

Relazione Progetto ModSem

Salvatore Latino, Pietro Francesco Pettinato

Anno 2021

salvatore.latino@edu.unito.it
pietro.pettinato@edu.unito.it

Contents

1	Motivazioni	3
2	Requirements	4
3	Descrizione del dominio	5
4	Documentazione sul dominio	6
4.1	Curriculum Vitae	6
4.2	Settori	8
4.3	Descrizione di alcuni individui	9
4.4	Allineamento con altre ontologie	10
4.5	Pattern utilizzati	11
4.6	Relazioni inverse, transitive e funzionali	11
4.7	Property chain	12
5	Visualizzazione dell'ontologia	13
5.1	Tassonomia delle classi	13
5.2	Visualizzazione grafo	13
6	Queries SPARQL	16
6.1	Query per le Aziende	17
6.1.1	Query 1	17
6.1.2	Query 2	18
6.1.3	Query 3	19
6.1.4	Query Federata Azienda - verso endpoint esterno	20
6.2	Query per i Candidati	24
6.2.1	Query 1	24
6.2.2	Query 2	24
6.2.3	Query 3	25
6.2.4	Query 4	26
6.2.5	Query 5	27
6.2.6	Query Federata Candidato - verso endpoint esterno	28
6.3	Query federata: inserimento nel grafo	29
7	Estensione - Applicazione Client	34
8	Estensione - Regole SWRL	35

1 Motivazioni

"L'Italia è una Repubblica democratica, fondata sul lavoro" purtroppo però, al giorno d'oggi molte persone non hanno impiego e soprattutto a causa della pandemia, la situazione si è ancora più aggravata.

Ci sono aziende che però sono costantemente alla ricerca di personale ed hanno difficoltà ad assumerne, spesso perché non riescono a mettersi in contatto con i candidati oppure perché non riescono a trovare una figura adatta a ciò che stanno cercando.

D'altra parte, chi è alla ricerca di lavoro ha difficoltà a trovare offerte adatte, trovandosi spesso in situazioni in cui occorre cambiare città o accettare un lavoro che non lo soddisfa perché non si addice ai suoi studi.

Con questo progetto d'esame, si vuole offrire la possibilità:

- alle aziende, di esporre in modo chiaro e preciso i requisiti di cui sono alla ricerca, offrendo la possibilità di selezionare i candidati sulla base dei dati esposti nel curriculum vitae
- ai candidati, di filtrare fra tutte le possibili offerte quelle più adatte alle loro competenze e più comode dal punto di vista del luogo di lavoro, oltre che a rendere disponibile il loro CV per attirare l'attenzione delle aziende

2 Requirements

L'obiettivo è quello di collegare le aziende con chi è in cerca di lavoro. Attraverso l'utilizzo dell'ontologia vengono offerti visibilità e collegamenti automatizzati sulla base di opportune caratteristiche.

Gli utenti che utilizzano la piattaforma sono di due categorie, *aziende* e *candidati*. Entrambi possono usufruire di vari servizi a seconda del contesto in cui si trovano.

Per le aziende (che vogliono assumere nuovi lavoratori) viene offerta la possibilità di:

- visualizzare la lista dei candidati che soddisfano i requisiti specificati
- pubblicare annunci con l'offerta di lavoro e selezionare i candidati più adatti fra quelli che vi aderiscono

Per i candidati (alla ricerca di lavoro) viene offerta la possibilità di:

- inserire il proprio CV in modo da essere selezionati dalle aziende
- cercare fra gli annunci pubblicati quello migliore in base alle proprie competenze

Il sistema consiglia anche annunci in base ad alcune caratteristiche comuni fra la proposta ed il candidato.

3 Descrizione del dominio

Il dominio che si è scelto di modellare è quello del lavoro, in particolare la parte relativa alla domanda e all'offerta. Occorre avere alcune informazioni per poter utilizzare il reasoner ed avere inferenze automatiche sui dati.

Per le **aziende** viene richiesta una sua *breve descrizione*, il *settore* in cui svolgono la loro attività lavorativa, i *paesi* in cui hanno sedi ed il loro numero. È risultato molto utile organizzare i settori seguendo il *sistema dei profili professionali ISTAT*¹.

Per i **candidati** è invece fondamentale l'inserimento del *curriculum vitae*, dal quale possono poi venire estratte le varie informazioni. Fra le più importanti abbiamo:

- le *informazioni personali* (nome e cognome, residenza, data di nascita...)
- le *informazioni sui titoli di studio* (laurea triennale, dottorato, certificati linguistici...)
- le *competenze possedute* (computer, artistica...)
- le *attività principali* svolte nel corso degli anni

Per organizzare i vari campi del CV con le loro relative informazioni, si è preso spunto dall'organizzazione dei curriculum *Europass*².

Il punto d'incontro tra Azienda e Candidato è rappresentato dall'**Annuncio**, pubblicato dall'azienda. Un annuncio è composto da:

- una *descrizione* dell'attività lavorativa
- una voce che indica l'*azienda pubblicante*
- le *competenze generali* richieste (computer, tecniche, artistiche ...)
- il tipo di *contratto proposto* (determinato, indeterminato ...)
- le *ore di lavoro giornaliere*
- il *salario mensile*
- i *benefits*
- le *conoscenze richieste* nel dettaglio (linguaggio Java...)
- il *paese* in cui si svolge il lavoro

¹<http://www.sistemapiemonte.it/vetrinaweb/secure/HomePage.do>

²<https://europa.eu/europass/it>

4 Documentazione sul dominio

4.1 Curriculum Vitae

Come detto in precedenza, i dati relativi al candidato sono stati organizzati seguendo il modello di CV Europass, disponibile al seguente link: <https://europa.eu/europass/it>.

Di seguito è riportato un esempio del documento.

The image displays two examples of the Europass Curriculum Vitae form. The left form is for Mattia Loy, an HR consultant, and the right form is for a user named S. Schiavini. Both forms show sections for personal information, education, and competencies.

Left Form (Mattia Loy):

- INFORMAZIONI PERSONALI:** Mattia Loy, Via ... n. 1, 00100, Roma, Italia, Telefono: +39 06 32000000, Email: europass@europa.eu, LinkedIn: https://www.linkedin.com/company/...
- ESPERIENZA PROFESSIONALE:** 2019 - in corso, HR consultant, Libero professionista indipendente, Roma (Italia).
 - Specialista Risorse Umane (attuale/desiderata)
 - Master in Risorse Umane
 - Supportare come HR consultant le PMI che fanno grande Italia tramite servizi d'eccellenza
- ISTRUZIONE E FORMAZIONE:** 2010 - in corso, Abilitazione alla professione di psicologo, Ordine nazionale degli psicologi - ONOP.
 - Abilitazione alla professione e dei servizi esclusivi ad essa associati, come il counselling psicologico e la somministrazione di test psicometrici
- COMPETENZE PERSONALI:** Lingue madre: Italiano. Altre lingue: Inglese (C1), Francese (C1), Spagnolo (C1).

Right Form (S. Schiavini):

- Competenze organizzative e gestionali:** Leadership (fino a 200 risorse), project management, organizzazione eventi, gestione del tempo, progettazione, diagrammi di Gantt, diagrammi di PERT.
- Competenze professionali:** Ricerca e selezione del personale, employer branding, metodo STAR, somministrazione di test psicometrici Luvina Learning 8, assessment centre (etc.).
- Competenze digitali:** Elaborazione delle informazioni, Comunicazione, Creazione di contenuti, Sicurezza, Risoluzione di problemi. Livelli: Utente avanzato, Utente avanzato, Utente avanzato, Utente avanzato, Utente avanzato.
- Altre competenze:** Software SPSS per l'analisi statistica in psicometria, Università di Cagliari, utente base.
- Altre competenze:** Scrittura professionale di CV.
- ULTERIORI INFORMAZIONI:** Pubblicazioni, ricerche scientifiche ed articoli.
 - Curriculum Vitae: 45 articoli su come scrivere il CV ed utilizzato per trovare lavoro.
 - Soddisfazione lavorativa e valori: lavoratori precari e a tempo indeterminato a confronto (modello di S. Schiavini) 2010.
 - Model di business a confronto: produzione snella e just in time di TOYOTA vs FIAT e industrializzazione di massa 2007.
 - Competenza ed occupabilità nel 2014: ricerca empirica sul futuro del mercato del lavoro in Italia e l'occupazione 2014.
 - Psicologia delle vendite: il Panoramico, costruito psicologico che influenza i processi decisionali degli acquirenti 2011.
- ALLEGATI:** Mantenere questa parte solo in caso di invio per concorso pubblico o cartaceo.
 - Copie delle lauree e qualifiche conseguite
 - Attestazione del datore di lavoro
 - Lettera di presentazione
- Dati personali:** Autorizzo il trattamento dei miei dati personali ai sensi del Decreto Legislativo 30 giugno 2003, n. 159 "Codice in materia di protezione dei dati personali" e GDPR.

Figure 1: Un esempio di CV Europass con le sezioni utilizzate all'interno dell'ontologia

Il documento sopra mostrato è stato interpretato nella nostra ontologia tramite la divisione in più sotto campi, appartenenti ad un unico curriculum. Tale suddivisione può essere osservata graficamente attraverso l'immagine alla pagina successiva (7).

Come si può notare per ogni Curriculum vitae (CV) si ha un gruppo di campi (campi-CV). Ogni gruppo contiene più sezioni (haCampo), ognuna delle quali rispecchia una parte del curriculum.

Nella fig.2 vediamo nel dettaglio la sezione "Informazioni personali".

In questo caso prendiamo l'individuo Jeff.Bezos, che ha inserito i dati del suo CV (CV-Jeff.Bezos). Se vogliamo vedere le sue informazioni personali possiamo andare nella rispettiva sezione (infoPersonali-Jeff.Bezos) e visualizzare i suoi dati (nome, cognome, nazionalità...).

Questi dati vengono utilizzati per effettuare collegamenti fra candidato e azienda. Un esempio di un possibile collegamento è quello relativo al settore di interesse fra i titoli di studio del candidato e la posizione offerta dall'azienda.

Description: CV_Jeff_Bezos
Types +
cv
Same Individual As +
Different Individuals +

Property assertions: CV_Jeff_Bezos
Object property assertions +
contiene campi_CV_Jeff_Bezos
CV_è_di Jeff_Bezos
Data property assertions +
Negative object property assertions +
Negative data property assertions +

Description: campi_CV_Jeff_Bezos
Types +
Campi_CV
Same Individual As +
Different Individuals +

Property assertions: campi_CV_Jeff_Bezos
Object property assertions +
haCampo dottorato_Jeff_Bezos
haCampo attivitaPrincipali_Jeff_Bezos
haCampo competenze_Jeff_Bezos
haCampo laurea_specialistica_Jeff_Bezos
haCampo certificatiLingua_Jeff_Bezos
haCampo infoPersonali_Jeff_Bezos
haCampo funzioniOccupate_Jeff_Bezos
èContenutoIn CV_Jeff_Bezos
Data property assertions +
Negative object property assertions +
Negative data property assertions +

Description: infoPersonali_Jeff_Bezos
Types +
InfoPersonali
Same Individual As +
Different Individuals +

Property assertions: infoPersonali_Jeff_Bezos
Object property assertions +
Image foto_Jeff_Bezos
haPersonalmailbox email_Jeff_Bezos
phone (foaf:haPhone) Phone_Jeff_Bezos
depiction foto_Jeff_Bezos
Data property assertions +
firstName "Jeff"
professione "Direttore"^^xsd:string
viveln "New York"
gender "Maschio"
anniCarriera 22
surname "Bezos"
nazionalita "USA"^^xsd:string
dataNascita "1964-01-12"^^xsd:date
Negative object property assertions +

Figure 2: A-BOX del cv di un candidato nel quale viene mostrato l’inserimento delle informazioni personali

4.2 Settori

Riguardo i settori, le classi individuate provengono in parte dal sistema dei profili professionali ISTAT della regione Piemonte, disponibile al seguente link: <http://www.sistemapiemonte.it/vetrinaweb/secure/HomePage.do>.



Figure 3: Settori del sistema ISTAT, in rosso sono quelli ricreati nell'ontologia

Oltre ad i settori presi dal sistema ISTAT sono state create delle classi aggiuntive, per meglio rappresentare alcune aree lavorative, come ad esempio il settore Economico e quello Informatico, con le relative sottoclassi.

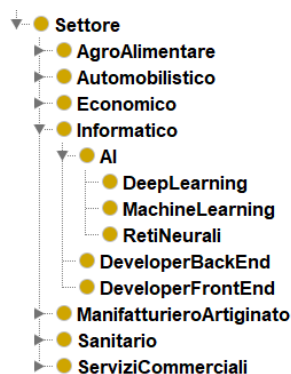


Figure 4: Settori dell'ontologia


Come si può notare il settore informatico si articola in un insieme di sottoclassi che meglio specificano l'ambito lavorativo.

4.3 Descrizione di alcuni individui

Ogni azienda deve inserire una sua descrizione insieme agli altri dati che la riguardano. Nello specifico le descrizioni aziendali inserite nell'ontologia sono state estrapolate da Wikipedia.

Amazon.com

Da Wikipedia, l'enciclopedia libera.
(Reindirizzamento da [Amazon](#))



Questa voce o sezione sull'argomento aziende è priva o carente di note e riferimenti bibliografici puntuali.

Sebbene vi siano una [bibliografia](#) e/o dei [collegamenti esterni](#), manca la contestualizzazione delle fonti con note a piè di pagina che indichino puntualmente la provenienza delle informazioni. Puoi [migliorare questa voce citando le fonti](#) più precisamente. Se [progetto di riferimento](#).

Amazon.com, Inc. è un'azienda di commercio elettronico statunitense, con sede a [Seattle](#) nello stato di [Washington](#). È la più grande *Internet company* al mondo^{[3][4][5]}. *Time Magazine* ha proclamato [Jeff Bezos](#), fondatore dell'azienda, [Uomo dell'anno](#) del 1999, a riconoscimento del successo di Amazon nel rendere popolare il commercio elettronico.

Figure 5: Descrizione dell'azienda su Wikipedia

La descrizione è stata copiata nell'apposita sezione dell'ontologia.

Description: azienda_Amazon

Types +

Azienda

Annuncio

Multinazionale

Same Individual As +

Different Individuals +

Property assertions: azienda_Amazon

haSedePrincipaleIn Italia

haSedePrincipaleIn USA

frapo:employs Jeff_Bezos

haLavoratore Jeff_Bezos

pubblica annuncio_Amazon

pubblica annuncio_Amazon_due

made annuncio_Amazon

made annuncio_Amazon_due

Data property assertions +

descrizioneAzienda "azienda di commercio elettronico statunitense, con sede a Seattle nello stato di Washington. È la più grande Internet company al mondo"

numeroSedi 35

Negative object property assertions +

Negative data property assertions +

Figure 6: Descrizione dell'azienda nell'ontologia

4.4 Allineamento con altre ontologie

Al fine di standardizzare il più possibile l'ontologia da noi implementata, la si è allineata con il vocabolario FOAF³ e con l'ontologia FRAP⁴, individuata su Linked Open Vocabularies (LOV)⁵.

L'allineamento è avvenuto principalmente attraverso l'utilizzo di RDF e OWL :

- **owl:EquivalentClass**: è una proprietà built-in che collega una descrizione di classe a un'altra descrizione di classe. Questo significa che le due descrizioni coinvolte contengono esattamente lo stesso insieme di individui, in altre parole stiamo dicendo che i due concetti sono correlati ma non uguali.

Nel nostro caso è stato utilizzato per allineare *strettamente* le nostre classi e le nostre proprietà con altre ontologie.

I class axioms **owl:equivalentClass** possono essere utilizzati anche per definire una classe enumerata. Pertanto la classe enumerata *Nazione* è stata realizzata attraverso l'utilizzo di *owl:equivalentClass*.

Utilizzando questo costrutto possiamo stabilire le condizioni necessarie e sufficienti per l'appartenenza ad una classe.

Da **notare** che l'uso di **owl:equivalentClass** non implica l'uguaglianza di classe (Class equality), la quale ci porterebbe a dire che le classi denotano lo stesso concetto. L'uguaglianza di classe reale può essere espressa solo con il costrutto **owl:sameAs**.

- **rdfs:SubClass Of**: le relazioni di sottoclasse forniscono le condizioni necessarie di appartenenza ad una classe, permettendo di costruire delle gerarchie.

rdfs:SubClass Of lo si è utilizzato per l'inclusione di un concetto in uno più generale (per esempio quando si collega un concetto nel proprio dominio a un concetto di un'ontologia top o mid-level).

Si è scelto di allineare le classi *Azienda* e *Candidato* con la classe **foaf:Agent** tramite una relazione di sottoclasse (**SubClassOf**), in quanto entrambe le classi sono un suo sottoinsieme.

FOAF FOAF è un progetto dedicato a collegare persone e informazioni utilizzando il web.

Dato che la classe **foaf:Agent** ha come sottoclassi **foaf:Person** e **foaf:Organization** si è scelto di allineare: *Candidato* con **rdfs:subClass Of Person** (in quanto un candidato non è altro che un sotto insieme del concetto più generale Persona) e *Azienda* tramite **Equivalent To Organization** (dato dal fatto che Organization ed Azienda sono equivalenti).

³<http://xmlns.com/foaf/spec/>

⁴<https://sparontologies.github.io/frapo/current/frapo.html>

⁵<https://lov.linkeddata.es/dataset/lov>

In maniera analoga a quanto fatto per *Candidato*, si è allineata la classe *CV* con la classe `foaf:Document` tramite `rdfs:subClass Of` (questo perché il CV non è altro che un particolare tipo di documento). Attraverso questi allineamenti è stato possibile utilizzare le Data properties e le Object properties di FOAF. Tra le Data properties utilizzate possiamo citare, *firstName*, *surname*, *gender*.... Tra le Object properties invece si è utilizzato *made* e *maker* per la realizzazione degli annunci da parte delle aziende.

FRAPO FRAPO è un'ontologia utilizzata per descrivere le informazioni amministrative del progetto di ricerca e per lavorare con CRIS (Current Research Information Systems). Può essere utilizzato per la caratterizzazione di domande di sovvenzione, enti di finanziamento, progetti, partner di progetto, ecc. Nel nostro caso FRAPO è stato utilizzato per allineare la nostra classe *Contratto* con la classe `frapo:Contract`.

Essendo FRAPO basato su FOAF è stato possibile utilizzare le Object property `frapo:isEmployedBy`, `frapo:employs`, `frapo:hasCurriculumVitae` allineate tramite `Equivalent To`, rispettivamente con *lavoraPer*, *haLavoratore*, *possiede*. Tra le Data properties, al fine di definire il paese in cui è presente l'offerta lavorativa, è stato utilizzato `frapo:hasCountry`.

4.5 Pattern utilizzati

Al fine di modellare schemi ricorrenti nella modellazione ontologia e data la necessità di dover associare un ruolo ai candidati è stato importato (attraverso il link di: Reusable OWL Building Block⁶) ed utilizzato il pattern *AgentRole*⁷. Tale pattern, quindi, permette di rappresentare gli agenti e i ruoli che giocano. In particolare, nell'ontologia realizzata, l'*Agent* corrisponde a *Funzioni Occupate*, mentre come *Role* vengono utilizzate le sue sottoclassi (*CEO*, *Direttore*, *Impiegato*...).

Inoltre la classe *Settore* può essere vista come un'implementazione del pattern *Collection* (senza l'utilizzo delle relative proprietà) in quanto ogni sottoclasse al suo interno viene intesa come un suo stesso membro.

4.6 Relazioni inverse, transitive e funzionali

Transitive Se una proprietà, *P*, è di tipo *transitive* allora per $\forall x, y, z$:

$$(xPy)(yPz) \rightarrow (xPz)$$

Tale proprietà è stata associata alle Object property: `haSettoreDinteresse` (la quale è una property chain) e a `contiene`.

⁶<http://www.ontologydesignpatterns.org/cp/owl/agentrole.owl>

⁷<http://ontologydesignpatterns.org/wiki/Submissions:AgentRole>

Funzionale Se una proprietà, P , è di tipo *functional* allora per $\forall x, y, z$:

$$(xPy)(yPz) \rightarrow (y = z)$$

Tale proprietà è stata associata alla Object property **haContratto** (in quanto un candidato prima di aderire ad una nuova proposta di lavoro può essere al massimo legato da un solo contratto) e a **haSettoreAzienda** (l'azienda può avere un solo settore in cui opera)

Inverse of Nell'ontologia sono presenti molte relazioni *inverse of*, come ad esempio per **pubblica**, **lavora per**, **employs**...

4.7 Property chain

Sono state realizzate 3 *property chain*:

- **lavoraNelSettore**: Tale property permette di dire che "se un Candidato lavora in un Azienda e quest'ultima opera in un determinato Settore, allora si può inferire che il Candidato lavora in tal Settore".

lavoraPer o **haSettoreAzienda** **subPropertyOf**: **lavoraNelSettore**

- **haSettoreAnnuncio**: In modo del tutto analogo alla precedente, tale property permette di affermare che "Se l'Azienda opera in un determinato Settore e l'Annuncio è pubblicato da tale Azienda, allora l'annuncio sarà nello stesso Settore in cui opera l'Azienda."

ePubblicato o **haSettoreAzienda** **subPropertyOf**:
haSettoreAnnuncio

- **haSettoreDinteresse**: Con tale property chain il settore d'interesse del candidato viene inferito sia dal settore di studi, sia dal settore in cui opera l'azienda per il quale esso lavora.

possiede o **contiene** o **haCampo** o **haSettoreStudi**
subPropertyOf: **haSettoreDiInteresse**

lavoraPer o **haSettoreAzienda** **subPropertyOf**: **lavoraNelSettore**

Da ciò si evince che un Candidato possa essere interessato a diversi annunci, anche con settore differenti tra di loro.

5 Visualizzazione dell'ontologia

5.1 Tassonomia delle classi

Di seguito viene mostrata la tassonomia delle classi, ordinate da sinistra verso destra.

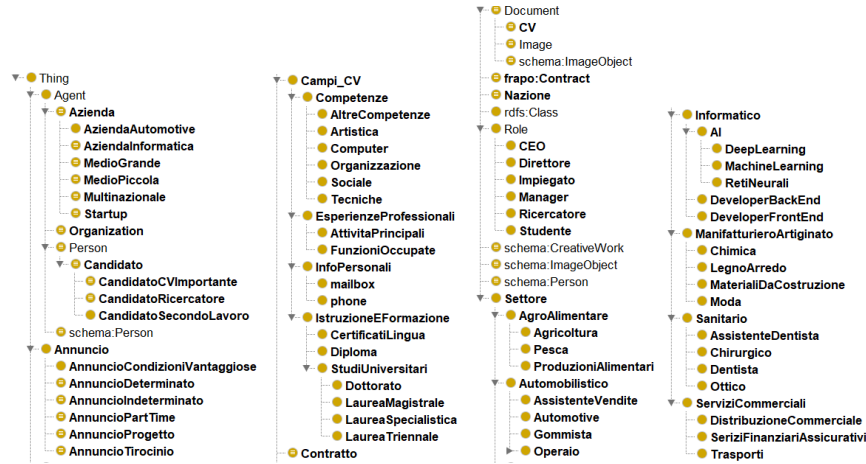


Figure 7: Tassonomia delle classi

5.2 Visualizzazione grafo

Per la visualizzazione grafica dell'intera ontologia (Tbox + Abox) è stato utilizzato il tool **OWLGrEd**. Di seguito viene indicato il link per la visualizzazione: http://owlgred.lumii.lv/online_visualization/e8s5

A-box Di seguito viene mostrato un esempio dettagliato di A-box tramite l'utilizzo di GraphDB.

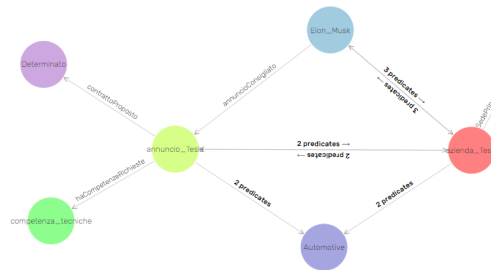


Figure 8: Vengono mostrate tutte le asserzioni dell'azienda Tesla e gli annunci ad essa correlati

Un CV è composto da una serie di campi che descrivono gli aspetti personali e professionali del candidato.

	subject	predicate	object	context
1	jso:campi_CV_Jeff_Bezos	jso:haCampo	jso:attivit�Principal�_Jeff_Bezos	http://modsem.com/graph/JSOProject
2	jso:campi_CV_Jeff_Bezos	jso:haCampo	jso:certificatiLingua_Jeff_Bezos	http://modsem.com/graph/JSOProject
3	jso:campi_CV_Jeff_Bezos	jso:haCampo	jso:competenze_Jeff_Bezos	http://modsem.com/graph/JSOProject
4	jso:campi_CV_Jeff_Bezos	jso:haCampo	jso:dottorato_Jeff_Bezos	http://modsem.com/graph/JSOProject
5	jso:campi_CV_Jeff_Bezos	jso:haCampo	jso:funzioniOccupate_Jeff_Bezos	http://modsem.com/graph/JSOProject
6	jso:campi_CV_Jeff_Bezos	jso:haCampo	jso:infoPersonali_Jeff_Bezos	http://modsem.com/graph/JSOProject
7	jso:campi_CV_Jeff_Bezos	jso:haCampo	jso:laurea_specialistica_Jeff_Bezos	http://modsem.com/graph/JSOProject
8	jso:campi_CV_Jeff_Bezos	jso:�ContenutoIn	jso:CV_Jeff_Bezos	http://modsem.com/graph/JSOProject
9	jso:campi_CV_Jeff_Bezos	rdf:type	jso:Campi_CV	http://modsem.com/graph/JSOProject
10	jso:campi_CV_Jeff_Bezos	rdf:type	owl:NamedIndividual	http://modsem.com/graph/JSOProject
11	jso:campi_CV_Jeff_Bezos	rdfs:comment	"campi del CV di Jeff_Bezos" ^{lit}	http://modsem.com/graph/JSOProject
12	jso:campi_CV_Jeff_Bezos	rdfs:label	"campi_CV_Jeff_Bezos" ^{lit}	http://modsem.com/graph/JSOProject

Figure 12: Tabella con le triple dell'individuo *CV_Jeff_Bezos*.

Nel CV sono presenti le varie sezioni, ognuna delle quali espressa come istanza di una delle sottoclassi della classe *Campi_CV*. Vengono mostrate nel dettaglio la sezione relativa alle informazioni personali.

	subject	predicate	object	context
1	jso:infoPersonali_Jeff_Bezos	jso:anniCarriera	"22" ^{xsd:integer}	http://modsem.com/graph/JSOProject
2	jso:infoPersonali_Jeff_Bezos	jso:dataNascita	"1964-01-12" ^{xsd:date}	http://modsem.com/graph/JSOProject
3	jso:infoPersonali_Jeff_Bezos	jso:nazionalit�	"USA"	http://modsem.com/graph/JSOProject
4	jso:infoPersonali_Jeff_Bezos	jso:professione	"Direttore"	http://modsem.com/graph/JSOProject
5	jso:infoPersonali_Jeff_Bezos	jso:viveln	"New York"	http://modsem.com/graph/JSOProject
6	jso:infoPersonali_Jeff_Bezos	rdf:type	jso:Campi_CV	http://modsem.com/graph/JSOProject
7	jso:infoPersonali_Jeff_Bezos	rdf:type	jso:infoPersonali	http://modsem.com/graph/JSOProject
8	jso:infoPersonali_Jeff_Bezos	rdf:type	owl:NamedIndividual	http://modsem.com/graph/JSOProject
9	jso:infoPersonali_Jeff_Bezos	rdfs:comment	"Info personali dell'individuo Jeff_Bezos" ^{lit}	http://modsem.com/graph/JSOProject
10	jso:infoPersonali_Jeff_Bezos	rdfs:label	"InfoPersonali_Jeff_Bezos" ^{lit}	http://modsem.com/graph/JSOProject
11	jso:infoPersonali_Jeff_Bezos	foaf:depiction	jso:foto_Jeff_Bezos	http://modsem.com/graph/JSOProject
12	jso:infoPersonali_Jeff_Bezos	foaf:firstName	"Jeff"	http://modsem.com/graph/JSOProject
13	jso:infoPersonali_Jeff_Bezos	foaf:gender	"Maschio"	http://modsem.com/graph/JSOProject
14	jso:infoPersonali_Jeff_Bezos	foaf:hasPersonalmailbox	jso:email_Jeff_Bezos	http://modsem.com/graph/JSOProject
15	jso:infoPersonali_Jeff_Bezos	foaf:hasPhone	jso:Phone_Jeff_Bezos	http://modsem.com/graph/JSOProject
16	jso:infoPersonali_Jeff_Bezos	foaf:img	jso:foto_Jeff_Bezos	http://modsem.com/graph/JSOProject
17	jso:infoPersonali_Jeff_Bezos	foaf:surname	"Bezos"	http://modsem.com/graph/JSOProject

Figure 13: Tabella con le triple dell'individuo *infoPersonali_Jeff_Bezos*.

6 Queries SPARQL

Vengono riportate di seguito le queries in linguaggio SPARQL, utilizzante per interrogare l'ontologia ed estrarre da essa le informazioni utili per aziende e candidati.

Nelle figure successive viene mostrata l'interfaccia per l'interazione tra l'utente e l'ontologia, tramite la Web Application.

Settore: Automobilismo ▼
Mostrare nome del settore:
☐ Si ☒ No
Studi universitari:
Laurea Specialistica ▼
Certificati Lingua: Inglese ▼
Cerca

Query Federata
Inserisci un settore e vedrai i candidati disponibili tramite dbpedia e JSO

Informatica 🔍

Figure 14: Interazione lato Azienda

Settore: Economico ▼
Competenze: Organizzative ▼
Aggiungi Conoscenze
Conoscenze candidato:
Gestione del magazzino
Salario mensile: 1000 - 1500 ▼
Possiedi un dottorato: ☐ Si ☒ No
Città: Torino ▼
Cerca

Query Federata
Inserisci una nazione per vedere tutte le aziende presenti tramite l'utilizzo di dbpedia e JSO.

Italia 🔍

Figure 15: Interazione lato Candidato

Di seguito vengono definiti i prefissi necessari all'esecuzione delle query.

```
PREFIX : <http://www.semanticweb.org/OntologiaRicercaLavoro#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX frapo: <http://purl.org/cerif/frapo/>
```

Nel codice delle query le "parole" in corsivo, vengono gestite come variabili nell'applicazione client, in modo da avere delle query più flessibili.

6.1 Query per le Aziende

6.1.1 Query 1

La seguente query viene utilizzata dalle aziende per visualizzare tutti i candidati con il titolo di studi specificato (nel seguente caso "Laurea Specialistica").

```
SELECT ?candidato ?ls ?nomeTitoloDiStudi
WHERE {
    ?candidato :possiede ?cv.
    ?cv :contiene ?campi_cv.
    ?campi_cv :haCampo ?ls.
    ?ls :nome ?nomeTitoloDiStudi.
    ?cv rdf:type :CV.
    ?campi_cv rdf:type :Campi_CV.
    ?ls rdf:type :LaureaSpecialistica.
}
```

I risultati che si ottengono eseguendola sull'ontologia allo stato attuale sono i seguenti.

candidato	ls	nomeTitoloDiStudi
Marco	laurea_specialistica_Marco	"Laurea Artificial Intelligence"
Jeff_Bezos	laurea_specialistica_Jeff_Bezos	"Laurea specialistica in Economia e Management"

Figure 16: Il risultato della query 1

Flowchart della query disponibile a pagina 22

6.1.2 Query 2

La query permette di cercare tutti i lavoratori che hanno almeno un titolo di studi universitario e che posseggono un certificato di lingua (nel caso mostrato di seguito "Inglese"), ordinando quest'ultimo dal livello più alto al livello più basso.

```
SELECT  distinct ?candidato    ?tipoCertificato
WHERE {
    ?candidato :possiede ?cv.
    ?cv :contiene ?campi_cv.
    ?campi_cv :haCampo ?studiUniversitari;
              :haCampo ?certificatoLingua.
    ?certificatoLingua :nome ?tipoCertificato.

    FILTER regex(str(?tipoCertificato), "inglese").

    ?cv rdf:type :CV.
    ?campi_cv rdf:type :Campi_CV.
    ?studiUniversitari rdf:type :StudiUniversitari.
    ?certificatoLingua rdf:type :CertificatiLingua.
}
ORDER BY DESC(?tipoCertificato)
```

I risultati che si ottengono eseguendola sull'ontologia allo stato attuale sono i seguenti.

?candidato	?tipoCertificato
:Vittoria	C2 inglese, B2 francese, B2 spagnolo, A2 russo ^{AA} xsd:string
:Jeff_Bezos	C2 inglese ^{AA} xsd:string
:Marco	C1 inglese, A2 francese ^{AA} xsd:string

Figure 17: Il risultato della query 2

Flowchart della query disponibile a pagina 22

6.1.3 Query 3

Ponendosi dal lato dell'azienda, tale query cerca tutti i candidati nel settore di studi universitario specificato (nel seguente caso "Informatico") e ne ritorna i relativi percorsi di studi. Per questa query occorre utilizzare il plugin *Snap SPARQL Query* di Protege.

```
SELECT distinct ?candidato ?studiUniversitari ?nomeTitoloDiStudi
WHERE {
    ?candidato :possiede ?cv.
    ?cv :contiene ?campi_cv.
    ?campi_cv :haCampo ?studiUniversitari.
    ?studiUniversitari :nome ?nomeTitoloDiStudi;
                        :haSettoreStudi ?settoreStudi.
    ?settoreStudi rdf:type :Informatico.
    ?cv rdf:type :CV.
    ?campi_cv rdf:type :Campi_CV.
    ?studiUniversitari rdf:type :StudiUniversitari.
}
ORDER BY ?Candidato
```

Il risultato della query è riportato in seguito.

?candidato	?studiUniversitari	?nomeTitoloDiStudi
Jeff_Bezos	dottorato_Jeff_Bezos	Ecommerce avanzato con utilizzo di AI^^xsd:string
Marco	laurea_triennale_Marco	Laurea In scienze informatiche^^xsd:string
Marco	laurea_specialistica_Marco	Laurea Artificial Intelligence^^xsd:string
Mark_Zuckerberg	laurea_magistrale_Mark_Zuckerberg	intelligenza artificiale e sistemi informatici^^xsd:string

Figure 18: Il risultato della query 3

Flowchart della query disponibile a pagina 23

6.1.4 Query Federata Azienda - verso endpoint esterno

La query federata realizzata permette di estrarre da DBpedia le prime 3 persone, con la relativa *data di nascita*, che lavorano nel campo "Computer" (quindi un campo del settore informatico dell'ontologia da noi sviluppata) e tutti i candidati presenti nell'ontologia da noi realizzata che hanno come settore d'interesse quello *Informatico*.

Da notare che la libreria *ARC2* (da noi utilizzata) per *php* non supporta l'utilizzo di query federate, ma permette comunque l'interrogazione di diversi endpoint. L'idea è stata quindi di andare ad interrogare separatamente i due endpoint (Dbpedia e JSO) e andare ad unire i risultati tramite semplici istruzioni java. La query mostrata, in realtà, per il suo reale utilizzo è stata divisa in due differenti query (una per Dbpedia e una per JSO)

```
SELECT distinct ?name/candidato ?birth/nascita
WHERE {
    ?candidato :possiede ?cv.
    ?cv :contiene ?campi_cv.
    ?campi_cv :haCampo ?infoPersonalì.
    ?infoPersonalì :dataNascita ?nascita.
    ?candidato :haSettoreDiInteresse ?settore.
    ?cv rdf:type :CV.
    ?campi_cv rdf:type :Campi_CV.
    ?infoPersonalì rdf:type :InfoPersonalì.
    ?settore rdf:type :Informatico.
    ?candidato rdf:type :Candidato.

    SERVICE <https://dbpedia.org/sparql> {
        ?p a dbo:Person.
        ?p dbo:occupation ?occupation.
        ?p foaf:name ?name .
        ?p dbo:birthDate ?birth.
        FILTER regex(str(?occupation), 'Computer')
        FILTER (langMatches(lang(?name), 'en'))
    }
}
```

I risultati della query sono i seguenti

Name	Data di nascita
Luis Falcón Martín	1970-09-03
Boleslaw Szymanski	1950-04-22
Alex Martelli	1955-10-05
Jeff_Bezos	1964-01-12
Marco	1993-09-11
Mark_Zuckernberg	1984-05-14
Mark_Zuckerberg	1984-05-14

Figure 19: Il risultato della query federata per l'azienda

Flowchart della query disponibile a pagina 23

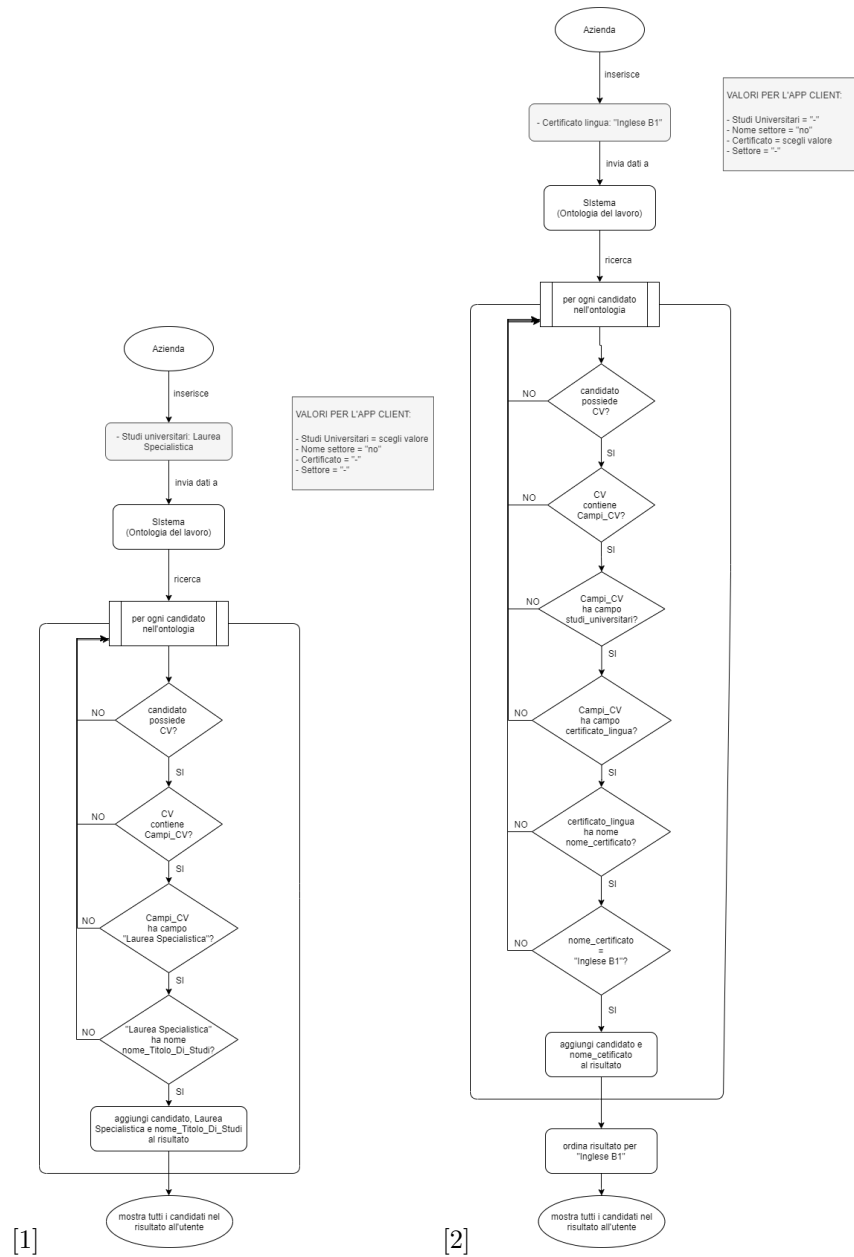


Figure 20: (1) flowchart query 1 (2) flowchart query 2

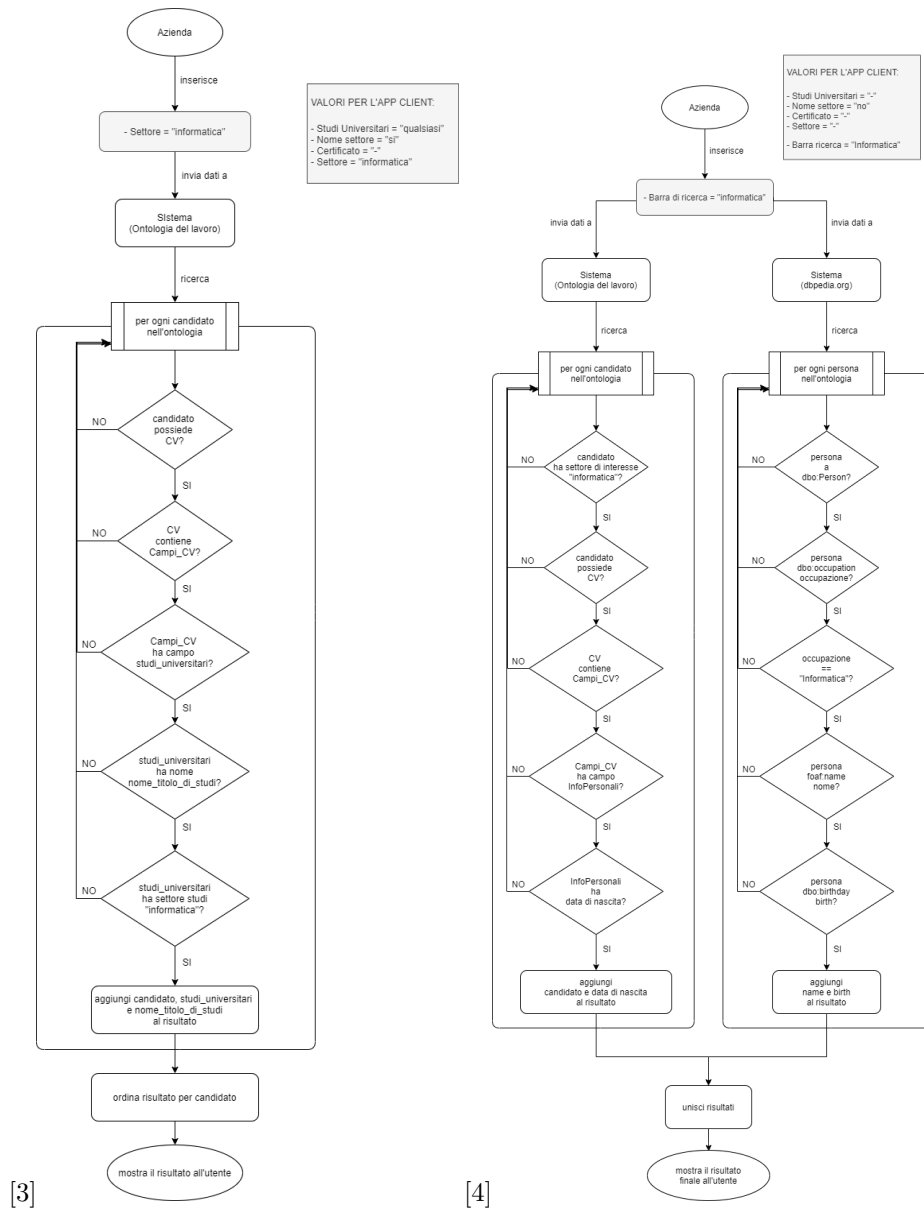


Figure 21: (3) flowchart query 3 (4) flowchart query federata Azienda

6.2 Query per i Candidati

6.2.1 Query 1

La seguente query permette al candidato di vedere nella bacheca, della Web Application, tutte le aziende che hanno pubblicato degli annunci.

```
SELECT ?azienda ?annuncio
WHERE {
    ?annuncio :ePubblicato ?azienda.
    ?annuncio rdf:type :Annuncio.
}
```

I risultati che si ottengono dalla sua esecuzione sono i seguenti.

azienda	annuncio
azienda_Amazon	annuncio_Amazon_due
azienda_Amazon	annuncio_Amazon
azienda_Apple	annuncio_Apple
azienda_Tesla	annuncio_Tesla
azienda_Orologel	annuncio_Orologel

Figure 22: Il risultato della query

Il flowchart della query è disponibile a pagina 30.

6.2.2 Query 2

La seguente query trova tutte le aziende che hanno pubblicato un annuncio nel settore selezionato dall'utente, con un salario mensile >1600 e con le competenze richieste uguali alle conoscenze indicate dall'utente (in questo caso la conoscenza di "Swift").

Per eseguirla occorre utilizzare il plugin *Snap SPARQL Query* di Protege.

```
SELECT ?azienda ?annuncio ?competenzeRichiesteDettagli
      ?contratto ?salario
WHERE {
    ?azienda :pubblica ?annuncio;
            :haSettoreAzienda ?settore.
    ?annuncio :salarioMensile ?salario;
            :competenzeRichiesteDettagli ?competenzeRichiesteDettagli;
            :contrattoProposto ?contratto.
    FILTER (?salario > 1600).
    FILTER regex(str(?competenzeRichiesteDettagli), 'Swift').
    ?settore rdf:type :Informatico.
    ?annuncio rdf:type :Annuncio.
}
```

Il risultato ottenuto è il seguente.

?azienda	?annuncio	?competenzeRichiesteDettagli	?contratto	?salario
:azienda_Apple	:annuncio_Apple	sviluppatore app iOS, Swift ^{AA} xsd:string	:Determinato	1800.0

Figure 23: Il risultato della query

Il flowchart della query è disponibile a pagina 30.

6.2.3 Query 3

Se si è ricercatori (candidato di tipo "candidato ricercatore"), vengono trovati tutti gli annunci che hanno lo stesso settore del candidato, mostrando le aziende che fanno ricerca nello stesso settore. Per eseguirla occorre utilizzare il plugin *Snap SPARQL Query* di Protege.

```

SELECT    ?annuncio ?azienda  ?contratto
WHERE{
    ?candidato :haSettoreDiInteresse ?settoreCandidato.
    ?annuncio :haSettoreAnnuncio ?settoreAnnuncio;
               foaf:maker ?azienda;
               :contrattoProposto ?contratto.

    ?candidato rdf:type :CandidatoRicercatore.
    ?annuncio rdf:type :Annuncio.
    ?azienda rdf:type :Azienda.
    ?settoreCandidato rdf:type :Settore.
    FILTER(?settoreCandidato = ?settoreAnnuncio)
}

```

Eseguendola si ottiene il seguente risultato.

?annuncio	?azienda	?candidato
:annuncio_Tesla	:azienda_Tesla	:Elon_Musk
:annuncio_Amazon	:azienda_Amazon	:Jeff_Bezos
:annuncio_Amazon_due	:azienda_Amazon	:Jeff_Bezos

Figure 24: Il risultato della query

Il flowchart della query è disponibile a pagina 31.

6.2.4 Query 4

La query restituisce l'annuncio, la sua descrizione, il contratto proposto e le ore giornaliere per tutti gli annunci che si trovano nella città scelta dall'utente con il settore richiesto. Per eseguirla occorre utilizzare il plugin *Snap SPARQL Query* di Protege.

```
SELECT DISTINCT ?annuncio ?descrizione ?contratto ?ore
WHERE{
    ?annuncio    :haSettoreAnnuncio ?settore;
                frapo:hasCountry ?countryAnnuncio;
                :descrizione ?descrizione;
                :contrattoProposto ?contratto;
                :oreDiLavoro ?ore.

    FILTER regex(str(?countryAnnuncio), 'Roma')
    ?annuncio rdf:type :Annuncio.
    ?settore rdf:type :Automobilistico.
}
```

Eseguendola si ottiene il seguente risultato.

?annuncio	?descrizione	?contratto	?ore
:annuncio_Tesla	Si propone un lavoro a tempo determinato per la relaiz...	:Determinato	7

Figure 25: Il risultato della query 4

Il flowchart della query è disponibile a pagina 31.

6.2.5 Query 5

Trova l'annuncio avente un monte ore giornaliero compreso tra 6 e 8, che riguardi il settore scelto dall'utente (nel seguente esempio "Economico") e che abbia come competenze richieste le competenze selezionate dall'utente (in questo caso "AltreCompetenze"). In fine ordina gli annuncio per ordine decrescente del salario.

```
SELECT  ?annuncio ?salario ?oreLavoro ?settore
WHERE{
    ?annuncio :haCompetenzeRichieste ?compRichieste;
    :salarioMensile ?salario;
    :oreDiLavoro ?oreLavoro;
    :haSettoreAnnuncio ?settore.

    FILTER(?oreLavoro > 5 && ?oreLavoro <9).
    ?annuncio rdf:type :Annuncio.
    ?compRichieste rdf:type :AltreCompetenze.
    ?settore rdf:type :Economico.
}
GROUP BY ?annuncio ?salario ?oreLavoro ?settore
ORDER BY DESC(?salario)
```

Eseguendola si ottiene il seguente risultato.

?annuncio	?salario	?oreLavoro	?settore
:annuncio_Amazon	2900.0	8	:settore_eCommerce
:annuncio_Amazon_due	1800.0	8	:settore_eCommerce

Figure 26: Il risultato della query

Il flowchart della query è disponibile a pagina 32.

6.2.6 Query Federata Candidato - verso endpoint esterno

Attraverso tale query vengono mostrate al candidato le prime 3 tre aziende, estratte da Dbpedia, locate in un preciso paese (in questo caso l'Italia) con la relativa descrizione. Il risultato ottenuto viene unito con le aziende, e la relativa descrizione, presenti nel medesimo paese, ma estratte dall'ontologia da noi realizzata (JSO).

Da notare che la libreria *ARC2* (da noi utilizzata) per *php* non supporta l'utilizzo di query federate, ma permette comunque l'interrogazione di diversi endpoint. L'idea è stata quindi di andare ad interrogare separatamente i due endpoint (Dbpedia e JSO) e andare ad unire i risultati tramite semplici istruzioni java. La query mostrata, in realtà, per il suo reale utilizzo è stata divisa in due differenti query (una per Dbpedia e una per JSO).

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX dbt: <http://dbpedia.org/resource/Template/>

SELECT distinct ?impresa/azienda ?country/sede ?description/descrizione
WHERE {
    ?azienda :pubblica ?annuncio;
            :haSedePrincipaleIn ?sede;
            :descrizioneAzienda ?descrizione.
    FILTER regex(str(?sede), 'Italia').
    ?azienda rdf:type :Azienda.

    SERVICE <https://dbpedia.org/sparql> {
        ?impresa a dbo:Company .
        ?impresa dbo:country ?country.
        ?impresa foaf:name ?name .
        ?impresa dbo:abstract ?description.
        FILTER( langMatches(lang(?description),'it') ).
        FILTER regex(str(?country), 'Italia').
    }
} LIMIT 6
```

I risultati ottenuti sono riportati nella fig.27.
Il flowchart della query è disponibile a pagina 32

Azienda	Sede	Descrizione
Vatican_Publishing_House	Italy	La Libreria Editrice Vaticana (LEV) è la casa editrice della Santa Sede. Dall'ottobre 2017 è diretta da fra Giulio Cesareo.
Fonit_Cetra	Italy	La Fonit Cetra è stata una casa discografica italiana, attiva tra il 1957 e il 1998.
Bluebell_Records	Italy	La Bluebell Records è stata una casa discografica attiva dal 1959 al 1970.
azienda_Amazon	Italia	azienda di commercio elettronico statunitense, con sede a Seattle nello stato di Washington. È la più grande Internet company al mondo
azienda_Orologi	Italia	Azienda italiana attiva nel settore alimentare fondata a Cesena e con sedi in diverse zone d'Italia.
azienda_Tesla	Italia	azienda statunitense specializzata nella produzione di auto elettriche, pannelli fotovoltaici e sistemi di stoccaggio energetico.

Figure 27: Il risultato della query federata

6.3 Query federata: inserimento nel grafo

La seguente query federata permette di interrogare l'endpoint esterno *WikiData* e permette di inserire nel grafo la tripla estratta.

La query trova tutte le aziende fondate (wtd:P112) da Jeff Bezos (wd:Q312556) tali per cui l'azienda sia un'istanza (wtd:P31) di impresa (wd: Q6881511).

Nel grafo viene quindi inserita una tripla per ogni azienda individuata. Tale tripla ci indica che Jeff Bezos *lavoraPer* le aziende individuate. Viene eseguita tramite *GraphDB*.

```

INSERT {
  GRAPH <http://modsem.com/graph/JSOProject> {
    jso:Jeff_Bezos jso:lavoraPer ?aziendaLabel
  }
} WHERE {
  SERVICE <https://query.wikidata.org/sparql> {
    ?azienda wtd:P112 wd:Q312556.
    ?azienda wtd:P31 wd:Q6881511.
    SERVICE wikibase:label {
      bd:serviceParam wikibase:language "it".
      ?azienda rdfs:label ?aziendaLabel.
    }
  }
}

```

Esplorando il grafo riusciamo a vedere come la tripla sia stata inserita correttamente.

474	jso:Jeff_Bezos	jso:lavoraPer	"Amazon"@it	http://www.semanticweb.org
475	jso:Jeff_Bezos	jso:lavoraPer	"Blue Origin"@it	http://www.semanticweb.org

Figure 28: Il risultato della query

Il flowchart della query è disponibile a pagina 33

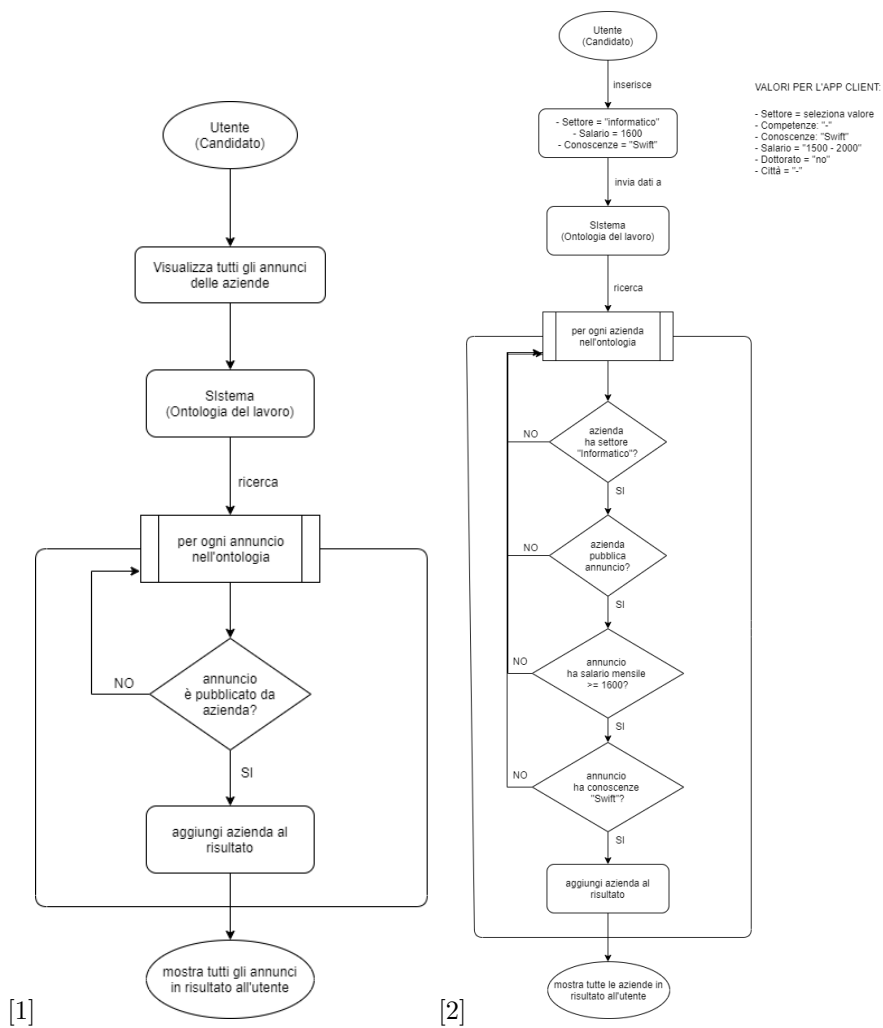


Figure 29: (1) flowchart query 1 (2) flowchart query 2

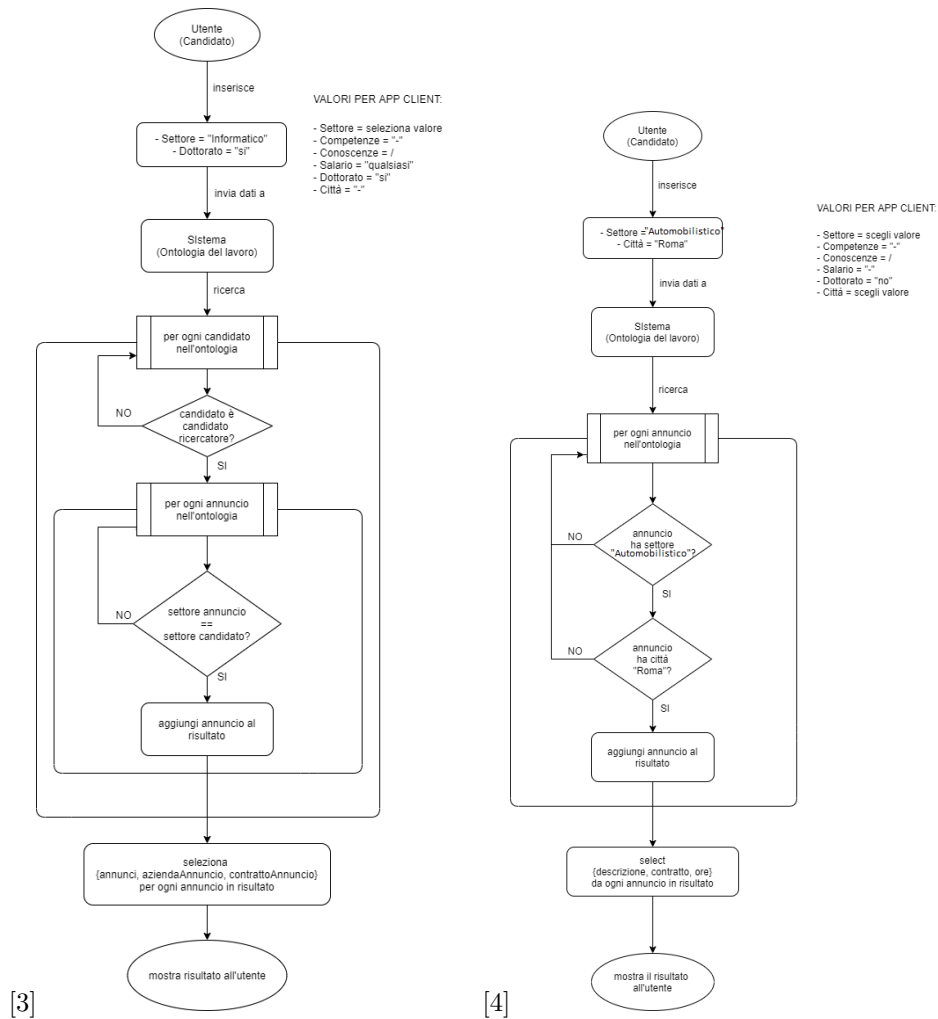


Figure 30: (3) flowchart query 3 (4) flowchart query 4

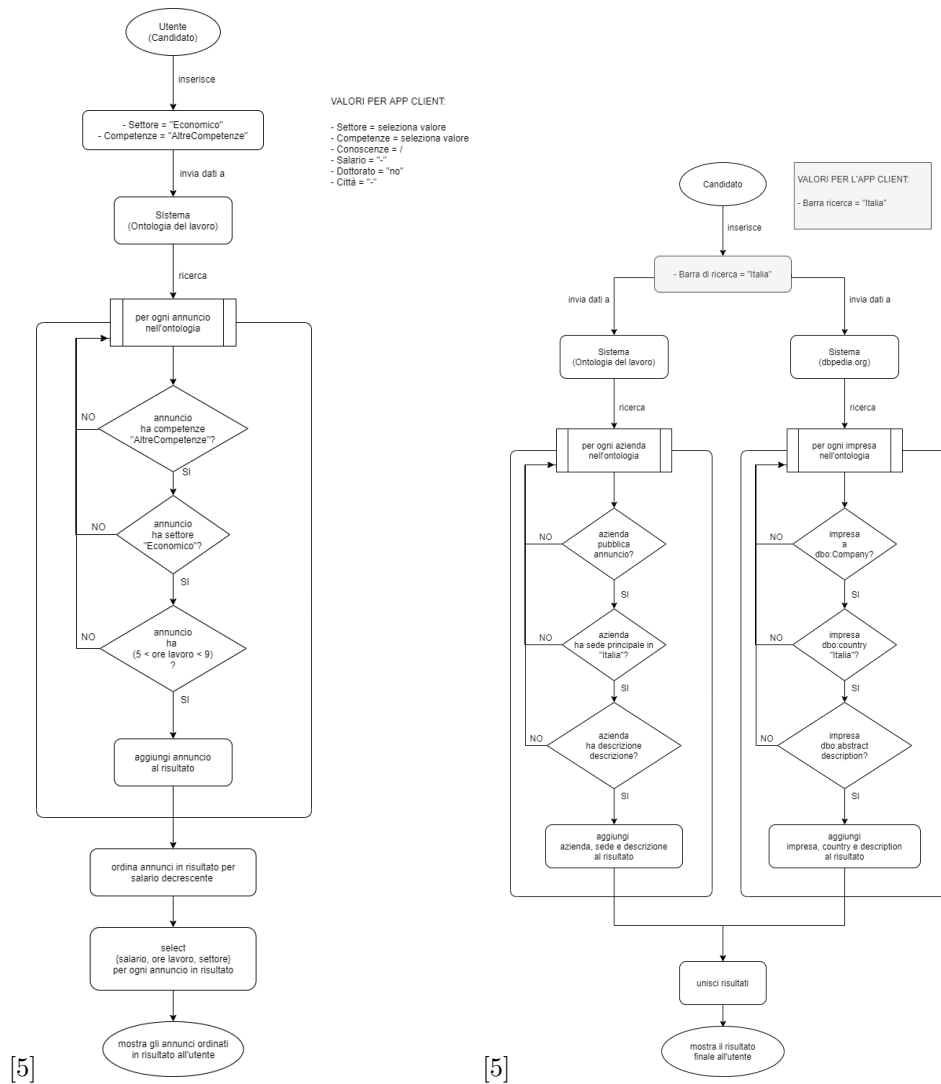


Figure 31: (5) flowchart query 5 (6) flowchart query federata1 candidato

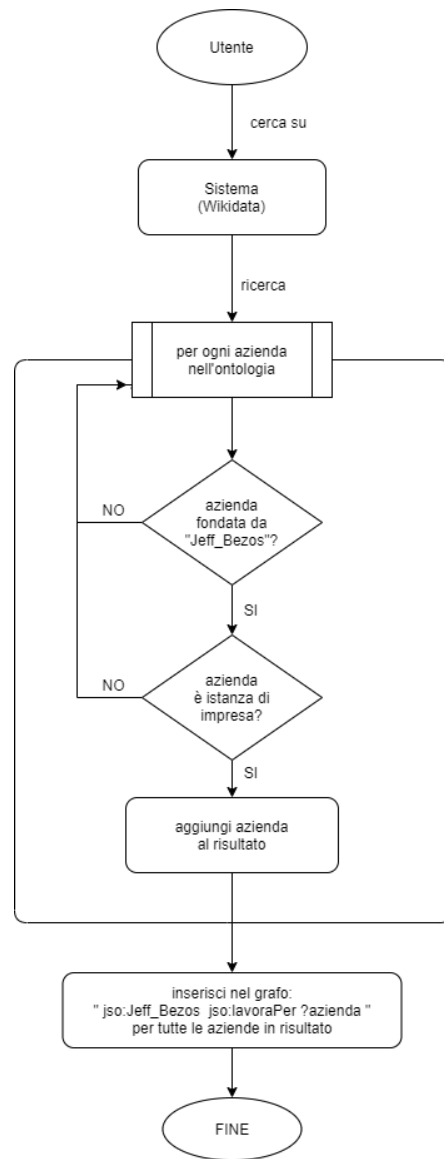


Figure 32: flowchart query federata per l'inserimento nel grafo

7 Estensione - Applicazione Client

Al fine della realizzazione dell'Applicazione Client è stata utilizzata la Linked Data Platform **GraphDB** e il suo SPARQL endpoint.

GraphDB è un servizio che permette di creare un repository su cui caricare l'ontologia. Fra le funzioni più utili troviamo la visualizzazione grafica in forma tabellare e attraverso un grafo dell'ontologia, la visualizzazione delle triple e la possibilità di eseguire query SPARQL. La sua semplice interfaccia lo rende molto intuitivo e ci ha permesso di capirne velocemente il funzionamento.

La Web Application è stata realizzata in *PHP* e per poter interrogare l'endpoint SPARQL si è reso necessario l'utilizzo della libreria *ARC2* di *Semsol*.

```
#!/php
/* ARC2 static class inclusion */
include_once('semo/ARC2.php');
// endpoint dell'ontologia JSO
$jsconfig = array(
    "remote_store_endpoint" => "http://192.168.184.1:7200/repositories/JobSearchOntology",
);
$store = ARC2::getRemoteStore($jsconfig);
if ($errs = $store->getErrors()) {
    echo "<h1>getRemoteStore error<h1>" ;
}

$query = "
    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
    SELECT ...
    WHERE {...}
    " ;
/* esecuzione della query */
$rows = $store->query($query, 'rows');
```

Figure 33: Utilizzo della libreria ARC2

L'applicazione permette agli utenti (sia Aziende che Candidati) di interagire con l'ontologia. Essi possono interrogarla (tramite le query SPARQL) specificando determinati parametri per ottenere le informazioni. Ad esempio se l'utente è:

- un candidato in cerca di lavoro, cercherà degli annunci specificandone le caratteristiche
- un'azienda in cerca di personale, cercherà fra i candidati che hanno inserito i loro dati

8 Estensione - Regole SWRL

SWRL è un linguaggio per la realizzazione di regole, che attraverso la produzione di regole di tipo dichiarativo permette di arricchire un'ontologia con nuove asserzioni. Si creano regole in forma IF-THEN, composte da antecedente e conseguente. È stato utilizzato il plugin *SWRL Tab* di Protege. Per l'interpretazione delle regole si è utilizzato il reasoner *Pellet* in modo da evitare che le asserzioni effettuate delle regole siano inserite nell'A-box dell'ontologia.

1. **Nome:** S1_AziendaAutomotive

Descrizione: Se l'azienda ha settore "Automotive" allora inferisce che l'azienda è un "AziendaAutomotive".

Regola:

```
jso:Azienda(?a) ∧ jso:Automotive(?sa) ∧  
jso:haSettoreAzienda(?a, ?sa) -> jso:AziendaAutomotive(?a)
```

2. **Nome:** S2_AnnuncioCondizioniVantaggiose

Descrizione: Individua se un annuncio ha delle condizioni di lavoro vantaggiose.

Regola:

```
jso:contrattoProposto(?x, ?contratto) ∧  
jso:salarioMensile(?x, ?salario) ∧  
swrlb:greaterThan(?salario, 2000) ->  
jso:AnnuncioCondizioniVantaggiose(?x)
```

3. **Nome:** S3_annuncioConsigliato

Descrizione: Segnala gli annunci che hanno lo stesso settore di interesse del candidato.

Regola:

```
jso:pubblica(?azienda, ?annuncio) ∧  
jso:haSettoreAnnuncio(?annuncio, ?settore) ∧  
jso:haSettoreDiInteresse(?candidato, ?settore) ->  
jso:annuncioConsigliato(?candidato, ?annuncio)
```

4. **Nome:** S4_gestisceAzienda

Descrizione: Individua se ci sono candidati che gestiscono un'azienda (sono CEO dell'azienda in cui lavorano).

Regola:

```
jso:possiede(?x, ?cv) ∧ jso:contiene(?cv, ?campi) ∧  
jso:haCampo(?campi, ?funzioniOccupate) ∧  
objectrole:hasRole(?funzioniOccupate, ?ceo) ∧ jso:CEO(?ceo)  
∧ jso:lavoraPer(?x, ?azienda) -> jso:gestisce(?x, ?azienda)
```

5. **Nome:** S5_CandidatoSecondoLavoro

Descrizione: Se la dataProperty "CercaLavoro" è "true" allora il candidato è in cerca di un secondo lavoro.

Regola:

$$\begin{aligned} & \text{jso:Candidato(?x)} \wedge \text{jso:cercaLavoro(?x, true)} \wedge \\ & \text{jso:lavoraPer(?x, ?azienda)} \wedge \text{jso:Azienda(?azienda)} \rightarrow \\ & \text{jso:CandidatoSecondoLavoro(?x)} \end{aligned}$$