,0:06:20,S3,3,0,0,0
AWD1,,,S4,4,0,0,0
AWD1,,,S5,5,0,0,0
AWD1,0:06:45,0:06:45,S6,6,0,0,0
```

stop_times.txt contiene le informazioni sugli orari delle fermate, come l'orario di partenza e di arrivo.

calendar.txt

```
service_id,monday,tuesday,wednesday,thursday,friday,saturday,sunday,start_date,end_date
WE,0,0,0,0,0,1,1,20060701,20060731
WD,1,1,1,1,1,0,0,20060701,20060731
```

In questo file sono definiti i giorni settimanali, la data di inizio e quella di fine in cui la linea di trasporto interessata è in servizio.
Ogni giorno è un booleano ('0' per dire che la linea è fuori servizio, '1' altrimenti) mentre la data è espressa nel formato *AnnoMeseGiorno* (*YYYYMMDD*).
In questo particolare esempio *WE* specifica che il servizio è attivo nel fine settimana mentre *WD* nei giorni che settimanali (dal Lunedì al Venerdì).

calendar_dates.txt

```
service_id,date,exception_type
WD,20060703,2
WE,20060703,1
WD,20060704,2
WE,20060704,1
```

Infine, nel file *calendar_dates.txt* sono specificate le eccezioni, ossia quando il servizio non c'è per una qualche problematica, come un giorno festivo o un'interruzione voluta, oppure che è attivo in un giorno festivo.
Anche in questo caso la data è nel formato *YYYYMMDD* e il tipo di eccezione (*exception_type*) può essere '1' (attivo in una particolare data) o '2' (interrotto).

Analizzando GTFS [3] nel completo, si può notare alcuni punti di forza: la quasi totale presenza di informazioni per ogni caratteristica del trasporto, come le fermate, i dati dell'azienda di trasporto, gli orari, le eccezioni, le coordinate geografiche delle fermate e via dicendo.
Ancora una volta però non c'è la possibilità di utilizzare le proprie formattazioni, come la data o l'ora.
Inoltre, per motivi di sicurezza, Google non permette la registrazione dell'URI di questa risorsa ma il caricamento dell'archivio zip, validato manualmente con un apposito validatore. Anche questa è

una grossa limitazione per le informazioni pubbliche, siccome per ogni modifica è necessario seguire alcuni passaggi quando questa procedura potrebbe tranquillamente essere fatta in automatico.
Infine il formato utilizzato (*CSV [26]*) non permette l'ampliamento tramite informazioni semantiche (*RDF*), limitando così le interrogazioni possibili.

Capitolo 3

La proposta

Per ovviare alle limitazioni delle soluzioni già esistenti, abbiamo ritenuto necessario creare un nuovo modello per l'interscambio dei dati.
L'idea è quella di usare un'ontologia che permetta di definire informazioni relative al documento, localizzazione e internazionalizzazione delle informazioni, un sistema di ereditarietà tra eventi (*venue*), un modo compatto di specificare le tabelle orario per ogni evento.
Vogliamo dare inoltre la possibilità di gestire diverse tipologie di eventi, unificando in un modello unico le altre soluzioni proposte.

L'elemento comune di ogni tipologia d'evento (attività commerciale, ente pubblico, trasporto, evento temporaneo, eccetera) è il tempo e in questo caso particolare le tabelle orario.
Ciò che può variare sono le informazioni relative che l'evento può portare con se nella descrizione, senza scendere troppo nel dettaglio. Ad esempio, una tournée potrebbe aver bisogno di specificare la posizione geografica delle sue tappe, mentre un cinema potrebbe specificare il prezzo del biglietto per un particolare film.

3.1 Specifiche

Per specificare l'ontologia si è voluto utilizzare una struttura XML [14] e si è deciso che il lavoro è preliminare, e che una sua formalizzazione in un linguaggio *OWL [20]* sarebbe appropriata, ma che si è preferito concentrarsi nella modellazione completa di tutti gli aspetti, non solo temporali ma anche geografici.

Di seguito la struttura base dell'ontologia:

```
<?xml version="1.0" encoding="UTF-8"?>
<timetable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.fabiovitali.it/tt Timetable-prova.xsd"
xmlns="http://www.fabiovitali.it/tt">
        <metadata>
                ...
        </metadata>
        <venues>
```

```
            ...
        </venues>
        <locale>
            ...
        </locale>
</timetable>
```

Lo schema sopra indicato è suddiviso in tre macro aree: *metadata*, *venues* e *locale*.
L'elemento *metadata* contiene tutte le informazioni relative al documento; un esempio:

```
<metadata>
        <this href="http://data.bibliotecasalaborsa.it/orario"/>
        <permanentUri href="http://data.bibliotecasalaborsa.it/2012"/>
        <title value="Orario della Biblioteca Sala Borsa di Bologna - anno 2012"/>
        <author href="http://www.bibliotecasalaborsa.it/me" showAs="Biblioteca Salaborsa"/>
        <created value="2011-11-01"/>
        <validFrom value="2011-11-01"/>
        <validTo value="2012-10-31"/>
        <authoringTool value="none"/>
        <authoritative value="yes"/>
        <previousDocument href="http://data.bibliotecasalaborsa.it/2011/orario"/>
        <source href="http://data.bibliotecasalaborsa.it/2012/orario"/>
</metadata>
```

Ogni sotto elemento di *metadata* specifica una particolare proprietà dell'intero documento:

- *this*: identifica l'URI corrente del documento
- *permanentUri*: identifica l'URI permanente del documento
- *title*: identifica il titolo del documento
- *author*: identifica le caratteristiche dell'autore del documento
- *created*: identifica la data di creazione
- *validFrom*: identifica l'inizio del periodo di validità
- *validTo*: identifica il termine del periodo di validità
- *authoringTool*:
- *authoritative*: specifica se l'*authoringTool* è anche autoritativo o meno
- *previousDocument*: identifica l'URI del documento precedente (si basa sull'ultima data specificata in validTo)
- *source*: identifica l'URI del documento sorgente (il file XML)

La seconda macro area, definita dall'elemento *venues*, contiene tutte le *venue* create dall'utente. Ogni *venue* rappresenta un evento ed è divisa in due parti: la parte *information* e la parte *hours*. Segue un esempio:

```
<venues>
	<venue id="8xis8i3" showAs="Gualerzi Pietro">
		<information>
			<type item="library" dict="#shoptypes" showAs="Negozio"/>
			<name value="Gualerzi Pietro - Linea Corredi dal 1905"/>
			<address value="Piazza Benderi 8, 42025 Cavriago RE Italy"/>
			<telephone value="0522372028"/>
			<coords value="44.697678,10.522596"/>
			<homepage href="http://www.gualerzipietro.it"/>
		</information>
		<hours>
			<hour id="d9ci9s" pattern="lun, mar, mer, ven, sab: 0830-1230,
1430-1730. gio: 0830-1230. dom: -." type="Weekly"/>
			<hour id="zzks92" pattern="#festivi: -." type="Festivity"/>
		</hours>
	</venue>
</venues>
```

In questo caso, il *type* indica che la *venue* è un negozio con una serie di caratteristiche e informazioni. L'elemento *information* contiene tutte le informazioni inerenti alla *venue*, ossia:

- *subvenueOf*: identifica la venue genitore da cui eredita l'attuale
- *type*: identifica il tipo di venue
- *subtype*: identifica il sottotipo
- *name*: identifica il nome della venue
- *address/telephone/coords/...*: ulteriori informazioni sulla venue

L'elemento subvenueOf è la chiave per l'ereditarietà fra venue.
Infatti, ogni venue può ereditare le informazioni dalla venue genitore specificata in subvenueOf, sovrascrivendo quelle differenti semplicemente assegnandogli un nuovo valore.
In questo modo, le informazioni non verranno replicate ogni volta e si potrà costruire una struttura ad albero specificando relazioni di parentela.

Nell'elemento *hours*, invece, sono specificate di fatto le tabelle orario, ognuna di esse identificate dall'elemento *hour*: possono coesistere più tabelle orario dello stesso tipo e posso contenere differenti attributi.
Di seguito la lista degli attributi necessari:

- *id*: identificativo univoco

- *pattern*: identifica l'orario
- *type*: tipo di tabella orario

Da prendere in considerazione è l'attributo *pattern*, uno degli elementi rivoluzionari della proposta: accetta una serie di regole, separate da un punto, ognuna delle quali è suddivisa in due parti separate dai due punti.
Il *pattern* viene così formalizzato:

```
el1, el2, el3, [...] : val1, val2, val3, [...] . el1, el2, el3, [...] : val1, val2, val3, [...] .
```

Nell'esempio precedente la *venue* ha due sole tabelle orario: *Weekly* e *Festivity*. La prima rappresenta l'orario settimanale mentre la seconda quello delle festività.
Quindi, nell'esempio, la *venue* specifica che il negozio è aperto tutti i giorni tranne la domenica e il giovedì pomeriggio, mentre i festivi è chiuso. In questo particolare caso i festivi si rifanno a un dizionario (spiegato più avanti). Inoltre, ci sono due fasce orario: dalle 08:30 alle 12:30 e poi dalle 14:30 alle 17:30. Il trattino indica un intervallo temporale.
Se avessimo voluto specificare che il martedì fa orario continuato, avremmo dovuto riscrivere il *pattern* come segue:

```
lun, mer, ven, sab: 0830-1230, 1430-1730. mar: 0830-1730. gio: 0830-1230. dom: -.
```

La terza macro area è identificata dall'elemento *locale*.
Ivi sono contenute tutte le informazioni locali, come il formato della data, i giorni della settimana, i mesi dell'anno e via dicendo, così come i dizionari. Di seguito un esempio:

```
<locale>
	<weekDays value="lun mar mer gio ven sab dom"/>
	<months value="gen feb mar apr mag giu lug ago set ott nov dic"/>
	<dateFormat value="D d/m/Y"/>
	<weekStartsWith value="1" showAs="Lunedì"/>
	<dicts>
		<dict id="placeTypes" href="http://www.fabiovitali.it/placeTypes"/>
		<dict id="shopTypes" href="http://www.fabiovitali.it/shopTypes"/>
		<dict id="festivi">
			<items>
				<item id="c" value="01/gen" showAs="Capodanno"/>
				<item id="e" value="06/gen" showAs="Epifania"/>
				<item id="p" value="31/mar" showAs="Pasqua"/>
				<item id="lda" value="01/apr" showAs="Lunedì dell'Angelo"/>
				<item id="lib" value="25/apr" showAs="Festa della Liberazione"/>

				<item id="lav" value="01/mag" showAs="Festa dei Lavoratori"/>
```

```
                        <item id="r" value="02/giu" showAs="Festa della Repubblica"/>
                        <item id="f" value="15/ago" showAs="Ferragosto"/>
                        <item id="sp" value="04/ott" showAs="San Petronio (patrono di
Bologna)"/>
                        <item id="t" value="01/nov" showAs="Tutti i Santi"/>
                        <item id="i" value="08/dic" showAs="Immacolata Concezione"/>
                        <item id="n" value="25/dic" showAs="Natale"/>
                        <item id="ss" value="26/dic" showAs="Santo Stefano"/>
                    </items>
                </dict>
        </dicts>
</locale>
```

In questo caso, sono specificati le informazioni locali secondo il formato italiano:

- *weekDays*: identifica i giorni della settimana
- *months*: identifica i giorni del mese
- *weekStartsWith*: identifica il giorno con cui inizia la settimana
- *dicts*: contiene una lista di dizionari

In particolare è possibile specificare nel locale una serie di dizionari (*dict*). Possono essere definiti internamente, tramite la seguente struttura:

```
<dict id="id_univoco">
        <item id="id_univoco" value="valore_da_usare" showAs="valore_da_rappresentare"/>
        ...
</dict>
```

Altrimenti è possibile definirli esternamente come link a risorse:

```
<dict id="id_univoco" href="URI_della_risorsa"/>
```

Per finire, un esempio completo dell'ontologia:

```
<?xml version="1.0" encoding="UTF-8"?>
<timetable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.fabiovitali.it/tt    Timetable-prova.xsd"
xmlns="http://www.fabiovitali.it/tt">
        <metadata>
                <this href="http://data.bibliotecasalaborsa.it/orario"/>
```

```
            <permanentUri href="http://data.bibliotecasalaborsa.it/2012"/>
            <title value="Orario della Biblioteca Sala Borsa di Bologna - anno 2012"/>
            <author href="http://www.fabiovitali.it/me" showAs="Fabio Vitali"/>
            <created value="2011-11-01"/>
            <validFrom value="2011-11-01"/>
            <validTo value="2012-10-31"/>
            <authoringTool value="none"/>
            <authoritative value="yes"/>
            <previousDocument href="http://data.bibliotecasalaborsa.it/2011/orario"/>
            <source href="http://data.bibliotecasalaborsa.it/2012/orario"/>
        </metadata>
        <venues>
            <venue id="salaborsa" showAs="Sala Borsa a Bologna">
                <information>
                    <type item="publicPlace" dict="#placetypes" showAs="Luogo
pubblico"/>
                    <subtype item="library" dict="#shoptypes" showAs="Biblioteca"/>
                    <name value="Biblioteca Sala Borsa"/>
                    <address value="Piazza del Nettuno 3, 40124 Bologna BO Italy"/>
                    <telephone value="0512194400"/>
                    <coords value="44.4947,11.3425"/>
                    <homepage href="http://www.bibliotecasalaborsa.it/home.php"/>
                    <wikipedia href="http://it.wikipedia.org/wiki/Sala_borsa"/>
                    <foursquare href="https://it.foursquare.com/v/biblioteca-sala-
borsa/4bb334de42959c74549e212c"/>
                </information>
                <hours>
                    <hour id="w1" pattern="mar, mer, gio, ven: 0830-1930. sab: 0900-
1845. lun,dom: -." type="Weekly"/>
                    <hour id="f1" pattern="c, e, p, lib, lav, n, ss: 0830-1230. r, f, t:
1000-1300" type="Festivity"/>
                </hours>
            </venue>
            <venue id="salaborsaragazzi" showAs="Sala Borsa Ragazzi">
                <information>
                    <subvenueOf href="#salaborsa"/>
                    <subtype item="library" dict="#shoptypes" showAs="Biblioteca per
ragazzi"/>
                    <name value="La sala borsa ragazzi"/>
                </information>
                <hours>
                    <hour id="w1" pattern="mar, mer, gio, ven: 1000-1900. sab: 1000-
1800. lun,dom: -." type="Weekly"/>
                    <hour id="f1" pattern="#festivi: -." type="Festivity"/>
                </hours>
            </venue>
            <venue id="ingressosalaborsa" showAs="Ingresso sala borsa">
                <information>
                    <subvenueOf href="#salaborsa"/>
                    <name value="Ingresso della sala borsa"/>
```

```
                    </information>
                    <hours>
                        <hour id="w1" pattern="mar, mer, gio, ven, sab: 0830-1930.
lun,dom: -." type="Weekly"/>
                        <hour id="f1" pattern="#festivi: -." type="Festivity"/>
                    </hours>
                </venue>
                <venue id="informazioni" showAs="Informazioni telefoniche">
                    <information>
                        <subvenueOf href="#salaborsa"/>
                        <name value="Informazioni telefoniche"/>
                        <telephone value="0512194426"/>
                        <coords value=""/>
                    </information>
                    <hours>
                        <hour id="w1" pattern="mar, mer, gio, ven, sab: 0830-1530.
lun,dom: -." type="Weekly"/>
                        <hour id="f1" pattern="#festivi: -." type="Festivity"/>
                    </hours>
                </venue>
                <venue id="camioncino" showAs="La sala borsa itinerante">
                    <information>
                        <subvenueOf href="#salaborsa"/>
                        <name value="La sala borsa itinerante"/>
                    </information>
                    <hours>
                        <hour id="t1" pattern="lun: 0800-1100, 1400-1700." type="Travel"
coords="44.483892,11.353340"/>
                        <hour id="t2" pattern="mar, mer: 0900-1100, 1400-1600."
type="Travel" coords="44.505933,11.364326"/>
                        <hour id="t3" pattern="gio: 0800-1100, 1400-1700." type="Travel"
coords="44.479116,11.309738"/>
                        <hour id="t4" pattern="ven: 0800-1200, 1400-1800." type="Travel"
coords="44.478626,11.375484"/>
                        <hour id="f1" pattern="#festivi: -." type="Festivity"/>
                    </hours>
                </venue>
            </venues>
            <locale>
                <weekDays value="lun mar mer gio ven sab dom"/>
                <months value="gen feb mar apr mag giu lug ago set ott nov dic"/>
                <dateFormat value="D d/m/Y"/>
                <weekStartsWith value="1" showAs="Lunedì"/>
                <dicts>
                    <dict id="placeTypes" href="http://www.fabiovitali.it/placeTypes"/>
                    <dict id="shopTypes" href="http://www.fabiovitali.it/shopTypes"/>
                    <dict id="festivi">
                        <items>
                            <item id="c" value="01/gen" showAs="Capodanno"/>
                            <item id="e" value="06/gen" showAs="Epifania"/>
```

```
                                <item id="p" value="31/mar" showAs="Pasqua"/>
                                <item id="lda" value="01/apr" showAs="Lunedì
dell'Angelo"/>
                                <item id="lib" value="25/apr" showAs="Festa della
Liberazione"/>
                                <item id="lav" value="01/mag" showAs="Festa dei
Lavoratori"/>
                                <item id="r" value="02/giu" showAs="Festa della
Repubblica"/>
                                <item id="f" value="15/ago" showAs="Ferragosto"/>
                                <item id="sp" value="04/ott" showAs="San Petronio
(patrono di Bologna)"/>
                                <item id="t" value="01/nov" showAs="Tutti i Santi"/>
                                <item id="i" value="08/dic" showAs="Immacolata
Concezione"/>
                                <item id="n" value="25/dic" showAs="Natale"/>
                                <item id="ss" value="26/dic" showAs="Santo Stefano"/>
                            </items>
                        </dict>
                    </dicts>
                </locale>
</timetable>
```

Il modello sopracitato permette quindi di gestire eventi differenti tenendo conto del tempo e del luogo del loro svolgimento. Si tratta di un modello altamente configurabile e adattabile alle esigenze della nazionalità dell'utente, senza dover per forza usare un formato internazionale o doversi adattare a qualche lingua straniera per i giorni della settimana o i mesi dell'anno.

3.2 Esempi

Seguiranno una serie di esempi pratici su come utilizzare la suddetta ontologia.

Ogni esempio prende in considerazione un particolare evento: potrà essere composto da più *venue* così come da più attributi per singole *hour* ma il *pattern* seguirà sempre lo stesso modello. Per far capire meglio sono stati realizzati esempi eterogenei per dare una migliore panoramica sulle potenzialità della nostra proposta.

Si è deciso di non specificare gli *id* di ogni *venue* e *hour* ma si da per scontato che vengano utilizzati per rendere unici gli eventi e le loro tabelle orario.

Infine, ogni esempio è stato realizzato prendendo casi reali per mostrare ancor di più la validità della nostra proposta: il negozio *Maisons du Monde*, il cinema multisala *Medusa*, il ciclo universitario della facoltà d'*Informatica* dell'università di Bologna e via dicendo.

3.2.1 Negozio

L'attività commerciale più comune è il negozio.
I commercianti di una città hanno bisogno di stilare l'orario di apertura e chiusura della loro attività.

Spesso viene fatto varie volte nel corso del tempo, usando diversi formati: a volte si stila una tabella su Excel, altre volte un'immagine, altre volte ancora un file PDF.
L'idea è quella di fare un po' di chiarezza, unificando tutto con l'ontologia proposta.

La venue in questione è formata da una serie di informazioni di base (che non verranno ripetute negli altri esempi, ndr) e da due tabelle orario: quella settimanale e quella delle festività.

```
<venue showAs="Maisons du Monde">
	<information>
		<type item="shop" dict="#shoptypes" showAs="Negozio"/>
		<name value="Maisons du Monde - Sede di Bologna"/>
		<address value="Via Rizzoli 28, 40133 Bologna BO Italy"/>
		<telephone value="0039051269623"/>
		<coords value="44.494516,11.345186"/>
		<homepage href="http://www.maisonsdumonde.com/"/>
		<wikipedia href="http://fr.wikipedia.org/wiki/Maisons_du_Monde"/>
		<facebook href="http://www.facebook.com/MaisonsduMonde"/>
		<twitter href="https://twitter.com/MDM_IT"/>
		<foursquare href="https://it.foursquare.com/v/maison-du-
monde/4cc2612a914137046642b455"/>
	</information>
	<hours>
		<hour type="Weekly" pattern="lun,mar,mer,gio,ven,sab: 1000-1930. dom: -."/>
		<hour type="Festivity" pattern="t: 1000-1930. c,e,p,lda,lib,lav,r,f,sp,i,n,ss: -."
dict="#festivi"/>
	</hours>
</venue>

<dict id="festivi">
	<items>
		<item id="c" value="01/gen" showAs="Capodanno"/>
		<item id="e" value="06/gen" showAs="Epifania"/>
		<item id="p" value="31/mar" showAs="Pasqua"/>
		<item id="lda" value="01/apr" showAs="Lunedì dell'Angelo"/>
		<item id="lib" value="25/apr" showAs="Festa della Liberazione"/>
		<item id="lav" value="01/mag" showAs="Festa dei Lavoratori"/>
		<item id="r" value="02/giu" showAs="Festa della Repubblica"/>
		<item id="f" value="15/ago" showAs="Ferragosto"/>
		<item id="sp" value="04/ott" showAs="San Petronio (patrono di Bologna)"/>
		<item id="t" value="01/nov" showAs="Tutti i Santi"/>
		<item id="i" value="08/dic" showAs="Immacolata Concezione"/>
		<item id="n" value="25/dic" showAs="Natale"/>
		<item id="ss" value="26/dic" showAs="Santo Stefano"/>
	</items>
</dict>
```

Nell'esempio viene trattato il negozio *Maisons du Monde* [28].
Dalle informazioni (*information*) si deduce che la *venue* rappresenta un negozio (*type*), che è situato in un determinato punto geografico (*coords*) e che ha una serie di riferimenti online.

L'elemento *hours* specifica gli orari di apertura: dal lunedì al sabato è aperto dalle 10:00 alle 19:30 mentre la domenica è chiuso, così come anche nei festivi, tranne in via del tutto eccezionale il giorno di *Tutti i Santi* cioè il *01/nov* come definito nel dizionario festivi.
L'uso del dizionario aiuta a evitare la ridondanza delle informazioni.

3.2.2 Farmacia

L'esempio di sopra è il più semplice che si possa mostrare ma è altrettanto possibile complicare la situazione.
Esistono altre attività che richiedono una gestione dell'orario di lavoro molto più schematizzata.
La farmacia, a differenza del negozio, può avere una serie di dipendenti ognuno dei quali svolge la propria attività in turni separati.
La necessità è quindi quella di creare una tabella orari mensile in cui ogni giorno è suddiviso in turni.

```
<venue showAs="Farmacia">
	<information>...</information>
	<hours>
		<hour event="Maria" type="Pharmacy" pattern="02/10, 07/10, 12/10, 17/10,
22/10, 27/10: m. 01/10, 06/10, 11/10, 16/10, 21/10, 26/10: p. 03/10, 08/10, 13/10, 18/10, 23/10,
28/10: n. 05/10, 10/10, 15/10, 20/10, 25/10, 30/10: r. 04/10, 09/10, 14/10, 19/10, 24/10, 29/10: s."
dict="#turniFarmacia"/>
		<hour event="Guido" type="Pharmacy" pattern="02/10, 07/10, 12/10, 17/10,
22/10, 27/10: s. 01/10, 06/10, 11/10, 16/10, 21/10, 26/10: n. 03/10, 08/10, 13/10, 18/10, 23/10,
28/10: r. 05/10, 10/10, 15/10, 20/10, 25/10, 30/10: m. 04/10, 09/10, 14/10, 19/10, 24/10, 29/10: p."
dict="#turniFarmacia"/>
		<hour event="Marco" type="Pharmacy" pattern="02/10, 07/10, 12/10, 17/10,
22/10, 27/10: p. 01/10, 06/10, 11/10, 16/10, 21/10, 26/10: r. 03/10, 08/10, 13/10, 18/10, 23/10,
28/10: m. 05/10, 10/10, 15/10, 20/10, 25/10, 30/10: s. 04/10, 09/10, 14/10, 19/10, 24/10, 29/10: n."
dict="#turniFarmacia"/>
		...
	</hours>
</venue>

<dict id="turniFarmacia">
	<item id="m" value="Mattino" showAs="0700-1300"/>
	<item id="p" value="Pomeriggio" showAs="1300-1900"/>
	<item id="n" value="Notte" showAs="1900-0700"/>
	<item id="s" value="Smonto Notte" showAs="-"/>
	<item id="r" value="Riposo" showAs="-"/>
</dict>
```

In questo caso il *pattern* è usato in maniera diversa rispetto al *Negozio*.
Invece di avere un intervallo temporale espresso in ore, qui è espresso in giorni e ogni *hour* identifica un dipendente e i suoi turni.
Prendendo Maria, si deduce che:

- il 2, il 7, il 12, il 17, il 22 e il 27 di Ottobre lavora al mattino, ossia dalle 07:00 alle 13:00
- l'1, il 6, l'11, il 16, il 21 e il 26 lavora al pomeriggio, ossia dalle 13:00 alle 19:00
- il 3, l'8, il 13, il 18, il 23 e il 28 lavora la notte, ossia dalle 19:00 alle 07:00
- il 5, il 10, il 15, il 20, il 25 e il 30 riposa
- mentre gli altri giorni finisce alla notte.

Come nel negozio, anche in questo caso l'uso di un dizionario è di grande aiuto per associare diverse informazioni a ogni singola proprietà.

3.2.3 Orario scolastico (studente)

Ora vogliamo spostarci sul panorama studentesco.
Non per forza l'orario deve rappresentare ore di lavoro ma può tranquillamente definire il ciclo di lezioni di uno studente universitario.
Uno studente quindi avrà la necessità di stilare il proprio orario scolastico in base ai corsi e ogni corso avrà un termine d'inizio e uno di fine.
E' anche possibile inserire le coordinate dell'aula dove viene svolta la lezione del tal corso ma si è deciso di trattare questa informazione nell'esempio della tournée, descritto più avanti.

```
<venue showAs="Orario scolastico - primo ciclo">
	<information>...</information>
	<hours>
		<hour type="Class" pattern="mer:0830-1130. ven:1030-1330." event="Linguaggi di Programmazione" limit="24/09-21/12"/>
		<hour type="Class" pattern="lun:1530-1730. ven:0830-1030." event="Ottimizzazione" limit="24/09-21/12"/>
		<hour type="Class" pattern="mar:0830-1130. gio:1330-1530." event="Reti di Calcolatori" limit="24/09-21/12"/>
		<hour type="Class" pattern="lun:1330-1530. mar:1130-1330." event="Sistemi Operativi" limit="24/09-21/12"/>
		<hour type="Class" pattern="mer:1130-1330. gio:1530-1830." event="Calcolo Numerico" limit="24/09-21/12"/>
	</hours>
</venue>
```

Ogni corso è rappresentato da un hour di tipo *Class*.
In particolare, oltre al *pattern* simile a quello usato nel tipo di tabella orario *Weekly*, viene utilizzato l'attributo *limit* che ha il compito di definire un intervallo temporale in più: in questo caso si tratta della durata totale del corso.
Per comprendere prendiamo il corso di *Linguaggi di Programmazione*:

- *limit* dice che inizia il 24 di Settembre e termina il 21 di Dicembre

- *pattern* definisce due lezioni settimanali: la prima al Mercoledì dalle 08:30 alle 11:30, la seconda il Venerdì dalle 10:30 alle 12:30

3.2.4 Orario scolastico (istituto)

Preso lo studente, passiamo al suo istituto [29].
Utilizzando la stessa ontologia, gli stessi procedimenti e quindi la stessa struttura è possibile rappresentare l'orario scolastico generale della facoltà del suddetto studente.
La differenza dal seguente caso rispetto al precedente sta nei cicli: ogni anno è diviso in due cicli, quindi una triennale avrà 6 cicli in tutto da gestire.
Ogni ciclo è identificato da una *venue* e ogni corso da una *hour*.

```
<venue showAs="Orario scolastico - Primo Anno - Primo ciclo">
	<information>...</information>
	<hours>
		<hour type="Class" pattern="lun:0830-1030. mer:0830-1030. ven:0830-1030."
event="Fisica" limit="24/09-21/12"
seeAlso="http://www.scienze.unibo.it/Scienze+Matematiche/Didattica/Insegnamenti/dettaglio.htm
?
AnnoAccademico=2011&IdComponenteAF=320576&CodDocente=015847&CodMateria=00405
"/>
		<hour type="Class" pattern="lun:1030-1230. mar:1130-1230. mer:1030-1230."
event="Algebra e Geometria" limit="24/09-21/12"
seeAlso="http://www.scienze.unibo.it/Scienze+Matematiche/Didattica/Insegnamenti/dettaglio.htm
?
AnnoAccademico=2011&IdComponenteAF=366975&CodDocente=033180&CodMateria=58414
"/>
		<hour type="Class" pattern="mar:0830-1130. gio:1330-1530." event="Reti di
Calcolatori" limit="24/09-21/12"
seeAlso="http://www.scienze.unibo.it/Scienze+Matematiche/Didattica/Insegnamenti/dettaglio.htm
?
AnnoAccademico=2011&IdComponenteAF=320580&CodDocente=016887&CodMateria=11145
"/>
		<hour type="Class" pattern="lun,mar,mer:1330-1630." event="Programmazione"
limit="24/09-21/12"
seeAlso="http://www.scienze.unibo.it/Scienze+Matematiche/Didattica/Insegnamenti/dettaglio.htm
?
AnnoAccademico=2011&IdComponenteAF=320574&CodDocente=030761&CodMateria=00819
"/>
		<hour type="Class" pattern="mar:0830-1130. gio:0830-1130." event="Analisi
Matematica" limit="24/09-21/12"
seeAlso="http://www.scienze.unibo.it/Scienze+Matematiche/Didattica/Insegnamenti/dettaglio.htm
?
AnnoAccademico=2011&IdComponenteAF=320573&CodDocente=016912&CodMateria=00013
"/>
		<hour type="Class" pattern="gio:1330-1630. ven:1030-1330." event="Architettura
degli Elaboratori" limit="24/09-21/12"
seeAlso="http://www.scienze.unibo.it/Scienze+Matematiche/Didattica/Insegnamenti/dettaglio.htm
```

```
?
AnnoAccademico=2011&IdComponenteAF=350960&CodDocente=031994&CodMateria=11925
"/>
        </hours>
</venue>
<venue showAs="Orario scolastico - Primo Anno - Secondo ciclo">
        <information>...</information>
        <hours>...</hours>
</venue>
<venue showAs="Orario scolastico - Secondo Anno - Primo ciclo">
        <information>...</information>
        <hours>...</hours>
</venue>
```

Questo esempio differisce dal precedente soltanto dal fatto che vi sono più *venue*, una per ogni ciclo.
Si sarebbe potuto fare una venue per ogni anno e due sotto *venue* per i suoi cicli, giusto per evitare la ridondanza delle informazioni tramite l'utilizzo dell'eredità tra *venue*.

Infine compare un nuovo attributo: *seeAlso*.
In questo attributo viene specificato l'URI del corso in questione: in questo modo si evita di legare specifiche informazioni direttamente all'orario.
Più avanti si vedrà come questo attributo sia essenziale per alcuni casi, come quello del cinema: non è corretto infatti inserire una serie di informazioni che non hanno a che fare direttamente con l'oggetto in questione.

3.2.5 Teatro

Perché non passare una bella serata a teatro?
Cosa c'è di meglio di fare una ricerca sulla nuova stagione in programma, recuperando gli spettacoli in previsione e i loro prezzi?
Ecco l'esempio che calza a dovere: il teatro [30]!

Ogni teatro ha la sua particolare programmazione di stagione e i prezzi del biglietto possono variare da zona a zona: le platee costeranno di più rispetto alle gallerie, le quali saranno più care rispetto alle balconate.
E' quindi necessario suddividere l'evento in programma per categorie.

```
<venue showAs="Stagione 2012-2013 - Teatro Manzoni">
        <information>
                <type item="publicPlace" dict="#placetypes" showAs="Luogo pubblico"/>
                <subtype item="theatre" dict="#theatretypes" showAs="Teatro"/>
                <name value="Stagione 2012-2013 - Teatro Auditorium Manzoni – Bologna"/>
                <homepage
href="http://www.auditoriumanzoni.it/presentazionestagione20122013/"/>
                ...
        </information>
```

```
        <hours>
                <hour event="FTCB – Vedernikov (direttore) – Maisky (violoncello) - Platea I e II"
type="Uniplex" pattern="13/11/2012: 2100-2300." price="45"
seeAlso="http://www.auditoriumanzoni.it/event/ftcb-vedernikov-direttore-maisky-violoncello/"/>
                <hour event="FTCB – Vedernikov (direttore) – Maisky (violoncello) - Platea III e
Galleria I" type="Uniplex" pattern="13/11/2012: 2100-2300." price="36"
seeAlso="http://www.auditoriumanzoni.it/event/ftcb-vedernikov-direttore-maisky-violoncello/"/>
                <hour event="FTCB – Vedernikov (direttore) – Maisky (violoncello) - Galleria II e
Balconata I" type="Uniplex" pattern="13/11/2012: 2100-2300." price="29"
seeAlso="http://www.auditoriumanzoni.it/event/ftcb-vedernikov-direttore-maisky-violoncello/"/>
                <hour event="FTCB – Vedernikov (direttore) – Maisky (violoncello) - Balconata II
e III" type="Uniplex" pattern="13/11/2012: 2100-2300." price="17"
seeAlso="http://www.auditoriumanzoni.it/event/ftcb-vedernikov-direttore-maisky-violoncello/"/>
                <hour event="Giovanni Allevi – Orchestra Carlo Felice di Genova - Platea I e II"
type="Uniplex" pattern="18/11/2012: 2100-2300." price="48"
seeAlso="http://www.auditoriumanzoni.it/event/giovanni-allevi-orchestra-carlo-felice-di-
genova/"/>
                <hour event="Giovanni Allevi – Orchestra Carlo Felice di Genova - Platea III e
Galleria I" type="Uniplex" pattern="18/11/2012: 2100-2300." price="40"
seeAlso="http://www.auditoriumanzoni.it/event/giovanni-allevi-orchestra-carlo-felice-di-
genova/"/>
                <hour event="Giovanni Allevi – Orchestra Carlo Felice di Genova - Galleria II e
Balconata I" type="Uniplex" pattern="18/11/2012: 2100-2300." price="32"
seeAlso="http://www.auditoriumanzoni.it/event/giovanni-allevi-orchestra-carlo-felice-di-
genova/"/>
                <hour event="Giovanni Allevi – Orchestra Carlo Felice di Genova - Balconata II e
III" type="Uniplex" pattern="18/11/2012: 2100-2300." price="19"
seeAlso="http://www.auditoriumanzoni.it/event/giovanni-allevi-orchestra-carlo-felice-di-
genova/"/>
        <hours>
</venue>
```

Un'unica *venue* rappresenta la stagione del teatro e ogni evento è rappresentato da un elemento *hour*.
Da notare la ripetizione degli eventi, distinti per area di esecuzione: *platea*, *galleria*, *balconata*. Ogni area ha il suo prezzo e in questo modo è possibile interrogare la *venue* per sapere quali siano gli spettacoli più o meno cari: l'attributo *price* serve a tale scopo.

3.2.6 Cinema multisala

Siete a capo di un cinema multisala [31] e avete la necessità di stilare un orario di apertura della struttura e orari differenti per ogni sala, così da gestire le varie proiezioni dei film in programma. Non solo, volete aggiungere informazioni per ogni sala, come ad esempio una propria pagina web, o addirittura un orario di apertura e chiusura per la sala stessa!
Ebbene, si utilizza un'altra delle caratteristiche innovative della nostra proposta: l'ereditarietà a cascata tra *venue*.
Un cinema multisala potrà gestire film differenti, con diverse proiezioni, in ogni singola sala.

```
<venue id="f98jkd893" showAs="Cinema Multisala Medusa">
	<information>...</information>
	<hours>
		<hour id="e9d309d" type="Weekly" pattern="lun,mar,mer,gio,ven,sab,dom: 1000-
0230."/>
		<hour id="f9iroaa" type="Festivity" pattern="#festivi: 1900-0100."/>
	</hours>
</venue>
<venue id="fg9clsscz" showAs="Sala 1">
	<information>
		<subvenueOf href="#f98jkd893"/>
		...
	</information>
	<hours>
		<hour id="zid8ocis" type="Uniplex" event="The wedding party"
pattern="25/10,26/10,27/10,28/10,29/10,30/10: 1550, 1805, 2020, 2240." duration="87"
seeAlso="http://www.thespacecinema.it/film/the-wedding-party-4303?cinema_from=3"/>
		<hour id="brlzckss" type="Uniplex" event="Skyfall 007"
pattern="31/10,01/11,02/11,03/11,04/11,05/11: 1600, 1915, 2230." duration="145"
seeAlso="http://www.thespacecinema.it/film/skyfall-007-4313?cinema_from=3"/>
	</hours>
</venue>
<venue id="91cz22scz" showAs="Sala 2">
	<information>
		<subvenueOf href="#f98jkd893"/>
		...
	</information>
	<hours>...</hours>
</venue>
```

Questo esempio mette in mostra la proprietà di ereditarietà delle *venue*.
La prima *venue* definisce il cinema multisala con due tabelle orario: *Weekly* e *Festivity*. Queste due tabelle riguardano l'intera struttura ma è possibile che, all'interno del multisala, le attività possano avere orari differenti. Discorso simile può essere fatto per un centro commerciale.
Ecco quindi la spiegazione delle altre *venue* che fanno uso della proprietà *subvenueOf*: riferendosi alla *venue* principale (la prima), ereditano tutte le informazioni specificate nell'elemento *information*. Così ogni sala sarà una sotto *venue* della principale.

In questo caso, viene utilizzata una nuova tabella orari: *Uniplex*.
Tale tabella, rappresenta un film con alcune informazioni associate ad esso:

- *event*: definisce il nome dell'evento (in questo caso il film)
- *duration*: definisce la durata del film
- *seeAlso*: definisce l'URI della scheda del film

Il *pattern* specifica le date in cui sarà proiettato il film e le proiezioni.
Prendendo come esempio il film *Skyfall 007*, si nota che sarà in proiezione dal 31/10 al 05/11 e ci sarà una proiezione alle 16:00, alle 19:15 e alle 22:30.

3.2.7 Tournée

E che dire infine delle tournée?
Oltre a gestire semplici orari come quelli di un negozio, o complessi come quelli di una farmacia a turni, o ben separati come quelli di un cinema multisala vogliamo anche definire una tabella orario per le tappe di una tournée, situate in differenti luoghi fisici.
Qui, la necessità è quella di associare ad ogni tappa un titolo ma soprattutto delle coordinate geografiche, così da poter rappresentare il dato su una mappa. (*Google Maps*, *Open Street Map*, ...).

In questo particolare caso verrà preso l'esempio del tour italiano che il gruppo *Depeche Mode* farà nel 2013 [32].

```
<venue showAs="Depeche Mode Tournée 2013">
	<information>
		<type item="publicPlace" dict="#placetypes" showAs="Luogo pubblico"/>
		<name value="Depeche Mode Tournée 2013 - Italy Tour"/>
		<homepage href="http://www.concertionline.com/concerti-rock/biglietti-tour-depeche-mode-a-milano-e-roma/"/>
	</information>
	<hours>
		<hour event="Stadio San Siro - Milano" type="Tournee" pattern="18/07/2013: 2100-0100." coords="45.478519,9.123931" seeAlso="http://www.eventinbus.com/autobus-concerto/depeche-mode-milano-18-07-2013_109.html"/>
		<hour event="Stadio Olimpico - Roma" type="Tournee" pattern="20/07/2013: 2200-0200." coords="41.933923,12.454748" seeAlso="http://www.eventinbus.com/autobus-concerto/depeche-mode-roma-20-07-2013_110.html"/>
	<hours>
</venue>
```

Una singola *venue* è specificata per l'intera tournée mentre le tappe sono identificate dalle *hour*. I *Depeche Mode* faranno tappa allo stadio di *San Siro* a *Milano* e allo stadio *Olimpico* di *Roma* in diverse date con possibili orari differenti. Il *pattern* come al solito è formato da due parti: una data (o una serie di date) e una fascia orario (o una serie di fasce orario).
Infine, l'attributo *coords* indica la posizione geografica precisa dell'evento.

A questo esempio, si sarebbero potuti aggiungere altri attributi, come il prezzo del biglietto, ma l'intenzione è quella di far notare la possibilità di fissare una certa locazione a una serie di eventi.

3.2.8 Trasporti pubblici

Come ultimo esempio, si vuole far vedere l'utilizzo della nostra proposta per gestire gli orari delle linee degli autobus di un'azienda di trasporti, in contrapposizione alla soluzione proposta da

Google (**GTFS [3]**).
I dati dell'esempio dato non sono da prendere come reali.

Un'azienda di autotrasporti pubblici (bus) decide di realizzare una tabella orario per le proprie linee urbane. Ogni linea deve transitare su una serie di fermate ad un certo orario.
Così, siccome ogni linea può avere più fermate e ogni fermata può avere più linee, l'azienda potrebbe stilare la tabella orario in due modi distinti ma con lo stesso potere espressivo:

1. ogni linea è vista come una *venue* e ogni fermata come un *hour*
2. ogni fermata è vista come una *venue* e ogni linea come un *hour*

Viene scelto il primo caso.
Le fermate dovranno avere un paio di coordinate geografiche.

```
<?xml version="1.0" encoding="UTF-8"?>
<timetable xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.fabiovitali.it/tt    Timetable-prova.xsd"
xmlns="http://www.fabiovitali.it/tt">
        <metadata>
                <this href="http://www.atc.bo.it/orario"/>
                <permanentUri href="http://www.atc.bo.it/2012"/>
                <title value="Orario delle linee urbane della ATC di Bologna - anno 2012"/>
                <author href="http://www.wilky.it/me" showAs="Vincenzo Ferrari"/>
                <created value="2011-11-01"/>
                <validFrom value="2011-11-01"/>
                <validTo value="2012-10-31"/>
                <authoringTool value="none"/>
                <authoritative value="yes"/>
                <previousDocument href="http://www.atc.bo.it/2011/orario"/>
                <source href="http://www.atc.bo.it/2012/orario"/>
        </metadata>
        <venues>
                <venue showAs="Linea 1">
                        <information>
                                <type item="bus" dict="#bustypes" showAs="Autobus"/>
                                <subtype item="line" dict="#linetypes" showAs="Linea Bus"/>
                                <name value="Linea 1"/>
                                <homepage href="http://www.atc.bo.it/linea-1"/>
                        </information>
                        <hours>
                                <hour event="Fermata Porta San Donato" type="Stop"
pattern="lun,mar,mer,gio: 0800,0830,0900,0930,1000,1030. ven,sab,dom:
0900,0930,1000,1030." coords="44.498281,11.356173"/>
                                <hour event="Fermata Porta San Donato - Festivi"
type="StopFestivity" pattern="#festivi: ." coords="44.498281,11.356173" dict="#festivi"/>
                                <hour event="Fermata Porta San Vitale" type="Stop"
pattern="lun,mar,mer,gio: 0815,0845,0915,0945,1015,1045. ven,sab,dom:
```

```
0915,0945,1015,1045." coords="44.493934,11.356988"/>
                        <hour event="Fermata Porta San Vitale - Festivi"
type="StopFestivity" pattern="c,n,ss: 0815,0845,0915,0945,1015,1045. lda,lib,lav:
0915,0945,1015,1045. e,p,r,f,sp,t,i: -." coords="44.493934,11.356988" dict="#festivi"/>
                        <hour event="Fermata Porta Mazzini" type="Stop"
pattern="lun,mar,mer,gio: 0830,0900,0930,1000,1030,1100. ven,sab,dom:
0930,1000,1030,1100." coords="44.490077,11.357546"/>
                        <hour event="Fermata Porta Mazzini - Festivi" type="StopFestivity"
pattern="e,p,r,f,sp,t,i: 0830,0900,0930,1000,1030,1100. lda,lib,lav: 0930,1000,1030,1100. c,n,ss:
-." coords="44.490077,11.357546" dict="#festivi"/>
                    <hours>
                </venue>
                <venue showAs="Linea 2">
                    <information>...</information>
                    <hours>
                        <hour event="Fermata Porta San Donato" type="Stop"
pattern="lun,mar,mer,gio: 1200,1300,1400,1500,1600,1700. ven,sab,dom:
1600,1700,1800,1900." coords="44.498281,11.356173"/>
                        <hour event="Fermata Porta San Vitale" type="Stop"
pattern="lun,mar,mer,gio: 1300,1500,1700,1900. ven,sab,dom: 1200,1600,1900."
coords="44.493934,11.356988"/>
                        <hour event="Fermata Porta Santo Stefano" type="Stop"
pattern="lun,mar,mer,gio: 1300,1400,1600,1700. ven,sab,dom: -."
coords="44.484504,11.356430"/>
                    <hours>
                </venue>
                <venue showAs="Linea 3">
                    <information>...</information>
                    <hours>
                        <hour event="Fermata Porta San Donato" type="Stop"
pattern="lun,mar,mer,gio: 0800,1030,1200,1630,1900. ven,sab: 0900,0930,1000,1030. dom: -."
coords="44.498281,11.356173"/>
                        <hour event="Fermata Porta Mascarella" type="Stop"
pattern="lun,mar,mer,gio: 0745,1015,1145,1615,1845. ven,sab: 0845,0915,0945,1015. dom: -."
coords="44.502413,11.352739"/>
                        <hour event="Fermata Porta Galliera" type="Stop"
pattern="lun,mar,mer,gio: 0715,0945,1115,1545,1815. ven,sab: 0815,0845,0915,0945. dom:
1000, 1300, 1500, 1700, 1900." coords="44.504800,11.345444"/>
                    <hours>
                </venue>
        </venues>
        <locale>
                <weekDays value="lun mar mer gio ven sab dom"/>
                <months value="gen feb mar apr mag giu lug ago set ott nov dic"/>
                <dateFormat value="D d/m/Y"/>
                <weekStartsWith value="1" showAs="Lunedì"/>
                <dicts>
                    <dict id="bustypes" href="http://www.atc.bo.it/bustypes"/>
                    <dict id="linetypes" href="http://www.atc.bo.it/linetypes"/>
                    <dict id="festivi">
```

```
                        <items>
                                <item id="c" value="01/gen" showAs="Capodanno"/>
                                <item id="e" value="06/gen" showAs="Epifania"/>
                                <item id="p" value="31/mar" showAs="Pasqua"/>
                                <item id="lda" value="01/apr" showAs="Lunedì
dell'Angelo"/>
                                <item id="lib" value="25/apr" showAs="Festa della
Liberazione"/>
                                <item id="lav" value="01/mag" showAs="Festa dei
Lavoratori"/>
                                <item id="r" value="02/giu" showAs="Festa della
Repubblica"/>
                                <item id="f" value="15/ago" showAs="Ferragosto"/>
                                <item id="sp" value="04/ott" showAs="San Petronio
(patrono di Bologna)"/>
                                <item id="t" value="01/nov" showAs="Tutti i Santi"/>
                                <item id="i" value="08/dic" showAs="Immacolata
Concezione"/>
                                <item id="n" value="25/dic" showAs="Natale"/>
                                <item id="ss" value="26/dic" showAs="Santo Stefano"/>
                        </items>
                </dict>
        </dicts>
    </locale>
</timetable>
```

Per migliorare il paragone con l'alternativa di *Google* è stato necessario portare un esempio completo di *metadata* e *locale*.

Come per *agency.txt* di **GTFS [3]**, *metadata* specifica tutte le informazioni inerenti al documento e all'azienda di trasporti pubblici.
Nella sezione *venues* sono presenti tutte le informazioni sulle linee e sulle loro fermate, come nei file *stops.txt*, *stop_times.txt*, *calendar.txt* e *calendar_dates.txt* di GTFS [3].
Infine, nella sezione *locale*, sono presenti le informazioni locali e i dizionari usati per definire gli orari.

Analizziamo la prima venue: *Linea 1*.
Dalla *information* notiamo che si tratta di un autobus, in particolare di una linea urbana della città di Bologna. Nelle altre *venue* queste informazioni non saranno ripetute.
Nella sezione *hours* invece è possibile visualizzare le fermate che la suddetta linea farà nei giorni feriali e in quelli festivi. Vediamole.

In tutto vi sono tre fermate: *Porta San Donato*, *Porta San Vitale* e *Porta Mazzini*.
La *Linea 1* passa per le porte nei seguenti orari dei seguenti giorni:

Giorni	Porta San Donato	Porta San Vitale	Porta Mazzini
Lunedì	08:00, 08:30, 09:00, 09:30, 10:00, 10:30	08:15, 08:45, 09:15, 09:45, 10:15, 10:45	08:30, 09:00, 09:30, 10:00, 10:30, 11:00

Martedì	08:00, 08:30, 09:00, 09:30, 10:00, 10:30	08:15, 08:45, 09:15, 09:45, 10:15, 10:45	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Mercoledì	08:00, 08:30, 09:00, 09:30, 10:00, 10:30	08:15, 08:45, 09:15, 09:45, 10:15, 10:45	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Giovedì	08:00, 08:30, 09:00, 09:30, 10:00, 10:30	08:15, 08:45, 09:15, 09:45, 10:15, 10:45	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Venerdì	09:00, 09:30, 10:00, 10:30	09:15, 09:45, 10:15, 10:45	09:30, 10:00, 10:30, 11:00
Sabato	09:00, 09:30, 10:00, 10:30	09:15, 09:45, 10:15, 10:45	09:30, 10:00, 10:30, 11:00
Domenica	09:00, 09:30, 10:00, 10:30	09:15, 09:45, 10:15, 10:45	09:30, 10:00, 10:30, 11:00
Capodanno	-	08:15, 08:45, 09:15, 09:45, 10:15, 10:45	-
Epifania	-	-	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Pasqua	-	-	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Lunedì dell'Angelo	-	09:15, 09:45, 10:15, 10:45	09:30, 10:00, 10:30, 11:00
Festa della Liberazione	-	09:15, 09:45, 10:15, 10:45	09:30, 10:00, 10:30, 11:00
Festa dei Lavoratori	-	09:15, 09:45, 10:15, 10:45	09:30, 10:00, 10:30, 11:00
Festa della Repubblica	-	-	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Ferragosto	-	-	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
San Petronio	-	-	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Tutti i Santi	-	-	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Immacolata Concezione	-	-	08:30, 09:00, 09:30, 10:00, 10:30, 11:00
Natale	-	08:15, 08:45, 09:15, 09:45, 10:15, 10:45	-
Santo Stefano	-	08:15, 08:45, 09:15, 09:45, 10:15, 10:45	-

I giorni festivi sono specificati con l'apposito dizionario e il *pattern* riesce in poche righe a esprimere una tabella così completa.
Da notare l'estrema facilità con cui si possono raggruppare tutti gli orari uguali nei singoli giorni.

Inoltre, vi è l'attributo *coords* che specifica le coordinate geografiche di ogni fermata.
Quindi, la *Linea 1* passa per la fermata di *Porta San Donato* tutti i giorni settimanali, esclusi quelli festivi (domeniche a parte), come *Google* utilizza nel file *calendar_dates.txt* per specificare le eccezioni di GTFS [3]: la differenza sta nel fatto che Google specifica solo due tipi di eccezioni (in servizio o fuori servizio) mentre in questo caso è possibile specificare anche orari differenti dall'orario settimanale.
Per le altre fermate invece, la *Linea 1* ha orari differenti e in alcuni giorni festivi è attiva mentre in altri è fuori servizio.

Nelle altre linee le informazioni sono simili quindi non staremo qui ad analizzarle tutte quante. Vogliamo però far notare con quale semplicità è possibile esprimere così tante informazioni, mantenuto nello stesso file in un formato aperto e comodo per l'interscambio di dati.

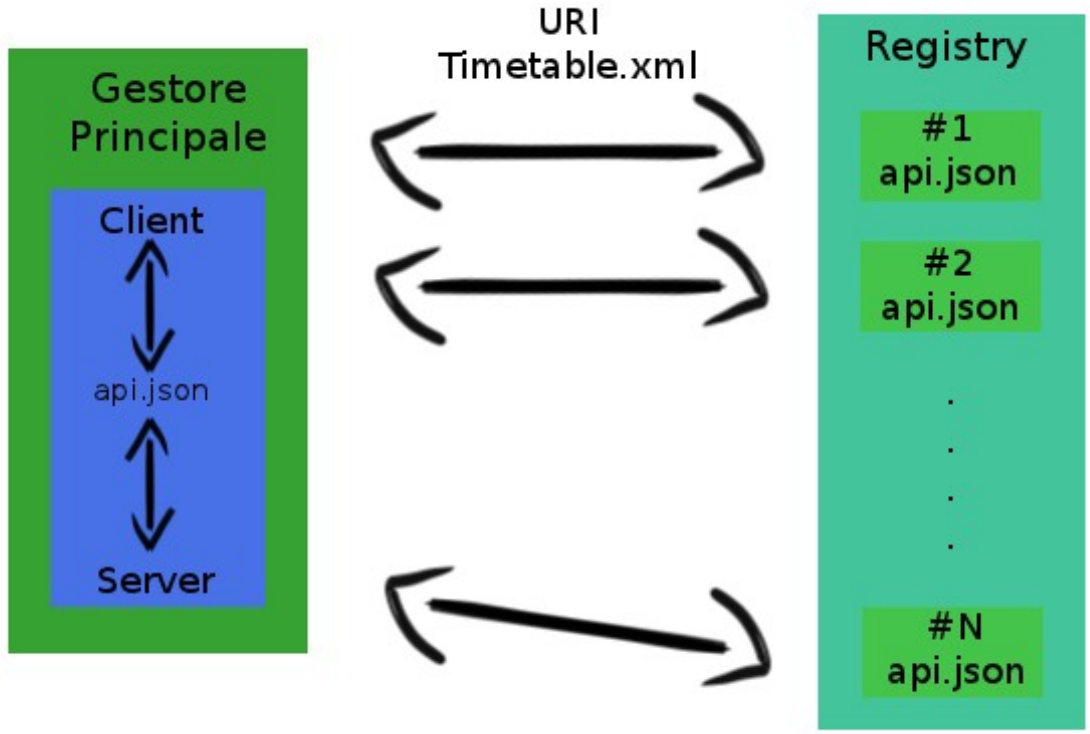
A questo punto, l'azienda di trasporti, non dovrebbe far altro che registrare l'URI della propria tabella orario nei *Registry* che più gli aggrada: per maggiori informazioni su questo punto, si guardi il **Capitolo 4**.

Capitolo 4

Universal Timetable Manager

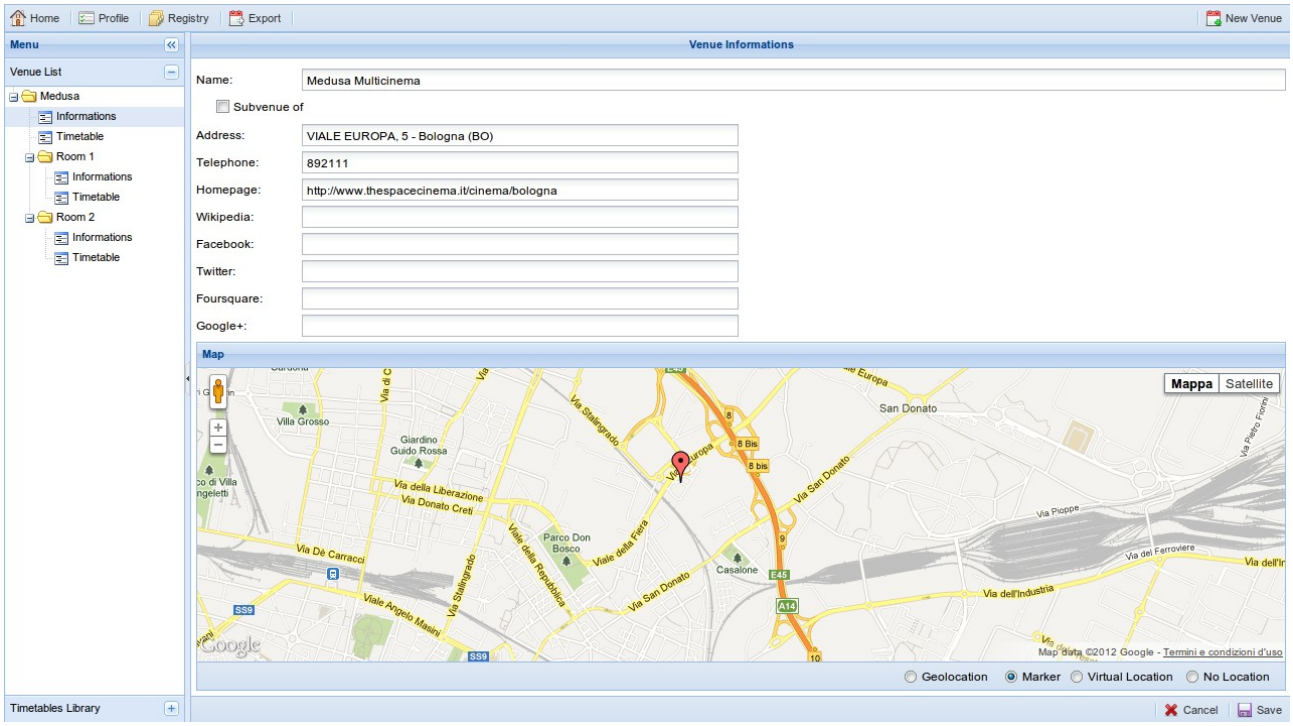
UTM (*Universal Timetable Manager*) è un'applicazione atta a mostrare le potenzialità e la gestione della proposta qui fatta.
Si tratta di un'applicazione web in grado di gestire una serie differente di orari di esercizi commerciali ed enti pubblici. Essa è in grado di configurare una lista di tabelle orario, suddivise in *venue* (locazioni).
L'applicazione è realizzata in due parti: il **gestore principale** (con cui si gestiscono le tabelle orari) e il **Registry** (un registro online di tabelle orario).

Di seguito una rappresentazione della comunicazione tra il gestore principale e il Registry:



4.1 Gestore principale

Il gestore è la parte principale dell'applicazione UTM e ha il compito di fornire all'utente un'interfaccia comoda e completa per la realizzazione e la gestione di tabelle orario di diverso tipo. E' suddiviso in un client (*front-end*) e un server (*back-end*): questi comunicano attraverso un protocollo definito in un file JSON [33] (*api.json*), rispettando così le specifiche **REST** (*Representational State Transfer*) [34].



Inizialmente, l'applicazione viene lanciata con una serie di configurazioni preimpostate che l'utente potrà modificare man mano.
Il client, realizzato con il **Model-View-Controller** di **ExtJS** [35], carica tutte le informazioni che risiedono sul server *(Timetable.xml* e *RegistryList.xml*) in memoria, utilizzando form dinamici per modificarle in locale, inviandole poi al server in un secondo momento.
Sia il client che il server caricano un file comune chiamato *api.json* che permetterà loro di comunicare tramite il protocollo HTTP.
Di seguito un esempio:

```
{
        "baseURI": "http://localhost:5000" ,
        "updateMetadata": {
                "url": "/metadata/update" ,
                "method": "POST"
        } ,
        "createItem": {
```

```
                "url": "/locale/dict/item/create" ,
                "method": "PUT"
        } ,
        ...
}
```

Ogni chiamata è identificata da un nome (come *updateMetadata*) ed è formata da due elementi: *url*, l'url della chiamata, e *method*, il metodo HTTP con cui si accede alla risorsa.

4.1.1 Client

Il client è stato realizzato con l'ausilio del framework Javascript **ExtJS** [35] e comunica col server tramite chiamate *AJAX* [36], trasmettendo e ricevendo dati in formati standard (*XML* & *JSON*).

Segue l'albero delle directory e dei file:

```
.
├── app
│   ├── controller
│   │   ├── Menu.js
│   │   ├── NewVenue.js
│   │   ├── Profile.js
│   │   ├── Registry.js
│   │   ├── Sidebar.js
│   │   └── venue
│   │       ├── Information.js
│   │       ├── timetable
│   │       │   ├── direct
│   │       │   │   └── Direct.js
│   │       │   ├── festivity
│   │       │   │   ├── Festivity.js
│   │       │   │   └── NewFestivity.js
│   │       │   ├── uniplex
│   │       │   │   ├── Show.js
│   │       │   │   └── Uniplex.js
│   │       │   └── weekly
│   │       │       └── Weekly.js
```

```
│   │       └── Timetable.js
│   ├── data
│   │   └── TimetableLibrary.json
│   ├── model
│   │   ├── Dict.js
│   │   ├── Hour.js
│   │   ├── Information.js
│   │   ├── Item.js
│   │   ├── Locale.js
│   │   ├── Metadata.js
│   │   ├── Registry.js
│   │   ├── Timetable.js
│   │   ├── TimetableLibrary.js
│   │   ├── venue
│   │   │   └── timetable
│   │   │       ├── festivity
│   │   │       │   └── Festivity.js
│   │   │       ├── uniplex
│   │   │       │   └── Uniplex.js
│   │   │       └── weekly
│   │   │           └── Weekly.js
│   │   ├── Venue.js
│   │   └── VenueList.js
│   ├── store
│   │   ├── LinearizedVenueList.js
│   │   ├── NewVenueTimetableLibrary.js
│   │   ├── registry
│   │   │   ├── Subscribed.js
│   │   │   └── Unsubscribed.js
│   │   ├── Registry.js
│   │   ├── TimetableLibrary.js
│   │   └── VenueList.js
│   ├── Util.js
│   └── view
│       ├── Menu.js
```

```
│   ├── NewVenue.js
│   ├── Profile.js
│   ├── Registry.js
│   ├── Sidebar.js
│   ├── venue
│   │   ├── Information.js
│   │   ├── timetable
│   │   │   ├── direct
│   │   │   │   └── Direct.js
│   │   │   ├── festivity
│   │   │   │   ├── Festivity.js
│   │   │   │   └── NewFestivity.js
│   │   │   ├── uniplex
│   │   │   │   ├── Show.js
│   │   │   │   └── Uniplex.js
│   │   │   └── weekly
│   │   │       └── Weekly.js
│   │   └── Timetable.js
│   └── Viewport.js
├── app.js
├── images/
└── index.html
```

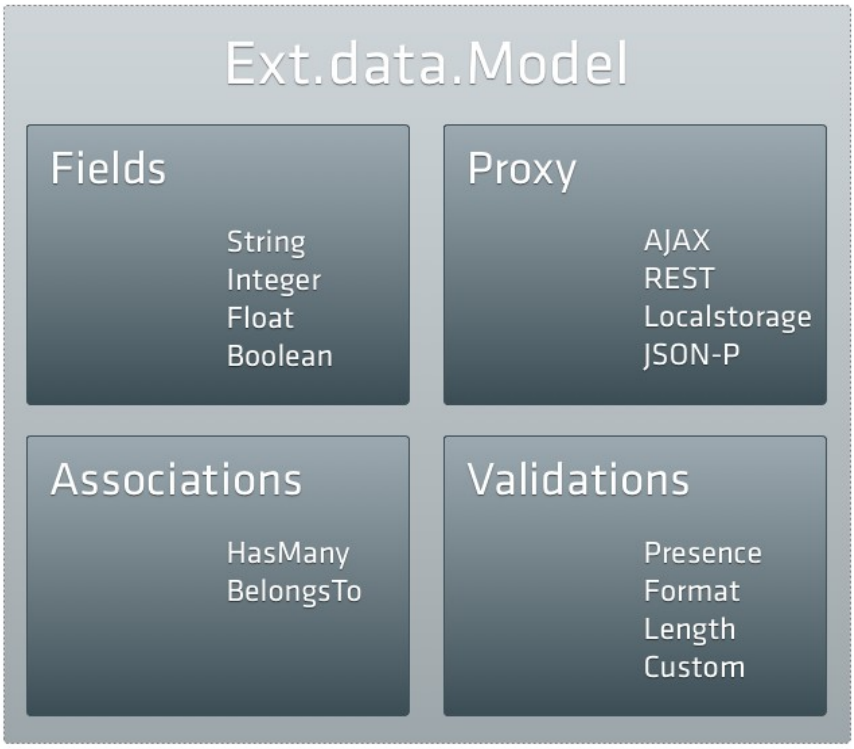
Il client è formato dalla cartella *app* (l'applicazione stessa), dalla cartella *extjs-4.1.1* (il framework javascript) e dai file *index.html* e *app.js*: questo ultimo file in particolare inizializza la struttura **MVC** del client.
app rappresenta l'**MVC** (*model-view-controller*) dell'applicazione in cui risiedono i *model*, le *view*, i *controller* delle *view* e gli *store* (aggregati di *model*).
Le quattro sotto cartelle principali (*controller, view, model, store*) sono speculari e ogni file è soprannominato con lo stesso nome: ciò che li differenzia è il percorso.
Per ogni *view* vi è un *controller* appropriato e solitamente anche un *model* e uno *store*.

Bisogna porre particolare attenzione ai seguenti modelli, siccome si tratta di una complicata associazione, a struttura gerarchica.:

- *Timetable*
- *Metadata*
- *Venue*

- *Information*
- *Hour*
- *Locale*
- *Dict*
- *Item*

Il modello principale di tale associazione è *Timetable*. Essa ha due associazioni '*uno a uno*' (*Metadata* e *Locale*) e una '*uno a molti*' (*Venue*).
Metadata rappresenta il modello per i metadati del documento.
Venue rappresenta le *venue* definite dall'utente e ve ne possono essere diverse. Ogni *Venue* ha due associazioni: una '*uno a uno*' (*Information*) e una '*uno a molti*' (*Hour*).
Information definisce le informazioni della *venue*, mentre *Hour* la lista delle tabelle orario (*timetable*) di cui la *venue* è fornita.
Infine *Locale* identifica i dati locali dell'applicazione e ha un'unica associazione '*uno a molti*': *Dict*.
Dict rappresenta la lista dei dizionari presenti e anche esso ha una relazione '*uno a molti*' con *Item*, siccome ogni *Dict* può avere più *Item* che sono gli elementi componenti del dizionario.



Ciò che l'utente visualizza inizialmente è la pagina centrale (*Homepage*) dove sono visualizzate le informazioni riassuntive e le linee guida dell'applicazione.
Da qui potrà spostarsi tramite il menu di navigazione nelle seguenti pagine: *Profile*, *Registry* ed *Export*.

Profile è la pagina che permette di modificare i metadati del documento *Timetable.xml*: questi dati riguardano l'utente e la sua attività. E' inoltre possibile specificare un intervallo di validità del documento: al termine di tale intervallo, il server provvederà a creare una nuova versione del file, rinominata con la data di scadenza (*versioning*).

Registry invece è composta da due griglie: la prima visualizza i registri a cui l'utente ha iscritto la sua attività mentre la seconda contiene i registri rimanenti a cui l'utente può eventualmente iscriversi.
Le informazioni visualizzate per registro sono: un titolo, l'url e una breve descrizione.

Export è la pagina da cui l'utente può scaricare direttamente il file *Timetable.xml* in diversi formati: *XML*, *HTML*, *CSV* e *JSON*.

Sulla barra sinistra sono disponibili due menu: *Venue List* e *Timetable Library*.

Venue List è la lista delle venue realizzate fino a quel momento dall'utente, visualizzate in una struttura ad albero: ogni venue è formata da due pagine (*Information* e *Timetable*) e da una serie di subvenue (venue ereditarie).
La pagina *Information* permette di modificare le informazioni relative alla venue (e alle sue sottovenue): è possibile modificare il nome, la venue da cui si eredita, una serie di piattaforme sociali online e le coordinate geografiche disposte su una mappa (*Google Maps*).
Ogni venue può avere una geolocazione fisica, virtuale o non averla.

Invece, la pagina *Timetable* è un pannello a schede e ogni scheda è un particolare form che rappresenta una tabella orario.
L'utente può decidere di aggiungere al volo nuovi form direttamente dal menu *Timetable Library*. Questo menu contiene tutti i form preconfezionati, disposti in collezioni: ogni collezione può avere uno o più form.

Infine, l'utente può decidere di creare una nuova venue da zero oppure ereditando le caratteristiche di un'altra venue.
Nel primo caso sarà necessario scegliere una collezione di base (*Shop*, *Multi Screen Cinema*, …) mentre nel secondo si potrà scegliere la venue genitore da cui ereditare tutte le informazioni.
Nel caso l'utente decidesse di voler aggiungere nuovi form (timetable), dovrà seguire la seguente procedura:

1. decidere il nome del form: *Festivity*
2. creare una nuova cartella nella directory delle view del client con lo stesso nome del form, tutto in minuscolo: *app/view/venue/timetable/festivity*
3. inserire nella suddetta cartella tutte le view che compongono il form: *Festivity.js, NewFestivity.js*
4. la view principale deve contenere un metodo chiamata **getHour** che, quando invocato, ha il compito di restituire l'elemento <*hour*> definito con pattern, type (necessari) e altri attributi
5. creare la cartella dei controller nella directory dei controller del client, sempre in minuscolo: *app/controller/venue/timetable/festivity*
6. creare i relativi controller nella suddetta cartella: *Festivity.js, NewFestivity.js*
7. lo stesso può essere fatto nel caso si vogliano usare *Model* e *Store*
8. aggiornare il file *app.js* nella cartella principale del client, aggiungendo alla lista le view, i controller, i model e gli store appena creati, in maniera tale che il client sappia quali file dover caricare
9. modificare il file *TimetableLibrary.json* situato nella cartella *app/data* per poter inserire il suo nuovo form nella lista delle timetable disponibili
10. lo stesso dovrà fare nel file *Collections.json* situato sul server nella cartella *data*

Se invece l'utente desidera creare una nuova collezione con i form già disponibili o con dei nuovi da
lui realizzati, dovrà modificare il file *data/Collections.json* situato sul server nel seguente modo:

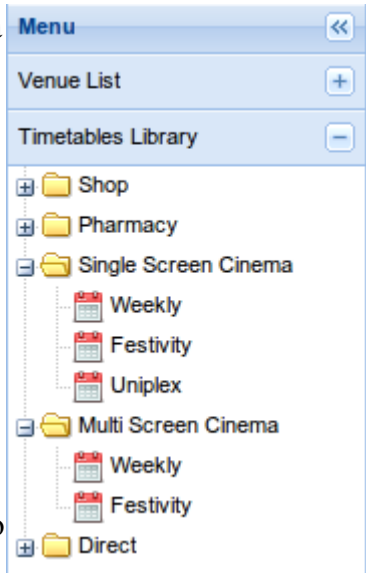
- l'elemento principale è il nome della collezione: *Multiplex*
- questo è composto da due sotto elementi: *venue* e *subvenues*
- *venue* specifica la venue principale, il cui nome sarà definito durante la compilazione, ed è un oggetto in cui è possibile specificare i form che lo comporranno attraverso l'elemento *timetables*: questo accetta un array di stringhe, i nomi delle timetable: *Weekly*, *Festivity*
- *subvenues* invece definisce le venue figlie che ereditano tutte le informazioni: questo è fatto per poter generare al volo una serie di sotto venue. Ogni subvenue ha un nome predefinito e una serie di form (timetable), come visto precedentemente.

Di base, l'utente può usare le seguenti tabelle orario:

- *Weekly*: è la tabella dell'orario settimanale. I giorni della settimana sono specificati nella sezione *locale* della *Timetable.xml*
- *Festivity*: è la tabella dell'orario delle festività. Vi è una lista preimpostate di festività, come il Natale, la Pasqua, Ferragosto, e via dicendo: l'utente potrà rimuoverle, modificarle e aggiungerne di nuove. Ogni festività è un *Item* del dizionario festivi, anch'esso situato nella sezione *locale*.
- *Uniplex*: è la tabella per specificare le date e l'orario delle proiezioni di un singolo film. Contiene inoltre informazioni come il prezzo, la durata e il titolo del film. E' possibile (e consigliato) usare più istanze di questo form per gestire i vari film in programma in una singola sala, sia che il cinema sia monosala che multisala.
- *Direct*: chiama anche l'**Arma X**, è il form puro, atto a scrivere direttamente l'elemento *hour* (la tabella orario) senza l'utilizzo di una struttura preimpostata. L'utente deve specificare a mano il tipo di tabella, il *pattern* degli orari e tutti gli altri attributi necessari e/o facoltativi.

Invece, le collezioni di base sono le seguenti:

- *Shop*: è formato dalle tabelle orario *Weekly* e *Festivity* ed è la collezione per le attività commerciali semplici, come negozi, alimentari, parrucchieri, ossia quegli esercizi che non hanno particolari esigenze.
- *Pharmacy*: di fatto è una copia della collezione *Shop* ma dà la possibilità all'utente di inquadrare meglio la propria attività. Questa collezione serve per mostrare l'utilizzo reale delle collezioni: essendo aggregati di tabelle orario, possono essere duplicati a piacere se questo aiuta l'utente a riconoscere meglio la sua attività nell'applicazione, senza dover rimanere nel generico.



- *Single Screen Cinema*: è formata da tre tabelle orario, ossia *Weekly*, *Festivity* e *Uniplex*. *Weekly* e *Festivity* riguardano l'orario d'apertura del cinema, mentre *Uniplex* è riferito all'unica sala presente nel cinema. Per ogni film in proiezione in tale sala sarà usato un diverso form *Uniplex*. Questa collezione è fatta per quei cinema che hanno una sola sala e hanno necessità di creare una tabella di proiezioni per più film.
- *Multi Screen Cinema*: è l'evoluzione della collezione *Single Screen Cinema*. Al momento della creazione di una nuova *venue* con tale collezione, vengono create due ulteriori sotto *venue*. La prima (la venue genitore) ha soltanto due tabelle orario, *Weekly* e *Festivity*, utili per specificare l'orario di apertura del multisala, mentre le sotto *venue* saranno dotate dell'unica tabella orario *Uniplex*, atta a definire le informazioni sulla proiezione del singolo film. Ogni sala è quindi rappresentata da una sotto *venue*, utile per specificare informazioni extra come una proprio pagina web, coordinate geografiche caso mai fosse locata in una zona differente, eccetera. Questa collezione è suggerita a tutti quei cinema che hanno due o più sale.
- *Direct*: contiene l'unica tabella orario *Direct*. Questa collezione serve per dare all'utente la possibilità di aggiungere al volo la suddetta tabella orario ad ogni *venue*.

4.1.2 Server

Il server invece è stato realizzato in **Python** [37], tramite l'utilizzo del microframework **Flask** [38] più l'aggiunta di librerie esterne per gestire file XML [14] e inoltrare richieste *HTTP* al Registry.
Il server è di fatto un'applicazione stand-alone e comunica col client tramite la tecnologia **CORS** (*Cross-Origin Resource Sharing*) [39] per ovviare ai problemi delle richieste *AJAX cross-domain*: infatti il server viene messo in ascolto su una generica porta e le richieste, seppur fatte da locale, vengono interpretate come richieste *cross-domain*.
Inoltre gestisce due file XML: *Timetable.xml* e *RegistryList.xml*.

Di seguito invece l'albero delle directory e dei file del server:

```
.
├── CORS.py
├── data
│   ├── Collections.json
│   ├── RegistryList.xml
│   └── Timetable.xml
├── Router.py
└── Server.py
```

Router.py è il file principale che richiama al suo interno tutti gli altri file.
Server.py è un modulo separato che gestisce tutte le chiamate definite in *Router.py*: ha anche il compito di gestire il 'database' XML (*Timetable.xml*) e le interazione col *Registry*.
CORS.py è invece una libreria esterna che permette di abilitare la tecnologia **CORS** sulle chiamate dal client verso il server.

Il server viene lanciato invocando il file *Router.py*:

```
user@home:/var/www/UTM/server$ python Router.py
```

Tale file ha il compito di gestire ogni singola chiamata alle risorse. Le chiamate sono di tipo **CRUD** (*Create, Read, Update, Delete*).

Timetable.xml è il cuore dell'applicazione: ivi risiedono tutte le informazioni relative alle tabelle orario e alle venue.
RegistryList.xml invece contiene la lista dei registri a cui l'utente può iscriversi attraverso il client. La sua struttura XML è siffatta:

```
<?xml version='1.0' encoding='utf-8'?>
<registries>
	<registry uri="url_del_registry" description="breve descrizione" showAs="titolo" subscribed="false"/>
	....
</registries>
```

Ogni registro è identificato da un URI (che fa anche da codice univoco), una descrizione, un titolo e un flag che specifica se l'attuale gestore è iscritto o no ad esso.
Il server modifica i suddetti file in base alle richieste del client.

Infine, il server gestisce il file delle collezioni: *Collections.json*.
Questo file di configurazione permette di sapere al server quali *venue* deve realizzare per una certa collezione e con quali tabelle orario.

Di seguito un esempio:

```
{
	"Pharmacy": {
		"venue": {
			"timetables": ["Weekly", "Festivity", "Direct"]
		}
	},
	"Shop": {
		"venue": {
			"timetables": ["Weekly", "Festivity", "Direct"]
		}
	},
```

```
        "Uniplex": {
                "venue": {
                        "timetables": ["Weekly", "Festivity", "Uniplex", "Direct"]
                }
        } ,
        "Multiplex": {
                "venue": {
                        "timetables": ["Weekly", "Festivity"]
                } ,
                "subvenues": [{
                        "name": "Room 1" ,
                        "timetables": ["Uniplex"]
                } , {
                        "name": "Room 2" ,
                        "timetables": ["Uniplex"]
                }]
        }
}
```

Quando il client chiede di creare una nuova *venue*, il server recupera il contenuto dell'oggetto e comincia a scandire le *venue* e le *subvenues*: per ognuna di essa controlla anche l'elenco delle tabelle orario (*timetables*) e ciò che viene generato è il codice XML della venue che sarà salvato nella *Timetable.xml*.
Se ad esempio volessimo creare una nuova venue avente collezione *Multiplex*, il client non dovrebbe far altro che inviare tale richiesta e il server creerebbe un pezzo di codice XML [14] con la seguente struttura:

```
<venue id=”f98jkd893” showAs="Nome della venue specificato dall'utente">
        <information>
                <name value=”Nome della venue”/>
        </information>
        <hours>
                <hour type="Weekly" pattern=""/>
                <hour type="Festivity" pattern=""/>
        </hours>
</venue>
<venue showAs="Sala 1">
        <information>
                <subvenueOf href="#f98jkd893"/>
                <name value=”Sala 1”/>
        </information>
        <hours>
                <hour type="Uniplex" event="" pattern="" duration="" seeAlso=""/>
        </hours>
</venue>
<venue showAs="Sala 2">
        <information>
                <subvenueOf href="#f98jkd893"/>
```

```
            <name value=”Sala 2”/>
        </information>
        <hours>
            <hour type="Uniplex" event="" pattern="" duration="" seeAlso=""/>
        </hours>
</venue>
```

Essendo una *venue* vergine è necessario che le tabelle orario (*Weekly*, *Festivity*, *Uniplex*) siano completamente da definire, di modo che l'utente possa riempirle con calma in un secondo momento.

4.2 Registry

Si tratta un'applicazione stand-alone che ha il compito di gestire i server registrati e di fornire informazioni su di essi. E' realizzata in *Python* con l'ausilio del micro framework **Flask**.

Lo scopo del Registry è quello di mantenere una lista di gestori di tabelle orario, dando possibilità a programmi di aggregazione (aggregatori) di poter recuperare le informazioni inerenti agli orari delle varie attività commerciali registrate. Come paragone si può prendere *Google* come registro per i feed **GTFS [3]**.

All'avvio carica due file principali: *api.json* e *data/URI.xml*.
Il primo definisce il protocollo che ogni gestore deve caricare per poter comunicare col registro: ivi sono definite le chiamate formate da un indirizzo url e un metodo.
Il secondo è il 'database' in cui sono salvati tutti i gestori iscritti; di seguito la struttura XML:

```
<?xml version='1.0' encoding='utf-8'?>
<list>
        <uri href="url_del_gestore" id="id_univoco" showAs="titolo" />
</list>
```

Ogni gestore è formato da:

- ID univoco: identificato dal tag id
- URI: identificato dal tag href
- titolo: identificato dal tag showAs

Il gestore può chiedere al *Registry* di iscriversi o di de-iscriversi.
Può anche chiedere di poter cambiare le proprie informazioni, come l'URI o il titolo.
Ogni altra applicazione può interrogare il Registry per sapere quali sono i gestori registrati e possono richiederlo in diversi formati: *XML*, *JSON* o *CSV*.

Conclusioni

Dulcis in fundo, le conclusioni.
Dopo aver visto le problematiche odierne date dalla mancanza di un reale modello per la condivisione e l'interscambio di dati di tabelle orario, si sono viste le limitazioni che hanno le attuali soluzioni proposte da aziende e associazioni internazionali.
Dopodiché, si è data la nostra proposta che coinvolge un nuovo modello più sofisticato e personalizzabile di tabella orario.
A differenza degli altri modelli, il nostro propone tre cose innovative:

1. *localizzazione*: l'utente finale può scegliere il proprio formato di data, di ora, i giorni della settimana così come quelli del mese, senza doversi rifare sempre a un unico formato internazionale (generalmente quello inglese).
2. *ereditarietà*: come si è visto nel terzo capitolo, è possibile stilare *venue* in forma gerarchica, permettendo così all'utente di suddividere le proprie tabelle orario senza dover ripetere ogni volta le informazioni associate ad essa. Tale strumento permette anche di generare tabelle orario complesse e di snellire il codice che, nell'ambito delle applicazioni web, significa diminuire enormemente il carico di dati che devono scorrere sulla rete.
3. *pattern efficiente*: un nuovo pattern per gestire intervalli temporali di diverso tipo, come ore e date. Questo nuovo modo di concepire gli orari settimanali di apertura di un negozio, così come le proiezioni di un film di una certa sala, e via dicendo, permette chiarezza e al tempo stesso semplicità nel rappresentare tali dati.

Questi tre punti sono il cuore di questa tesi e invitano a riflettere sulla possibilità di gestire gli orari con il formato e le esigenze del singolo legato alla possibilità di interagire con orari stilati in formati differenti ed esigenze differenti, permettendo un migliore interscambio di dati tra applicazioni.

Infine, si è dimostrata la validità di questa tesi tramite la creazione di un'applicazione web apposita.
Compito dell'applicazione non è soltanto di stilare il documento finale ma anche quello di gestirlo, creare nuove versioni a piacimento (*versioning*) e poter comunicare con registri online (*Registry*). Questo ultimo sforzo serve anche per far capire che ogni attività può comodamente mantenere le informazioni sul proprio server, evitando di dover aggiornare i propri dati su altri servizi: si prenda l'esempio di dover ogni volta cambiare la tabella orario su siti differenti che ospitano informazioni inerenti all'attività interessata.
Così, viene a definirsi ancor meglio l'idea di gestione federata dei dati, dove ogni creatore di dati è anche il gestore a cui fare riferimento, fornendo quindi una versione costantemente aggiornata.

Uno sguardo al futuro

Questa tesi è la base di un cambiamento più grande.
Il lavoro infatti non è finito qui e il futuro ci propone una prospettiva migliore.
Oggi giorno, soprattutto coi nuovi dispositivi mobili e la potenza di calcolo e di rappresentazione dei dati che abbiamo a disposizione, risulta necessario avere la possibilità di avere a che fare con dati aggiornati in tempo reale, così come la comodità nel recuperarli e raggrupparli.

Ecco allora che si delinea la necessità di migliorare il lavoro fatto fin'ora: la creazione di un'ontologia formale è il primo passo da compiere per dare a chiunque la possibilità di scrivere applicazioni che utilizzino il nostro modello: *OWL [20]* è il linguaggio proposto per tale compito.

Il secondo step è quello di ampliare il modello con il *Semantic Web*, o meglio con *Open Linked Data*.
Introduciamo quindi l'uso di ontologie esterne, come *FOAF [19]* per definire le persone, *Dublin Core [18]* per specificare i metadati, *geoRDF [40]* per definire meglio le coordinate geografiche delle singole *venue*, e via discorrendo.
L'uso di *RDF [15]* diventa quindi necessario per la stesura di una nuova versione del modello da noi proposto.

Infine, la creazione di *aggregatori* che mostrino le totali potenzialità che questo modello può dare.
Immaginate di essere a casa, seduti comodamente davanti al vostro PC, desiderosi di conoscere quali film ci sono presso tutti i cinema di Bologna: lanciate il programma di aggregazione, riempite un semplice filtro di ricerca con le parole '*cinema*' e '*Bologna*', selezionate la data di oggi, ed eccovi comparire lì, davanti a voi senza sforzo alcuno, la lista di tutti i film di tutti i cinema di Bologna in programmazione la sera stessa.
E che dire del viaggiatore che ha sempre bisogno di conoscere l'orario del prossimo aereo o del prossimo treno? Eccolo tirare fuori dalla tasca il suo *smartphone*, lanciare l'aggregatore e ritrovarsi immediatamente il primo mezzo di trasporto disponibile più vicino a lui.

E' questo il percorso che il sottoscritto desidera intraprendere nell'immediato futuro, per dimostrare l'efficacia del nostro modello e per dare la possibilità a tutti gli interessati di usarne e, spero, abusarne il prima possibile.

Bibliografia

[0] RDF Calendar - an application of the Resource Description Framework to iCalendar Data
http://www.w3.org/TR/rdfcal/
Retrieved: [18-10-2012]

[1] Restaurant Recommendation - A look at Chef Moz
http://www.w3.org/wiki/RestaurantRecommendation
Retrieved: [18-10-2012]

[2] unimaner - University Manager - is a program to manage your time at university
https://code.google.com/p/unimaner/
Retrieved: [18-10-2012]

[3] Google Transit - General Transit Feed Specification
https://developers.google.com/transit/
Retrieved: [18-10-2012]

[4] Federfarma Bologna
http://www.federfarma-bo.it/
Retrieved: [19-10-2012]

[5] Federfarma Bologna - Farmacie di turno
http://www.federfarma-bo.it/index.php?id=23&sez=11
Retrieved: [19-10-2012]

[6] Cineflash - Cinema Multisala
http://www.cineflash.it/
Retrieved: [19-10-2012]

[7] Adobe Flash Player
http://get.adobe.com/it/flashplayer/
Retrieved: [20-10-2012]

[8] API - Application Programming Interface
http://it.wikipedia.org/wiki/Application_programming_interface
Retrieved: [20-10-2012]

[9] HTML5
http://www.w3.org/TR/2011/WD-html5-20110525/
Retrieved: [21-10-2012]

[10] CSS - Cascading Style Sheets
http://www.w3.org/Style/CSS/current-work
Retrieved: [21-10-2012]

[11] Javascript – ECMA
http://www.ecma-international.org/publications/standards/Ecma-262.htm
Retrieved: [21-10-2012]

[12] Orari I Ciclo - Informatica Triennale Unibo
http://orari.web.cs.unibo.it/cgi-bin/2012-2013/I-ciclo/inf/primo.php
Retrieved: [21-10-2012]

[13] CMS - Content Management System
http://it.wikipedia.org/wiki/Content_management_system
Retrieved: [21-10-2012]

[14] XML - eXtensible Markup Language
http://www.w3.org/XML/
Retrieved: [22-10-2012]

[15] RDF - Resource Framework Description
http://www.w3.org/RDF/
Retrieved: [22-10-2012]

[16] W3C - World Wide Web Consortium
http://www.w3.org
Retrieved: [23-10-2012]

[17] RDF Example - Suggestions and Questions
http://www.w3.org/wiki/RestaurantRecommendation
Retrieved: [24-10-2012]

[18] DC - Dublin Core
http://dublincore.org/
Retrieved: [24-10-2012]

[19] FOAF - Friend Of A Friend
http://www.foaf-project.org/
Retrieved: [24-10-2012]

[20] OWL - Web Ontology Language
http://www.w3.org/TR/owl-features/
Retrieved: [24-10-2012]

[21] OWL Example
http://jmvidal.cse.sc.edu/talks/xmlrdfdaml/owlexample.html
Retrieved: [24-10-2012]

[22] RDFCalendar - A Simple Example
http://www.w3.org/TR/rdfcal/#exsim
Retrieved: [24-10-2012]

[23] Opening Hours Use Case
http://www.w3.org/wiki/OpeningHoursUseCase
Retrieved: [24-10-2012]

[24] ODP - Open Directory Project
http://www.dmoz.org/
Retrieved: [24-10-2012]

[25] Google Transit - GTFS Reference
https://developers.google.com/transit/gtfs/reference
Retrieved: [25-10-2012]

[26] CSV : Comma Separated Value
http://tools.ietf.org/html/rfc4180
Retrieved: [25-10-2012]

[27] Google Transit - GTFS Examples
https://developers.google.com/transit/gtfs/examples/gtfs-feed
Retrieved: [25-10-2012]

[28] Maisons du Monde
http://www.maisonsdumonde.com
Retrieved: [25-10-2012]

[29] Facoltà d'Informatica - Università di Bologna
http://corsi.unibo.it/informatica/Pagine/orari-delle-lezioni.aspx
Retrieved: [26-10-2012]

[30] Teatro Manzoni – Bologna
http://www.auditoriumanzoni.it
Retrieved: [26-10-2012]

[31] Medusa Cinema - Multisala Bologna
http://www.thespacecinema.it/cinema/bologna
Retrieved: [26-10-2012]

[32] Depeche Mode Tour 2013
http://www.daringtodo.com/lang/it/2012/10/23/depeche-mode-tour-2013-ecco-tutte-le-date/

Retrieved: [26-10-2012]

[33] JSON : Javascript Object Notation
http://www.json.org/
Retrieved: [26-10-2012]

[34] REST : REpresentational State Transfer
http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
Retrieved: [26-10-2012]

[35] Sencha ExtJS
http://www.sencha.com/products/extjs
Retrieved: [27-10-2012]

[36] AJAX : Asynchronous Javascript and XML
http://www.w3.org/TR/XMLHttpRequest/
Retrieved: [28-10-2012]

[37] Python Language
http://www.python.org/
Retrieved: [28-10-2012]

[38] Flask : A Python Microframework
http://flask.pocoo.org/
Retrieved: [28-10-2012]

[39] CORS : Cross Origin Resource Sharing
http://www.w3.org/TR/cors/
Retrieved: [29-10-2012]

[40] geoRDF
http://www.w3.org/wiki/GeoRDF
Retrieved: [29-10-2012]

Ringraziamenti

E' doveroso fare dei ringraziamenti al termine di questa tesi e più in generale di questo percorso di studi.

Ringrazio i miei genitori per primi perché senza di loro non avrei potuto iscrivermi all'università e arrivare fino in fondo, perciò ringrazio loro e i miei famigliari per il loro sostegno e la loro fiducia.

Ringrazio la mia fidanzata per essermi stata sempre a fianco in tutti questi anni di studio e di non aver mai dubitato di me ma anzi di avermi sempre spronato in ogni situazione, soprattutto in quelle più difficili.

Ringrazio tutti i miei professori che mi hanno non solo trasmesso le conoscenze tecniche e teoriche che hanno fatto di me un degno programmatore ma anche di avermi formato con i loro insegnamenti, in particolare di quelli relativi al software libero e open source.
Tra questi ringrazio di cuore il mio relatore Fabio Vitali per avermi sopportato, spronato, aiutato in questi ultimi mesi per la riuscita del tirocinio e la stesura della tesi; lo ringrazio anche per avermi fatto comprendere e capire la mia strada nel campo dello sviluppo software.
Ringrazio Renzo Davoli, un altro grande professore, che sin dal primo giorno di facoltà ha suscitato in me l'interesse per la condivisione e l'accessibilità della conoscenza.
Ringrazio il professor Vittorio Ghini per il suo modo d'interagire con noi studenti e per il suo modo rivoluzionario e stravagante di fare lezione.

Ringrazio i miei compagni di facoltà che mi hanno accompagnato per il percorso formativo.
In particolare voglio ringraziare Barbara Iadarola, una grande compagna di studi ma anche grande amica con la quale ho sviluppato i migliori progetti di facoltà.
Ringrazio anche Vittorio Lanzara e Andrea Mancini, due colleghi che ho conosciuto soltanto al termine del corso di laurea, ma che hanno condiviso con me gli esami più difficili e con cui ho realizzato una grande amicizia.

Infine, ma non per ultimi, ringrazio tutte quelle persone, tra colleghi, professori, amici e famigliari, che mi sono state accanto, che mi hanno educato, che mi hanno formato, che mi hanno supportato in ogni momento in questi anni.