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

Indice

1 Introduzione **6**

2 Web Semantico **7**
2.1 Resource Description Framework (RDF) 8
2.2 Logiche descrittive 11

3 Mappe on-line per siti web **14**
3.1 Caricamento mappa 14
3.2 Creazione delle icone, dei markers e dei popups 17

4 SPARQL **18**
4.1 Il modello Turtle 18
4.2 Comandi principali 19
4.3 Libreria javascript per interrogazioni SPARQL 20

5 Ontologie per la rappresentazione dei servizi pubblici **21**
5.1 Menù gerarchici 22
5.2 Codifica JSON 24
5.3 Chiamata AJAX 25

6 Presentazione e codice dell'applicazione **28**
6.1 Programmi utilizzati 28
6.2 HTML 28
6.3 Css 28
6.4 Il problema delle richieste Cross-Domain 28

1 Introduzione

2 Web Semantico

Il termine Web Semantico 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[1](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**[2] 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,**[3] 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**[4], 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(vedi Paragrafo 2.1) ed elaborata attraverso il linguaggio di programmazione Javascript, ma ne parleremo più avanti.
Analizzato 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**

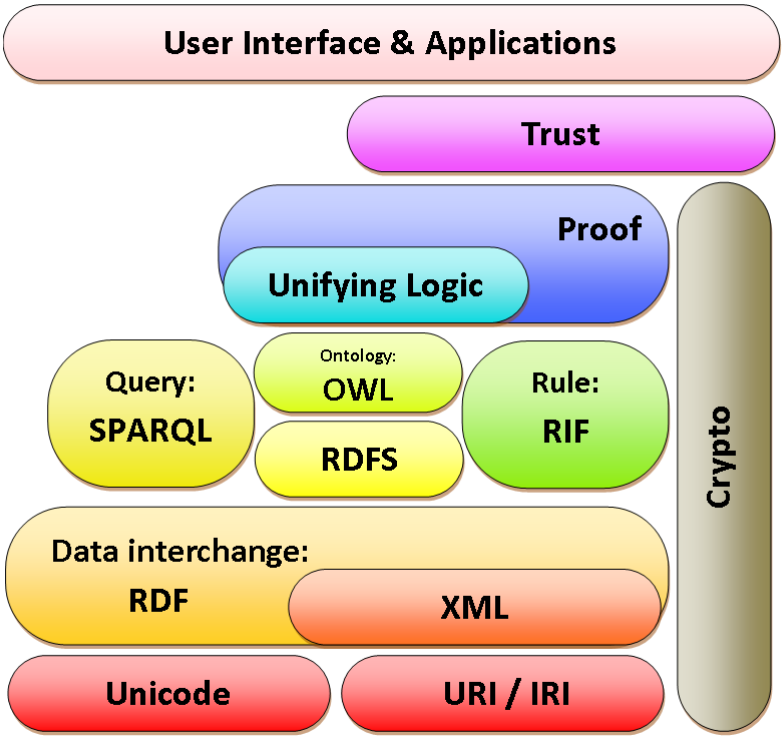

Figura 1: Rappresentazione a livelli del Web Semantico.

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

I livelli principali[5] sono:

- URI/IRI:[6] 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:[7] 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 un URI;
- RDF:[8] 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:[9] Estensione di RDF, fornisce elementi di base per la descrizione di ontologie, chiamate vocabolari per le risorse RDF;
- OWL:[10] 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 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 2**.

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

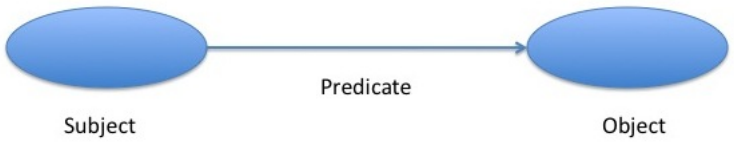

Figura 2: Relazione degli statement.

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 e soggetto di 2 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 in questo un grafo dove il soggetto e l'oggetto rappresentano i nodi mentre al predicato corrisponde l'arco.

Figura 3: Esempio di Grafo.

Il grafo indicato in Figura 3 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 però possono assumere qualsiasi forma, le principali forme assunte sono:

- **URI/IRI**: come ad esempio la risorsa Napoleone può essere contenuta, per esempio, su DBpedia;[11]
- **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.[12]

Per rappresentare semanticamente il modello RDF, si utilizza un'estensione chiamata RDF Schema, che fornisce meccanismi per gruppi di risorse correlate e descrive le risorse con le classi, proprietà e 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 viene descritto 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.

[11] 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/`

[12] Fonte Wikipedia.

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.

2.2 Logiche descrittive

La logica descrittiva[13] 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**: costanti usate solo nelle asserzioni.

Una base di conoscenza per le DL è costituito 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**. Un linguaggio descrittivo è il 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.[14]

[13]Description Logics
[14] `http://homes.di.unimi.it/~ghilardi/logica2/DL.pdf`). Introduzione alle Logiche Descrittive di Silvio Ghilardi

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 $\top$(sempre soddisfatto) e per il concetto contraddittorio $\bot$(mai 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** 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 $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

Una volta fatta la premessa sul linguaggio descrittivo, definiamo adesso la logica descrittiva di base chiamata $\mathcal{ALC}$[15] e definiamo i seguenti concetti:

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$;
- $\bot^{\mathcal{I}} = 0$;
- $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$;
- $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$;
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$;
- $(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^I | (\forall [a,b] \in R^{\mathcal{I}})(b \in C^I)\}$;
- $(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^I | (\exists [a,b] \in R^{\mathcal{I}}) \wedge (b \in C^I)\}$.

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$\in$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:

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

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

[15] Attribute Language with Complement

Per estendere il linguaggio si dice che $\mathcal{I}$ è modello $\models$ di:
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 $(\mathcal{A}, \mathcal{T})$ e l'interpretazione $\mathcal{I}$ è modello della base di conoscenza 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**, accessibile digitando l'indirizzo url `http://leafletjs.com/`.
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 con soli circa 33 KB di memoria, inoltre utilizzando la tecnologia **HTML5** e **CSS3** 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 più 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 il file scritto in 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'.

Come mostrato nel seguente esempio:

```
<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>
```

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. Per fare questo occorre registrarsi al sito web fornendo:

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

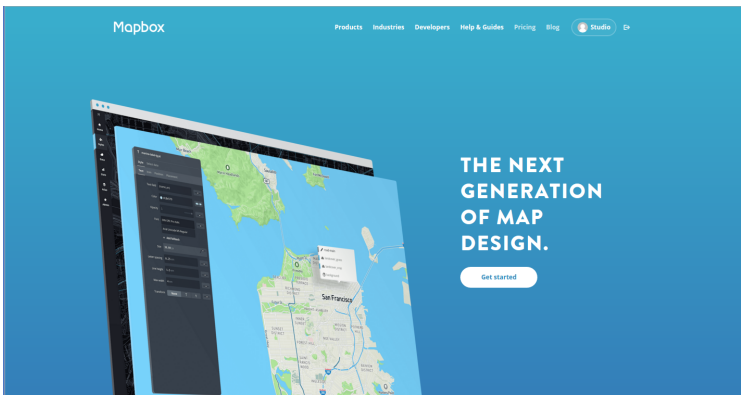

Figura 4: Sito di Mapbox.

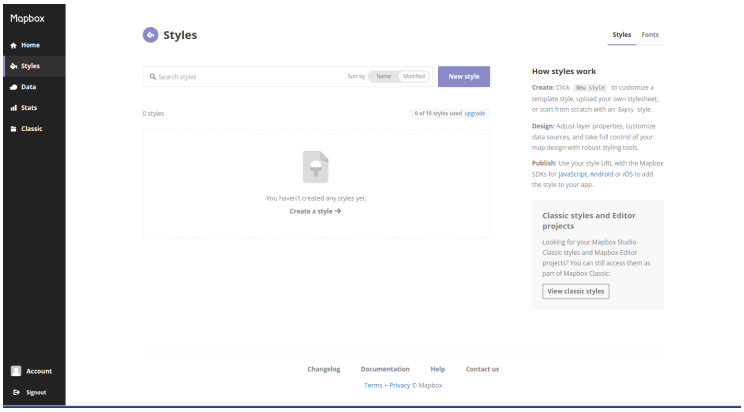

Figura 5: Pannello Styles.

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 creerà una nuova mappa da poter modellare a seconda delle proprie necessità. Mostrato in Figura 4 e Figura 5.
Una volta creata la mappa, cliccare sul pannello **Edit**, in questo modo 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 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 6 è mostrato la schermata **Edit**.

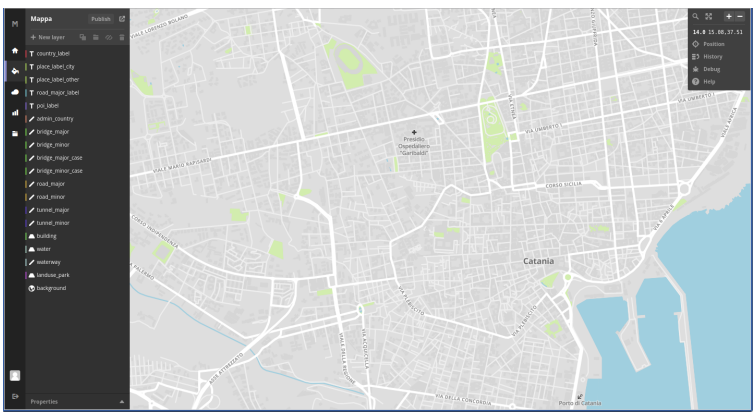

Figura 6: Pannello Edit.

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 reperibile sul pannello Edit come mostrato in Figura 7, e devono essere inserite nella sezione body della file **index.html**.

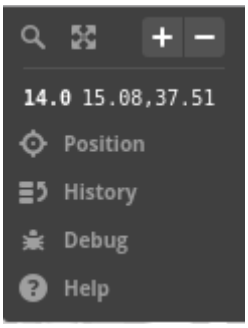

Figura 7: Coordinate e zoom mappa.

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>
```

Dalla 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={pk.eyJ1IjoiYW5kcmVhY29zdGF6emEiLCJhIjoiN3VXTWIxayJ9.'+
    'iG0HHcB2nWD2-I9-tx0vhA}', {
  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: 'pk.eyJ1IjoiYW5kcmVhY29zdGF6emEiLCJhIjoiN3VXTWIxayJ9.'+
  'iG0HHcB2nWD2-I9-tx0vhA'
        }).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 5 accanto alla voce Edit. Infine alla voce **accessToken**, si deve indicate 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 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

Il linguaggio **SPARQL**[16] 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 in tutto il mondo.
La struttura del database è un insieme di triple soggetto-predicato-oggetto, però a differenza dell'RDF, le 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 che fa riferimento a rdf.[17]
I dati ottenuti, facendo una query[18] allo SPARQL, è una tabella molto simile a quelle utilizzate nei database relazionali in SQL; si tratta di una tabella dove le colonne sono composte dal risultato del soggetto, del predicato e dell'oggetto.

4.1 Il modello Turtle

La sintassi utilizzata nello SPARQL è simile a Turtle[19] 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 6. 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 8: Esempio Codice Turtle.

attraverso il comando **@prefix**, a cui viene associato un nome che può essere richiamato in diversi parti del file.
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.[20]

[16] SPARQL Protocol and RDF Query Language
[17] rdf:`http://www.w3.org/TR/rdf-schema/`
[18] Interrogazione
[19] Terse RDF Triple Language
[20] La pubblicazione si trova al sito https://www.w3.org/TR/turtle/

4.2 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.

Come in SQL, anche in SPARQL esiste il comando **FROM**, contenente un indirizzo IRI, che identifica tutto il set di dati presenti, dove l'interrogazione dovrà essere svolta; è spesso accompagnato dal comando NAMED usato per specificare più indirizzi IRI. 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.

4.3 Libreria javascript per interrogazioni SPARQL

Per fare un interrogazione in SPARQL, JavaScript ha bisogno di una chiamata AJAX[21] e della codifica JSON[22]. 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
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:

```
<body>
<div id='map'>
<script type="text/javascript">
//Created SPARQL query
function createSparqlQuery(endpoint,query,map,callback){
 var querypart = "query=" + escape(query);
 // Get our HTTP request object.
 var xmlhttp = getHTTPObject(); //Called AJAX
 //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); //JSON code
         }else
        // Error
        alert("Error: Status: "+ xmlhttp.status + "Response: "
        + xmlhttp.responseText);
        }
 };
 // Send the query to the endpoint.
 xmlhttp.send(querypart);
}
</script>
</div>
</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.

[21] Vedi Capitolo 5.2
[22] Vedi Capitolo 5.3

5 Ontologie per la rappresentazione dei servizi pubblici

L'Agenzia per l'Italia Digitale (AgID[23]) 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**. Tale metodo consiste nello strutturare i dati in modo da poter essere interconnessi e diventare più utili attraverso delle interrogazioni, fatte per esempio dallo 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**.[24]. 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.

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 scompone 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:

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 degli stati di XMLHttpRequest.

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

Tali vocabolari servono per organizzare al meglio la struttura organizzativa di aziende, società e comuni. Nel progetto è stato utilizzato un vocabolario creato dal **Dott.Cristiano Longo** che fa riferimento allo macro struttura del Comune di Catania. Tale vocabolario si trova al seguente sito `http://dydra.com/cristianolongo/comune-di-catania/sparql`.
Tramite questo endpoint, si fa una specifica interrogazione in SPARQL, attraverso una chiamata AJAX. I dati ottenuti sono codificati tramite la codifica JSON.

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 S.O.[25]. 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 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 l'albero N-ario.[26] Un albero N-ario ha i due elementi fondamentali che sono la **radice**,[27] che ha n nodi che vengono chiamati figli e le **foglie**,[28] che sono i nodi senza figli. Tutti gli altri nodi vengono chiamati nodi interni[29]. 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,[30], 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**, struttura dati complessa che contiene tre variabili che sono:
  - **uri**: Indirizzo IRI/URI di un valore;
  - **name**: Nome di un valore;
  - **homepage**: Indirizzi IRI/URI 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.

[25] Sistemi Operativi.
[26] Albero con n figli
[27] Root.
[28] Leaf.
[29] Fonte Wikipedia
[30] Struttura dati complessa composto da celle di lunghezza n.

- Classe **Storage**, che è un contenitore di tutti i nodi, compresa la radice con i seguenti metodi:
  - **add(Description)**: istanza classe Tree(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()**, che crea un lista utilizzando i tag:
  - definisce una lista non ordinata;
  - creare la liste non ordinate, è sempre associato al tag <ul>.
- **createMenu()**, che 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[31] è un formato convenzionale che serve per lo scambio di dati fra client e server. Ha la stessa nomenclatura dell'XML, ma a differenza di quest'ultimo, è 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.[32]. 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"},
{"secondUri": "www.example2.com","secondValue": "Example2",
"secondComment": "Second example page"}]

[31] JAVASCRIPT Object Notatio
[32] Vedi Paragrafo 5.3

5.3 Chiamata AJAX

Il metodo AJAX[33] 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**[34] 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:

```
function getHTTPObject(){
var xmlhttp;
if(!xmlhttp && typeof XMLHttpRequest != 'undefined'){
 try{
  // Code for old browser
  xmlhttp=new ActiveXObject('Msxml2.XMLHTTP');
 }
 catch(err){
  try{
   // Code for IE6, IE5
   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
  catch(err2){
   try{
         // Code for IE7+, Firefox, Chrome, Opera, Safari
         xmlhttp=new XMLHttpRequest();
   }
   catch(err3){
    xmlhttp=false
   }
  }
 }
}
 return xmlhttp;
}
```

Ogni Browser web ha la sua richiesta, nel caso di Browser datati, come IE,[35] l'oggetto XMLHttpRequest, viene restituito da **ActiveXObject**, mentre i moderni browser tale oggetto è supportato nativamente.

[33] Asynchronous JAVASCRIPT and XML
[34] Indica una componente che accede ai servizi o alle risorse di un'altra componente detta server. Fonte Wikipedia
[35] 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 posso 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 headers,[36] dato dal comando:

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

La variabile **onreadystatechange** chiama la funzione ogni volta che il valore ottenuto dal metodo readyState cambia. 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.

[36] serie di coppie chiave/valore specifici per uno scambio dati via interne

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;
- ...

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 che poi, successivamente, si applicherà la codifica JSON. Tuttavia però tale chiamata non sempre va a buon fine. I browser moderni hanno una sorta di restrizione[37], 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**, che verrà affrontata nel Capitolo 6.4.

[37] Same-domain-policy

6 Presentazione e codice dell'applicazione

6.1 Programmi utilizzati

6.2 HTML

6.3 Css

6.4 Il problema delle richieste Cross-Domain