In the Footsteps: Trentino's Famous People

Background tecnico e tecnologico

L'applicazione oggetto della relazione vuole rendere evidente il vantaggio ottenibile dalla fruizione di Linked Open Data (LOD) e pertanto sfrutta tutte le tecnologie che ruotano attorno a questo nuovo modo di organizzare dati e accedervi. Nello specifico, grazie alla piattaforma Fusepool P3, ci è stata data la possibilità di accedere a diversi datasets del Trentino: "Historical Characters", "Points of Interest", "Architectural and Artistic Heritage" e "Restaurants (Osterie Tipiche)". Il punto di partenza, quello su cui si focalizza l'intera applicazione, è rappresentato proprio dagli Historical Characters, ciascuno descritto dai seguenti termini : name, jobTitle, birthDate, deathDate, birthPlace, deathPlace entity-reference e description.

Mediante una query SPARQL all'endpoint di Fusepool, che ne consente il recupero all'indirizzo *http://sandbox.fusepool.info:8181/sparql/select*, l'applicazione mostra all'utente una lista completa dei personaggi storici del Trentino, in ordine alfabetico, con associato un simbolo che ne indica la categoria (ad esempio: artista, poeta, vescovo, nobile ecc.).

Per ottimizzare il recupero di queste risorse, come le altre che vedremo più avanti, è stato implementato un server ReST.

<<E' noto come la presenza di server in applicativi di questo tipo sia dettata non solo dagli effettivi miglioramenti che ne derivano in termini di elaborazione, velocità di risposta, scalabilità...ma, se vogliamo, soprattutto da un concreto fine funzionale. In assenza di un server , infatti, il dispositivo mobile non sarebbe in grado di supportare un eccessivo quantitativo di dati, non sarebbe sufficientemente interattivo e l'applicativo perderebbe del tutto senso>>.

Entrando nel dettaglio del sistema proposto, si tratta di un server Apache Tomcat, implementato in Java tramite libreria Jersey, che è richiamato tramite l'url *http://server-footsteps.rhcloud.com/ProvaTomcat-1.0-SNAPSHOT/rest/* e che ha l'obiettivo di inoltrare queries a Fusepool e restituire i dati in formato JSON, tutto mediante richieste HTTP.
Ciò che in sostanza avviene, ad esempio nel caso della lista dei personaggi storici, è la richiesta da parte dell'applicativo dello specifico servizio del server così programmato (url *http://server-footsteps.rhcloud.com/ProvaTomcat-1.0-SNAPSHOT/rest/listapersonaggi)* per ottenere name e jobTitle, ove presenti. Quest'ultimo , poi, inoltra a Fusepool dinamicamente, o meglio in real-time, la seguente query SPARQL:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

PREFIX schema: <http://schema.org/>

PREFIX dbo: <http://www.dbpedia.org/ontology/>

PREFIX fam: <http://vocab.fusepool.info/fam#>


SELECT DISTINCT  ?person  ?name  ?job
```

```
FROM<http://sandbox.fusepool.info:8181/ldp/historical-
characters/personaggi_storici_trentino-refine-csv-csv-transformed>

WHERE {

  ?person a schema:Person ;

          schema:name ?name ;

         schema:jobTitle ?job;

          fam:entity-reference ?ref.

}

ORDER BY ?name
```

Come si può osservare, la query restituisce ?person che rappresenta l'uri associato a ogni entità del dataset in questione (un identificativo unico che è utile per la visualizzazione delle informazioni di un singolo elemento), name che rappresenta la stringa del nome del personaggio e jobTitle che indica la categoria di appartenenza. E' stato indispensabile inserire nella query la variabile ?ref che è associata agli entity references, ovvero uri di tutte le entità principali contenute nella descrizione del personaggio; questa scelta è dovuta al fatto che s'intende prelevare solo dati di personaggi che abbiano una descrizione associata (elaborazione che avviene lato server una volta che questo abbia già interrogato Fusepool).

Output esemplificativo della query sulla lista dei personaggi:

?person	**?name**	**?job**
http://www.trentinocultura.net/asp_cat/main.asp?IDProspettiva=19&TipoVista=Scheda&IdObj=50552&Pag=3&IdSel=1	Abbondanzio	Vescovo di Trento
...	...	...

Allo stesso modo avviene la richiesta del servizio di visualizzazione del singolo elemento, ovvero quando l'utente clicca sul personaggio d'interesse con l'intento di ricavarne dettagli; in questo caso, però, l'url del servizio del server associato è *http://server-footsteps.rhcloud.com/ProvaTomcat-1.0-SNAPSHOT/rest/InfoPersonaggio/<URI-parziale>* mentre la query SPARQL diviene:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX schema: <http://schema.org/>
PREFIX dbo: <http://www.dbpedia.org/ontology/>
PREFIX fam: <http://vocab.fusepool.info/fam#>
SELECT ?name ?title ?birthPlace ?deathPlace ?sameAs ?description
FROM <http://sandbox.fusepool.info:8181/ldp/historical-
characters/personaggi_storici_trentino-refine-csv-csv-transformed>
WHERE {
    <URI> schema:name ?name.
    OPTIONAL { <URI> schema:jobTitle ?title }
```

```
  OPTIONAL { <URI> schema:birthPlace ?birthPlace }
  OPTIONAL { <URI> schema:deathPlace ?deathPlace }
  OPTIONAL { <URI> schema:sameAs ?sameAs }
  OPTIONAL { <URI> schema:description ?description }
}
```

dove <URI> indica l'uri completo dell'entità che nella query precedente è stato richiesto tramite la variabile ?person; quindi, nel caso del Vescovo Abbondanzio, il servizio richiesto sarà associato all'url:*http://server-footsteps.rhcloud.com/ProvaTomcat-1.0-SNAPHOT/rest/InfoPersonaggio/IDProspettiva=19&TipoVista=Scheda&IdObj=50552&Pag=3&IdSel=1*
in cui l'ultima parte rappresenta un uri parziale, mentre l'uri completo che è indicato nella query sarà:
http://www.trentinocultura.net/asp_cat/main.asp?IDProspettiva=19&TipoVista=Scheda&IdObj=50552&Pag=3&IdSel=1. Come si può notare, tale query restituisce ?name ?title ?birthPlace ?deathPlace ?sameAs ?description che rappresentano rispettivamente il nome del personaggio scelto, l'equivalente di job ovvero il "titolo" a egli associato, luogo di nascita, luogo di decesso, l'uri di DBpedia del personaggio e infine la sua descrizione, con la possibilità di recuperare anche gli attributi con valore inesistente (motivo per cui la query è caratterizzata da una serie di Optional). Ciò che è mostrato all'utente tuttavia fa a meno della variabile sameAs data la sua relativa utilità informativa ai fini di quest'ultimo. Di fatto, quest'ultima variabile potrebbe essere del tutto eliminata dalla query, ma è stata lasciata per ragioni di completezza.

Output esemplificativo della query sul singolo personaggio:

?name	**?title**	**?birthPplace**	**?deathPlace**	**sameAs**	**?description**
Abbondanzio	Vescovo di Trento				Abbondanzio fu il secondo vescovo della Chiesa di Trento, successore di Giovino e diretto predecessore del più celebre San Vigilio

Dopo aver ricavato queste informazioni e averle mostrate all'utente, prima di procedere in avanti con il flusso logico dell'applicativo, che prevede la visualizzazione della mappa con i marker posizionati sui luoghi correlati al personaggio, da un punto di vista del tutto trasparente all'utente, ecco cosa accade: dal momento in cui l'utente desidera "navigare" all'interno della mappa del Trentino sulle "orme" del personaggio storico selezionato, il sistema entra in contatto con ulteriori "datasets" il primo dei quali è DBpedia Italia, il più grande Database di risorse semantiche che si è sviluppato a partire dalle informazioni di Wikipedia. Per chiarire il motivo per il quale quest'ultimo sia stato inserito all'interno del progetto, è opportuno fare una piccolo passo indietro.

Com'è stato già detto precedentemente, ogni personaggio possiede, tra i tanti parametri caratterizzanti, una descrizione; il dataset di Fusepool, a ogni descrizione, associa un campo entity-

references con gli uri delle entità principali contenute all'interno, ad esempio: Vescovo, Trento, ecc...

Di tutte queste entità, il nostro sistema è "interessato" a estrapolare solo ed esclusivamente i luoghi appartenenti alla regione del Trentino (infatti non è detto che un personaggio storico, seppur significativo per il Trentino, non abbia vissuto e/o operato altrove).

Per realizzare opportunamente un itinerario specifico per la regione, il sistema va a ricavare dagli entity-references un vettore di oggetti JSON (le entità nella descrizione), tramite una query al dataset di DBpedia Italia ottiene una lista di luoghi della regione (città, province, composti da una etichetta, un uri, latitudine e longitudine di valore decimale) e, infine, visualizza all'utente le sole entità del vettore che rientrano in questo insieme di luoghi. Di seguito è riportata la query SPARQL al dataset di DBpedia:

```
SELECT DISTINCT ?place
WHERE{

        { ?place <http://dbpedia.org/ontology/wikiPageWikiLink>
                <http://it.dbpedia.org/resource/Provincia_di_Trento>;
                 http://airpedia.org/typeWithConfidence#1>
                        <http://dbpedia.org/ontology/Place> .
        }


        UNION
        { ?place <http://dbpedia.org/ontology/administrativeDistrict>
                        <http://it.dbpedia.org/resource/Trentino-Alto_Adige> . }
        UNION
        { ?place <http://dbpedia.org/ontology/administrativeDistrict>
                        <http://it.dbpedia.org/resource/Trento> . }
        UNION
        { ?place <http://dbpedia.org/ontology/administrativeDistrict>
                        <http://it.dbpedia.org/resource/Bolzano> . }

        FILTER (!CONTAINS('Provincia',?place))
}
```

Il sistema sta richiedendo, in questo modo, l'insieme di tutti i luoghi che sono Provincia di Trento unito all'insieme di tutti i luoghi che appartengono al distretto amministrativo Trentino-Alto-Adige, unito all'insieme del distretto Trento, unito, a sua volta, al distretto Bolzano, escludendo dal risultato tutte le entità che contengono 'Provincia'; quest'ultima operazione è inserita solo per evitare che, come nel caso del Vescovo di Trento Abbondanzio, si facciano visualizzare due marker: quello effettivo di Trento e quello di 'Provincia di Trento' che risulterebbe ridondante.

Ora, anche se place è un oggetto composto da label, uri, latitudine e longitudine, all'utente sono mostrati solo i marker, o il singolo nel caso, con la rispettiva label. Le altre informazioni sono utilizzate per posizionare questi o quest'ultimo; ma a questo scopo è necessaria una ulteriore query SPARQL a DBpedia:

```
PREFIX dbpprop-it:<http://it.dbpedia.org/property/>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
SELECT ?latG ?latM ?latS ?longG ?longM ?longS ?label
WHERE{
        <uri> dbpprop-it:latitudineGradi ?latG.
        <uri> dbpprop-it:latitudineMinuti ?latM.
        <uri> dbpprop-it:latitudineSecondi ?latS.
        <uri> dbpprop-it:longitudineGradi ?longG.
        <uri> dbpprop-it:longitudineMinuti ?longM.
        <uri> dbpprop-it:longitudineSecondi ?longS.
        <uri> rdfs:label ?label.
}
```

Cosicché, per ogni luogo (?label), si ottengono grado, minuto primo e minuto secondo rispettivamente per la latitudine e per la longitudine.

Ora che l'utente può visualizzare la mappa con le entità geografiche d'interesse può davvero "navigare" e scovare Points of Interest (POI), Ristoranti/osterie tipiche e Hotels. Anche per questi tre datasets, com'è facile intuire, sono necessarie tre query tra server e Fusepool che sono riportate di seguito.

La ricerca dei punti d'interesse avviene con la chiamata di servizio all'url:
http://server-footsteps.rhcloud.com/ProvaTomcat-1.0-SNAPHOT/rest/RicercaLuoghiInteresse/

che avvia l'esecuzione della query SPARQL che segue:

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT DISTINCT ?label ?lat ?long ?description
FROM <http://sandbox.fusepool.info:8181/ldp/trentino-architectural-cultural-
heritage-enriched-ttl>
WHERE {
        ?building a schema:TouristAttraction;
           rdfs:label ?label;
           dct:description ?description;
           geo:lat ?lat ;
           geo:long ?long .
    FILTER (?lat >= \""<latitudine-prec>\"^^xsd:double && ?lat <=
                        \""<latitudine+prec>\"^^xsd:double && ?long >=
                \"<longitudine-                    prec>\"^^xsd:double &&
                ?long <=        \""<longitudine+prec> ^^xsd:double)
    FILTER (!sameTerm(\"CASA\", ?description))
 }
```

Ogni qualvolta si richiede il servizio sono passati tre parametri: latitudine, longitudine e precisione(prec) relativi alla regione di mappa correntemente visualizzata (quindi, per avere una idea di quella che sarebbe la chiamata completa al servizio, l'url sarà : http://server-

footsteps.rhcloud.com/ProvaTomcat-1.0-SNAPHOT/rest/RicercaLuoghiInteresse/?lat=45.88333333333333&long=11.034166666666666&prec=0.01); questo avviene in quanto è stato pensato di far visualizzare i marker associati ai POI per "range", dove per range s'intende ogni possibile riquadro della mappa. Quindi, spostandosi all'interno della cartina geografica, l'utente dovrà di volta in volta ricliccare sul bottone dei luoghi d'interesse e questi saranno nuovamente posizionati mediante l'estrapolazione di latitudine, longitudine e precisione, opportunamente ricalcolati come mostra la query. Si fa notare che, poiché i datasets dei POIs del Trentino risultano molto ricchi e completi, è stato necessario introdurre nella query una operazione di filtro, onde evitare di mostrare all'utente entità, come le abitazioni/case, che, come intuibile, non sarebbero poi così d'interesse.

Similmente, per consentire la visualizzazione degli Hotels, cliccando sull'apposito bottone l'utente può effettuare la chiamata di servizio che eseguirà la seguente query:

```
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX schema: <http://schema.org/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?h ?name ?category ?lat ?long ?comment ?phone ?loc ?via ?sito
?mail
FROM <http://sandbox.fusepool.info:8181/ldp/trentino-point-of-interest-ttl>
WHERE {
    VALUES ?category
                {
               "Bed & Breakfast"
                "Hotel"
              }
    ?h schema:category ?category;
      rdfs:label  ?name;
      geo:lat ?lat;
      geo:long ?long.
    OPTIONAL{ ?h rdfs:comment ?comment.
    FILTER LANGMATCHES(LANG(?comment),  \"IT\")}
    OPTIONAL { ?h  foaf:homepage ?sito. }
    OPTIONAL { ?h  foaf:mbox ?mail. }
    OPTIONAL { ?h foaf:phone ?phone. }
    OPTIONAL { ?h  schema:streetAddress ?address .
           ?address  schema:addressLocality ?loc;\n
                 schema:streetAddress ?via.
          }
  FILTER (?lat >= \""<latitudine-prec>\"^^xsd:double && ?lat <=
                        \""<latitudine+prec>\"^^xsd:double && ?long >=
              \"<longitudine-                          prec>\"^^xsd:double &&
              ?long <=          \""<longitudine+prec> ^^xsd:double)

}
```

Ha senso sottolineare in merito a quest'ultima query sugli hotels che, in primo luogo, non è stato possibile inserire i commenti in italiano (?comment) per problemi dovuti al fatto che le stringhe vi-

sualizzate producevano simboli che rendevano il testo poco chiaro se non del tutto incomprensibile, e, in secondo luogo, è stato apprezzato da parte del team di sviluppo la possibilità che Fusepool offre, rispetto a DBpedia ad esempio, nell'inserire all'interno delle query il campo values, che, in questo specifico caso, serve per definire una variabile ?category con due possibili valori: Bed & Breakfast e Hotel. Quanto all'output, esso consiste in una lista di attributi quali: ?h ?name ?category ?lat ?long ?comment ?phone ?loc ?via ?sito ?mail di cui, ove presenti, sono mostrati all'utente solo ?name, ovvero il nome del "luogo di soggiorno", ?category, la categoria di appartenenza tra le due possibili sopracitate, ?phone, il contatto telefonico, ?loc e ?via, l'indirizzo, ?sito, il sito internet, e infine ?mail, l'indirizzo mail.

Giunti alla fine, si riporta la query SPARQL relativa all'ultimo dataset: ristoranti e osterie.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
SELECT ?name ?locality ?street ?locality ?lat ?long
FROM <http://sandbox.fusepool.info:8181/ldp/trentino-restaurants-1/osterie-
csv-csv-transformed-1>
WHERE {
        ?restaurant a schema:Restaurant ;
           schema:name ?name ;
           schema:address ?address ;
           schema:geo ?geo .
        ?geo a schema:GeoCoordinates ;
                  schema:latitude ?lat ;
                  schema:longitude ?long .

        ?address a schema:PostalAddress ;
        schema:streetAddress ?street ;
         schema:addressLocality ?locality .
        FILTER (?lat >= \"<latitudine-prec>"^^xsd:double && ?lat <=
        \""<latitudine+prec>\"^^xsd:double && ?long >= \<longitudine-
        prec>\"^^xsd:double &&          ?long <=
\"<longitudine+prec>\"^^xsd:double)
}

ORDER BY ?lat
```

Solo in quest'ultimo caso è stato necessario ricorrere, per fornire maggiori dettagli all'utente, che vuole ricavare informazioni sui ristoranti limitrofi la zona d'interesse, al secondo dataset esterno che prima si menzionava, ovvero quello di Yelp. Si tratta di un applicazione mobile che, come Trip Advisor, ma forse meno noto, raccoglie e fornisce informazioni su pub, ristoranti, pizzerie e quant'altro, indicando agli utenti descrizioni, contatti e valutazioni e dando la possibilità di leggere e di scrivere commenti e recensioni. In particolare il sistema proposto interagisce con Yelp tramite le API che quest'ultimo fornisce per recuperare la valutazione (rating), una breve descrizione (snippet_text), una immagine o fotografia (image_url) e il contatto telefonico per ogni ristorante (phone).

Possibili sviluppi futuri

Il server, per ragioni dettate dall'efficienza, in termini di reperibilità delle informazioni e di velocità di risposta, dovrebbe esser munito di un meccanismo di caching mediante ad esempio MongoDB, o altri, tramite il quale gli oggetti JSON restituiti dal medesimo possano essere memorizzati per un periodo di N giorni, intervallo limite oltre il quale il server dovrebbe eliminare i dati in locale. Tuttavia è indispensabile evidenziare che, con il crescere della scalabilità dell'applicazione e, nello specifico, del numero di utenti che ne fanno uso, il server potrebbe dover essere riconfigurato con nuovi parametri; potrebbe inoltre essere necessario l'intervento di più server o comunque di macchine con capacità più elevate rispetto all'attuale di cui si usufruisce per il prototipo dell'applicativo: tutte considerazioni che rientrano nell'ottica di possibili sviluppi futuri.

Background Mobile

Com’è già stato fatto notare, l'applicazione ruota intorno al lavoro del server, il quale ne diviene elemento centrale per il corretto funzionamento. Di fatto, sia la versione per Android sia per iOS, di cui forniamo le versioni sugli appositi stores, consistono in elementi d’interfaccia con l'utente per la visualizzazione e la fruizione interattiva con i dati del server. Ci limitiamo quindi a fornire i seguenti dettagli tecnici:

1) le librerie utilizzate ovvero: Alamofire, SwiftyJSON, Urk (libreria proprietaria).

2) I linguaggi di programmazione adottati sono Swift per iOS e Java per Android.

3) l'ambiente di sviluppo (l'IDE) è Xcode per iOS, Android Studio per Android.