Università degli Studi di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea in Informatica triennale

Andrea Costazza

# Librerie JavaScript per il trattamento di ontologie del Web Semantico con informazioni geolocalizzate

Relazione progetto finale

Relatore
**Prof. Domenico Cantone**

Correlatore
**Dott. Cristiano Longo**

Anno Accademico 2015/16

Università degli Studi di Catania
Dipartimento di Matematica e Informatica
Corso di Laurea in Informatica triennale

Andrea Costazza

# Librerie JavaScript per il trattamento di ontologie del Web Semantico con informazioni geolocalizzate

Relazione progetto finale

Relatore
**Prof. Domenico Cantone**

Correlatore
**Dott. Cristiano Longo**

Anno Accademico 2015/16

Questa tesi è stata realizzata nell'ambito del progetto `opendatahacklab`,[1] in un accordo di collaborazione e ricerca scientifica tra il Dipartimento di Matematica e Informatica dell'Università degli Studi di Catania e l'associazione Hackspace Catania (convenzione del 18 Settembre 2015, prot. 113964).

[1] `http://opendatahacklab.org`

# Indice

# 1 Introduzione

Lo scopo di questa tesi di laurea è di realizzare una libreria JavaScript per il trattamento delle ontologie del Web Semantico. Tale libreria è divisa in due parti. Nella prima parte si concentra sulla informazioni geolocalizzate, attraverso il sito web Leaflet, che fornisce librerie per integrare una mappa on-line su una pagina web (come spiegato nel terzo capitolo), mentre nella seconda parte utilizza SPARQL, dove attraverso una query otteniamo i dati tramite la codifica JSON (come spiegato nel quarto capitolo). Il resto della tesi è strutturato come segue: nel secondo capitolo vedremo i concetti principali del Web Semantico e delle logiche descrittive. Nel quinto capitolo si parlerà dei Linked Open Data, che sono dei vocabolari messi a disposizione dall'AgID coi quali si realizzerà, insieme alla base di conoscenza del dott.Longo Cristiano, che rappresenta la macrostruttura del Comune di Catania, sia il menù gerarchico sia le query necessarie per la creazione dei markers. Per ogni ufficio della macrostruttura del Comune di Catania è indicato una latitudine e una longitudine e ci serviremo dei markers, che sono dei puntini di colore blu, dove cliccandoci sopra si apre una tendina contenente tutte le informazioni relative all'ufficio, per associare tali coordinate. Infine verranno indicati tutti gli strumenti adoperati, quali i programmi e i siti web, per la realizzazione del progetto, descritti nel capitolo sei.

## 2 Web Semantico

Il termine Web Semantico[3, 2, 5] è un concetto nato solo da pochi anni, dalla mente di Tim Berners-Lee, il quale non solo ha ideato il Word Wide Web (WWW) e il W3C[2](World Wide Web Consortium), ma lo ha anche trasformato in qualcosa di rivoluzionario. L'idea di base era quella di associare a tutti i documenti caricati nel web, dei **metadati**[3] in modo che qualsiasi macchina, motore di ricerca e applicazione fosse in grado di elaborarli con estrema facilità.
Inizialmente si adoperava il semplice **collegamento ipertestuale,**[4] le macchine si limitavano solamente a trasmettere il contenuto, senza possibilità di capire com'era strutturata la pagina. A tal proposito si utilizza il concetto di **rappresentazione della conoscenza**. I sistemi di rappresentazione della conoscenza permettono di usare semantiche formali e simbolismi di carattere matematico, cioè una serie di costrutti sia per definire la sintassi del dominio di interesse, sia una serie di operatori che permettano di dare un significato alle asserzioni. Con questo ragionamento possiamo quindi costruire una **base di conoscenza**,[5] che permette agli applicativi software e agli agenti automatici di scaricare diverse informazioni che sono relative, per esempio, ad aziende oppure enti culturali e utilizzarle in maniera più adeguata. In questo progetto si utilizza un base di conoscenza realizzata in RDF[6] ed elaborata attraverso il linguaggio di programmazione Javascript, ma ne parleremo più avanti.
Analizzati i concetti di base del Web Semantico vediamo adesso come è strutturato. Lo si può pensare come un sistema a livelli gerarchico, dove ogni livello è arricchito con nuovi costrutti e simbolismi come mostrato in **Figura 1.**

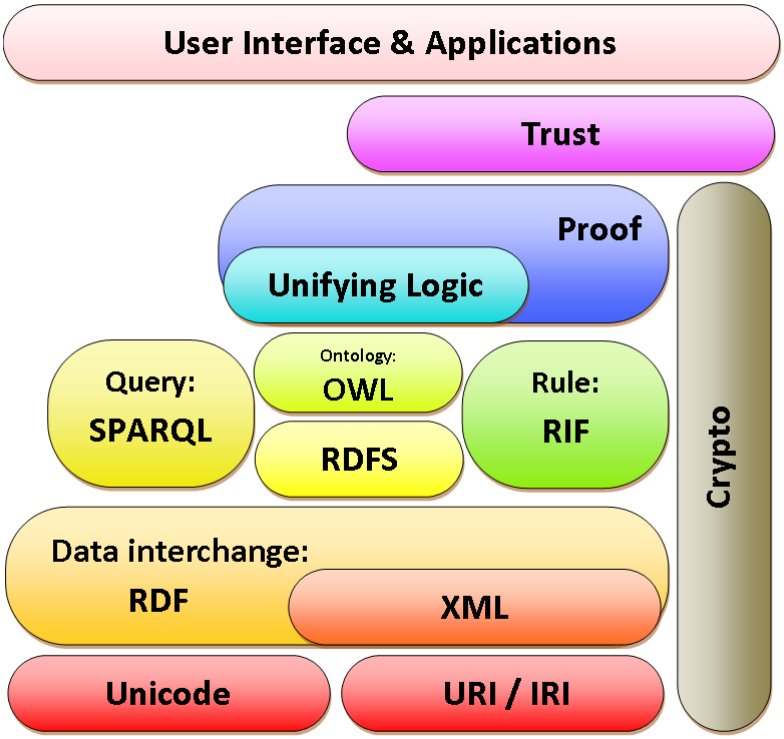


Figura 1: Rappresentazione a livelli del Web Semantico.

[2] `http://www.w3.org/`
[3] Informazione che descrive un insieme di dati. Fonte Wikipedia.
[4] Rinvio da un'unità informativa su supporto digitale ad un'altra. Fonte Wikipedia.
[5] Ambiente volto a facilitare la raccolta, l'organizzazione e la distribuzione della conoscenza. Fonte Wikipedia.
[6] Vedi Capitolo 2.2

I livelli principali[7] sono:

- URI/IRI:[8] Il livello degli indirizzi dove indicano una risorsa generica come ad esempio un pagina web;
- Unicode: Standard di codifica dei set di caratteri internazionali. Questo livello permette che tutte le lingue del mondo possano essere utilizzate per qualsiasi pagina web;
- XML:[9] è un linguaggio di markup, simile all'HTML, che è stato progettato per memorizzare e trasportare i dati. Utilizza dei tag che devono rispettare determinate regole e molto spesso fanno uso dei "namespace", una sorta di vocabolario di termini, a cui si fa riferimento per esprimere delle URI;
- RDF:[10] è un linguaggio, proposto dal W3C per rappresentare informazioni sulle risorse attraverso una struttura a grafo. In questo linguaggio le informazioni sono esprimibili con asserzioni (statement) costituite da triple formate da soggetto, predicato e oggetto (identificati come subject, predicate e object, rispettivamente);
- RDFS:[11] Estensione di RDF, fornisce elementi di base per la descrizione di ontologie, chiamate vocabolari per le risorse RDF;
- OWL:[12] è un linguaggio che deriva dalle logiche descrittive, e offre più costrutti rispetto a RDFS. OWL si suddivide in tre categorie: OWL Lite per tassonomie e vincoli semplici, OWL DL per il pieno supporto della logica descrittiva e OWL Full per la massima espressività e la libertà sintattica di RDF;
- SPARQL: è un linguaggio di interrogazione per tipologie di dati rappresentati in RDF; è uno degli elementi cardine per sviluppare il web semantico e consente di estrarre informazioni dalle basi di conoscenza distribuite sul web.

## 2.1 Il modello Turtle

La sintassi utilizzata nello SPARQL è simile a Turtle[13] per esprimere modelli di query.
Il Turtle è anch'esso un formato per esprimere i dati nel **Resource Description Framework (RDF)**, anche in questo caso le informazioni vengono rappresentate attraverso le triple, ciascuna delle quali è costituito da un soggetto, un predicato, e un oggetto. Ogni elemento è espresso come indirizzo URI/IRI come mostrato in **Figura 2**. Gli indirizzi URI vengono racchiusi tra < ed > e possono essere abbreviati

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#>.

<http://www.w3.org/People/EM/contact#me>
  rdf:type contact:Person;
  contact:fullName "Eric Miller";
  contact:mailbox <mailto:em@w3.org>;
  contact:personalTitle "Dr.".
```

Figura 2: Esempio Codice Turtle.

attraverso il comando **@prefix**, a cui viene associato un nome che può essere richiamato in diversi parti del file.

[7] Tratto da Realizzazione di moduli JAVA per il trattamento del web semantico di Andrea Costazza
[8] Uniform Resource Identifier/Internationalized Resource Identifier
[9] eXtensible Markup Language
[10] Resource Description Framework
[11] RDF Schema
[12] Ontology Web Language
[13] Terse RDF Triple Language

Turtle è un'alternativa a RDF/XML, ma a differenza di quest'ultimo, Turtle non si basa su XML ed è più facile da modificare, inoltre risulta molto più leggibile.
Nel 2011, il World Wide Web Consortium (W3C) ha iniziato a lavorare su una versione aggiornata di RDF, ed ha pubblicato la documentazione il 25 febbraio 2014.[14]

## 2.2 Resource Description Framework (RDF)

Il **Resource Description Framework(RDF)**, proposto dal W3C, è lo strumento base per la realizzazione del Semantic Web. Esso esprime informazioni sulle risorse che possono essere di qualunque tipo, come ad esempio persone o cose.
I dati espressi possono anche essere, ad esempio, informazioni sulle pagine web, contenuti per i motori di ricerca oppure biblioteche elettroniche, sia aziendali che comunali, contenenti moltissime informazioni.
Per descrivere queste informazioni RDF utilizza:

- Risorse: come già descritto può essere una qualsiasi cosa;
- Proprietà: è una relazione utilizzata per descrivere una risorsa;
- Valore: è il valore assunto dalla proprietà; può essere anche una risorsa.

Le combinazioni tra Risorse, Proprietà e Valori prendono il nome di **Asserzioni** (statement), cioè una tripla composta da un soggetto (risorsa), un predicato (proprietà) e un oggetto (valore), come mostrato in **Figura 3**.


Figura 3: Relazione degli statement.

[14]La pubblicazione si trova al sito https://www.w3.org/TR/turtle/

Il soggetto e l'oggetto rappresentano le due risorse, mentre il predicato rappresenta la natura del loro rapporto.
Ecco un esempio di triple in RDF:
<Napoleone> <è nato ad> < Ajaccio>
<Napoleone> <perse a> < Waterloo>.
<Jacques-Louis David> <dipinse l'incoronazione di> < Napoleone>.
Come si evince dall'esempio, Napoleone è soggetto di due asserzioni e oggetto dell'altra; questa caratteristica importante, cioè quella di avere la stessa risorsa sia in posizione di soggetto e sia in posizione di oggetto, permette di trovare connessioni tra triple e quindi reperire più informazioni. Si realizza un grafo dove il soggetto e l'oggetto rappresentano i nodi mentre al predicato corrisponde l'arco.


Figura 4: Esempio di Grafo.

Il grafo indicato in **Figura 4** si traduce nel linguaggio rdf come segue:

```
<rdf:Description rdf:about="http://www.host.html/">
  <s:personaggio rdf:resource="http://persona.com/id/0001"/>
</rdf:Description>
<rdf:Description rdf:about="http://persona.com/id/0001">
  <s:nome>Napoleone</s:Nome>
  <s:luogo di Nascita>Ajaccio</s:Luogo di Nascita>
</rdf:Description>
```

Le risorse, i predicati possono essere solo URI/IRI mentre i valori possono essere:

- **URI/IRI**: come ad esempio la risorsa Napoleone può essere contenuta, per esempio, su DBpedia;[15]
- **Literals**: sono dei valori costanti, come ad esempio date, numeri o anche stringhe.

Un'altra caratteristica importante del modello RDF e che è le risorse possono essere raggruppate in strutture che prendono il nome di **Contenitori**. In RDF un contenitore può essere di tre tipi:

- **Bag**, è una lista non ordinata di Risorse e Literals;
- **Sequence**, a differenza di Bag, l'insieme è ordinato;
- **Alternative**, è una lista di risorse che definiscono un'alternativa per il valore singolo di una proprietà.[16]

[15]Sito internet che offre un progetto aperto e collaborativo per l'estrazione e il riutilizzo di informazioni semi-strutturate dalla Wikipedia in italiano. `http://it.dbpedia.org/`
[16]Fonte Wikipedia.

Per definire schemi e vocabolari per i metadati, si utilizza un'estensione di RDF chiamata RDF Schema, che fornisce meccanismi per gruppi di risorse correlate e descrive le risorse con le classi, le proprietà e i valori.
Il sistema di classi e delle proprietà di RDF Schema fornisce elementi di base per la descrizione delle ontologie, per vincolare domini e codomini delle relazioni, definire classi di oggetti e relazioni tra classi.
Nelle seguenti tabelle vengono descritte le classi e le proprietà utilizzate dal RDF Schema:

| **Nome Classe** | **Commento** |
|---|---|
| rdfs:Resource | The class of literal values, e.g. textual strings and integers. |
| rdfs:Literal | The class of language-tagged string literal values. |
| rdf:langString | La classe language-tagged string literal values. |
| rdf:HTML | The class of HTML literal values. |
| rdf:XMLLiteral | The class of XML literal values. |
| rdfs:Class | The class of classes. |
| rdf:Property | The class of RDF properties. |
| rdfs:Datatype | The class of RDF datatypes. |
| rdf:Statement | The class of RDF statements. |
| rdf:Bag | The class of unordered containers. |
| rdf:Seq | The class of ordered containers. |
| rdf:Alt | The class of containers of alternatives. |
| rdfs:Container | The class of RDF containers. |
| rdfs:ContainerMembershipProperty | The class of container membership properties. |
| rdf:List | The class of RDF Lists. |

Tabella 1: Tabella delle classi dell'RDF Schema.

| **Property name** | **comment** | **domain** | **range** |
|---|---|---|---|
| rdf:type | The subject is an instance of a class. | rdfs:Resource | rdfs:Class |
| rdfs:subClassOf | The subject is a subclass of a class. | rdfs:Class | rdfs:Class |
| rdfs:subPropertyOf | The subject is a subproperty of a property. | rdf:Property | rdf:Property |
| rdfs:domain | A domain of the subject property. | rdf:Property | rdfs:Class |
| rdfs:range | A range of the subject property. | rdf:Property | rdfs:Class |
| rdfs:label | A human-readable name for the subject. | rdfs:Resource | rdfs:Literal |
| rdfs:comment | A description of the subject resource. | rdfs:Resource | rdfs:Literal |
| rdfs:member | A member of the subject resource. | rdfs:Resource | rdfs:Resource |
| rdf:first | The first item in the subject RDF list. | rdf:List | rdfs:Resource |
| rdf:rest | The rest of the subject RDF list after the first item. | rdf:List | rdf:List |
| rdfs:seeAlso | Further information about the subject resource. | rdfs:Resource | rdfs:Resource |
| rdfs:isDefinedBy | The definition of the subject resource. | rdfs:Resource | rdfs:Resource |
| rdf:value | Idiomatic property used for structured values. | rdfs:Resource | rdfs:Resource. |
| rdf:subject | The subject of the subject RDF statement. | rdf:Statement | rdfs:Resource |
| rdf:predicate | The predicate of the subject RDF statement. | rdf:Statement | rdfs:Resource |
| rdf:object | The object of the subject RDF statement. | rdf:Statement | rdfs:Resource |

Tabella 2: Tabella delle proprietà dell'RDF Schema.

## 2.3 Logiche descrittive

La logica descrittiva (DL,[17] vedi [1]) è una notazione formale utilizzata nella rappresentazione della conoscenza. Ogni nodo, oggetto o categoria, è caratterizzato da un elenco di proprietà. Dato un particolare dominio di conoscenza, la logica descrittiva individua i concetti primari, le categorie più rilevanti, e successivamente analizza le proprietà degli oggetti, al fine di migliorare la descrizione delle classificazioni e delle sotto-classificazioni del dominio di conoscenza. Ogni DL è basata da blocchi sintattici di base che sono:

- **Concetti**: corrispondenti a predicati unari che, combinati tra loro, danno origine a predicati complessi;
- **Ruoli**: corrispondenti a predicati binari ed eventualmente operatori;
- **Individui**: entità astratte o concrete usate nelle asserzioni.

Una base di conoscenza per le DL è costituita da:

- un insieme finito di assiomi **Tbox** (terminalogical box);
- un insieme finito di asserzioni **Abox** (assertional box).

Il TBox contiene frasi che descrivono gerarchie di concetti o di ruoli, mentre l'Abox è un insieme finito di asserzioni di concetto o di ruolo (ad esempio, le relazioni tra gli individui e concetti).
Introduciamo adesso un concetto sintattico molto importante per lo sviluppo delle logiche descrittive, cioè il **linguaggio descrittivo**. Si tratta del linguaggio attraverso cui si esprimono prescrizioni, aventi la funzione di indirizzare il comportamento degli individui.
**Definizione**.
Un linguaggio descrittivo consiste di una terna di insiemi finiti **(C, R, Ob)**. Gli elementi di **C** sono indicati con le lettere A, B, ... e sono chiamati concetti atomici; gli elementi di **R** sono indicati con le lettere R, S, ... e sono detti ruoli, mentre gli elementi di **Ob** sono indicati con le lettere a, b, ... e sono detti nomi degli oggetti.[18]
Fra i costruttori di concetti annoveriamo certamente gli operatori booleani che indichiamo con $\neg$ (negazione), $\sqcap$ (intersezione), $\sqcup$ (unione).
Ci riserveremo anche di usare rispettivamente per il concetto universale $\top$ (mai soddisfatto) e per il concetto contraddittorio $\bot$ (sempre soddisfatto).
Un linguaggio descrittivo si può scrivere nel seguente modo $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, dove $\Delta^{\mathcal{I}}$ rappresenta il dominio dell'interpretazione, mentre $\cdot^{\mathcal{I}}$ rappresenta la funzione di interpretazione, che assegna:

- ad ogni concetto atomico A$\in$**C** un insieme $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$;
- ad ogni ruolo R$\in$**R** una relazione binaria $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ x $\Delta^{\mathcal{I}}$;
- ad ogni nome di oggetto a$\in$**Ob** un elemento $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

Dopo il linguaggio descrittivo, definiamo adesso la logica descrittiva di base chiamata $\mathcal{ALC}$[19] e definiamo i seguenti concetti:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$;
- $\bot^{\mathcal{I}} = 0$;
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;

[17] Description Logics
[18] `http://homes.di.unimi.it/~ghilardi/logica2/DL.pdf`). Introduzione alle Logiche Descrittive di Silvio Ghilardi
[19] Attribute Language with Complement

- $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^I | (\forall [a,b] \in R^{\mathcal{I}})(b \in C^I)\}$;
- $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^I | (\exists [a,b] \in R^{\mathcal{I}}) \wedge (b \in C^I)\}$.

La logica descrittiva $\mathcal{ALC}$ si può estendere per creare logiche più complesse e articolate; tra i vari costrutti importanti abbiamo:

1. $\mathcal{N}$: si introducono i costruttori $\geq nR$, $\leq nR$, detti restrizioni numeriche, dove n∈N, che sono interpretati come segue:

$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^I | \#\{b \in \Delta^I | [a,b] \in R^{\mathcal{I}}\} \geq n\}$$
$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^I | \#\{b \in \Delta^I | [a,b] \in R^{\mathcal{I}}\} \leq n\}$$

   dove $\#\{...\}$ si indica la cardinalità dell'insieme $\{...\}$

2. $\mathcal{Q}$: si introducono i costruttori$\geq nR.C$, $\leq nR.C$, detti restrizioni qualificate

$$(\geq nR.C)^{\mathcal{I}} = \{a \in \Delta^I | \#\{b \in C^I | [a,b] \in R^{\mathcal{I}}\} \geq n\}$$
$$(\leq nR.C)^{\mathcal{I}} = \{a \in \Delta^I | \#\{b \in C^I | [a,b] \in R^{\mathcal{I}}\} \leq n\}$$

3. $\mathcal{O}$: si introducono gli insiemi finiti di elementi detti nominals ($\{a\} o \{a_1, ...a_n\}$) interpretati come segue:

$$\{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\}$$

$$\{a_1, ...a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, ...a_n^I\}$$

Per estendere il linguaggio si dice che $\mathcal{I}$ è modello di (in simboli: $\models$):
**TBox** se:

$$\mathcal{I} \models C \sqsubseteq D \text{ se e soltanto se } C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$$

$$\mathcal{I} \models \mathcal{T} \text{ se e soltanto se } \mathcal{I} \models \Phi \forall \Phi \in \mathcal{T}$$

**ABox** se:

$$\mathcal{I} \models a : C \text{ se e soltanto se } a^{\mathcal{I}} \in C^{\mathcal{I}}$$

$$\mathcal{I} \models (a,b) : R \text{ se e soltanto se } (a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$$

$$\mathcal{I} \models \mathcal{A} \text{ se e soltanto se } \mathcal{I} \models \phi \forall \phi \in \mathcal{A}$$

Infine si definisce **Base di Conoscenza K** la coppia ordinata $(\mathcal{A}, \mathcal{T})$. Un'interpretazione $\mathcal{I}$ è modello della base di conoscenza **K** se:

$$\mathcal{I} \models \mathbf{K} \text{ se e soltanto se } \mathcal{I} \models \mathcal{T} \text{ e } \mathcal{I} \models \mathcal{A}$$

# 3 Mappe on-line per siti web

La realizzazione del portale è stata resa possibile grazie alle librerie fornite dal sito web **Leaflet**.[20] Leaflet è una moderna libreria open-source realizzata in JavaScript e ha lo scopo di rendere interattive le mappe per utilizzarle in qualsiasi piattaforma si voglia, che sia desktop o mobile. Lo sviluppatore di tale libreria è Vladimir Agafonkin che, con l'aiuto di un team di collaboratori dedicati, ha realizzato una semplice e versatile libreria grande circa 33 KByte. Inoltre utilizzando la tecnologia **HTML5** e **CSS3** essa è accessibile sui browser moderni quali Chrome, Firefox, Safari e Internet Explorer. Può essere anche accessibile per i browser più datati e ha anche una buona e facile documentazione on-line. Infine ha un'estesa e vasta gamma di plugin, che si possono facilmente integrare rendendo il codice molto compatto possibile e di facile intuizione.

## 3.1 Caricamento mappa

Per preparare il sito web con la mappa interattiva occorre realizzare le seguenti principali procedure:

- Inserire nel codice HTML nella sezione **head** il riferimento al file **"'leaflet.css"'**;
- Includere la libreria JavaScript **"'leaflet.js"'**;
- Inserire un elemento div che ha come parametro **"'id=map"'** nella sezione **body**;
- Settare attraverso la tecnologia CSS3 le caratteristiche della mappa attraverso l'id "'map"'.

Di seguito mostriamo un esempio di pagina HTML che include una mappa relizzata con Leaflet:

```
1  <head>
2   <meta charset=utf-8 />
3   <title>MAPPA</title>
4   <link rel="stylesheet" href="./css/leaflet.css"/>
5   <link rel="stylesheet" href="./css/menu.css"/>
6   <script src="./js/leaflet.js"></script>
7   <script src="./js/client.js"></script>
8  </head>
9  <body>
10  <div id='map'>
11  <div id="loading"><p class="loading">Loading...<p></div>
12  </div>
13  <div id="navigation"></div>
14  <script type="text/javascript">
15   launch();
16  </script>
17 </body>
```

Il secondo passaggio è quello di creare una mappa interattiva. Per farlo occorre collegarsi al sito **www.mapbox.com**, che fornisce un portale gratuito per creare o modificare una mappa secondo le caratteristiche che si vogliono. Occorre preliminarmente registrarsi al sito web fornendo:

- Username;
- Cognome;
- Nome;
- Email;
- Password.

Una volta effettuato l'accesso bisogna cliccare sul pulsante **Studio**, situato in alto a destra, poi sul pannello **Styles** e, infine su **Create a Style**: in questo modo verrà creata una nuova mappa da poter modellare a seconda delle proprie necessità. In **Figura 5** è mostrata la pagina principale del sito web, mentre nella **Figura 6** il pannello **Styles** con i comandi principali.

[20] http://leafletjs.com/

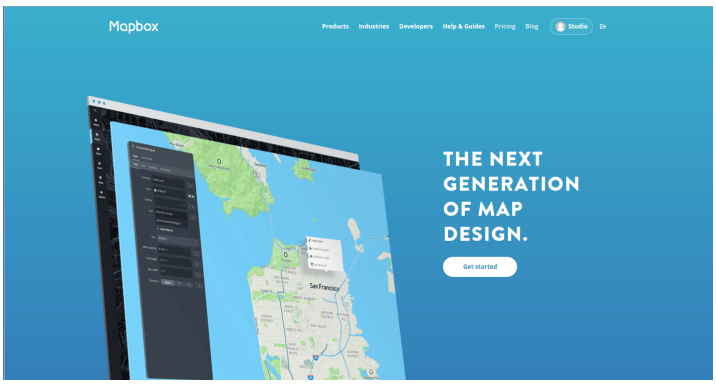


Figura 5: Sito di Mapbox.

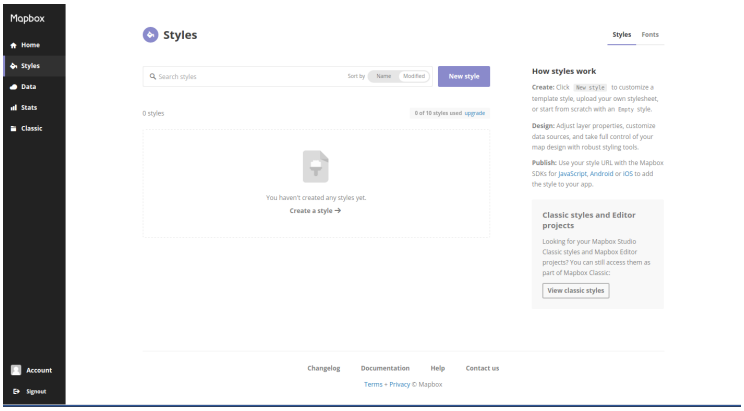


Figura 6: Pannello Styles.

Una volta creata la mappa, cliccando sul pannello **Edit**, si aprirà una nuova pagina. Sul pannello a sinistra è indicata la consolle per modellare la mappa: possiamo scegliere i colori delle strade, dell'acqua, della terra e pianure e dello sfondo.
Sul pannello di destra, invece, è indicato lo zoom e le coordinate, cioè la latitudine e la longitudine. Sistemate tutte le modifiche per confermare il progetto della mappa, occorre cliccare sul pulsante **Publish**. Cliccando sul nome della mappa, nel pannello **Styles**, a destra è indicato l'indirizzo url per poter pubblicare la mappa. In **Figura 7** è mostrato la schermata **Edit**.

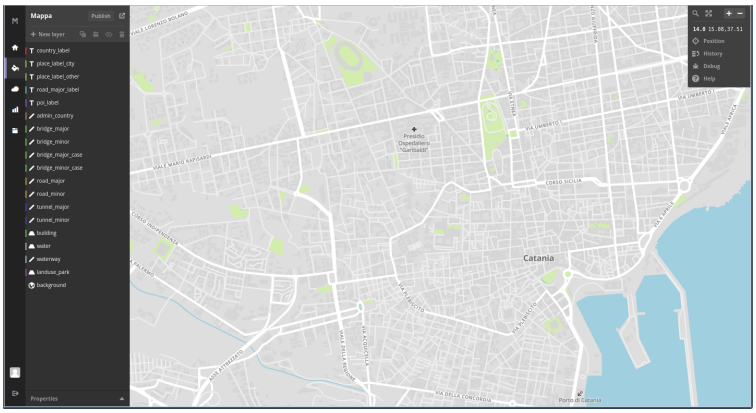


Figura 7: Pannello Edit.

Per inserire la mappa sul codice sorgente e caricarla nella pagina web, bisogna inizializzarla fornendo le coordinate geografiche della posizione e il livello dello zoom. Tali informazioni sono reperibili sul pannello Edit come mostrato in **Figura 8**, e devono essere inserite nella sezione body della file **index.html**.

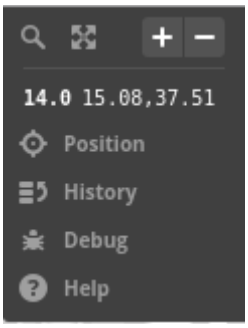


Figura 8: Coordinate e zoom mappa.

Il codice da inserire e da mettere nella sezione body ed è il seguente:

```
<body>
<div id=`map'>
<script type="text/javascript">
 var auxmap = L.map(`map').setView([37.51, 15.08], 14);
</script>
</div>
</body>
```

Dal codice si evince che all'interno delle parentesi, sono inserite le coordinate geografiche presenti nella **Figura 7**, inoltre lo script utilizzato è sempre JavaScript. Il prossimo passaggio è quello di inserire la mappa utilizzando il comando **tileLayer**. Tale comando permette di inserire la mappa creata sul sito di Mapbox, definendo lo zoom massimo possibile, gli attributi, l'identificativo della mappa e l'access token. Di seguito abbiamo il codice sorgente:

```
<body>
<div id=`map'>
<script type="text/javascript">
 var auxmap = L.map(`map').setView([37.51, 15.08], 14);
 L.tileLayer(`https://api.tiles.mapbox.com/v4/{id}/{z}/{x}/{y}.png?'+
  `access_token={accessToken}', {
 maxZoom: 18,
 attribution: `Map data &copy;
 <a href="http://openstreetmap.org">OpenStreetMap</a> contributors, '+
  `<a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, '+
  `Imagery c <a href="http://mapbox.com">Mapbox</a>',
 id: `mapbox://styles/andreacostazza/cijyrg3yt00zsbpm3orpxkj2l'
 accessToken: `XXXXXXXXXXX'
  }).addTo(map);
</script>
</div>
</body>
```

Sul codice precedente per il caricamento della mappa, si nota che nella prima voce è indicato il collegamento ipertestuale della mappa, alla voce **attribution** sono indicate le licenze, mentre alla voce **id** è specificato l'identificativo della nostra mappa, che si ricava dal sito Mapbox alla sezione **Styles**, come mostrato in **Figura 6** accanto alla voce Edit. Infine, alla voce **accessToken** si deve indicare quel valore disponibile, sempre nel pannello **Styles**, a destra.
Seguendo le procedure appena indicate, la mappa verrà visualizzata correttamente nella nostra pagina web.

### 3.2 Creazione delle icone, dei markers e dei popups

Attraverso il metodo **L.Icon** è possibile creare un icona che identifica un determinato punto della mappa. Per prima cosa occorre scegliere un'immagine, dopodiché attraverso **iconUrl** è possibile caricarla specificando il percorso del file; se si dispone anche di un'immagine con l'ombra, bisogna caricarla con **shadowUrl** sempre specificando il percorso del file. Poi bisogna specificare la grandezza dell'icona e dell'ombra, attraverso i parametri **iconSize** e **shadowSize**, come è evidenziato nel codice seguente. Infine è possibile caricare il marker attraverso il metodo **L.marker**, dove vengono specificate le coordinate.

```
<body>
<div id='map'>
<script type="text/javascript">
var map=L.map('map')setView([37.583,14.071],9);
L.tileLayer('https://{s}.tiles.mapbox.com/v3/{id}/{z}/{x}/{y}.png',{
        maxZoom: 18,
        attribution: 'Map data &copy;
  <a href="http://openstreetmap.org">OpenStreetMap</a> contributors,'+
 '<a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>,'+
  'Imagery c <a href="http://mapbox.com">Mapbox</a>',
  id: 'andreacostazza.ik9ap86i'
  }).addTo(map);
var iconBlue= L.icon({
  iconUrl: './icon/marker-icon.png',
  shadowUrl: './icon/marker-shadow.png',

  iconSize: [25,41],
  shadowSize: [41,41],
  iconAnchor:[lat,lon],
  shadowAnchor:[lat,lon],
  popupAnchor:[-25,-10]
  });

  var marker = L.marker([lat, lon],{icon:iconBlue});
  marker.addTo(map);
</script>
</div>
</body>
```

# 4 SPARQL

**SPARQL**[21] è un linguaggio di interrogazione per dati rappresentati tramite il **Resource Description Framework (RDF)**. SPARQL è un elemento essenziale per lo sviluppo del web semantico e consente di estrarre informazioni dai dataset RDF.
Una query SPARQL è un insieme di triple "soggetto-predicato-oggetto", però a differenza dell'RDF, gli elementi delle triple possono essere delle variabili, che sono indicate con il punto interrogativo ?, oppure costanti.

**?title rdf:label ?name**

Nell'esempio proposto il soggetto e l'oggetto sono delle variabili, mentre il predicato è una costante. Il risultato di una query SPARQL contro un dataset RDF è una tabella molto simile a quelle utilizzate nei database relazionali in SQL; si tratta di una tabella con una colonna per ogni variabile presente nella query. Ogni riga della tabella rappresenta una sostituzione *variabile -> valore* tale che, se applicata alle triple indicate nella query, fa sì che le triple ottenute appartengano al dataset contro cui è eseguita la query.

## 4.1 Comandi principali

Il linguaggio SPARQL è composto da tre comandi principali:

- **PREFIX:** dichiara prefissi e namespace e, come nel Turtle, per abbreviare il percorso URI si associa un nome che verrà richiamato all'interno di WHERE (ad es., gr:offers può essere scritto anche <http://purl.org/goodrelations/v1offers>)
- **SELECT:** identifica le variabili che verranno visualizzate, sotto forma di tabella, dal risultato dell' interrogazione. È spesso accompagnato da *, che seleziona tutte le variabili di una interrogazione, e DISTINCT, che esclude dal risultato i valori duplicati;
- **WHERE:** seleziona il criterio di selezione da applicare ai dati; è composto da un blocco, delimitato dalle parentesi graffe ({...}), dove al suo interno può esserci una o più triple, separati da un punto fermo.

Inoltre SPARQL fornisce diversi comandi per una maggiore flessibilità sulle interrogazioni. Nella tabella seguente sono riportate quelle più utilizzate:

| **Nome Comando** | **Cosa fa** | **Attributi** |
|---|---|---|
| FILTER: | Filtra i valori visualizzati. | **regex** si utilizza per espressioni regolari. |
| OPTIONAL: | Prevede l'assenza di alcuni termini | Le variabili compariranno prive di valore. |
| ORDER BY: | Ordina il risultato in base ad una variabile specifica. | DESC ordinamento decrescente. |
| LIMIT: | Limita la visualizzazione. | Accompagnata da valori numerici |
| a: | Visualizza le classi a cui appartiene una variabile. | Alternativa a rdf:type |

Tabella 3: Tabella dei comandi principali di SPARQL.

[21] `https://www.w3.org/standards/techs/sparql`

## 4.2 Libreria javascript per interrogazioni SPARQL

Per fare un interrogazione in SPARQL, JavaScript ha bisogno di una chiamata AJAX[22] e della codifica JSON.[23] Nel progetto sono date rispettivamente dalla variabile **xmlhttp**, che fa la chiamata AJAX tramite il metodo **getHTTPObject()** e restituisce il risultato tramite **responseText**. La richiesta AJAX, inoltre, utilizza:

```
1 setRequestHeader("Accept", "application/sparql-results+json");}
```

in questo modo i risultati ottenuti sono visualizzati secondo la codifica JSON. Il codice utilizzato nel progetto è il seguente:

```
1  <body>
2  <div id='map'>
3  <script type="text/javascript">
4  //Created SPARQL query
5  function createSparqlQuery(endpoint,query,map,callback){
6   var querypart = "query=" + escape(query);
7   // Get our HTTP request object.
8   var xmlhttp = getHTTPObject(); //Called AJAX
9   //Include POST OR GET
10  xmlhttp.open('POST', endpoint, true);
11  xmlhttp.setRequestHeader('Content-type',
12   'application/x-www-form-urlencoded');
13  xmlhttp.setRequestHeader("Accept",
14   "application/sparql-results+json");
15  xmlhttp.onreadystatechange = function() {
16   if(xmlhttp.readyState==4 ){
17    if(xmlhttp.status==200){
18    callback(xmlhttp.responseText,map); //JSON code
19    }else
20   // Error
21   alert("Error: Status: "+ xmlhttp.status + "Response: "
22   + xmlhttp.responseText);
23   }
24  };
25  // Send the query to the endpoint.
26  xmlhttp.send(querypart);
27 }
28 </script>
29 </div>
30 </body>
```

Basta scrivere la query utilizzando la nomenclatura dello SPARQL descritta precedentemente e fornire l'indirizzo URI della base di conoscenza; la variabile utilizzata nel progetto per scrivere la query è **query**, mentre per far riferimento alla base di conoscenza si utilizza **endpoint**.

[22] Vedi Capitolo 5.2
[23] Vedi Capitolo 5.3

# 5 Ontologie per la rappresentazione dei servizi pubblici

L'Agenzia per l'Italia Digitale (AgID[24]) ha l'obiettivo di pubblicare dati e documenti relativi ai servizi, alla struttura e alle attività svolte sul sito e di fornire a pubbliche amministrazioni, imprese e cittadini le tecniche utilizzate dal Web Semantico. In pratica vengono forniti vocabolari, sviluppati tramite la pubblicazione **Linked Open Data**[4]. Tale metodo consiste nello strutturare i dati in modo da poter essere interconnessi e diventare più utili attraverso delle interrogazioni, fatte per esempio in linguaggio SPARQL. Nella tabella successiva vengono indicati alcuni dei principali vocabolari utilizzati.

| **Vocabolario** | **Namespace** | **URL** |
|---|---|---|
| Friend Of A Friend | foaf | http://xmlns.com/foaf/spec/ |
| Core Location | locn | http://www.w3.org/ns/locn |
| Organization Ontology | org | http://www.w3.org/TR/vocab-org/ |

Tabella 4: Tabella Vocabolari.

Analizzeremo in dettaglio i vocabolari **Core Location** e **Organization Ontology**. Il **Core Location** è un vocabolario utilizzato per rappresentare qualsiasi luogo in termini di nome, di indirizzo o di coordinate. Tale vocabolario fa parte del progetto **ISA**.[25] L'idirizzo URI utilizzato è `http://www.w3.org/ns/locn#` ed usa il prefisso **locn**.
Il vocabolario è composto da tre classi che sono:

- **Location**: utilizzata per identificare luoghi, città, stati;
- **Address**: utilizzata per gli indirizzi. Questa classe fa riferimento anche ad altre informazioni quali la casella postale, al numero civico, al codice di avviamento postale e così via;
- **Geometry**: fornisce latitudine e longitudine.

Di seguito viene mostrato un semplice esempio:

```
<rdf:Description rdf:about="http://www.w3.org/ns/org#hasSite">
  <rdfs:subPropertyOf rdf:resource="http://www.w3.org/ns/locn#location"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.w3.org/ns/org#siteAddress">
  <rdfs:subPropertyOf rdf:resource="http://www.w3.org/ns/locn#address"/>
</rdf:Description>
<rdf:Description rdf:about="http://purl.org/NET/c4dm/event.owl#place">
  <rdfs:subPropertyOf rdf:resource="http://www.w3.org/ns/locn#location"/>
</rdf:Description>
```

**Organization Ontology**, invece, è utilizzato per modellare le pubbliche amministrazioni, fornendo indicazioni relative ad organizzazioni, membri, macrostrutture e strutture fisiche. L'indirizzo URI utilizzato è `http://www.w3.org/TR/vocab-org/#` ed usa il prefisso **org**.
La classe principale è **Organizzation** che rappresenta le organizzazioni in strutture gerarchiche. Mentre la classe **FormalOrganizzation** scompone un'organizzazione che è riconosciuta in tutto il mondo, come ad esempio i governi. Per rappresentare un'organizzazione, come ad esempio un dipartimento o unità di supporto, che è parte di una organizzazione più grande, si usa la classe **OrganizationalUnit**.
Organization Ontology estende e utilizza termini da altri vocabolari, riportati nella seguente tabella:

[24] `www.agid.gov.it`
[25] Interoperability Solutions for European Public Administrations `http://ec.europa.eu/isa/`

| Prefisso | Namespace |
|---|---|
| foaf | http://xmlns.com/foaf/0.1/ |
| gr | http://purl.org/goodrelations/v1# |
| prov | http://www.w3.org/ns/prov# |
| owl | http://www.w3.org/2002/07/owl# |
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| rdfs | http://www.w3.org/2000/01/rdf-schema# |
| time | http://www.w3.org/2006/time# |
| skos | http://www.w3.org/2004/02/skos/core# |
| vcard | http://www.w3.org/2006/vcard/ns# |
| dct | http://purl.org/dc/terms/ |

Tabella 5: Tabella dei namespace più diffusi.

Tali vocabolari servono per organizzare al meglio la struttura organizzativa di aziende, società e comuni. Nel progetto è stata utilizzata una ontologia, resa disponibile attraverso un dataset RDF ed un corrispondente endpoint sparql, che fa riferimento allo macro struttura del Comune di Catania.[26] Tramite questo endpoint, si fa una specifica interrogazione in SPARQL, attraverso una chiamata AJAX. I dati ottenuti sono codificati tramite la codifica JSON.

[26] `http://dydra.com/cristianolongo/comune-di-catania/sparql`

## 5.1 Menù gerarchici

Un menu gerarchico è una struttura ad albero che simula l'aspetto e il comportamento della struttura a cartelle e sottocartelle utilizzato in tutti i sistemi operativi. Questo tipo di menù viene utilizzato nella stragrande maggioranza dei siti web proprio perché è facile da implementare e non occupa moltissimo spazio, inoltre in questo modo si possono realizzare strutture annidate e complesse ricche di voci, che sono ideali per l'organizzazione.
Nel progetto il menù gerarchico ha lo scopo di raggruppare gli uffici della base di conoscenza relativa alla macrostruttura del Comune di Catania sotto forma di struttura ad albero gerarchico. Esso è creato dinamicamente, cioè ad ogni valore che viene passato si controlla il genitore e i figli associati. Per comprendere meglio elenchiamo i metodi e le classi, realizzati in Javascript, per la creazione del menù:

- Classe **Tree**, che implementa un struttura ad albero. La struttura ad albero ha due elementi fondamentali che sono la **radice**, che ha n nodi, che vengono chiamati figli, e le **foglie**, che sono i nodi senza figli. Tutti gli altri nodi vengono chiamati nodi interni. La classe Tree ha le seguenti variabili:
  - **value**: può essere qualsiasi valore; in questo progetto la si associa ad un indirizzo URI/IRI o a un nome della gerarchia o a una struttura complessa contenete sia URI che nomi;
  - **children**: un array, contenente i figli di un nodo.

  ed i seguenti metodi:
  - **addChild(value)**: aggiunge ad un nodo un figlo;
  - **getChild()**: stampa a video di tutti i figli di un nodo;
  - **preOrder(), postOrder()**: metodi utilizzati per la visita dell'albero;
  - **high(), frontier()**: metodi informativi dell'albero.
- Classe **Description**, in questa classe ad ogni ufficio, appartenente alla base di conoscenza della macrostruttura del Comune di Catania, viene associato tre variabili che sono:
  - **uri**: corrispondente al suo indirizzo IRI/URI;
  - **name**: corrispondente al suo nome;
  - **homepage**: corrispondente la sua homepage che si collega alla pagina principale del sito web.

  i metodi di tale classe sono:
  - **getUri()**: restituisce il valore di uri;
  - **getName()**: restituisce il valore di name;
  - **getHomepage()** restituisce il valore di homepage.
- Classe **Storage**, che è un contenitore di tutti i nodi, compresa la radice con i seguenti metodi:
  - **add(Description)**: crea nodo all'interno dello storage;
  - **get(Uri)**: restituisce nodo con la URI, se esiste; altrimenti NULL;
  - **getTrees()**: restituisce tutte le foglie.

Il menù gerarchico nel progetto è creato sfruttando le classi appena elencate, in più è arricchito con due metodi che sono:

- **createLiMenu()**, crea un lista utilizzando i tag:
  - <ul> definisce una lista non ordinata;
  - <li> definisce gli elementi delle liste, è sempre associato al tag <ul>.

- **createMenu()**, crea il menu dinamico.

Di seguito viene specificato, attraverso lo pseudo-codice, la creazione di tale menu:
**Pseudo codice**

- Attraverso la chiamata AJAX e la codifica JSON otteniamo i dati dalla base di conoscenza;
- Vengono create le variabili parent e child;
- A parent viene associato il nodo, mentre a child il figlio;
- Per ogni coppia di parent e child:
  - ricerca di parent all'interno di Storage;
  - se lo trova va avanti, altrimenti lo crea(metodo add di Storage);
  - ricerca di child all'interno di Storage;
  - se lo trova va avanti, altrimenti lo crea(metodo add di Storage);
  - associa child a parent;
- restituisce l'albero dallo Storage;
- crea il menu attraverso il metodo createLiMenu.

Il risultato sarà il seguente menù:


Figura 9: Il menù gerarchico del progetto.

## 5.2 Codifica JSON

La codifica JSON[27] è un formato convenzionale che serve per lo scambio di dati fra client e server. E' molto leggero da analizzare, ed è anche molto semplice da realizzare. Il JSON è scritto interamente in JavaScript ed è utilizzato, principalmente, per le chiamate AJAX.[28] I dati della codifica JSON possono essere di qualunque tipo. I principali valori sono:

- boolean;
- stringhe;
- interi;
- array sia semplici che associativi;
- null.

Quindi tale codifica può essere utilizzata in un qualsiasi linguaggio di programmazione. Inoltre è facilmente leggibile da quasi tutti i browser moderni, visto che hanno il supporto nativo a JSON, mentre per quelli sprovvisti occorre utilizzare il comando **eval**.
Di seguito vediamo un esempio di codifica JSON

```
{
  "uri": "www.example.com",
  "name": "Example",
  "comment": "A example page",
  "group": [
    {
      "firstUri": "www.example1.com",
      "firstValue": "Example1",
      "firstComment": "First example page"
    },
    {
      "second_uri": "www.example2.com",
      "second_value": "Example2" ,
      "second_comment": "Second example page"
    }
  ]
}
```

Ogni dato ha la forma coppia "nome" : "valore", ed è separato dall'altro tramite la virgola e, inoltre sia i nomi che i valori sono racchiusi tra doppie virgolette.
Le parentesi graffe contengono oggetti:

```
{
  "firstUri": "www.example1.com",
  "firstValue": "Example1",
  "firstComment":"First example page"
 }
```

Le parentesi quadre contengono array:

```
{
   "group": [
    {
      "firstUri": "www.example1.com",
      "firstValue": "Example1",
      "firstComment": "First example page"
    },
```

[27] JAVASCRIPT Object Notatio
[28] Vedi Capitolo 5.3

```
8       {
9         "second_uri": "www.example2.com",
10        "second_value": "Example2" ,
11        "second_comment": "Second example page"
12      }
13    ]
14  }
```

## 5.3 Chiamata AJAX

Il metodo AJAX[29] è stato creato da Jesse Garrett ed è un modo di riutilizzare gli standard internet esistenti. Attraverso il linguaggio di Javascript, che si interfaccia con XML, un **client**[30] richiama le informazioni in modo veloce e trasparente, cioè in maniera asincrona. Per effettuare una chiamata AJAX abbiamo bisogno di un oggetto specifico chiamato **XMLHttpRequest** che serve per lo scambio di dati in modo asincrono con un server. Tale oggetto viene utilizzato nel progetto dal metodo **getHTTPObject()** di cui riportiamo il codice:

```
1  function getHTTPObject(){
2  var xmlhttp;
3  if(!xmlhttp && typeof XMLHttpRequest != 'undefined'){
4   try{
5    // Code for old browser
6    xmlhttp=new ActiveXObject('Msxml2.XMLHTTP');
7   }
8   catch(err){
9    try{
10    // Code for IE6, IE5
11    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
12   }
13   catch(err2){
14    try{
15    // Code for IE7+, Firefox, Chrome, Opera, Safari
16    xmlhttp=new XMLHttpRequest();
17    }
18    catch(err3){
19     xmlhttp=false
20    }
21   }
22  }
23 }
24  return xmlhttp;
25 }
```

Ogni Browser web ha la sua richiesta, nel caso di Browser datati, come IE,[31] l'oggetto XMLHttpRequest, viene restituito da **ActiveXObject**, mentre i moderni browser tale oggetto è supportato nativamente.

[29] Asynchronous JAVASCRIPT and XML
[30] Indica una componente che accede ai servizi o alle risorse di un'altra componente detta server. Fonte Wikipedia
[31] Internet Explorer

Per inviare una richiesta ad un server, bisogna utilizzare il seguente codice:

```
var xmlhttp = getHTTPObject();
//Include POST OR GET
xmlhttp.open('POST', endpoint, true);
xmlhttp.setRequestHeader('Content-type',
  'application/x-www-form-urlencoded');
xmlhttp.setRequestHeader("Accept",
  "application/sparql-results+json");
xmlhttp.onreadystatechange = function() {
 if(xmlhttp.readyState==4 ){
  if(xmlhttp.status==200){
   callback(xmlhttp.responseText,map);
  }else
  // Error
  alert("Error: Status: "+ xmlhttp.status + "Response: "
    + xmlhttp.responseText);
 }
};
// Send the query to the endpoint.
xmlhttp.send(querypart);
```

Una volta creato un oggetto di tipo **getHTTPObject()**, la variabile **xmlhttp**, con i metodi **open** e **send**, si apre e si invia la richiesta al server.
Con open si possono effettuare due tipi di chiamata. La prima può essere fatta tramite GET, mentre la seconda può essere fatta tramite una chiamata POST. Per una chiamata di tipo POST però bisogna impostare gli header,[32] attraverso il comando:

```
xmlhttp.setRequestHeader('Content-type','application/x-www-form-urlencoded')
```

Alla variabile **onreadystatechange** viene associata una funzione; tale funzione dipende dal valore assunto dal metodo readyState. Il metodo **readyState** tiene traccia dello stato di XMLHttpRequest e può assumere determinati valori:

| Valore | Commento |
|---|---|
| 0 | richiesta non inizializzata. |
| 1 | connessione server stabilita. |
| 2 | richiesta ricevuta. |
| 3 | richiesta in processo. |
| 4 | richiesta finita e responso è pronto. |

Tabella 6: Tabella degli stati di XMLHttpRequest.

Infine con il metodo **status** si controlla la risposta dal server e può assumere diversi valori:

- 200 Il server è stato trovato;
- 403 Forbidden;
- 404 Server Not Found;
- 500 Internal Server Error;
- ...

[32] Serie di coppie chiave/valore specifici per uno scambio dati via internet.

Quando si verifica che **readyState** ha valore 4 e **status** ha valore 200, il client comunica con il server e il risultato, dato dal metodo **responseText** viene convertito in una stringa alla quale, successivamente, si applicherà la codifica JSON. Tuttavia però tale chiamata non sempre va a buon fine. I browser moderni hanno una sorta di restrizione,[33] cioè che non si può fare una richiesta HTTP da risorse che si trovano su server diversi rispetto a quello iniziale che ha inviato lo script.
Tale richiesta viene chiamata richiesta **Cross-domain**.[34]

## 5.4 Il problema delle richieste Cross-Domain

Quando si effettua una chiamata AJAX verso domini diversi da quello da cui proviene la pagina, i moderni browser applicano una restrizione che impedisce l'esecuzione dello script. Una tecnica per aggirare il problema è il **JSONP**,[35] che consiste nel realizzare una funzione che permetta di inserire nuovi tag <script> all'interno della pagina e una funzione che gestisca i dati ricevuti dal server come mostrato nel seguente codice:

```
1  function requestJSONP(endpoint,query,callback){
2    var queryUrl = endpoint+"?query="+ encodeURIComponent(query) +"&format=json";
3
4    $.ajax({
5          type:"GET",
6          dataType: "jsonp",
7          url: queryUrl,
8          success: callback
9      });
10 }
```

La sintassi utilizzata in questo breve codice è **jQuery**, che è un framework JavaScript. Con il jQuery è possibile realizzare, attraverso pochissime righe di codice, script molto complessi. Il punto di forza di questo framework è che è compatibile con tutti i browser in circolazione.
Analizzando il codice notiamo

- **queryUrl**: è la variabile che inserisce i tag all'interno della URL, tra cui la query e l'endpoint;
- **type**: è il metodo utilizzato dal client per comunicare con il server; in questo caso GET;
- **callback**: è la funzione che si intende eseguire; come ad esempio una creazione di una tabella.

In questo modo riusciamo ad aggirare il problema e ad ottenere la codifica JSON da un dominio diverso da quello di partenza.

[33] Same-domain-policy
[34] Capitolo 5.4
[35] JSON with Padding

# 6 Presentazione e codice dell'applicazione

In questo ultimo capitolo verranno illustrati sia l'elenco dei programmi utilizzati, sia il codice del progetto.

## 6.1 Programmi utilizzati

Il progetto è stato realizzato utilizzando due sistemi operativi:

- **Linux Mint 17.1 Rebecca**: per lo sviluppo del codice e per la stesura della tesi;
- **Windows 10**: per la pubblicazione su internet.

I programmi utilizzati su piattaforma Linux Mint 17.1 Rebecca per la realizzazione della tesi e per lo sviluppo del codice sono:

1. **Notepadqq**:[36] è un editor di testo open-source sviluppato su Linux capace di riconoscere più di 100 linguaggi di programmazione diversi. Raggruppa il codice e evidenzia con colori diversi gli environment dei linguaggi, come ad esempio variabili o funzioni. È possibile, inoltre, cercare il testo utilizzando la nomenclatura delle espressioni regolari. La parte codice è stata realizzata attraverso il Notepaqq.

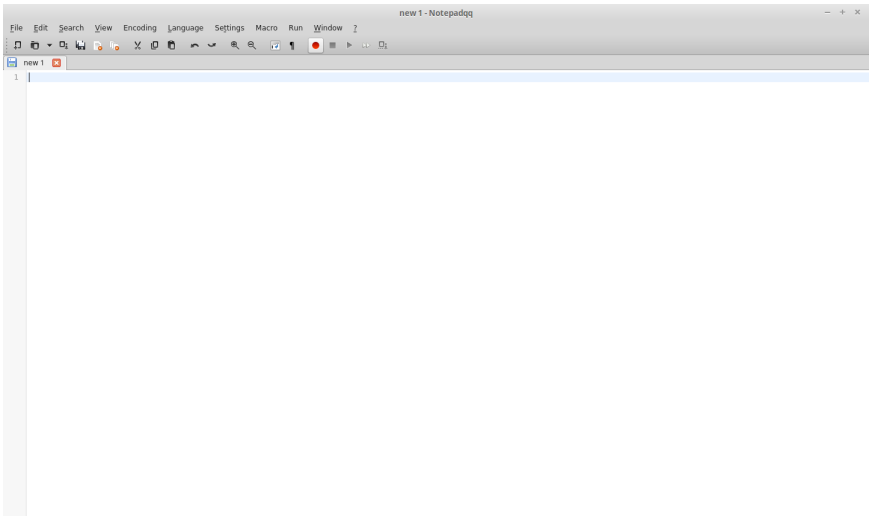


Figura 10: Notepadqq.

[36] http://notepadqq.altervista.org/wp/

2. **TexMaker**:[37] è un editor LaTeX gratuito per Linux, MacOSX e Windows rilasciato sotto la licenza **GNU General Public License**.[38] Integra molti strumenti necessari per sviluppare documenti con il linguaggio LaTeX, quali il supporto unicode, il controllo ortografico, il completamento automatico, il raggruppamento del codice e un visualizzatore di PDF. La scrittura della tesi in LaTeX è realizzata attraverso questo programma.

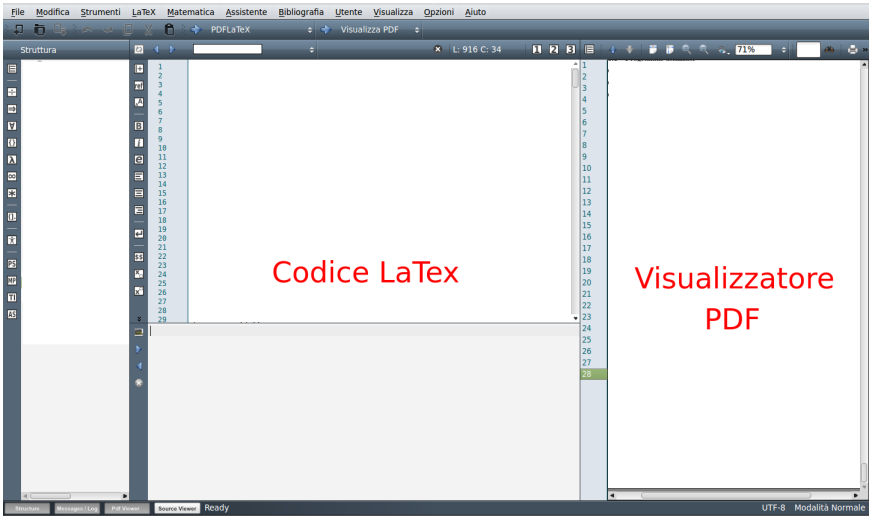


Figura 11: Texmaker.

La parte della pubblicazione su internet è stata effettuata su Windows 10; il codice del progetto è stato pubblicato su un **free-hosting**,[39] ed è stato scelto **GitHub**,[40]un portale web ideale per ospitare progetti software privati, gratuiti e open-source. Basato sul sistema **Git**[41] GitHub è stato lanciato nel 2009 da Tom Preston-Werner, Chris Wanstrath e PJ Hyett, diventando ben presto la più autorevole piattaforma per lo sviluppo collaborativo di software.
Git è un sistema di controllo di versione, ciò vuol dire che controlla e gestisce gli aggiornamenti di un progetto senza sovrascrivere nessuna parte del progetto stesso. In questo modo è possibile ritornare indietro se l'aggiornamento del progetto non dovesse andare a buon fine.

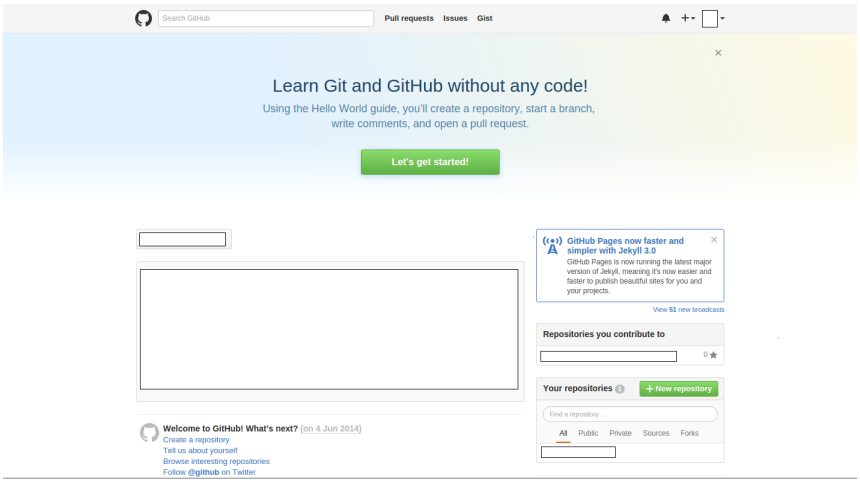


Figura 12: GitHub.

Questo portale web è un mezzo molto potente, perché si può collaborare con altre persone che non si trovano nella stessa sede; in più, in caso di malfunzionamento del pc, non si perde il lavoro svolto;

[37] `http://www.xm1math.net/texmaker/`
[38] Licenza per software libero `https://gnu.org/licenses/gpl.html`.
[39] Un servizio di rete che consiste nell'allocare su un server web pagine web o un'applicazione web
[40] `https://github.com/`
[41] Sviluppato da Linus Torvalds il creatore di Linux.

infatti è ideale per la stesura di una tesi di laurea.
Una volta effettuato l'accesso al portale web, possiamo creare i **repository**[42] e gestirli, come mostrato in **Figura 13**.

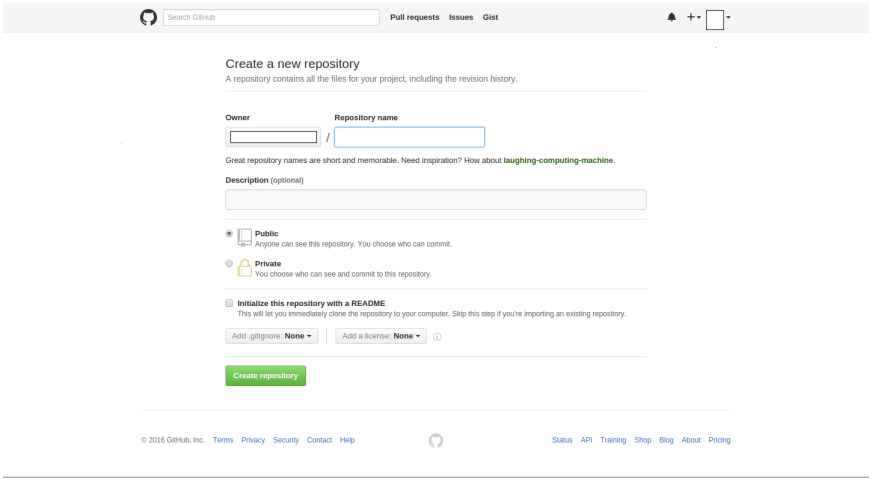


Figura 13: Creazione di un repository.

Bisogna fornire il nome del repository e scegliere se renderlo pubblico o privato. Una volta creato il repository, bisogna pubblicare il codice su GitHub attraverso un comando Commit eseguito su un Terminale.[43] Per agevolare questo processo e per evitare errori è possibile utilizzare un client grafico, ad esempio, **SourceTree**.[44] SourceTree è basato su Git e Mercurial e gestisce tutti i repository di GitHub attraverso un'interfaccia semplice. Una volta fatta la registrazione, fornendo un email e la password è possibile creare, clonare, eseguire commit, con un semplice clic. Per caricare il progetto

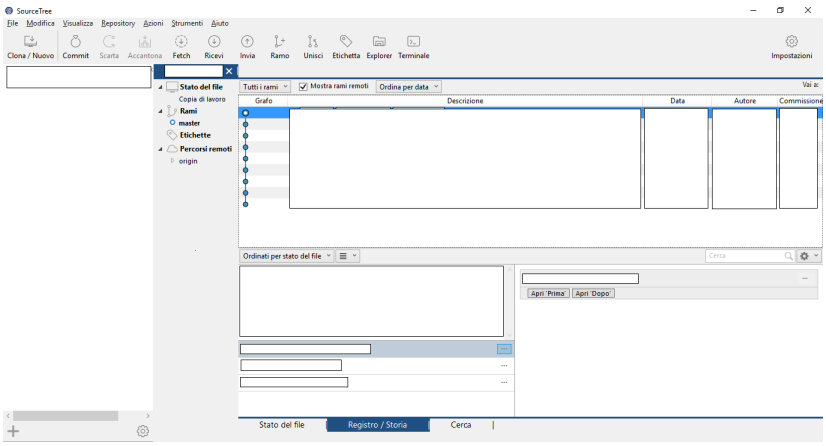


Figura 14: SourceTree.

non bisogna far altro che premere il pulsante **Clona/Nuovo**, nella schermata successiva fornire il percorso sorgente, cioè quello di GitHub, il percorso di destinazione e cliccare su Clona, come mostrato in **Figura 15**.

[42] E' un ambiente di un sistema informativo, in cui vengono gestiti i metadati, attraverso tabelle relazionali. Fonte Wikipedia.
[43] cmd per windows, shell per Linux
[44] `https://www.sourcetreeapp.com/`

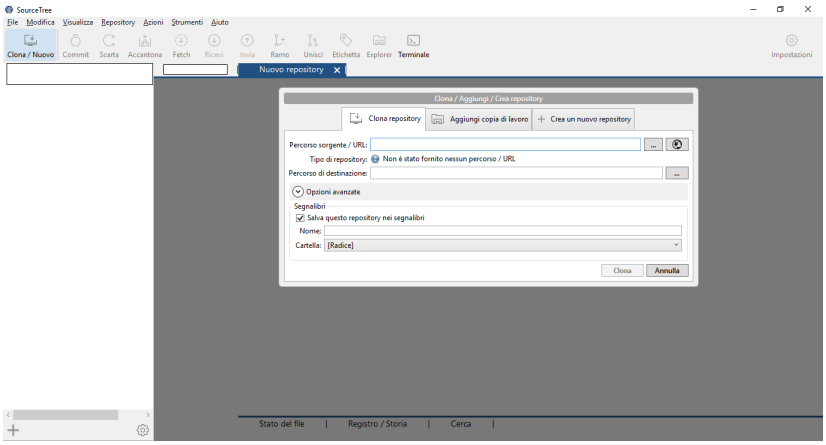


Figura 15: Pannello Clona di SourceTree.

Da questo momento in poi se nella cartella di destinazione c'è un file modificato SourceTree rileva la modifica e bisogna effettuare il commit, specificando le modifiche che si sono effettuate. Infine per mandare i file su GitHub bisogna premere il pulsante **Invia**.

## 6.2 HTML

### index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset=utf-8 />
    <title>MAPPA</title>
    <link rel="stylesheet" href="./css/leaflet.css"/>
    <link rel="stylesheet" href="./css/menu.css"/>
    <script src="./js/leaflet.js"></script>
    <script src="./js/client.js"></script>
  </head>
  <body>
    <div id='map'>
    <div id="loading"><p class="loading">Loading...<p></div>
    </div>
    <div id="navigation"></div>
    <script type="text/javascript">
      launch();
    </script>
  </body>
</html>
```

## 6.3 Css

### menu.css

```
1
2  /*Loading*/
3  #loading{
4        /*opacity:0.80;
5        filter:alpha(opacity=80); /* For IE8 and earlier */
6        margin: 0px auto;
7        width:1000px;
8        height: 600px;
9        background-color: #e3e3e3;
10       z-index: 2;
11       position:relative;
12       padding-top:20px;
13       display:block;
14       }
15
16 p.loading{
17       text-align:center;
18       line-height: 500px;
19       color:#272a80;
20       font-size:37px;
21       margin: 0px auto;
22       padding-top:20px;
23       display:block;
24       }
25 /*Stile mappa*/
26 #map {
27     height: 600px;
28     width: 1000px;
29     display: block;
30     margin: 0px auto;
31     text-align: center;
32     z-index: 0;
33     }
34
35 /* Fix IE. Hide from IE Mac \*/
36 * html ul li { float: left; }
37 * html ul li a { height: 1%; }
38 /* End */
39
40 #navigation{
41       visibility: hidden;
42       margin: 0px auto;
43       display: block;
44       position:ab4;
45       top: 0px;
46       left: 0 px;
47       z-index: 1;
48       }
49 #nav{
50   font-size: 12px;
51   margin: 0px auto;
52   display: block;
53   position:relative;
54   top: 10px;
55   left: 20px;
56 }
57 /* CSS Document */
58 #nav, #nav ul {
59   list-style:none;
60   padding:0;
61   margin:0;
62   position:absolute;
63 }
```

```
#nav li {
  float:left;
  position:relative;
  line-height:2.5em;
  width:30em;
}
#nav li ul {
  position:absolute;
  margin-top:-1em;
  margin-left: 1em; /* for IE */
  display:none;
}
#nav ul li ul {
  margin-top:-1.5em;
  margin-left:7em;
}
/* ********************************************************************** */
/* SHOW SUBMENU  1 */
#nav li:hover ul, #nav li.over ul {
  display:block;
}
#nav li:hover ul ul, #nav li.over ul ul {
  display:none;
}
/* SHOW SUBMENU  2 */
#nav ul li:hover ul, #nav ul li.over ul {
  display:block;
}
/* ********************************************************************** */
/* STYLING UP THE LINKS */
#nav a {
  display:block;
  border-right:1px solid #fff;
  background:#FCF800;
  color:#000000;
  font-weight:bold;
  text-decoration:none;
  padding:0 10px;
  border-radius: 10px 10px 10px 10px;
}
#nav a:hover {
  background-color:#5798B4;
  color:#fff;
}
#nav ul {
  border-top:1px solid #fff;
  border-radius: 10px 10px 10px 10px;
}
#nav ul a {
  border-right:none;
  border-right:1px solid #fff;
  border-bottom:1px solid #fff;
  border-left:1px solid #fff;
  background:#FF0F0F;
}
#nav ul ul a{
  border-right:none;
  border-right:1px solid #fff;
  border-bottom:1px solid #fff;
  border-left:1px solid #fff;
  background:#005DE0;
  }

/* ********************************************************************** */

#nav {
```

```
130   z-index:1;
131 }
132 #nav ul {
133   z-index:2;
134 }
135 #nav ul ul {
136   z-index:3;
137 }
```

## 6.4 JavaScript

```
1  function launch(){
2    var x=location.search.split('?select=');
3    var selectGet;
4    if(x.length<2){
5      selectGet=null;
6    }
7    else{
8      selectGet=decodeURIComponent(x[1]);
9    }
10   //Create Map
11   var map=createMap();
12   //Include knowledge base
13   var endpoint="http://dydra.com/cristianolongo/comune-di-catania/sparql";
14   //Created query
15   var query =
16         "PREFIX org:<http://www.w3.org/ns/org#> \n"+
17         "PREFIX foaf:<http://xmlns.com/foaf/0.1/>\n"+
18         "PREFIX locn:<http://www.w3.org/ns/locn#>\n"+
19         "PREFIX geo:<http://www.w3.org/2003/01/geo/wgs84_pos#>\n"+
20
21         "select ?site ?address ?lat ?lon ?dir ?dir_name ?dir_homepage ?dir_tel ?
               dir_mail where\n"+
22         "{?dir org:hasPrimarySite ?site . \n"+
23         "?site locn:address ?a ."+
24         "?a locn:fullAddress ?address ."+
25         "?site locn:geometry ?g .\n"+
26         "?g geo:lat ?lat .\n"+
27         "?g geo:long ?lon .\n"+
28         "?dir rdfs:label ?dir_name .\n";
29         if(selectGet){
30           query+="?dir org:transitiveReflexiveSubOrganizationOf <"+selectGet+">.\
                 n"
31         }
32         query+="optional {?dir foaf:homepage ?dir_homepage} .\n"+
33         "optional {?dir foaf:phone ?dir_tel} .\n"+
34         "optional {?dir foaf:mbox ?dir_mail}\n"+
35         "}order by ?site ?dir"  ;
36   var query_menu=
37         "PREFIX org:<http://www.w3.org/ns/org#> \n"+
38
39         "select ?u ?label ?homepage ?child ?childlabel ?childhomepage where { \n"
               +
40         "?u a org:Organization . \n"+
41         "?u rdfs:label ?label . \n"+
42         "optional { ?u foaf:homepage ?homepage .} \n"+
43         "?u org:hasSubOrganization ?child .\n"+
44         "optional { ?child foaf:homepage ?childhomepage .} \n"+
45         "?child rdfs:label ?childlabel .\n"+
46         "} order by ?u";
47   createSparqlQuery(endpoint,query,map,createMarker);
48   createSparqlQuery(endpoint,query_menu,map,createMenu);
49 }
50
51 //Created SPARQL query
```

```
52 function createSparqlQuery(endpoint,query,map,callback){
53   var querypart = "query=" + escape(query);
54   // Get our HTTP request object.
55   var xmlhttp = getHTTPObject();
56   //Include POST OR GET
57   xmlhttp.open('POST', endpoint, true);
58   xmlhttp.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
59   xmlhttp.setRequestHeader("Accept", "application/sparql-results+json");
60   xmlhttp.onreadystatechange = function() {
61     if(xmlhttp.readyState==4 ){
62       if(xmlhttp.status==200){
63           callback(xmlhttp.responseText,map);
64       }else
65         // Error
66         alert("Error: Status: "+ xmlhttp.status + "Response: "
67             + xmlhttp.responseText);
68     }
69   };
70   // Send the query to the endpoint.
71   xmlhttp.send(querypart);
72 }
73 //Created Menu
74 function createMenu(response,map){
75   //Request accept
76   var jsonObj=eval('(' + response + ')');
77   var cut=jsonObj.results.bindings;
78   var storage=new Storage();
79
80   var descriptionParent="";
81   var descriptionChild="";
82
83   var homepageChild="";
84   var homepageParent="";
85
86   var searchParent="";
87   var searchChild="";
88
89   for(var i = 0; i< cut.length; i++) {
90     try{
91       homepageParent = cut[i].homepage.value;
92     }
93     catch(err){
94       homepageParent = ""
95     }
96     try{
97       homepageChild = cut[i].childhomepage.value;
98     }
99     catch (err){
100      homepageChild = "";
101    }
102    descriptionParent = new Description (cut[i].u.value, cut[i].label.value,
         homepageParent);
103    descriptionChild = new Description (cut[i].child.value, cut[i].
         childlabel.value, homepageChild);
104
105    searchParent=storage.getNode(descriptionParent.getUri());
106    searchChild=storage.getNode(descriptionChild.getUri());
107
108    if(searchParent === undefined){
109      searchParent=storage.add(descriptionParent);
110    }
111    if(searchChild === undefined){
112      searchChild=storage.add(descriptionChild);
113    }
114    searchParent.nodetree.addChild(searchChild.nodetree);
115    searchParent.parent= true;
```

```
116       searchChild.parent= true;
117     }
118     document.getElementById("navigation").innerHTML="<ul id=\"nav\"></ul>";
119     createLiMenu(storage.nodes[0].nodetree.value,storage.nodes[0].nodetree.children
            ,"nav");
120   }
121   //Create Li menu'
122   function createLiMenu(value,children,id){
123     var node=""
124     var current=""
125     var next=""
126     var ul=""
127     var ul2="";
128     var a=""
129     /*First element*/
130     if(id.localeCompare("nav")==0){
131       a=document.createElement("a");
132       next=document.createElement("LI");
133       next.setAttribute("id", value.getName());
134       a.textContent=value.getName();
135       a.setAttribute('href', "index.html?select="+encodeURIComponent(value.getUri()
              ));
136       next.appendChild(a);
137       document.getElementById(id).appendChild(next);
138     }
139     /*Children*/
140     if(children.length >0 ){
141       ul=document.createElement("UL");
142       ul.setAttribute("id","nav_"+value.getName());
143       for (var i in children) {
144         a=document.createElement("a");
145         current=document.createElement("LI");
146         current.setAttribute("id",children[i].value.getName());
147         a.textContent = children[i].value.getName();
148         a.setAttribute('href', "index.html?select="+encodeURIComponent(children[i].
              value.getUri()));
149         current.appendChild(a);
150         ul.appendChild(current);
151         }
152       document.getElementById(value.getName()).appendChild(ul);
153       }
154     for (var i in children){
155       createLiMenu(children[i].value,children[i].children,ul.getAttribute("id"));
156     }
157   }
158   //Create Marker
159   function createMarker(response,map){
160     //Request accept
161     var jsonObj=eval('(' + response + ')');
162     var t=0;//Pointer in the head of json
163     var str=""
164     var email=""
165     var table= new Array()
166     //Create Marker
167     for(var i = 1; i<  jsonObj.results.bindings.length; i++) {
168       //Address and site are equals
169       if (jsonObj.results.bindings[t].site.value.localeCompare(
              jsonObj.results.bindings[i].site.value)==0 &&
170           jsonObj.results.bindings[t].address.value.localeCompare(
                  jsonObj.results.bindings[i].address.value)==0 ){
171         try{
172           //Clone telephone number
173           str+=jsonObj.results.bindings[t].dir_tel.value+"<br>";
174           }
175
176         catch (err){
```

```
177           //Telephone number not found
178           str+="";
179           }
180       }
181       else{
182         //Change pointer
183         try{
184           str+=jsonObj.results.bindings[t].dir_tel.value+"<br>";
185         }
186         catch(err){
187           str+=""
188         }
189         //Insert email
190         try{
191           email=jsonObj.results.bindings[t].dir_mail.value+"<br>";
192         }
193         catch(err){
194           email=""
195         }
196         //Create table
197         table[table.length]= new Array("<a href=\""+jsonObj.results.bindings[t].
              dir_homepage.value+"\">"+jsonObj.results.bindings[t].dir_name.value+"</a
              ></b>",
198                   jsonObj.results.bindings[t].address.value,email,str,
                        jsonObj.results.bindings[t].lat.value,
199                   jsonObj.results.bindings[t].lon.value);
200         str="";
201       }
202       t=i;
203       //Last element
204       if (t==jsonObj.results.bindings.length-1){
205         try{
206           str+=jsonObj.results.bindings[t].dir_tel.value+"<br>";
207         }
208         catch(err){
209           str+=""
210         }
211         //Insert email
212         try{
213           email=jsonObj.results.bindings[t].dir_mail.value+"<br>";
214         }
215         catch(err){
216           email=""
217         }
218         table[table.length]= new Array("<a href=\""+jsonObj.results.bindings[t].
              dir_homepage.value+"\">"+jsonObj.results.bindings[t].dir_name.value+"</a
              ></b>",
219                   jsonObj.results.bindings[t].address.value,email,str,
                        jsonObj.results.bindings[t].lat.value,
220                   jsonObj.results.bindings[t].lon.value);
221         str="";
222
223       }
224
225     }
226     createMessage(table,map);
227   }
228   //Create Message of PopUp
229   function createMessage(table,map){
230     var message="";
231     var control="no";
232     for (var i in table) {
233       for (var j in table){
234         if (j!=i && (table[i][4].localeCompare(table[j][4])==0 && table[i][5].
              localeCompare(table[j][5])==0)){
235           if(table[i][0].localeCompare(table[j][0])==0){
```

```
236             message=table[i][0]+"<br>"+table[i][1]+"<br>"+table[j][1]+"<br>"+
237                     table[i][3]+table[i][2]+"<br>";
238             control="yes";
239             createIcon(map,table[i][4],table[i][5],message);
240
241           }
242           else{
243             message=table[i][0]+"<br>"+table[i][1]+"<br>"+
244                 table[i][3]+table[i][2]+"<br>"+
245                 table[j][0]+"<br>"+table[j][1]+"<br>"+
246                 table[j][3]+table[j][2]+"<br>";
247             control="yes";
248             createIcon(map,table[i][4],table[i][5],message);
249           }
250         }
251       }
252       if(control.localeCompare("no")==0){
253         message=table[i][0]+"<br>"+table[i][1]+"<br>"+
254             table[i][3]+table[i][2]+"<br>";
255         createIcon(map,table[i][4],table[i][5],message);
256       }else
257         control="no";
258     }
259     document.getElementById("navigation").style.visibility = "visible";
260     document.getElementById("loading").style.visibility = "hidden";
261   }
262
263   //Request HTTP
264   function getHTTPObject(){
265     var xmlhttp;
266     if(!xmlhttp && typeof XMLHttpRequest != 'undefined'){
267       try{
268         // Code for old browser
269         xmlhttp=new ActiveXObject('Msxml2.XMLHTTP');
270         }
271       catch(err){
272         try{
273           // Code for IE6, IE5
274           xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
275         }
276         catch(err2){
277           try{
278             // Code for IE7+, Firefox, Chrome, Opera, Safari
279             xmlhttp=new XMLHttpRequest();
280           }
281           catch(err3){
282             xmlhttp=false
283           }
284         }
285       }
286     }
287     return xmlhttp;
288   }
289   //Created Map
290   function createMap(){
291     var auxmap = L.map('map').setView([37.506, 15.079], 14);
292     L.tileLayer('https://{s}.tiles.mapbox.com/v3/{id}/{z}/{x}/{y}.png', {
293           maxZoom: 18,
294           attribution: 'Map data &copy; <a href="http://openstreetmap.org">
                OpenStreetMap</a> contributors, '+
295           '<a href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>, '
                +
296           'Imagery <a href="http://mapbox.com">Mapbox</a>',
297           id: 'andreacostazza.ik9ap86i'
298         }).addTo(auxmap);
299     return auxmap
```

```
}
//Created Icon
function createIcon(map,lat,lon,message){
  //Inserted an icon
  var iconBlue= L.icon({
    iconUrl: './icon/marker-icon.png',
    shadowUrl: './icon/marker-shadow.png',

    iconSize: [25,41],
    shadowSize: [41,41],
    iconAnchor:[lat,lon],
    shadowAnchor:[lat,lon],
    popupAnchor:[-25,-10]
  });

  var marker = L.marker([lat, lon],{icon:iconBlue});
  marker.addTo(map);
  marker.bindPopup(message);
}

//Create Object Description with uri,name,homepage attributes
function Description(uri,name,homepage){
  this.uri=uri
  this.name=name;
  this.homepage=homepage;

  //Return uri
  this.getUri=
    function(){
      return this.uri;
  }
  //Return name
  this.getName=
    function(){
      return this.name;
  }
  //Return homepage
  this.getHomepage=
    function(){
      return this.homepage;
  }
}
//Creation class Storage
function Storage(){
  this.nodes = [];

  //Create instance of Tree class and add node into Storage
  this.add=
    function(description){
      var tree=new Tree(description);
      this.nodes.push({
        nodetree: tree,
        parent: false
      });
      return this.nodes[this.nodes.length-1];

  }
  //Return node corresponding to specified uri
  this.getNode=
    function(uri){
      if (this.nodes === undefined || this.nodes.length == 0) {
        // empty
      return undefined;
      }
      for(var i in this.nodes){
          if(this.nodes[i].nodetree.value.getUri().localeCompare(uri)==0)
```

```
366             return this.nodes[i];
367       }
368       return undefined;
369   }
370   //Return nodes without parent
371   this.getNodesWithoutParent=
372     function(index){
373         if(this.nodes.length==index )
374           return "";
375         if(this.nodes[index].parent== false ){
376           return this.nodes[index].nodetree.value.getName()+
                  this.getNodesWithoutParent(index+1) + " ";
377         }
378         else
379           return this.getNodesWithoutParent(index+1) + " ";
380     }
381 }
382 //Create N-ary Tree
383 function Tree(value){
384   this.value=value;
385   this.children= [] ;
386
387   this.addChild =
388     function (child){
389       this.children.push(child);
390     }
391
392   this.getChild =
393     function (){
394       var current=""
395       var next=""
396       current+=this.value.getUri() +" has children: ";
397       for (var i in this.children) {
398         current+= this.children[i].value.getUri() + " ";
399       }
400       for (var i in this.children){
401         next+="<br>"+this.children[i].getChild()+" ";
402       }
403       return current+next;
404     }
405   this.preOrder =
406     function(){
407       var visit = "";
408       for(var i in this.children){
409         visit+=this.children[i].preOrder()+" ";
410       }
411       return this.value +" " +visit + " ";
412     }
413
414   this.postOrder =
415     function(){
416       var visit ="";
417       for(var i in this.children){
418         visit+=this.children[i].postOrder()+" ";
419       }
420       return visit + this.value + " ";
421     }
422
423   this.high =
424     function(){
425       if(this.children.length==0){
426         return 0;
427       }
428       var maxHigh =this.children[0].high();
429       for(var i=1 ; i<this.children.length; i++){
430         var highSon =this.children[i].high();
```

```
431           if(highSon >maxHigh){
432             maxHigh=highSon
433           }
434         }
435         return maxHigh + 1;
436       }
437
438     this.frontier=
439       function(){
440         if(this.children.length == 0){
441           return this.value + " ";
442         }
443         var f="";
444         for (var i in this.children){
445           f += this.children[i].frontier();
446         }
447         return f;
448       }
449 }
```

# A Siti Utili

- http://www.opendatahacklab.org/site/projects.html
- https://github.com/opendatahacklab/cityservices
- https://it.wikipedia.org/wiki/Pagina_principale
- http://www.w3.org/
- https://www.w3.org/ns/locn
- http://www.w3.org/TR/vocab-org/
- https://www.w3.org/standards/techs/sparql#w3c_all
- http://www.di.unipi.it/~ambriola/pw/radice.htm
- http://http://homes.di.unimi.it/~ghilardi/logica2/DL.pdf
- http://dot-maui.blogspot.it/2014/02/javascript-leggere-i-parametri-in-get.html
- http://leafletjs.com/
- http://www.agid.gov.it/
- http://semapps.blogspot.it/2012_05_01_archive.html
- http://www.html.it/guide/guida-ajax/
- http://www.html.it/articoli/jsonp-e-le-richieste-cross-domain-1/
- http://cosenonjaviste.it/jsonp-e-jquery-conosciamoli-meglio/
- http://cosenonjaviste.it/jquery-tutorial-parte-i/
- http://www.xm1math.net/texmaker/
- https://github.com/
- http://notepadqq.altervista.org/wp/
- https://www.sourcetreeapp.com/
- https://www.overleaf.com/latex/examples/listings-code-style-for-html5-css-html-javascript
  htstpdbpnpmt#.VsO4YOzhBpS
- http://www.html.it/articoli/jsonp-e-le-richieste-cross-domain-1/
- http://www.dmi.unict.it/~longo/comunect/publicServicesLOD.pdf
- http://dydra.com/cristianolongo/comune-di-catania/@query
- http://www.w3schools.com/
- http://it.dbpedia.org/

## Riferimenti bibliografici


[1] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.

[2] Tim Berners-Lee. A roadmap to the Semantic Web. `http://www.w3.org/DesignIssues/Semantic.html`, 1998.

[3] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American Magazine*, May 2001.

[4] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[5] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006.