ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Learning Path Recommendation

Studio e progettazione di modelli di raccomandazione di learning paths e di risorse didattiche.

Laurea Magistrale in
Ingegneria e Scienze Informatiche

Relatore: Prof.ssa Antonella Carbonaro

Presentata da: Sokol Guri

Anno accademico 2020-2021
Cesena

Indice

Capitolo 1

Introduzione

Il progresso nell'era digitale degli ultimi anni ha influenzato su tutti gli aspetti della vita quotidiana. Il progresso fatto dalle innovazioni tecnologiche sta ridefinendo e ristrutturando le metodologie di apprendimento. Con tale progresso e miglioramento, oggi gli utenti scelgono l'autoapprendimento attraverso l'internet. Al centro di questo cambiamento radicale sta l'E-learning.

Nei giorni d'oggi l'utilizzo di sistemi e-learning che forniscono risorse educative digitalizzate agli utenti, è diventata una normalità. L'e-learning porta tantissimi vantaggi rispetto ai metodi di apprendimento tradizionali con un insegnate che svolge il ruolo principale. Il vantaggio principale è l'aumento dell'accessibilità e della disponibilità delle risorse, riducendo i costi e rispettando la flessibilità degli utenti. L'e-learning è un tipo di apprendimento a distanza che viene svolto tramite internet dove l'utente può accedere alle risorse in qualsiasi momento da qualsiasi posto.

Le metodologi tradizionali di apprendimento incentrate sull'insegnante sono state utilizzate per tantissimi anni come la soluzione più efficacie e fattibile da implementare per ottenere i risultati migliori. Con il progresso fatto a livello tecnologico sono state rilevate tantissime problematiche che i metodi tradizionali portavano. Nella maggior parte degli utenti tali metodi portavano un disorientamento all'utente, fornendo una varietà di risorse disorganizzate e non strutturate. Questa problematica risultava critica se l'utente aveva un esperienza di apprendimento limitata.

Da tantissime ricerche fatte sull'ambito dell'apprendimento è stato affermato che oltre alla digitalizzazione delle risorse, anche l'ordine delle risorse ha un grande impatto sulla qualità dell'apprendimento. Gradualmente alla metodologia di e-learning, sono stati associatati modelli per l'organizzazione

delle risorse didattiche strutturate in una sequenza di materiali che rispettano l'ordine. Tale sequenza rappresenta un percorso di apprendimento personalizzato per un utente che lo guiderà al raggiungimento degli obbiettivi prefissati. L'utilizzo di E-learning con i modelli di raccomandazione potrebbe ridurre significativamente il tempo necessario per raccogliere e organizzare le risorse e in questo modo migliorare l'esperienza dell'apprendimento.

I modelli di raccomandazione forniscono una sequenza di materiali didattici come *learning-path*, però possono essere applicati su diversi domini. Uno di questo domini, sempre collegato con l'insegnamento e l'apprendimento, è academic-program-path. I modelli applicati a tale dominio prevedono un percorso educativo adatto per un utente in base alle informazioni che si forniscono al modello. Rispettando l'organizzazione strutturale delle istituzioni educative come le università, questi modelli prevedono le scuole, i dipartimenti e le facoltà da raccomandare all'utente in base alle sue caratteristiche (passioni, obbiettivi, stile di apprendimento, etc).

Il progetto di tesi propone un modello di raccomandazione del percorso accademico e del percorso di apprendimento per un utente basato sulla modellazione semantica del dominio. *Pathadora* è l'ontologia progettata per rappresentare e modellare i componenti di questo dominio, incorporando ontologie già esistenti sulla accessibilità e l'organizzazione strutturale delle istituzioni educative. Il modello di raccomandazione si basa su regole che estendono e inferiscono nuove relazioni semantiche tra i componenti dell'ontologia. Per ricevere le richieste di interrogazione e manipolazione della *knowledge* dell'ontologia è stato implementato *pathadora-recommender*, che svolge il ruolo di una engine sempre in esecuzione in attesa per computare la risposta alle richieste ricevute.

In questo documento di relazione verranno introdotto una varietà di modelli di raccomandazione che utilizzano diverse metodologie di progettazione, oltre alle regole semantiche. In seguito verrà spiegato in profondità la soluzione ontologica scelta per il modello di raccomandazione e le problematiche incontrate, focalizzandosi sugli vantaggi e svantaggi di tale scelta. Una sezione della relazione sarà dedicata alla progettazione del *pathadora-client* da parte di Andrea, che interagisce con l'engine e fornisce le richieste aspettando la risposta.

La relazione è organizzata come segue:

- Sezione 1- Introduction: introduce senza approfondire il problema e il tema del progetto, paragonando le soluzione tradizionali con quelle più innovative.
- Sezione 2- Background: *to be done*
- Sezione 3- Web Semantico: *to be done*
- Sezione 4- Analisi: *to be done*
- Sezione 5- Progettazione: *to be done*
- Sezione 6- Implementazione: *to be done*
- Sezione 7- Risultati: *to be done*
- Sezione 8- Conclusioni: *to be done*

Capitolo 2

Background

2.1 Ambito

E-learning è uno strumento importantissimo per il miglioramento della qualità dell'istruzione e formazione, basandosi sulle tecnologie d'informazione. La mancanza di adattamento dei contenutati educativi per gli studenti è un problema che e-learning ha portato sin dall'inizio. Mettere a disposizione degli studenti le stesse risorse allo stesso modo, non è la soluzione migliore e più efficiente possibile.

Provando di risolvere il problema della personalizzazione del modello di raccomandazione di risorse didattiche è nato il concetto di *learning path*. Un learning path (percorso di apprendimento) consiste nella progettazione di una sequenza di attività di apprendimento che aiutino lo studente a raggiungere gli obbiettivi prefissati. Durante gli anni la personalizzazione dei learning paths è diventata un problema importante da risolvere, perché tale soluzione si potrebbe applicare in vari domini, oltre all'aspetto didattico. Oltre alla raccomandazione delle risorse didattiche, si potrebbe raccomandare una sequenza di corsi da seguire, di video tutorial da vedere, di libri da leggere, di medicine da prendere, etc. La dinamicità di applicazione lo rende un problema molto complesso e complicato da trovare una soluzione unica e condivisa per tutti.

Dalla fine degli anni 60', i ricercatori hanno tentato di affrontare la personalizzazione, utilizzando diversi parametri, approcci e algoritmi. In seguito si spiegano i sistemi e le tecniche di raccomandazione studiate e implementate seguendo diverse soluzioni e modelli di progettazione.

2.2 Sistemi e tecniche di raccomandazione

Lo sviluppo di modelli di raccomandazione inizia dagli anni 60' utilizzando una sequenza direzionale di risorse didattiche basandosi sui meccanismi di sequenza del curriculum presentato. Questi meccanismi generavano i learning paths, offrendo un unica soluzione per tutti, siccome fornivano le stesse risorse didattiche. Come già accennato prima, le soluzione "*one-fit-all*" causavano tanti problemi, ignorando la diversità degli studenti.

Per risolvere il problema di una soluzione unica, sono stati introdotti *"personalized learning paths"*, focalizzandosi sulla personalizzazione e l'adattamento della soluzione allo specifico studente. In base ad alcuni paramenti iniziali che rappresentano le caratteristiche dello studente, i modelli di raccomandazione generavano diverse soluzioni.

Con l'introduzione di nuove tecniche di raccomandazione comparivano nuove problematiche. Tramite le personalized learning paths, veniva ignorato il progresso che lo studente stava facendo durante tale percorso. Questo problema influenzava negativamente l'efficienza del percorso di apprendimento. Non tenendo in considerazione una tale problematica si rischiava che lo studente non utilizzasse più questo sistema mostrando una mancanza correlazione tra il processo e il metodo di apprendimento.

Secondo Nabizadeh, Jorge, and Leal (2015) and Nabizadeh et al. (2017), i metodi di personalizzazione dei percorsi possono essere classificati in due macro categorie:

- *Path Generation*: questi metodi si focalizzano sulla generazione dell'intero percorso in unica raccomandazione e la valutazione dell'esperienza viene fatta alla fine del percorso.
- *Path Sequence*: questi metodi si focalizzano nella generazione di una sequenza di passi, che presi insieme formano il learning path. La raccomandazione del percorso intero viene fatta man mano che l'utente valuta l'andamento fino a quel punto.

2.2.1 Path Generation

Il processo di generazione del percorso di apprendimento intero consiste in diverse fasi. Durante la prima fase si stabiliscono le caratteristiche e i requisiti che l'utente dovrebbe specificare come parametri iniziali. Durante

la seconda fase, in base a tali parametri allo studente viene generato e consigliato un unico percorso.

Al posto di generare un percorso per ogni studente, alcune soluzioni tendono di raggruppare gli studenti tramite alcuni parametri in comune e poi raccomandare lo stesso percorso per tutti. Kardan et al. ha chiamato tale metodo come ACO-Map e consiste in una soluzione con due fasi principali. All'inizio si applica l'algoritmo K-means per dividere e raggruppare gli studenti e successivamente si ottimizza la soluzione tramite Ant-Colony-optimizzation in modo da generare un percorso per ogni gruppo.

GLPD (*Groupized learning path discovering*) è un altra tecnica di raccomandazione di gruppo, dove inizialmente viene generato un grafo di topics e poi si raccolgono le conoscenze e preferenze dello studente. Il modello dovrà stimare i limiti temporali per tutti i gruppi di studenti sul completamento di un percorso. In base a tali limiti una certa strategia viene scelta per generare il percorso.

Basandosi sulla teoria dei grafi, Belacel propone una tecnica di raccomandazione dove i vertici rappresentano le risorse e gli archi le relazioni e le dipendenze tra le risorse. La fase iniziale consiste nella potatura del grafo, togliendo i vertici irrilevanti e ragruppando quelli simili. L'algoritmo Branch-and-Bound viene utilizzato per individuare il percorso più breve.

Seguendo la logica di Belacel, CourseNavigator è un altro metodo della categoria di Path Generation che genera un percorso di apprendimento tramite i grafi. CourseNavigator si basa sulla logica dei grafi con un algoritmo di ricerca sul un insieme di percorsi generata dalle caratteristiche dichiarate dell'utente. Le caratteristiche servono come vincoli nell'elenco dei percorsi generati.

Seguendo la stessa logica Xu ha progettato un metodo di raccomandazione che ha come obbiettivo finale il completamento del percorso nel minor tempo possibile massimizzando il voto finale.

ECM (*Educational Concept Map*) viene considerato come uno dei metodi più efficienti della categoria Path Generation per la raccomandazione dei percorsi di apprendimento. Presentata da Adorni e Koceva nel 2015, si focalizza sulla conoscenza iniziale che uno studente ha, prima di iniziare il percorso. Lo studente definisce la sua conoscenza selezionando un insieme di argomenti nella mappa dei concetti. ECM utilizza l'algoritmo ENCODE

di Koceva, per linearizzare la mappa in base agli argomenti iniziali e target scelti dallo studente.

RUTICA è un metodo introdotto da Nabizadeh che ha come obbiettivo finale la generazione di un percorso che massimizza il voto finale allo studente con un limite temporale. Utilizzando l'algoritmo DFS (*Depth First Search*) si identificano tutti i possibili percorsi tenendo in considerazione il tempo. Per ogni percorso generato viene stimato e calcolato il voto finale.

Utilizzano i Markov chain, Xia ha progettato un sistema di raccomandazione basandosi su una sequenza di domande che lo studente dovrà rispondere. Le domande sono personalizzate in base allo studente e si focalizzano ad estrarre lo storico dell'esercitazione dello studente e altri concetti comuni che lo studente potrebbe avere con altri studenti.

Christudas, nel 2018, ha proposto una tecnica di raccomandazione basandosi sul CGA (*Compatible Genetic algorithm*). Tale tecnica si basa sullo stile di apprendimento, la conoscenza e l'interattività dello studente.

In uno degli studi più recenti in questo campo, Liu utilizza complex-network-theory definendo i percorsi in tre scenari diversi basandosi sullo storico di apprendimento dello studente.

I metodi di Path Generation sono utilizzati ampiamente anche se portano alcuni svantaggi. Tali metodi ignorano e non prendono in considerazione step intermediari durante il percorso generato. Nel caso dei percorsi di apprendimento, i modelli non considerano l'andamento dello studente e un probabile aggiornamento dei parametri dichiarati. Il percorso è statico ed è impossibile aggiornarlo step-by-step. Un percorso non efficiente porterebbe uno spreco di risorse e un risultato non efficiente per l'utente finale.

2.2.2 Path Sequence

Diversamente dai metodi Path Generation, i metodi Path Sequence raccomandano gli step del percorso man mano che lo studente avanza tenendo traccia dell'andamento. Per risolvere tale problema sono stati utilizzati diversi approcci come: ALN (*Association Link Network*), EAs (*Evolutionary Algorithms*), IRT(*Item Response Theory*), Bayes, etc.

Govindarajan, nel 2016, applico un algoritmo EA, in questo caso PSO *(Parallel Particle Swarm Optimizatio)* per restituire un percorso personaliz-

zato e dinamico per gli studenti. Govindarajan raggruppo gli studenti in base alla loro competenza, ovvero mappando una misura numerica alla possibilità di raggiungimento di un obbiettivo target e il cambiamento di tale unità con l'andamento dello studente. La dinamicità del percorso consiste nella raccomandazione dello step successivo basandosi sul cambiamento di tale unità di misura.

Seguendo la logica presentata da Govindarajan, Li sviluppo un metodo per la generazione dei percorsi dinamici basandosi su due algoritmi, MLE *(Maximum Likelihood Estimation)* e GA *(Genetic Algorithm)*. La fase iniziale del metodo consisteva nella creazione di una sequenza di risorse definendo la loro difficoltà in base ai feedback dello studente. Successivamente viene applicato MLE per analizzare le abilità e gli obbiettivi di ogni studente. Alla fine un algoritmo PSO viene utilizzato per generare il percorso tramite i risultati ottenuti negli step precedenti. Per ogni step compiuto, i feedback dello studente servono ad aggiustare il livello di difficoltà delle risorse ed aggiornare le abilità dello studente.

Nel 2013, Yarandi pubblicava un studio sulla modellazione dei learning path utilizzando meccanismi del web semantico. Yarandi proponeva un sistema adattivo di e-learning utilizzando il modello ontologico del dominio focalizzandosi sulle abilità dello studente. Tale sistema riceveva come parametri d'input le abilità dello studente, la conoscenza, lo stile di apprendimento e le preferenze e alla fine generava un percorso basandosi su questi parametri. L'analisi delle risposte veniva fatta tramite IRT e successivamente aggiornava le abilità dell'utente.

Nello stesso anno, anche Salahli aveva proposto un sistema di raccomandazione utilizzando IRT. Il sistema inizialmente identificava i temi, le loro relazioni e la loro difficoltà. L'algoritmo IRT viene applicato per calcolare il grado di comprensione dei temi e il livello di conoscenza necessario per ogni tema. Quando un utente utilizza il sistema, il suo livello di conoscenza e la difficoltà del tema selezionato sono i parametri che servono per stimare il grado di comprensione.

La generazione di percorsi dinamici offre risultati efficienti nella risoluzione del problema di learning path, però porta alcuni svantaggi per quanto riguarda l'istante di tempo quando si devono aggiornare i profili degli studenti in base ai risultati ottenuti. L'utilizzo di un tempo statico e prefissato potrebbe risultare non efficiente in tutti i casi e porterebbe un risultato non corretto. L'aggiornamento continuo dei profili degli studenti nella maggior

parte dei casi porterebbe uno spreco di risorse computazionali.

2.2.3 IRT (Item Response Theory)

La teoria IR (Item Response) viene nota come la teoria della risposta latente, si riferisce a un insieme di modelli matematici con l'obbiettivo di spiegare le relazioni, i risultati, le risposte e le prestazioni tra tratti latenti. Questa teoria stabilisce un collegamento tra le proprietà degli elementi, gli individui che rispondono e il tratto misurato. IRT si è basata sulla relazione tra le prestazioni degli individui su un elemento di prova e i livelli di prestazione dei partecipanti al test su una misura complessiva dell'abilità che l'elemento è stato progettato per misurare. Ogni risposta a un determinato elemento fornisce un grado in inclinazione sul livello o sull'abilità dell'individuo del tratto latente. L'abilità di un individuo è la probabilità di approvare la risposta come corretta e maggiore è l'abilità, maggiore è la probabilità di una risposta corretta.

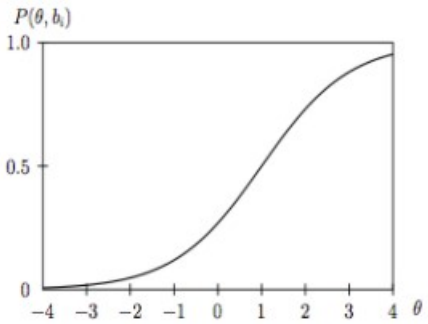

Figura 2.1: Curva caratterista dell'elemento.
La forma del grafico consiste in una S (Sigmoide/Ogiva). La probabilità di ottenere una risposta corretta dipende dalla abilità del intervistato la quale nelle applicazioni rispetta il range −3 and +3.

2.2.3.1 Assunzioni del IRT

- *Monotonicità*: l'assunzione specifica che la probabilità di ottenere una risposta corretta aumenterebbe aumentando il livello del tratto;
- *Unidimensionalità*: si assume che sia un tratto latente dominante da misurare che guida le risposte osservate per ogni elemento della misura;
- *Indipendenza locale*: si assume che le risposte date dagli elementi in un test sono reciprocamente indipendenti dato un livello di abilità;

- *Invarianza*: si possono stimare i paramenti di un elemento da qualsiasi gruppo di soggetto che ha già risposto all'elemento.

Basandosi sulle ipotesi iniziali e considerandole valide, la variazione del tratto latente da parte degli intervistati definisce le differenze nell'osservazione delle risposte corrette. Il modello prevede le risposte degli intervistati su uno strumento basato su elementi tramite la loro posizione nel grafico continuo del tratto latente e alle caratteristiche di tali elementi. Queste caratteristiche vengono considerate paramenti d'input per il modello.

2.2.3.2 Parametri degli oggetti

Le abilità e le capacita delle persone variano e sono molto dinamiche e per questo motivo la loro posizione nel grafico cambia. Il campione degli intervistati e i parametri degli oggetti definiscono la posizione precisa nel grafico.

- *Item Difficulty (bi)*: parametro che definisce il comportamento dell'oggetto lungo la scala delle abilità. Viene calcolato dalla capacità con cui il 50% degli intervistati approvano una risposta come corretta. Gli oggetti che è difficile approvarli sono spostati più a destra, mentre quelli più facili più a sinistra;

- *Item Discrimination (ai)*: parametro che definisce la velocità con cui la probabilità di approvare un oggetto corretto cambia in base ai livelli di abilità. Per ottenere un misura precisa, vengono inclusi elementi con discriminazione alta;

- *Guessing (ci)*: parametro che tiene in considerazione le ipotesi su un determinato elemento limitando la probabilità di approvare la risposta se l'abilità va negativamente all'infinito.

2.2.4 Three parameter logistic model

Nel modello logistico a tre parametri (3PL), la probabilità di una risposta corretta a un oggetto dicotomico i per una domanda a scelta multipla, è:

$$p_i(\theta) = c_i + \frac{1 - c_i}{1 + e^{-a_i(\theta - b_i)}}$$

dove:

- θ indica che le abilità degli intervistati sono definite in base ad una distribuzione normale per stimare i parametri dell'oggetto. In seguito sono stimate le abilità di ogni intervistato;

- a_i, b_i, c_i sono i parametri dell'oggetto;

Il cambiamento dei parametri viene rappresentato come il cambiamento della forma di una funzione logistica:

$$P(t) = \frac{1}{1 + e^{-t}}$$

dove:

- a - discriminazione, scala e pendenza massimale: $p'(b) = a \cdot (1 - c)/4$; Tale parametro rappresenta il grado della discriminazione dell'oggetto tra regioni diverse del grafico. Per esempio le persone con capacità basse hanno un probabilità molto minore di rispondere correttamente alla domanda.

- b - difficoltà e posizione dell'oggetto: $p(b) = a(1 + c)/2$. $p(b)$ in questo caso è il punto medio tra c_i (min) e 1 (max). In questo punto la pendenza è massimale.

- c - ipotesi: minimo asintotico $p(-\infty) = c$. Il parametro tende a spiegare il caso quando un intervistato con base capacità sceglie la risposta corretta.

Quando $c = 0$, $p(b) = 1/2$ e $p'(b) = a/4$, b è uguale 50% probabilità di avere successo (difficoltà) e a è la pendenza massimale (discriminazione).

2.3 Accessibilità

La potenza del Web sta nella sua universalità.
L'accesso da parte di tutti indipendentemente dalla disabilità è un aspetto essenziale. | Tim Berners-Lee |

Le persone disabili possono utilizzare facilmente le applicazioni e i siti web se sono state progettate e implementate correttamente, considerando i problemi di accessibilità. Molte applicazioni non gestiscono bene tale problema, portando barriere di accessibilità per gli utenti disabili e rendono questi applicazioni difficilmente utilizzabile da loro. Progettare applicazioni con livello alto di accessibilità, porta vantaggi per gli individui, le aziende e la società e per questo motivo sono state definite dei standard internazionali per risolvere tale problematica.

Il Web è stato progettato per essere utilizzato da tutte le persone, indipendentemente dal loro hardware, software, lingua, posizione o capacità. Il Web rimuove le barriere alla comunicazione e all'interazione che molte persone affrontano nel mondo fisico, però quando i siti Web, le applicazioni, le tecnologie o gli strumenti sono progettati male, possono creare barriere che escludono le persone dall'utilizzo del Web.

L'accessibilità al Web significa che i siti Web, le risorse, le informazioni, i tool e le tecnologie sono state progettate e implementate in modo che le persone con disabilità possono comprendere, percepire, navigare, utilizzare e contribuire al web. Le disabilità delle persone comprendono problemi uditivi, cognitivi, neurologici, fisici, di discorso, visivi, etc.

2.3.1 Linee guida per l'accessibilità

WCAG (*Web Content Accessibility Guidelines*) è un documento pubblicato da Accessibility Guidelines Working Group e copre una lista di raccomandazioni per rendere i contenuti Web più accessibili per le persone con disabilità diverse. Le linee guida specificate nel documento riguardano l'accessibilità dei contenuti sui diversi dispositivi elettronici e il rispetto di queste linee guida renderà i contenuti Web più fruibili per gli utenti in generale.

Le WCAG vengono utilizzate da individui che variano ampiamente e per poter soddisfare le diverse esigenze di individui diversi, vengono forniti diversi livelli di guida tra cui:

- *principi generali*: I principi più importanti che forniscono la base dell'accessibilità sono la percepibilità, l'operabilità, la comprensibilità e la robustezza;
- *linee guida generali*: WCAG introduce 13 linee guida che forniscono gli obbiettivi di base verso i quali gli autori dovrebbero focalizzasi per rendere i contenti e le risorse web più accessibili;
- *criteri di successo verificabili*: Per ogni linee guida vengono forniti criteri di successo verificabili per consentire l'utilizzo di WCAG.
- *una ricca raccolta di tecniche*: Per ciascuna delle linee guida e dei criteri di successo nel documento WCAG 2.0 stesso, il gruppo ha documentato un'ampia varietà di tecniche informative che si dividono in due categorie:
  - quelle sufficienti per soddisfare i criteri di successo;
  - quelle consultive che oltre quanto richiesto dai singoli criteri di successo, consentono e richiedono agli autori di affrontare meglio le linee guida;

2.3.2 Accessibilità delle risorse

Una risorsa accessibile è un risorsa creata e studiata per essere facilmente accessibile da un utente indipendentemente, dalla sua disabilità o dal formato della risorsa. Rendere accessibile una risorsa è più semplice quando siamo nelle fasi iniziali della creazione. Quando parliamo di risorse online di diversi tipi (testuali, immagini, video, audio) ci sono alcune considerazioni che dobbiamo verificare in modo da offrire una risorsa accessibile per utenti diversi. Nel caso delle risorse testuali, il livello d'accessibilità di un documento dipende da:

- *la struttura*: una struttura standardizzata di documenti migliora l'accessibilità, perché consente agli utenti di trovare i contenuti in una parte specifica della pagina, senza aver bisogno di leggere tutto il documento;
- *le intestazioni*: i titoli devono essere descrittivi e in un ordine coerente;
- *i floating elements*: questi elementi, comprese anche le immagini, vanno inseriti all'interno della sezione di appartenenza;
- *la risoluzione*: i documenti dovrebbero essere accessibili ai lettori che utilizzano dispositivi con schermi piccoli o ai lettori che utilizzano monitor a bassa risoluzione;

- *il testo*: le dimensioni dei caratteri ridotti o ingranditi, il tipo dei caratteri, l'indentazione e lo spazio sono parametri che dovrebbero essere utilizzati in modo da creare un stile da rispettare per il documento e ciò migliora l'accessibilità della risorsa;

- *i colori*: definiscono un alto livello di accessibilità se definiti con accuratezza, ma non utilizzare testo o sfondo colorato perché lettori non vedenti che accedono tale risorsa tramite una stampa o un dispositivo senza schermo a colori non riceveranno tali informazioni;

- *i blocchi di elementi*: gli elementi dell'elenco non si devono separare lasciando righe vuote o interruzioni di colonna tabulari tra di loro;

Le considerazioni sopraindicate aiutano al miglioramento della accessibilità dei documenti testuali, però ogni tipo di risorsa dovrà rispettare le linee guida specificate. Le immagini che non sono puramente decorative dovrebbero includere un attributo *alt* che sostituisce l'informazione dell'immagine per i lettori non vedenti. Per le immagini è una buona prassi includere una didascalia utilizzando la sintassi dell'immagine incorporata, descrivendo in modo conciso il significato dell'immagine e le informazioni essenziali che trasmette. Il posizionamento delle immagini può essere diverso in base al dispositivo o al formato del documento e per questo motivo si deve evitare il riferimento statico delle immagini come di sinistra o destra, si deve utilizzare invece le didascalie per identificarle e riferirle.

Per rendere una risorsa video accessibile, si possono aggiungere i sottotitoli in un formato di testo temporizzato, scaricabile a parte. I sottotitoli serviranno per la trascrizione delle informazioni trasmesse dal video. La stessa cosa deve valere anche per le risorse nel formato audio, dove trascrivendo il discorso tramite i sottotitoli, renderebbe la risorsa facilmente accessibile da un gruppo di utenti più ampio.

Capitolo 3

Web Semantico

Il web semantico è il grafo della conoscenza formato combinando dati collegati di Linked Data con contenuti intelligenti, metadati e altri oggetti informativi su larga scala per facilitare la comprensione e l'elaborazione dei contenuti da parte delle macchine.
https://simplea.com/Articles/what-is-the-semantic-web

3.1 La nascita del web semantico

Con lo sviluppo della tecnologia e del mondo d'internet, è stato possibile navigare diversi testi grazie ai link, che permettono di accedere altre informazioni cliccando tali link. L'internet non consiste solo nella gestione di grandi blocchi di informazioni, ma anche a scoprire e aggregare le associazioni tra tali blocchi, collegate tramite i link. Le relazioni tra i blocchi sono state rilevate da una serie di algoritmi che hanno rivoluzionati il Web da un contenitore di contenuti, ad un ambiente dove facilmente si possono trovare i contenuti richiesti tramite le associazioni tra informazioni e dati. Queste associazioni permettono di rintracciare tutto quanto concesso ad un concetto, un dato o una parola.

Per tanti anni si parlava di web di documenti, considerando i documenti come l'elemento più importante della navigazione nei sistemi web based. Nei giorni d'oggi si menziona sempre di più, web di dati, come nei database. L'obbiettivo del web di dati è di abilitare i compilatori ad effettuare un lavoro più efficiente e di sviluppare sistemi che possano supportare iterazioni sicure nella rete.

Il termine *web semantico* rappresenta una grande quantità di dati interconnessi tramite relazioni e facilmente estendibili e accessibili dagli utenti. L' articolo Scientific American considera il web semantico come *"un'estensione del web attuale dove alle informazioni viene dato un significato ben definito, consentendo ai computer e alle persone di lavorare in cooperazione."*

I dati, arricchiti con semantica, struttura e collegamenti significativi e interpretabili dalla macchina, consentono ai computer di trovare e manipolare le informazioni con maggiore precisione.

3.2 Lo sviluppo di web semantico

Per molti ricercatori web semantico è stato evoluto durante due fasi importantissime, la prima fino al 2006 e la seconda fino nei giorni d'oggi. Durante la prima fase di sviluppo si è lavorato sulla progettazione di un metodo sintetico oppure chiamato diversamente delle ontologie. Durante la seconda fase si è stato utilizzato il metodo analitico, o dei linked data. Il processo di sviluppo si è diviso in due fasi, perché il metodo sintetico era insufficiente a ottenere risultati concreti e intelligenti richiesti dagli utenti. L'obbiettivo finale del mondo del web semantico è quello di permettere ai calcolatori di comprendere al meglio la semantica delle ricerche dell'utente.

Qualsiasi algoritmo di ricerca è caratterizzato da tre elementi fondamentali:

- *organizzazione della conoscenza*: consiste nell'identificazione degli elementi che serviranno per la ricerca;
- *rappresentazione della conoscenza*: consiste nella descrizione dei dati;
- *strumentazione per accedere alla conoscenza*: tramite strumenti software sarà effettuata la ricerca di tutte le rappresentazioni di un dato che si trovano in rete.

Secondo Berners-Lee, i due componenti principali per sviluppare una repository d'informazione sono i metadati e il risultato di un collegamento ipertestuale. L'introduzione dei metadati permetteva ottenere informazioni sulle informazione trovate sul web che potevano essere interpretate dai calcolatori. L'utilizzo di valori ipertestuali è stato utile ad indirizzare le ricerche sulla rete. Il web semantico potrebbe essere definito come uno sviluppo del Web 3.0, che si focalizza sul miglioramenti dell'infrastruttura dei dati, in particolare sull'etichettamento dei dati in modo sa supportare le ricerche in

linguaggio naturale.

Aggiungere informazioni sulle informazioni, per molti ricercatori è stato considerato come una operazioni pesante e intensiva per il sistema, ma tale idea è stata implementata da Yahoo nel 2008.

3.3 Stack del web semantico

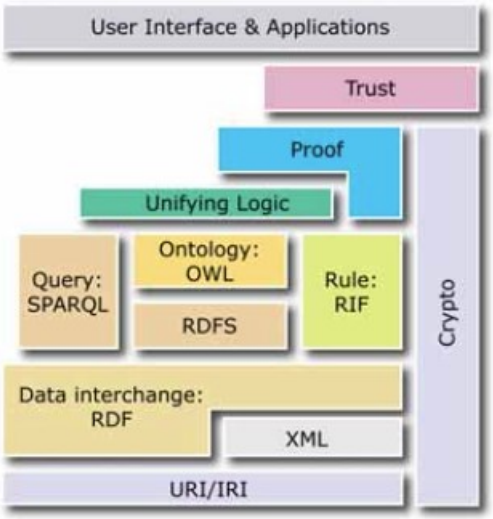

Figura 3.1: Stack del web semantico

La struttura del web semantico si rappresenta da tre livelli importanti, dove ogni strato viene suddiviso in sotto livelli:

- *livello base*: comprende le basi dei protocolli di comunicazione via Web (IRI, URI, Unicode);
- *livello corse*: comprende una serie di linguaggi con lo scopo di descrivere semanticamente le informazioni (RDF, OWL, SPARQL, etc);
- *livello finale*: comprende meccanismi che sono ancora in fase di sviluppo, con tendono a sincronizzare la progettazione semantica con intelligenza artificiale.

3.3.1 IRI e URI

Il livello base racchiude le tecnologie ipertestuali che forniscono le basi per il web semantico. IRI (*Internationalized Resource Identifier*), permette l'i-

dentificazione univoca delle risorse condivise nella rete. Le IRI aiutano nella generazione dei URI (*Uniform Resource Identifier*). Un URI è una stringa di caratteri che viene utilizzata per identificare univocamente una risorsa generica su Internet. Questo meccanismo di identificazione permette l'iterazione in rete con le risorse usando protocolli specifici.

Il mondo del web semantico necessita questo meccanismo di identificazione di informazioni per consentire la ricerca e la manipolazione delle risorse dai livelli superiori.

3.3.2 Unicode

La manipolazione e la rappresentazione delle informazione in diversi linguaggi viene effettuato tramite lo strumento di unicode. Tale strumento è uno standard informatico che permettere ai compilatori di rappresentare in maniera consistente e di manipolare i testi espressi in diverse lingue del modo. Non è altro che un sistema di codifica che assegna dei bit ad ogni carattere indipendentemente dal programma, dalla piattaforma o dalla lingua. Nello standard sono stati codificati i caratteri di tutte le lingue, i simboli matematici e chimici, i segni cartografici, gli ideogrammi, i simboli musicali, etc. In totale sono stati codificati più di 107.000 caratteri.

3.3.3 XML

Il livello core, racchiude le tecnologie più importanti che modellano semanticamente le risorse nella rete. XML (*eXtensible Markup Language*) è un metalinguaggio modella la struttura dei documenti e delle informazioni tramite un insieme di regole semantiche.

3.3.4 Namespace

Un namespace consiste in un insieme di attributi identificati univocamente da un identificatore e vengono utilizzati per dichiarare diversi sorgenti d'informazione per lo stesso concetto. L'associazione delle informazioni tra di loro necessità il riferimenti di più fonti.

In modo da utilizzare una lista di termini, si dovrebbe specificare una indicazione precisa dei vocabolari che verranno utilizzati. In questo modo si

ottiene un significato non ambiguo dagli identificatori e il documento ha una leggibilità migliore.

3.3.5 RDF

RDF (*Resource Description Framework*) permette la definizione di informazioni descrittive sulle risorse documentali tramite un insieme di regole. RDF viene utilizzato per la modellazione delle informazioni utilizzando le notazioni sintattiche e formati di serializzazione dei dati.

Il modello RDF assomiglia con i classici approcci di modellazione concettuale. Il modello si basa sulle asserzioni che vengono rappresentate tramite triple (soggetto, predicato, oggetto) che effettuano una relazione binaria tra gli elementi. Il predicato denota caratteristiche della risorsa ed esprime la relazione tra il soggetto e l'oggetto.

3.3.5.1 Classi e proprietà di RDF

RDF Schema fornisce strumenti per combinare tra loro le asserzione e le descrizioni in un singolo vocabolario. Gli elementi principali che permettono la definizione di un vocabolario sono:

- le classi:
  - `rdf:XLMLiteral` : la classe dei valori XLM;
  - `rdf:Property` : la classe delle properità;
  - `rdf:Statement` : la classe delle dichiarazioni;
  - `rdf:Alt, rdf:Bag, rdf:Seq` : le classi che rappresentano i contenitori di diversi tipi (di ordinati, non ordinati);
  - `rdf:List` : la classe che modella le liste;
  - `rdf:nil` : istanza di `rdf:List` che rappresenta una lista vuota;
  - `rdfs:Resource` : la classe delle risorse;
  - `rdfs:Literal` : la classe dei valori;
  - `rdfs:Class` : la classe delle classi;
  - `rdfs:Datatype` : la classe dei tipi di dato RDF;
  - `rdfs:Container` : la classe dei contenitori RDF;

- le proprietà:
  - `rdfs:type`: istanza di `rdf:Property` utilizzata per affermare che tale risorsa è istanza di una certa classe;
  - `rdfs:first`: il primo elemento di una lista RDF;
  - `rdfs:rest`: il resto della lista, ignorando il primo elemento;
  - `rdfs:value`: proprietà utilizzata per i valori strutturati;
  - `rdfs:subject`: il soggetto si una dichiarazione;
  - `rdfs:predicate`: il predicato di una dichiarazione;
  - `rdfs:object`: l'oggetto di una dichiarazione;
  - `rdfs:subClassOf`: il soggetto è una sottoclasse di una classe;
  - `rdfs:subPropertyOf`: il sogetto è una sotto proprietà di una proprietà;
  - `rdfs:domain`: dominio del soggetto di una proprietà;
  - `rdfs:range`: l'intervallo del soggetto di un proprietà;
  - `rdfs:label`: il nome del soggetto;
  - `rdfs:comment`: la descrizione del soggetto della risorsa;
  - `rdfs:isDefinedBy`: la definizione del soggetto di un a risorsa;

3.3.5.2 Serializzazione dei documenti RDF

Diversi formati di serializzazione sono stati progettatati per la modellazione semantica tramite RDF delle risorse nel web.

Turtle (*Terse RDF Triple Language*) è un formato di sintassi e di documento per rappresentare le informazioni tramite il modello di dati RDF. Questo formato prevede un raggruppamento di URI dei componenti della tripla (soggetto, predicato, oggetto) abbreviando l'informazione, per esempio aggregando le parti in comune delle URI.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .

[ foaf:name "Alice" ] foaf:knows [
   foaf:name "Bob" ;
   foaf:knows [
      foaf:name "Eve" ] ;
   foaf:mbox <bob@example.com> ] .
```

N-Triples è un altro formato per effettuare la memorizzazione e il trasferimento dei dati. Progettato per offrire una serializzazione più semplice del formato Turtle, più facile da parsare dai componenti software e rappresentare meglio le risorse innestate.

```
_:alice <http://xmlns.com/foaf/0.1/knows> _:bob .
_:bob <http://xmlns.com/foaf/0.1/knows> _:alice .
```

JSON-LD (*JavaScript Object Notation for Linked Data*) è un altro metodo per rappresentare la sintassi dei file che contengono informazioni semanticamente modellate. Il vantaggio di tale formato rimane la facilita di trasformazione dei file JSON in un JSON-LD, offrendo ai sviluppatori la possibilità di manipolare questi file utilizzando i metodi tradizionali.

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
    "Person": "http://xmlns.com/foaf/0.1/Person"
  },

  "@id": "https://me.example.com",
  "@type": "Person",
  "name": "John Smith",
}
```

RDF/XML è una dei formati di serializzazione maggiormente utilizzato e condiviso nel mondo del web semantico. Tale formato rappresenta un grafo RDF come un documento XML.

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:si="https://www.w3schools.com/rdf/">
   <rdf:Description rdf:about="https://www.w3schools.com">
      <si:title>W3Schools</si:title>
      <si:author>Jan Egil Refsnes</si:author>
   </rdf:Description>
</rdf:RDF>
```

Il meccanismo dei RDF per descrivere le risorse, è un componente importantissimo del web semantico che ha permesso la possibilità di archiviare, scambiare, manipolare e utilizzare le informazioni del web in un formato machine-readable. In questo modo è stato consentito agli utenti di gestire le risorse con maggiore efficienza, certezza e facilità. La collezione di dichiarazioni nel formato RDF, rappresenta un multi-grafo diretti e etichettati, rendendo il modello di dati più adatto certi tipi di rappresentazione della conoscenza.

3.3.6 OWL

OWL (*Ontology Web Language*) è un linguaggio di markup utilizzato per rappresentare il significato delle informazioni utilizzando le relazioni tra le informazioni e i vocabolari. OWL è stato progettato per rappresentare la conoscenza arricchita e complessa sugli elementi, gruppi di elementi e le relazioni tra loro. Secondo W3C, OWL è un linguaggio computazionale basato sulla logica, in modo che la conoscenza potrebbe essere utilizzata dai compilatori e dai programmi. I documenti OWL, vengono conosciuti come ontologie e una volta pubblicate nel WWW, potranno riferire o essere riferite da altre ontologie. La versione più recenti di OWL è OWL 2, sviluppata e pubblicata nel 2012.

Lo modellazione ontologica assomiglia con l'approccio del paradigma ad oggetti, organizzando le informazioni il classi e gerarchie di classi. Tramite le classi, le ontologie modellano informazioni e risorse che cambiano costantemente e velocemente nel web. Le ontologie riescono a rappresentare i dati provenienti da diversi fonti e sorgenti in modo molto flessibile.

Un dominio rappresentato da una ontologia tramite OWL, viene interpretato come una lista di *"individual"* e una lista di asserzioni di proprietà. Le asserzioni stabiliscono connessione tra diversi individui. Un'ontologia è costituita da un insieme di assiomi che definiscono vincoli su insiemi di individui e sui tipi di relazioni consentite tra di essi. Questi assiomi forniscono la semantica consentendo ai sistemi di dedurre ed inferire informazioni aggiuntive basate sui dati forniti esplicitamente.

3.3.6.1 Sottolinguaggi di OWL

Il linguaggio OWL prevede una varietà di sottolinguaggi con un maggiore livello di espressione, tra i quali:

- *OWL Lite*: linguaggio che supporta gli utenti a cui serve una gerarchia di classificazione e semplici vincoli di funzionalità.
- *OWL DL*: linguaggio che supporta gli utenti che vorrebbero massimizzare l'espressività senza perdere la completezza e la correttezza computazionale e la decidibilità del sistema di ragionamento. OWL DL, prende il nome dalla Description Logics, che si focalizza sulla decidibilità basandosi sulla logica del primo ordine;
- *OWL Full*: linguaggio che maggiore espressività e un sintassi svincolata con nessuna garanzia a livello computazionale.

3.3.6.2 Contenuto dell'ontologia

Un documento owl che rappresenta una ontologia inizia con la dichiarazione del namespace con gli identificatori e le URI dei vocabolari. Una volta dichiarato il namespace, viene specificata un insieme di asserzioni sotto `owl:Ontology`, aggiungendo informazioni sull'ontologia. Queste informazioni sono metadati che servono per controllare la versione del documento, le ontologie importate, eventuali commenti etc.

Importando una ontologia, si potrebbe accedere è utilizzare tutti i componenti di tale ontologia, definendo relazioni tra elementi di diverse ontologie. Le ontologie importate saranno sincronizzate con il namespace dichiarato. A volte l'importo di una ontologia potrebbe non avere successo siccome il sistema dovrebbe cercare le risorse condivise nella rete, che non è sempre possibile.

In seguito verranno spiegate gli elementi e i tag più importanti che OWL offre per la modellazione di un ontologia.

3.3.6.3 OWL classes

Le classi in OWL, sono uno strumento di astrazione importantissimo per raggruppare le risorse con caratteristiche simili. Come nelle classi RDF, ogni classe OWL è associata ad un insieme di individui, che vendono considerati come istanze della classe o estensione della classe. Il significato di una classe non è correlato con il significato della sua estensione, perché due classi possono avere la stessa estensione ma sono comunque diverse.

Le classi nell'ontologia vengono descritte tramite un insieme di assiomi utilizzando il nome della classe oppure specificando l'estensione della classe

come individuo di una classe anonima. OWL offre diversi modi per descrivere le classi tramite:

- un riferimento URI della classe: la descrizione viene fatta tramite il nome della classe.
- una enumerazione esaustiva degli individui che raggruppati insieme formano un istanza della classe;
- una restrizione di proprietà: l'insieme gli individui soddisfano la restrizione;
- l'intersezione, l'unione o il complemento di diverse classi.

3.3.6.4 OWL object properties

Le proprietà permettono di specificare caratteristiche associate ai membri di una certa classe e agli individui. Le proprietà degli oggetti definiscono relazioni tra istanze di due classi. Un assioma di proprietà definisce le caratteristiche della proprietà. La dichiarazione di una relazione tra le istanze si potrebbe vincolare definendo delle assiomi:

- sui costrutti del schema RDF: il domino e l'intervallo dei possibili valori utilizzando rispettivamente `rdf:domain` e `rdf:range`;
- sulle relazioni con altre proprietà utilizzando `owl:equivalentProperty` oppure `owl:inverseOf`;
- sulle cardinalità globali in questo caso utilizzando `owl:FunctionalProperty` e `owl:InverseFunctionalProperty`;
- sulle caratteristiche logiche della proprietà tramite `owl:SymmetricProperty` e `owl:TransitiveProperty`.

Le assiomi sopraelencato sono quelle più importanti, però OWL offre un insieme molto ampio di assiomi per poter aggiungere caratteristiche e informazioni sulle proprietà.

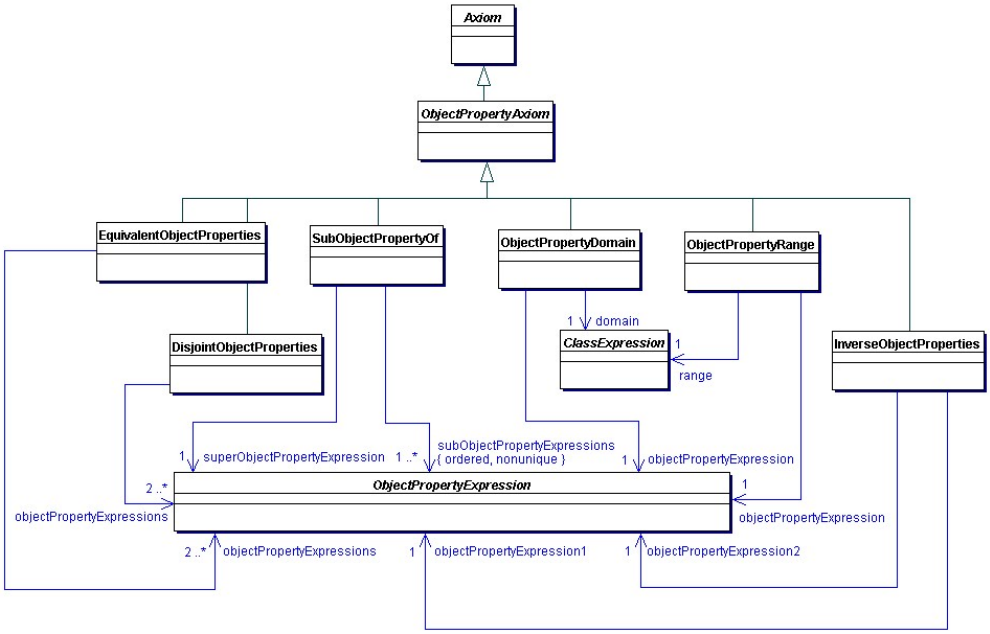

Figura 3.2: Le assiomi più importanti delle proprietà di oggetti

3.3.6.5 OWL data type properties

Le proprietà nelle ontologie vengo divise in base agli elementi che si riferiscono. Quando le proprietà si riferiscono a tipi di dati, parliamo di data type properties e si basano sui tipi di dati definiti tramite RDF e XML Schema. I tipi di dati più utilizzati con OWL sono: `xsd:string`, `xsd:decimal`, `xsd:integer`, `xsd:long`, `xsd:boolean`, `xsd:byte`, etc.

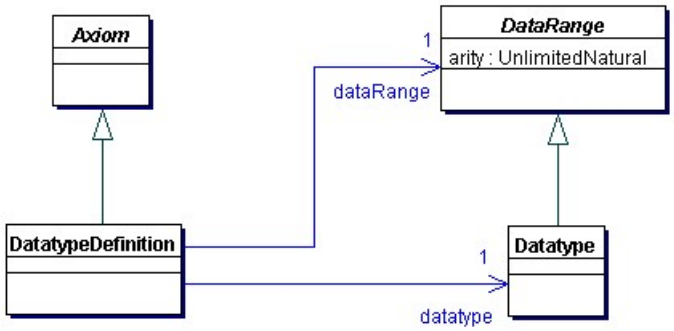

Figura 3.3: Definizione dei tipi di dato

3.3.6.6 OWL annotation properties

OWL offre le annotazioni come strumento per associare nuove informazioni all'ontologia, alle entità, alle assiomi, etc. La sintassi di OWL offre un meccanismo per incorporare i commenti nei documenti dell'ontologia.

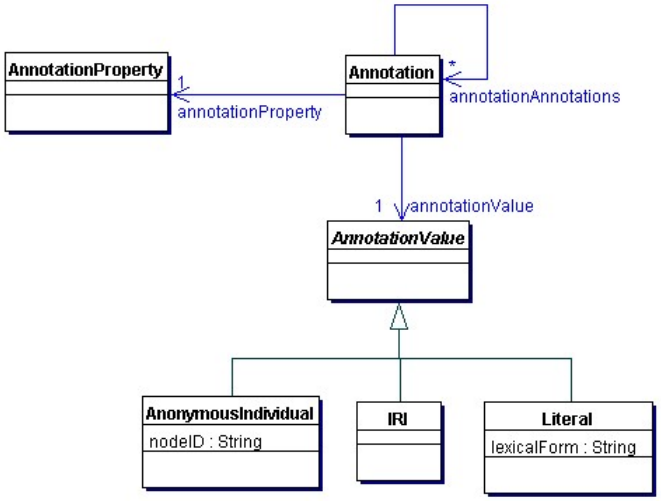

Figura 3.4: Annotazione delle ontologie in OWL

3.3.6.7 OWL individuals

Gli individui in OWL 2, rappresentano oggetti del dominio e possono essere di due tipi:

- *named individuals*: agli individui è stato dato un nome esplicito;
- *anonymous individuals*: gli individui non hanno un nome globale e sono locali all'ontologia alla quale fanno parte.

Gli individui nominati vengono identificato tramite il riferimento IRI perché vengono considerati come entità. Nel caso degli individui anonimi che non debbano essere utilizzati al di fuori dell'ontologia, loro vengono identificati tramite un nodo locale.

3.3.7 SPARQL

Lo SPARQL (*SPARQL Protocol and RDF Query Language*) è un linguaggio semantico per le interrogazioni dei database offrendo la possibilità di ricerca e di manipolazione dei dati memorizzati nei documenti rappresentati tramite il formato RDF.

Questo linguaggio potrebbe essere utilizzato per esprimere query tra diverse sorgenti di dati indipendentemente dal formato dei documenti. Tutto questo è possibile tramite i middleware che sono applicazioni che si interpongono in modo da offrire l'accessibilità sui diversi formati di documenti. Lo SPARQL offre la possibilità di fare query su diversi modelli di grafo messi insieme applicando su di loro delle operazioni. Il risultato delle interrogazioni potrebbe essere un insieme di dati oppure grafi RDF.

La sintassi e la semantica del linguaggio per interrogare i documenti si basa strettamente su due specifiche:

- `SPARQL Protocol for RDF`: definisce il protocollo per fare ed eseguire le query e ottenere il risultato da remoto;
- `SPARQL Query Result XML Format`: definisce il formato per rappresentare i risultati delle query eseguite.

Uno dei vantaggi più importati del linguaggio SPARQL sta nel fato che permette all'utente di applicare query non ambigue siccome si basa su identificatori URI non ambigui.

3.3.7.1 Sintassi delle interrogazioni

La maggior parte delle query rispettano il formato a triple, chiamato anche "*basic graph pattern*". Tale formato è molto simile alle triple RDF, ma con la differenza che il soggetto e l'oggetto possono essere variabili. L'accoppiamento tra basic graph pattern e un sottografo di informazioni RDF succedere quando gli elementi del sottografo possono essere sostituti al posto delle variabili del pattern e il risultato sarebbe un grafo RDF equivalente al sottografo.

```
<http://example.org/book/book1>
<http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

Listing 3.1: Esempio con i dati nel formato RDF

```
SELECT ?title
WHERE
{
   <http://example.org/book/book1>
   <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Listing 3.2: Esempio di una semplice query SPARQL

?title
"SPARQL Tutorial"

Tabella 3.1: Il risultato prodotto dall'applicazione della query SPARQL sull'insieme di dati sopraelencati

Il risultato di una query viene aggregato in una lista di soluzioni basandosi sulla corrispondenza del graph pattern con il sottografo dei dati in RDF. SPARQL offre la possibilità di utilizzare anche i literal RDF, i tipi numerici, i tipi di dati arbitrari nella struttura della query.

La struttura delle query non è uniforme perché SPARQL permette alle interrogazioni di assumere diverse forme. Le interrogazioni `SELECT` restituiscono un legame che i dati dichiarati nei documenti hanno con le variabili utilizzate nelle query. Le interrogazioni `CONSTRUCT` restituiscono un grafo RDF costruito basandosi su un modello che è stato utilizzato per generare triple RDF in base al matching del graph pattern con il dati.

SPARQL offre altri costrutti come `FILTER` per restringere il risultato della query solo a quelle tuple che soddisfano la condizione del filtro, `ORDER BY` per ordinare il risultato in base ad una regola, `DISTINCT` per ottenere un risultato con elementi con duplicati, etc.

3.3.8 SKOS

Lo SKOS (*Simple Knowledge Organization System*) è un insieme di linguaggi formali che fornisce un modello per rappresentare la struttura e il contenuto di schemi concettuali come glossari, tassonomie, classificazioni e altri tipi di vocabolari strutturati. Per molti ricercatori SKOS, viene considerato un vocabolario RDF per rappresentare la conoscenza semi formale. Il fato che SKOS si basa su RDF lo rende machine-readable.

Lo SKOS fornisce un linguaggio leggero, semplice ed intuitivo per i modelli concettuali permettendo di sviluppare e distribuire in rete nuovi KOS. Lo scopo di tale meccanismo e di fornire un percorso a basso costo per portare gli sistemi verso il web semantico. Lo SKOS svolge un ruolo da intermediario tra i formalismi ben organizzati dei linguaggi ontologici e quelli informali, scarsamente strutturati.

Il modello di dati SKOS è stato definito come una ontologia di tipo OWL Full e i dati SKOS sono stati rappresentanti come triple RDF da essere serializzate utilizzando qualsiasi sintassi. I concetti SKOS possono essere collegati ad altri concetti tramite le relazioni semantiche. Tali concetti possono essere raggruppati insieme in collezioni, a loro volta manipolate tramite l'ordinamento, etichettamento, etc.

Come nel caso di RDF, anche per il modello di dati SKOS, gli elementi più importanti della rappresentazione delle informazioni sono le classi e le proprietà. Le caratteristiche logiche delle classi e proprietà e loro dipendenze definiscono la struttura e l'integrità del modello. Tuttavia, lo SKOS non è un linguaggio formale per la rappresentazione della conoscenza, la quale viene espressa come un insieme di assiomi e fatti.

3.3.9 RIF

Il RIF (*Rule Interchange Format*) è uno standard ancora in fase di sviluppo, però un componente fondamentale dello stack del web semantico. Lo scopo principale di questo progetto è lo scambio delle regole tra diversi linguaggio di regole che già esistono. Nel 2005, fu istituito il *RIF Working Group* che aveva come obbiettivo di attirare coloro che creavano regole nei mercati commerciali sviluppando un formato di scambio tra i sistemi delle regole già esistenti.

La regola è un costrutto di tipo `IF-THEN` ed è probabilmente la nozione più semplice d'informatica. Questo costrutto descrive lo step da processare in caso una condizione viene verificata. I sistemi basati sulle regole utilizzano una nozione di predicato per rappresentare una certa informazione. Gli oggetti che sono argomenti della regola, vengono mantenuti connessi in base a tale predicato.

La progettazione e l'implementazione di motori di ricerca che inferiscono e processino le informazioni in base alle regole, è più facile siccome la conoscenza viene dichiarata tramite i predicati. Un sistema di regole è basato su un insieme di regole semantiche includendo quantificatori esistenziali, funzioni logiche disgiunzione, unione, negazione, etc. Nei giorni d'oggi tali sistemi vengono chiamati *sistemi esperti*, però il loro sviluppo data dagli anni '70.

RIF Group ha specificato tre dialetti standard del RIF:

- `Core`: dialetto che include un sottoinsieme comune di molte rule engines;
- `BLD`: estende il dialetto Core, aggiungendo caratteristiche che non sono direttamente disponibili;
- `PRD`: aggiunge la nozione della regola "forward-chaining".

3.3.10 SWRL

SWRL (*Semantic Web Rule Language*) è un linguaggio basato sulle regole per il mondo del web semantico ed è una combinazione dei sottolinguaggi dell'OWL con quelli del Rule Markup. Il linguaggio rimane una proposta da parte di W3C e non è stata approvata dal Consorzio.

La struttura delle regole si basa su due componenti principali: testa (consequent) e corpo (antecedent). La regola consiste in alcune condizioni espresse nell'antecedent e se le condizioni sono vere, allora dovrebbero essere vere anche le condizioni nella parte consequent. L'antecedent e il consequent consistono in zero o più atomi. Nel caso di un antecedent vuoto verrà considerato come sempre vero, mentre un consequent vuoto viene considerato come sempre falso e non sarà soddisfatto da nessuna interpretazione.

3.3.10.1 La sintassi delle regole

La sintassi del SWRL viene estratta da qualsiasi sintassi di documenti OWL per consentire un utilizzo più facile ed efficiente. Come discusso prima, qualsiasi ontologia OWL nella sua sintassi astratta contiene una sequenza di assiomi e di fatti. Tale ontologia potrebbe essere estesa definendo assiomi di regole come $axiom ::= rule$.

```
rule ::= 'Implies(' [URIreference ]{annotation }
antecedent consequent ')'
antecedent ::= 'Antecedent(' {atom }')'
consequent ::= 'Consequent(' {atom }')'
```

Listing 3.3: La struttura di un assioma di regole

```
atom ::= description '(' i-object ')'
   | dataRange '(' d-object ')'
   | individualvaluedPropertyID '(' i-object i-object
     ')'
   | datavaluedPropertyID '(' i-object d-object ')'
   | sameAs '(' i-object i-object ')'
   | differentFrom '(' i-object i-object ')'
   | builtIn '(' builtinID {d-object }')'
builtinID ::= URIreference
```

Listing 3.4: Composizione dell'atomo di una regola

Gli atomi in queste regole possono essere della forma:

- $C(x)$: valido se x è una istanza della classe descritta C;
- $P(x, y)$: valido se x è associato a y tramite la proprietà P;
- $sameAs(x, y)$: valido se x e y sono interpretatati come lo stesso individuo;
- $differentFrom(x, y)$: valido se x e y sono interpretatati come individui differenti;
- $builtin(r, x, \dots)$: valido se la relazione *builtin* di r è valida sulle interpretazione degli argomenti;

I valori di x e y potrebbero essere variabili, individui OWL o tipi di dati OWL.

3.3.11 Proof, Trust, Digital Firms

Il livello finale e la cima dello stack del web semantico consiste in un insieme di tecnologie che anche nei giorni d'oggi sono in fase si sperimentazione e di sviluppo senza avere una pubblicazione concretata e condivisa dalla maggior parte della comunità scientifica. Lo scopo delle tecnologie di questo livello è la realizzazione di sistemi in grado di formulare principi logici e permettere alle macchine di ragionare utilizzando tale principi.

Lo sviluppo dei sistemi logici permetterebbe di sfruttargli per provare e dimostrare la verità delle informazioni e le relazioni tra gli elementi. Il risultato delle dimostrazioni logiche deve restituire informazioni valide e vere. Qualsiasi utente che avrà la possibilità di accedere in rete, potrebbe scrivere istruzioni logiche e le machine potrebbero utilizzare tali riferimenti semantici per costruire le dimostrazioni. La costruzione delle dimostrazioni è molto difficile a livello operativo, ma potrebbero esser facilmente controllabili. Una volta ottenuto questo passaggio, il web potrà essere considerato come un meccanismo in grado di processare informazioni.

In una visione futura potremo vedere processori euristici che utilizzando le regole e gli istruzioni forniti dagli utenti, riusciranno ad aggregare autonomamente delle conclusioni. Dalla comunità scientifica viene chiamato

procedimento euristico un approccio alla soluzione dei problemi che si basa sulla intuizione e allo stato temporaneo delle istanze al fine di generare nuova conoscenza senza seguire un percorso chiaro e strutturato.

La firma digitale nel livello finale serve per garantire autenticità e sicurezza durante l'esecuzione dei processi. L'autenticità delle asserzioni viene garantita tramite un sistema di crittografia, dove la responsabilità del materiale pubblico cade sull'ente, sull'individuo o sull'organizzazione che lo ha condiviso. La firma digitale sarà univocamente associata ai documenti che si trovano nella rete, assicurando l'utente sulla provenienza di tale risorsa.

Il termine *"Web of Trust"* viene introdotto per rappresentare un modo della rete concentrato sulla fiducia, riservatezza e sicurezza. Condividendo la fiducia che due utenti avranno tra di loro, permetterà lo sviluppo di tale mondo associando a tutte queste relazioni un grado di fiducia. Alla fine sarà il livello di *User interface* che permetterà a tutti gli utenti di usufruire le applicazioni del web semantico.

3.4 Knowledge graph

Dopo avere parlato dei componenti principali dello stack del web semantico, adesso verranno discussi gli argomenti più importanti che riguardano i knowledge graph.

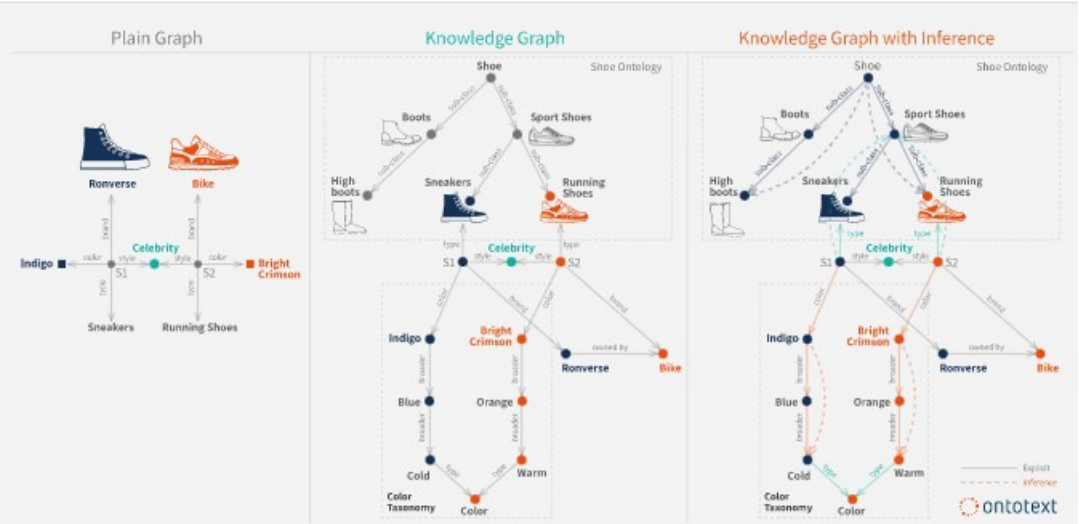

Figura 3.5: Esempio di una grafo di conoscenza

3.4.1 Cos'è un grafo di conoscenza?

Un knowledge graph oppure un grafo di conoscenza rappresenta un insieme enorme di entità che sono connesse tra di loro. Tutte queste relazioni formano una rete di entità intraconnesse e interconnesse. L'informazione viene memorizzata in database specifici per la gestione dei grafi di conoscenza e per la visualizzazione di tale informazione.

Come in tutti i casi, quando parliamo di un grafo, parliamo di una struttura composta da nodi e archi. Le etichette vengono utilizzate per facilitare la gestione e la leggibilità del grafo, aggiungendo informazioni e strumenti di ricerca per i nodi e gli archi. Gli archi permettono di definire relazioni tra diversi nodi che compongono il grafo.

Nella comunità scientifica c'è molta discussione sulle differenze tra le ontologie e i grafi di conoscenza, dove alcuni gli considerano la stessa cosa. La maggior parte condivide il fato che le ontologie servono per creare un formato di rappresentazione delle entità del grafo. Secondo *OntoText*, un grafo di conoscenza rappresenta una collezione di descrizioni di entità interconnesse, degli oggetti e eventi del mondo reale, e degli concetti astratti dove:

- le descrizioni hanno una semantica formale che permette agli utenti e ai processori di processarle in un modo efficiente e non ambiguo;
- le descrizioni delle entità contribuiscono a formare una rete di entità, dove ogni entità rappresenta una parte della descrizione associata a tale entità e fornisce contesto per l'interpretazione.

I knowledge graph combinano un insieme di caratteristiche di diversi paradigmi per la gestione dei dati tra i quali:

- *base di dati*: perché i dati possono ricercati, aggiornati e manipolati tramite interrogazioni strutturate;
- *grafi*: perché i dati possono essere analizzati e visualizzati come qualsiasi formato di dati in rete;
- *base di conoscenza*: perché portano una semantica formale, utile per interpretare i dati e dedurre nuova conoscenza.

Il knowledge graph rappresentato tramite RDF, offre il miglior framework per l'integrazione dei dati, la loro unificazione e riutilizzo. I grafi di conoscenza trovano un ampio dominio applicativo siccome riescono a combinare:

- *l'espressività*: Il core dei knowledge graph sono gli standard del web semantico, RDF e OWL, che vengono utilizzati per una rappresentazione fluida e semplice dei diversi tipi di dati e contesti. Gli schemi di dati, le tassonomie, i vocabolari e i metadati servono a modellare le informazioni in modo efficacie;

- *la prestazione*: Utilizzando il knowledge graph è stato provato nella pratica che consente una gestione efficiente di grafi su una grande quantità di fatti e proprietà;

- *l'interoperabilità*: Avendo una varietà di specifiche per la serializzazione dei dati, l'accesso (protocollo SPARQL per gli endpoint), la gestione (SPARQL Graph Store) e la federazione è possibile l'integrazione e la sincronizzazione con diversi sorgenti d'informazione. L'uso di identificatori univoci globali facilita l'integrazione e la pubblicazione dei dati;

- *la standardizzazione*: Tutte le caratteristiche sopra elencate sono state standardizzate dalla comunità scientifica, in questo caso W3C, per garantire che i requisiti siano sempre soddisfati.

Non tutti i grafi RDF e knowledge base sono grafi di conoscenza. Un elemento importantissimo dei knowledge graph è che le descrizioni delle entità devono essere interconnesse con una l'altra. La definizione di una entità include una altra entità e tale connessione forma il grafo di conoscenza. Le basi di conoscenza che non rispettano una struttura semantica formale, non possono essere considerate come grafi di conoscenza.

Nella rete sono miglia di enormi knowledge graph pubblici e condivisi, tra i quali DBPedia, Geonames, Wordnet, etc.

3.4.1.1 DBPedia

Questo progetto sfrutta la struttura inerente agli infobox di Wikipedia per creare un enorme dataset e un'ontologia che ha una copertura enciclopedica di entità come persone, luoghi, film, libri, organizzazioni, specie, malattie, ecc. Questo set di dati è al centro del movimento Open Linked Data. Questo progetto è stato importantissimo per le organizzazioni in modo da progettare i propri grafi di conoscenza interna con milioni di entità. DBpedia consente agli utenti di interrogare semanticamente le relazioni e le proprietà delle risorse di Wikipedia, inclusi i collegamenti ad altri set di dati correlati.

DBpedia estrae informazioni dalle pagine di Wikipedia, consentendo agli utenti di trovare risposte a domande in cui le informazioni sono distribuite su più articoli di Wikipedia. L'utente sarebbe in grado di accedere alle informazioni utilizzando un linguaggio di query simile a SQL per RDF, SPARQL.

```
PREFIX dbprop: <http://dbpedia.org/ontology/>
PREFIX db: <http://dbpedia.org/resource/>
SELECT ?who, ?WORK, ?genre
WHERE {
   db:Tokyo_Mew_Mew dbprop:author ?who .
   ?WORK dbprop:author ?who .
   OPTIONAL { ?WORK dbprop:genre ?genre } .
}
```

Listing 3.5: Esempio di una query SPARQL interrogando il dataset sulla serie di manga giapponese *Tokyo Mew Mew* per trovare i generi di altre opere scritte dalla sua illustratrice Mia Ikumi.

DBpedia ha una vasta gamma di entità che coprono diverse aree della conoscenza. Questo lo rende uno strumento naturale e importantissimo per la connessione di fonti di dati, dove i fonti di dati esterni potrebbero collegarsi ai suoi concetti. Il set di dati DBpedia è interconnesso a livello RDF con vari altri set di dati Open Data sul Web, consentendo alle applicazioni di arricchire le informazioni di DBpedia.

3.4.1.2 GeoNames

Fondato nel 2005, il progetto GeoNames è un database geografico modificabile e accessibile dagli utenti tramite vari servizi web. Sotto Creative Commons, gli utenti del set di dati Geonames hanno accesso a 25 milioni di entità geografiche. Tutte le informazioni sono classificate e raggruppate in nove classi e ulteriormente sotto-categorizzati in base a 645 codici.

Ogni entità di GeoNames è rappresentata come una risorsa web identificata da un URI in modo da fornire accesso attraverso la navigazione del contenuto, sia alla pagina wiki HTML, sia a una descrizione RDF della caratteristica, utilizzando elementi dell'ontologia GeoNames. Questa ontologia descrive le proprietà delle caratteristiche di GeoNames utilizzando il linguaggio dell'ontologia Web, le classi di caratteristiche e i codici descritti nel linguaggio SKOS. Attraverso l'URL degli articoli di Wikipedia collegati nelle

descrizioni RDF, i dati di GeoNames sono collegati ai dati di DBpedia e ad altri dati collegati a RDF.

3.4.1.3 WordNet

Il progetto Wordnet è uno dei database lessicali più conosciuti per le lingue, che fornisce definizioni e sinonimi, spesso utilizzato per migliorare le prestazioni della PNL e delle applicazioni di ricerca. WordNet è un database lessicale di relazioni semantiche tra parole in più di 200 lingue, collegando le parole in relazioni semantiche inclusi sinonimi e iponimi. I sinonimi sono raggruppati in synset con brevi definizioni ed esempi di utilizzo. WordNet può quindi essere visto come una combinazione ed estensione di un dizionario e di un thesaurus.

Sebbene sia accessibile agli utenti umani tramite un browser web, il suo uso principale è nell'analisi automatica del testo e nelle applicazioni di intelligenza artificiale. WordNet è stato creato per la prima volta in lingua inglese e gli strumenti software sono stati rilasciati con una licenza BSD.

3.4.2 Come funziona un grafo di conoscenza?

Il knowledge graph è composto da un insieme di dataset provenienti da diversi fonti o sorgetti che utilizzano diversi formati e strutture per rappresentare le informazioni memorizzate. Gli schemi dei dati, gli identificatori e il contesto vengono mischiati insieme per fornire una struttura comune ai dati diversi.

I knowledge graph che vengono alimentati tramite machine learning effettuano un processo di arricchimento semantico per fornire una visone completa dei nodi, archi ed etichette. Quando una sorgente di dati fornisce informazioni, il processo consente l'identificazione dei singoli oggetti e la comprensione delle loro relazioni. La conoscenza generata e inferita tramite questo processo viene confrontata con altri dataset. Gli sistemi che attendono risposte delle interrogazioni fate, riceveranno il risultato appena il grafo della conoscenza è completo e consente le query. L'integrazione dei dati e dei risultati potrebbe supportare la creazione di nuova conoscenza definendo relazioni tra dati che prima sarebbe stato impossibile definire.

3.4.3 Definizione di un grafo di conoscenza

Come già menzionato prima, un grafo della conoscenza è un grafo diretto etichettato costituito da nodi, archi ed etichette. Qualsiasi concetto potrebbe essere rappresentato come un nodo, ad esempio persone, aziende, eventi, luoghi, etc. Un arco effettua il collegamento tra una coppia di nodi e cattura la relazione d'interesse tra di loro, mentre le etichette catturano il significato della relazione.

Secondo la definizione formale, dato in un insieme di nodi N e un insieme di etichette L, un knowledge graph è un sottoinsieme del prodotto $N \times L \times N$. Ciascun elemento di questo insieme di dati viene rappresentato come una tripla.

Figura 3.6: Visualizzazione di due nodi connessi del grafo

Un grafo come quello sopraindicato viene chiamato un grafo di dati. Nel caso i nodi rappresentano classe di oggetti e gli archi catturano le relazioni dei sottoclassi, tale grafo viene chiamata tassonomia.

La navigazione sul grafo porta lo vantaggio di ridurre e risparmiare molti calcoli, che inizialmente sembrerebbero pesanti computazionalmente. In un knowledge graph della conoscenza che modella le amicizie, se vogliamo calcolare gli amici di un amico di una persona A, basterebbe navigare il grafo da A a tutti i nodi B collegati ad esso tramite una relazione etichettata *amico* e continuare la navigazione ricorsiva su tutti i nodi C, collegati a B tramite la stessa relazione.

Un *path* (percorso) in un grafo di conoscenza G è un insieme di nodi $(v_1, v_2, ..., v_n)$ dove per ogni $i \in N$ con $1 \leq i \leq n$, esiste un arco da v_1 a $v_i + 1$. Un percorso con nodi non ripetuti e distinti viene chiamato percorso semplice, mentre se il primo e l'ultimo nodo corrispondo viene chiamato percorso ciclico. Durante la navigare e l'attraversamento dei nodi è possibile definire numerose proprietà aggiuntive (ad esempio componenti connessi, fortemente connessi, etc).

3.4.4 Use case di knowledge graph

I grafi della conoscenza sono un meccanismo avanzato ed innovativo e negli ultimi anni ha trovato numerose applicazione nel mondo della ricerca e in diverse industrie. In seguito verranno spiegate le applicazione dei knowledge graph che hanno portato un aumento della sua popolarità recentemente.

3.4.4.1 Organizzare la conoscenza nella rete

L'utilizzo principale dei gradi della conoscenza è l'organizzazione delle informazioni in rete in modo da essere facilmente accessibili, manipolabili ed interrogabili. Wikidata è un esempio di come potrebbe essere strutturata la conoscenza in rete. Tale progetto offre un archivio centrale per i dati strutturati che si trovano nel Wikipedia.

Un esempio dell'utilità del Wikidata viene mostrato da una semplice ricerca su Wikipedia della città di Winterthur in Svizzera. Nella pagina Wiki di questa città vengono elencate alcune città gemelle che si trovano in Europa. Anche se non era specificato nella pagina di Winterthur, nella pagina Wiki di Ontario in California viene specificato la città Winterthur come una città sorella. Per gli utenti le relazioni sorella e gemella vengono riconosciute come uguali, ma per il sistema no. Al contrasto di Wikipedia, la rappresentazione di Wikidata mappa queste due relazioni come simili e siccome l'inferenza della informazioni è automatica, Ontario e Winterthur sono città gemelle e specificate nelle loro istanze in Wikidata.

Il grafo della conoscenza di Wikidata è il più grande grafo della conoscenza disponibile oggi. Molti dati in Wikidata possono provenire da informazioni estratte automaticamente, ma devono essere facilmente compresi e verificati secondo le politiche editoriali di Wikidata. Il progetto di Wikidata si focalizza esplicitamente sulla definizione semantica di diversi nomi di relazioni utilizzando una varietà di vocabolari. In questo modo Wikidata migliora l'esperienza di navigazione e di ricerca per gli utenti sul web.

3.4.4.2 Integrazione dei dati nelle industrie

L'integrazione dei dati è un processo di combinazione di dati provenienti da vari fonti, fornendo all'utente una visone strutturata, unificata e chiara dei dati. In diverse industrie, i dati vengono gestiti e memorizzati utilizzando diversi meccanismi di persistenza, aumentando il rischio e i costi. La progettazione e l'implementazione di uno schema globale condiviso da tutti per la

modellazione dei dati è un processo difficile. Gli ricercatori si sono focalizzati sulla risoluzione del problema seguendo un approccio diverso, conversione dei basi di dati in uno schema generico di triple. L'accumulazione di tutte le triple permetterebbe la creazione di un grafo di conoscenza.

3.4.4.3 Intelligenza artificiale

Sin dall'inizio, i knowledge graph sono stati utilizzati come rappresentazione dell'intelligenza artificiale. Negli ultimi anni i grafi della conoscenza sono stati rappresentati tramite i grafi concettuali, le logiche descrittive e linguaggi di regole.

Le due sfide principali dell'intelligenza artificiale sono la rappresentazione della conoscenza e l'acquisizione. sua catturazione tramite la rappresentazione scelta. Il processo dell'acquisizione consiste nella catturazione tramite la rappresentazione scelta in modo semplice e scalabile. Le tecniche utilizzate nei giorni d'oggi si basano sull'apprendimento induttivo e la generazione automatica dell'apprendimento.

I knowledge graph vengono utilizzati per la memorizzare e la visualizzazione delle conoscenze automaticamente apprese. La visualizzazione delle informazioni servirebbe per migliorare il processo dell'apprendimento automatico.

3.4.4.4 Input e output di machine learning

3.4.4.4.1 Input

I modelli di machine learning si basano maggiormente su un input numerico e richiede che qualsiasi struttura di dato potrebbe essere convertita in una rappresentazione numerica. La rappresentazione numerica applicata sull'input testuale, chiamata word embedding, migliora notevolmente le prestazioni dei task di *natual language processing* includendo l'estrazione delle entità e delle classi, il parsing, etc.

Word embedding viene utilizzato nel mondo dell'web semantico per autocompletare le interrogazioni di ricerca, prevedendo le parole che probabilmente seguiranno la query parziale che l'utente avrebbe già digitato. Calcolando il numero delle occorrenze di una parola nel testo, è possibile rappresentare il testo come un grafo di conoscenza dove ogni parola è un nodo e tra due parole consecutive si trova un arco etichettato. L'obbiettivo è di rappresentare

ogni nodo del grafo tramite un vettore, in modo che le similarità tra i nodi potranno essere calcolate come le differenze dei loro vettori. Questo processo viene chiamato come *graph embeddings*.

In modo da calcolare i graph embeddings per ogni nodo del grafo, viene definito un metodo di encoding che consiste in una funzione che calcola le similarità tra i nodi e poi applica un algoritmo di ottimizzazione. Il processo di encoding di un nodo viene chiamato come *node embedding*.

3.4.4.4.2 Output

I knowledge graph vengono utilizzati come rappresentazione dell'output target per l'elaborazione del linguaggio naturale e gli algoritmi di visione artificiale. L'estrazione di entità e l'estrazione di relazioni dal testo sono due obbiettivi fondamentali nell'elaborazione del linguaggio naturale. I grafi della conoscenza forniscono un mezzo naturale per estrarre informazioni correlate da più parti del testo.

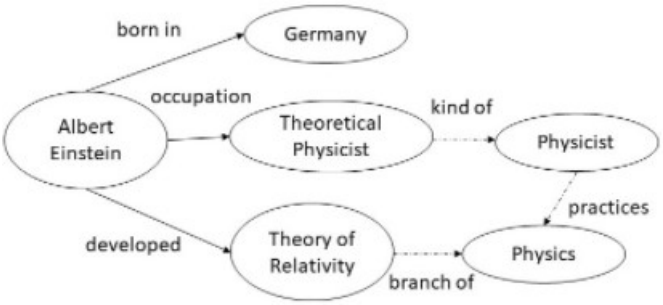

Figura 3.7: *Albert Einstein was a German-born theoretical physicist who developed the theory of relativity.* Passando come input l'informazione precedente, possiamo estrarre i concetti della frase e rappresentarle semanticamente costruendo un grafo.

Lo scopo della visone artificiale è la comprensione di un'immagine creando un modello in grado di rilevare e descrivere gli oggetti presenti specificando le relazioni.

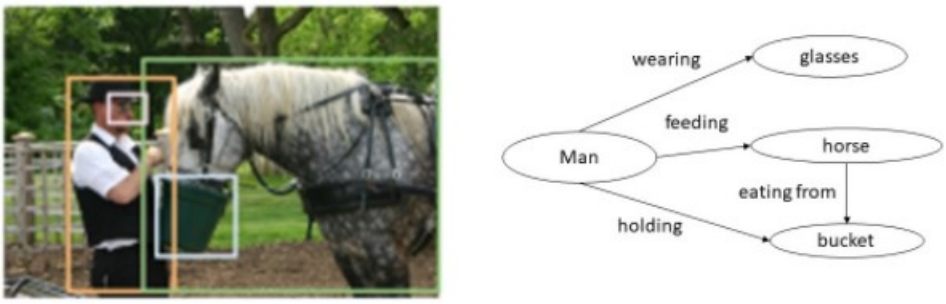

Figura 3.8: Esempio di un sistema di comprensione delle immagini che produce un grafo della conoscenza, dove i nodi sono gli output di un rilevatore di oggetti e gli archi le relazioni tra gli oggetti.

3.5 Linked Data

Il progetto Linked Data consiste in un insieme di principi di progettazione per la condivisione di dati machine-readable sul Web. La fusione di tale progetto con Open Data si chiama LOD (*Linked Open Data*). LOD è in grado di gestire enormi set di dati provenienti da fonti diversi e collegarli agli Open Data, il che aumenta la scoperta della conoscenza e fornisce strumenti per un analisi efficiente basata sui dati.

Il Web semantico ha lo scopo di offrire meccanismi per la creazione di collegamenti tra set di dati comprensibili non solo per gli esseri umani, ma anche per le macchine. I Linked Data forniscono le migliori pratiche per rendere possibili questi collegamenti. In altre parole, i Linked Data sono un insieme di principi di progettazione per la condivisione di dati interconnessi leggibili dalla macchina sul Web.

Quando parliamo di Web of Data, è fondamentale avere a disposizione un enorme quantità di dati in un formato standard, raggiungibile e gestibile dagli strumenti del Web semantico. Le relazioni tra i dati dovrebbero essere rese disponibili per creare un Web of Data, diversamente da una semplice raccolta di set di dati. Questa raccolta di set di dati interconnessi sul Web può anche essere indicata come Linked Data.

Per ottenere e creare Linked Data, dovrebbero essere disponibili tecnologie per un formato comune come RDF, in modo da effettuare la conversione e l'accesso al volo a database esistenti (relazionali, XML, HTML, ecc.). La configurazione degli endpoint delle interrogazioni è importantissima per poter accedere a tali dati velocemente e in modo più conveniente. W3C fornisce

una gamma di tecnologie (RDF, GRDDL, POWDER, RDFa, il prossimo R2RML, RIF, SPARQL) per accedere ai dati.

Capitolo 4

Analisi

4.1 Scopo del progetto

Pathadora ha come obbiettivo finale la generazione dei learning paths dinamici ed efficienti basandosi sugli parametri iniziali specificati dall'utente e dalle relazioni generate tra le istanze del sistema.

4.1.1 Modellazione ontologica

Pathadora deve modellare tramite gli strumenti del web semantico diversi domini tra i quali:

- **l'istituzione accademica**: l'organizzazione dell'università in scuole, dipartimenti, facoltà e corsi. La modellazione dell'istituzione tramite una gerarchia permetterebbe di generare una sequenza si dipartimenti, facoltà e corsi tendono in considerazione le associazioni tra tali componenti. Le informazioni dichiarati dallo studente, permettono al sistema di iniziare la ricerca in cima della gerarchia, ovvero le scuole per poi scendere step-by-step fino ai corsi.
- **le informazioni dello studente**: tale informazioni possono essere:
  - generali: informazioni che specificano i dati anagrafici, le passioni, le lingue che parla, etc;
  - personali: informazioni sulle disabilità, sugli obbiettivi, sul metodo di apprendimento, etc;
  - accademici: informazioni che enfatizzano gli obbiettivi raggiunti durante il suo percorso accademico.

- le risorse didattiche: modellazione dei materiali didattici focalizzando sulle informazioni che riguardano l'accessibilità della risorsa e il topic del corso associato.

- l'accessibilità delle risorse; rappresenta l'accessibilità come un macro concetto e si focalizza sulla modellazione di informazioni sulla accessibilità che possono essere utilizzati per usufruire diversi tipi di risorse con diverse caratteristiche. L'ontologia mappa metodi di accessibilità di una risorsa e metodi di conversione di una risorsa in modo da offrire un altra soluzione di accessibilità.

4.1.2 Rule and query base model

La logica del Pathadora si dovrebbe basare sulle regole in modo da creare un sistema *rule-based*. Tale sistema sarebbe in grado di memorizzare e manipolare la conoscenza e interpretare le informazioni in diversi modi. Pathadora utilizzerà questo sistema per fare deduzioni o scelte durante la generazione dei learning path basandosi sulle informazioni inferite dall'applicazione delle regole.

4.1.3 Pathadora engine

Pathadora-engine sarà una engine sempre in esecuzione che servirà le richieste ricevute da parte dell'utente. Tale engine offrirà i meccanismi necessari per interrogare la knowledge dell'ontologia.

4.2 Analisi dei requisiti

In questa sezione verranno spiegati i requisiti funzionali e non funzionali del progetto.

4.2.1 Requisiti funzionali

Il sistema finale del progetto dovrà soddisfare i seguenti requisiti funzionali:

- le ontologie:
  - pathadora-ontology: L'ontologia che modellerà tutto il dominio del sistema, racchiudendo i concetti dell'organizzazione dell'università, dei studenti, delle risorse e la loro accessibilità;

  - accesible: Sincronizzare l'ontologia dell'accessibilità con l'ontologia del sistema, pathadora-ontology;
  - lom: Sincronizzare l'ontologia che modella i metadati per le risorse didattiche con l'ontologia del sistema.

- gestire le richieste: L'utente potrà interagire con il sistema mandando delle richieste dal interfaccia web e il sistema dovrà essere in grado di accettare e servire queste richieste;

- manipolazione delle ontologie: Il sistema dovrà offrire meccanismi di manipolazione delle ontologie con l'inserimento di nuove istanze oppure i loro aggiornamento;

- manipolazione delle regole: L'utente deve essere in grado di personalizzare e parametrizzare le regole da applicare sulle ontologie presenti;

- manipolare le interrogazioni: L'utente deve essere in grado di manipolare le interrogazioni che verranno fatte al sistema di persistenza;

- Pathadora Recommender: Progettazione dell'engine che racchiude tutti i componenti del sistema.

4.2.2 Requisiti non funzionali

Il sistema finale del progetto dovrà soddisfare i seguenti requisiti non funzionali:

- la prestazione: Il sistema deve essere in grado di gestire la richiesta e produrre una risposta nel minor tempo possibile, sfruttando al massimo le risorse computazionali messe a disposizione;

- la scalabilità: A livello di codice il sistema deve essere facilmente estendibile se sarebbe necessario l'introduzione di nuovi componenti;

- la disponibilità: Pathadora deve essere sempre in ascolto e servire le richieste, definendo tecniche di fault-tolerance;

- la consistenza: Il sistema deve essere in grado di mantenere la consistenza nel caso di inserimento o aggiornamento della knowledge, verificando lo stato degli strumenti utilizzati per la persistenza.

Capitolo 5

Progettazione

Pathadora è un sistema di raccomandazione di facoltà e corsi da seguire da uno studente in base alle caratteristiche specificate. Il sistema oltre alle facoltà e corsi, genera una sequenza di risorse didattiche da consigliare per un determinato corso scelto. Il learning path delle risorse dipenderà principalmente dal livello di accessibilità delle risorse e dalle eventuali disabilità dichiarate dello studente.

Pathadora gestisce le caratteristiche che riguardano gli studenti, le facoltà, i corsi e le risorse tramite la modellazione ontologica utilizzando strumenti e meccanismi del web semantico. Sfruttando tali strumenti, Pathadora è in grado di produrre un risultato dinamico e aggiornabile con nuove entità o caratteristiche che vengono aggiunte nel sistema. La generazione di relazioni tra le entità del sistema offre la possibilità di ottenere percorsi non uniformi e uguali per tutti, aumentando l'efficacia del sistema.

Gli utenti possono interagire con il sistema, tramite l'interfaccia web che Pathadora offre in modo da catturare e servire le richieste. *Pathadora-engine* si occupa della gestione degli strumenti ontologici del sistema e la produzione dei learning paths e la verifica di tali risultati.

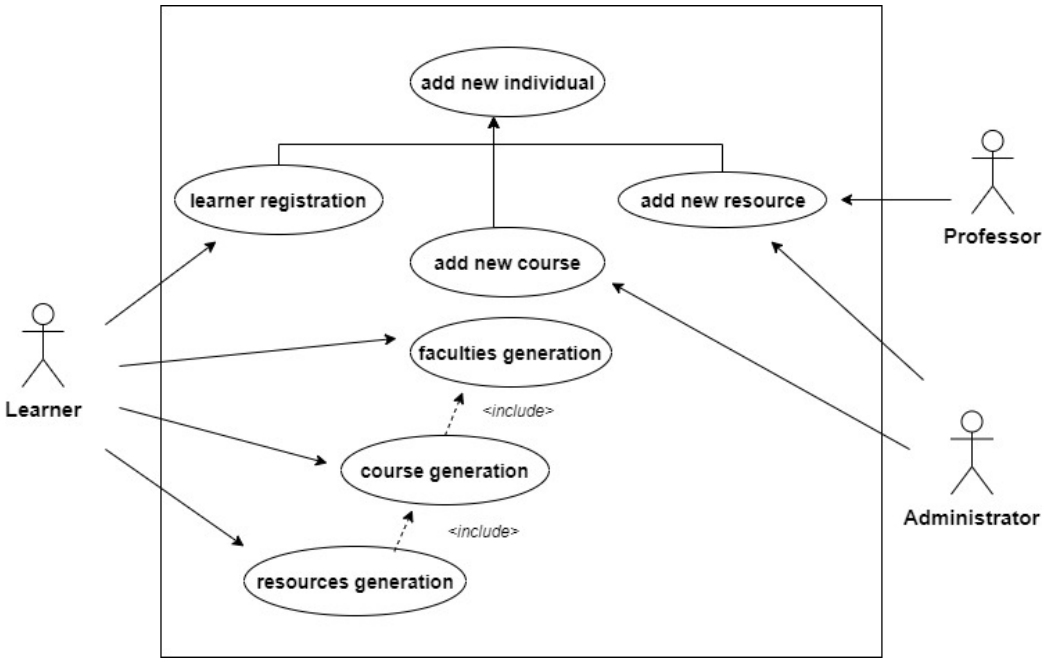

Figura 5.1: Pathadora Recommender Engine: Use case diagram

Uno studente tramite l'interfaccia web potrebbe richiedere la sua registrazione, fornendo le informazioni che dovranno essere inserite successivamente nell'ontologia. Lui potrebbe richiedere la generazione delle facoltà, dei corsi e delle risorse. Le richieste che potrà mandare dovranno rispettare la sequenzialità logica.

Un docente potrebbe inserire le risorse associate ad un determinato corso. Nella sua richiesta dovrà specificare le caratteristiche associate a tale risorsa, focalizzandosi sul livello di accessibilità.

Un amministratore del sistema ha i permessi di eseguire qualsiasi richiesta di inserimento a gestire, apprate quella dello studente. Un amministratore ha una visione completa del sistema e delle informazioni inserite. A tale ruolo è associata la gestione d'autenticazione degli utenti. Eventuali problematiche con i dati e i permessi nel sistema vengono risolte manualmente da parte dell'amministratore.

5.1 Architettura

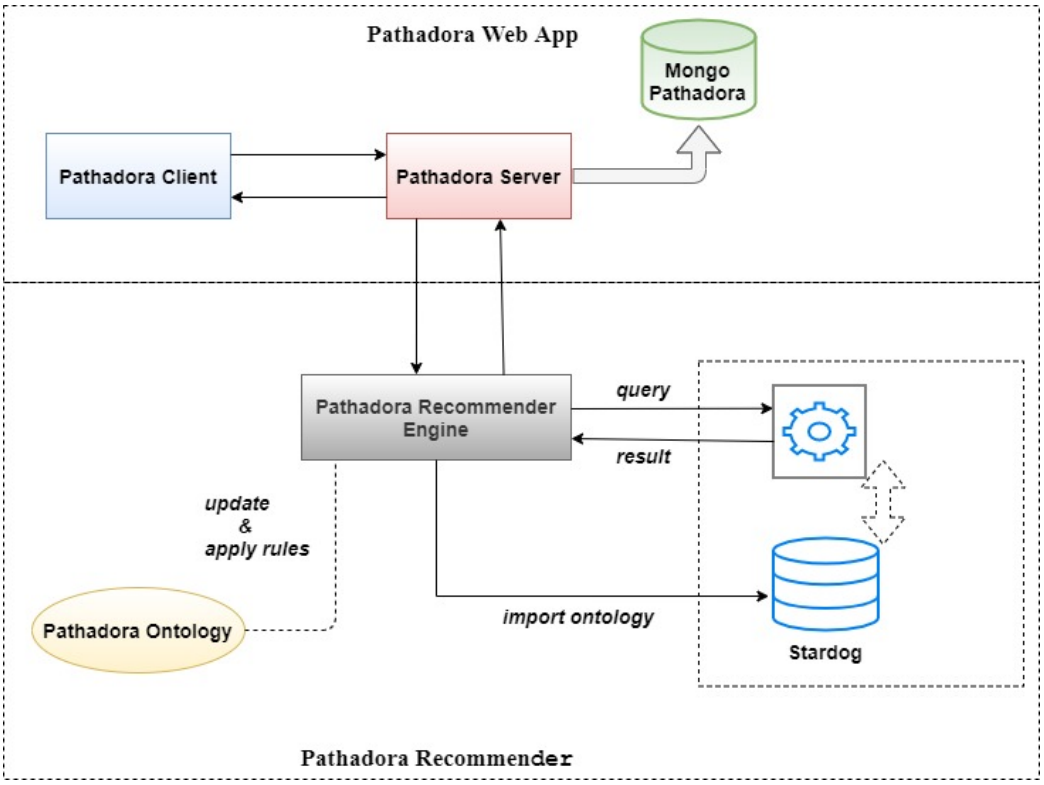

Figura 5.2: Architettura di Pathadora

La rappresentazione semantica della conoscenza è stata considerata come il fondamenta della soluzione del problema. Nei capitoli precedenti si è parlato dei vantaggi che gli strumenti del web semantico hanno portato nella gestione e manipolazione di informazioni. Il sistema di raccomandazione Pathadora, si doveva focalizzare nell'utilizzo degli strumenti semantici per la modellazione della conoscenza e l'aggregazione del risultato. Le relazioni semantiche tra i componenti del sistema sono state considerate il core del progetto e per questo motivo *pathadora-ontology* mirava la rappresentazione della conoscenza utilizzando tale ontologia. La conoscenza del grafo includeva qualsiasi informazione indipendentemente dal formato, dal sorgente o dalla correttezza.

La manipolazione della conoscenza viene effettuata utilizzando un insieme di regole e interrogazioni personalizzate in base alla richiesta dell'utente. Il sistema utilizza la potenza del reasoning nell'inferire informazioni sconosciute per ottenere risultati corretti e completi. Un insieme di regole SWRL

e interrogazioni SPARQL applicata sul grafo della conoscenza del sistema permette di generare i learning path, estraendo le relazioni tra le entità che rappresentano l'informazione richiesta.

Il sistema Pathadora consiste in due macro concetti principali, *Pathadora Web App* e *Pathadora Recommender*. Pathadora Web App offre un'interfaccia web permettendo all'utente di interagire con il sistema e visualizzare le risposte ottenute. L'applicazione web è composta da tre componenti importanti: il client, il server e il database.

Pathadora Ontology è un componente del sistema che ha come obbiettivo di gestire e manipolare l'ontologia utilizzata per rappresentare il dominio del progetto. Tale componente dovrebbe interagisce con l'engine quando richieste di aggiornamento o interrogazione vengono fatte. La gestione dell'ontologia racchiude operazioni di inserimento e aggiornamento di istanze presenti o non presenti nel grafo della conoscenza. Il sistema dovrebbe permettere l'inserimento di nuova informazione, anche nei casi quando l'informazione non è completa oppure non è nel formato giusto.

Pathadora Rule Model ha il compito di parametrizzare le regole ed effettuare la loro applicazione, notificando l'engine sull'andamento di tale operazione. Questa attività viene eseguita con l'inizializzazione del Pathadora Recommender e in quel momento il sistema è pronto a ricevere richieste di interrogazione.

Per connettere le informazioni a livello di elaborazione e non di archiviazione, è stata utilizzata una piattaforma esterna, Stardog. Stardog è un database che ospita grafi di conoscenza, permettendo di interrogare e manipolare una grande quantità d'informazione. Tale componente ha l'obbligo di memorizzare il knowledge graph del sistema e preparare le risposte delle richieste fatte.

Pathadora Recommender racchiude tutta la modellazione e la logica dietro la generazione dei learning path. Questa engine rimane sempre in esecuzione ad intercettare le richieste e inoltrale al componente che ha il compito di gestirle.

5.2 Gestione delle iterazioni e richieste

Il sistema dovrà gestire le richieste che arrivano da parte del cliente e istanziare il componente adeguato a computare la risposta.

5.2.1 Inserimento

La prima tipologia di richiesta da gestire è l'inserimento di nuove istanze nell'ontologia. In questa richiesta dovranno essere specificate tutte le informazioni che serviranno all'engine per aggiornare la knowledge graph. La richiesta potrebbe dichiarare relazioni tra elementi che non sono presenti e in tal caso, l'engine prima creerà questi elementi per poi definire la relazione tra loro. Nel caso della richiesta di inserimento tutti gli elementi verranno aggiunti nell'ontologia, indipendentemente se esistono, se rispettano il formato giuso oppure se le relazioni sono semanticamente corrette.

L'utente, in base al ruolo, potrebbe inserire nell'ontologia diversi tipi di individui, ad esempio le informazioni di uno studente, di un corso oppure di una risorsa. Nella richiesta l'utente specificherà le caratteristiche dell'entità che devono essere modellate e inserite nel grafo della conoscenza. Nel caso diversi utenti si riferiscono alla stessa risorsa, il sistema andrà ad aggiornare le proprietà di quella esistente senza produrre duplicati.

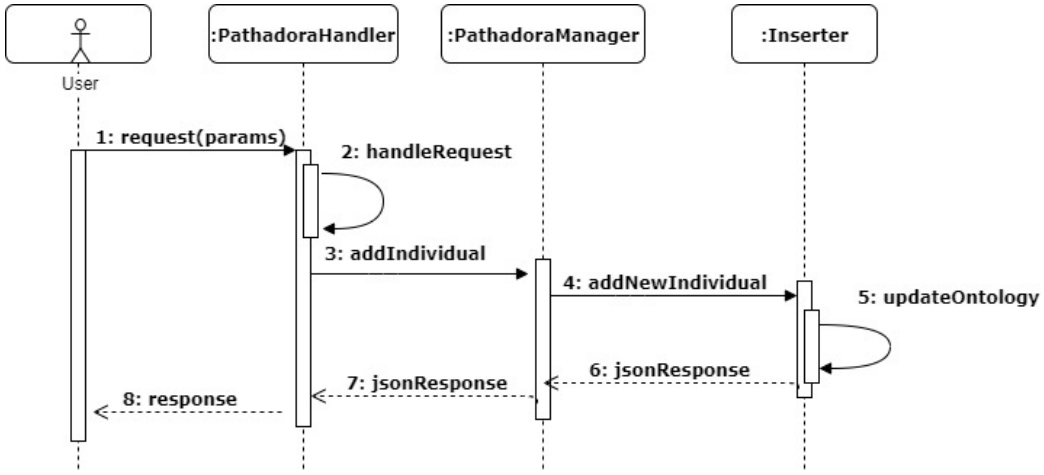

Figura 5.3: Diagramma di sequenza: Richiesta di inserimento di un individual

5.2.2 Generazione di facoltà

La seconda tipologia di richiesta consiste nella generazione del percorso delle facoltà. Lo studente nella richiesta di generazione dovrà specificare il tipo di diploma che vorrebbe conferire come parametro d'input per il sistema. L'aggregazione delle facoltà verrà fatta applicando una regola, parametrizzata con il tipo di diploma. Oltre al tipo di diploma, il reasoner utilizzerà tutte le informazioni e le proprietà dichiarate dallo studente durante la registrazione per aggregare una lista di facoltà.

Le informazioni specificate dall'utente aiuteranno il sistema a produrre un risultato corretto e congruente con le caratteristiche dello studente stesso. L'applicazione delle regole, implicherà il cambiamento dell'ontologia e della conoscenza, inferendo informazione che prima erano sconosciute. Alla fine di questa operazione, il sistema ottiene una conoscenza completa per quanto riguarda lo studente ed il sistema è pronto ad aspettare richieste d'interrogazione sui corsi e sulle risorse.

In un secondo tentativo di generare un elenco di facoltà da consigliare, lo studente potrebbe specificare nuovi parametri d'input oppure nuove caratteristiche, e otterrebbe un nuovo risultato, indipendentemente dai risultati precedentemente restituiti.

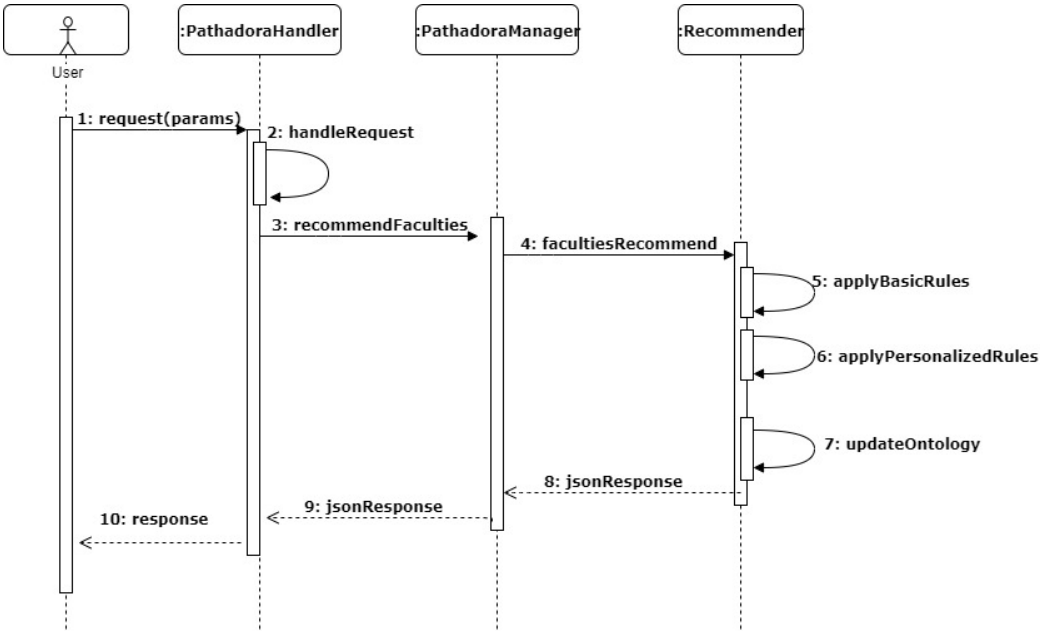

Figura 5.4: Diagramma di sequenza: Richiesta di raccomandazione di facoltà

5.2.3 Generazione di corsi

La generazione dei corsi, viene fatta sempre dopo la generazione delle facoltà. L'utente dovrà interagire con il sistema per fare delle scelte che serviranno per generare il risultato del prossimo step. La generazione del percorso di studio, step-by-step, necessita che lo studente scegliesse la facoltà preferita dall'elenco delle facoltà raccomandate. Fatta tale scelta, lo studente dovrà specificare anche l'anno dei corsi che vorrebbe ricevere come raccomandazione. Il sistema Pathadora, una volta ricevuto tale scelta, inizializzerà il knowledge graph database e lo interrogherà parametrizzando la query con le informazioni dello studente.

Il database avrà come grafo di conoscenza la versione più aggiornata dell'ontologia, avendo anche le informazioni generate dall'applicazione della regola SWRL delle facoltà. L'interrogazione del database sarebbe una query avendo come parametri l'input e le caratteristiche dello studente. Il database include anche le informazioni da diversi sorgenti, in questo caso diverse ontologie, inferendo nuova conoscenza e producendo un risultato più completo.

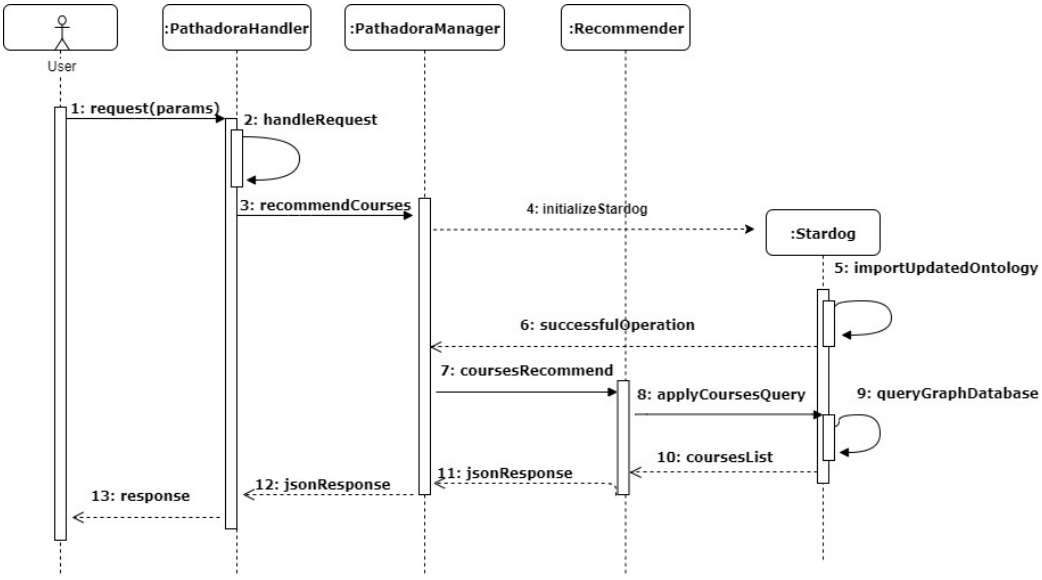

Figura 5.5: Diagramma di sequenza: Richiesta di raccomandazione di corsi

5.2.4 Generazione di risorse

L'ultima richiesta da gestire consiste nella raccomandazione delle risorse, focalizzandosi sull'accessibilità delle risorse e sulle disabilità dello studente. Per ogni risorsa verranno definite informazioni che rappresentano il livello di accessibilità utilizzando coefficienti che specificano tale livello.

Lo studente durante la registrazione definirà le disabilità, specificando un livello a scala numerica. Pathadora-engine dovrebbe mappare la correlazione tra questi coefficienti di disabilità e di accessibilità per aggregare una lista di risorse da raccomandare allo studente. Oltre alla accessibilità, la raccomandazione delle risorse si focalizzerà sulle caratteristiche dello studente come stile di apprendimento, obbiettivi, etc. Anche in questo caso, la richiesta della generazione delle risorse, dovrà essere fatta dopo la generazione dei corsi, in modo che lo studente scegliesse il corso per il quale vorrebbe ricevere una lista di risorse.

Il database che ospita il grafo della conoscenza conterrà una versione completa ed aggiornata dell'ontologia, modificata e manipolata dagli step precedenti. L'utente otterrebbe diversi percorsi di apprendimento se per le richieste successive utilizzasse diversi parametri di personalizzazione delle interrogazioni.

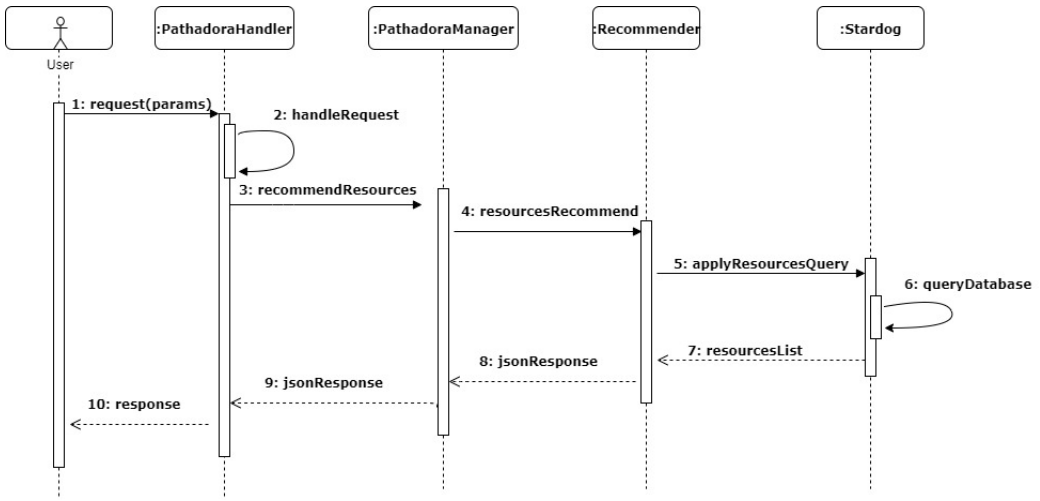

Figura 5.6: Diagramma di sequenza: Richiesta di raccomandazione di risorse

5.2.5 La gestione di una richiesta

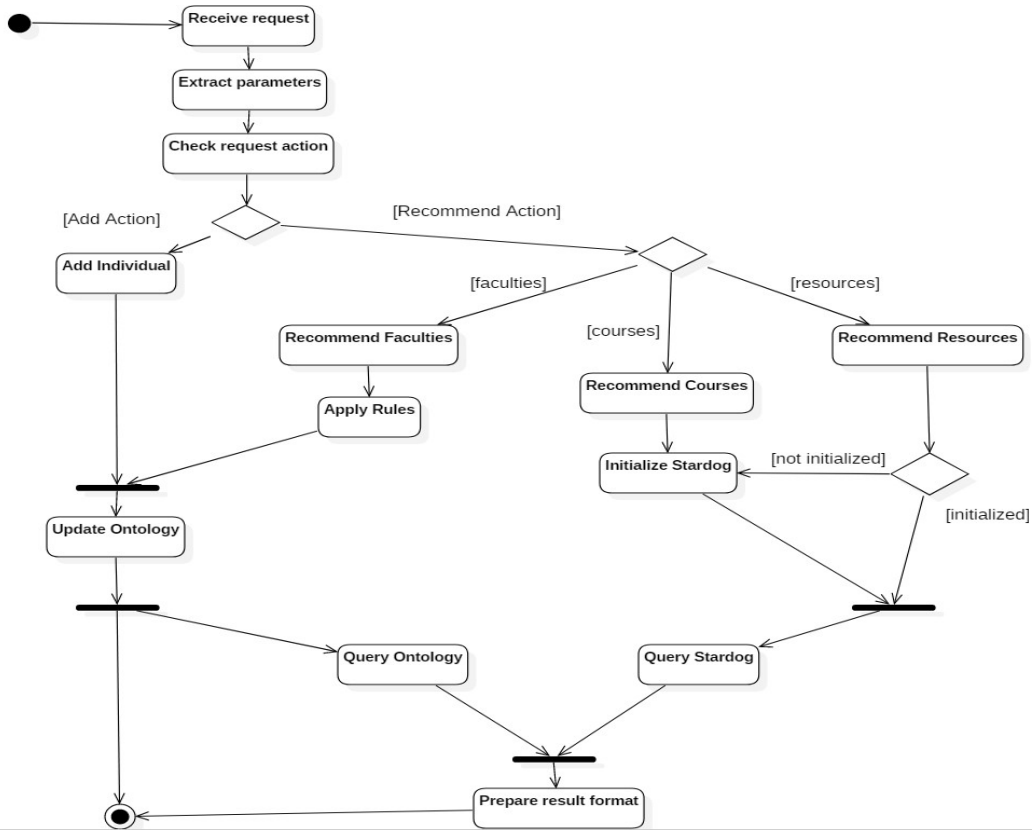

Figura 5.7: Diagramma di attività: La gestione di una richiesta

Come mostrato nel diagramma sopraelencato, appena riceve una richiesta il sistema dovrebbe estrarre i parametri passati come input. Lo step successivo consiste nel controllo dell'azione specificata nella richiesta: `add` o `recommend`, rappresentando rispettivamente un azione di inserimento o di generazione dei percorsi da raccomandare.

Nel caso di una azione `add`, il sistema inserisce l'individuo e aggiorna l'ontologia. Nel caso di una azione `recommend`, il sistema dovrebbe aggregare il concetto per qui l'utente sta richiedendo la raccomandazione, ad esempio facoltà, corsi oppure risorse. Se le facoltà vengono richieste, il sistema applica le regole personalizzate e come step finale aggiorna l'ontologia. Se i corsi o le risorse vengono richieste, il sistema dovrebbe interagire con il database del grafo della conoscenza per effettuare le interrogazioni. Il risultato sarebbe convertito in un formato specifico richiesto dal client.

5.2.6 Il sistema Pathadora

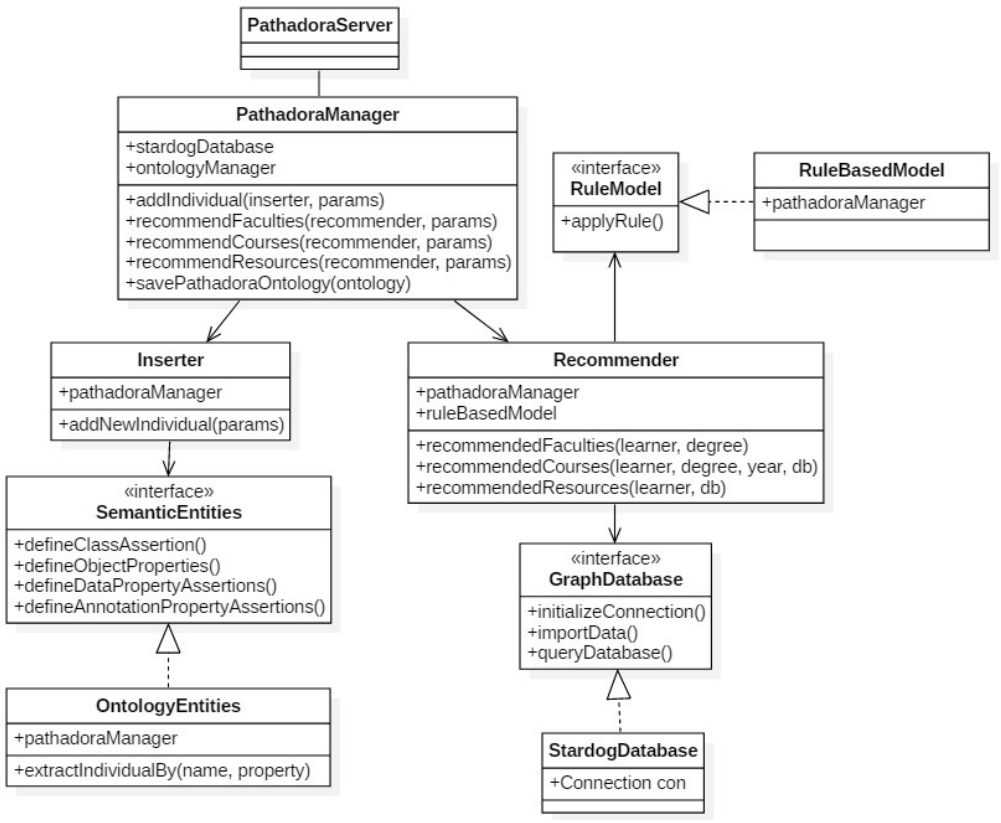

Figura 5.8: Diagramma delle classi: Pathadora Recommender

Il diagramma delle classi descrive i componenti principali del sistema focalizzandosi su `PathadoraManager`, come entità più importante. Tale entità gestisce tutte le richieste che vengono effettuate dall'utente, tenendo traccia dell'ontologia e del database di knowledge graph. Qualsiasi informazione viene passata al sistema tramite questo componente, che prima di richieder la gestione, controlla la correttezza dell'informazione e della richiesta.

`PathadoraManager` garantisce al sistema l'inizializzazione corretta dei componenti utili per computare il risultato desiderabile. `PathadoraManager` inoltra le informazioni necessari a tali componenti, garantendo la correttezza e la completezza di questi parametri. `PathadoraManager` utilizza `Inserter` e `Recommander` per computare la risposta e il risultato delle richieste.

`Inserter` ha come scopo di gestire le richieste `add` e di inserire gli individui nella conoscenza del sistema. Per poter svolgere questa funzionalità, utilizza `SemanticEntities` che gli offre un API in modo da manipolare l'ontologia. Alla fine dell'operazione, `Inserter` notifica il manager sull'andamento dell'operazione, e in caso positivo `PathadoraManager` salva l'ontologia.

`Recommender` ha come obbiettivo di gestire e computare il risultato di una richiesta di raccomandazione. In base al tipo di richiesta, tale componente utilizza diverse strategie per preparare il risultato. Nel caso le facoltà vengono richieste, `Recommender` utilizza un entità `RuleModel` che gli offre la possibilità di applicare le regole SWRL. Anche in questo caso, `Recommender` notifica `PathadoraManager` sull'andamento di questa operazione, per poi proseguire con l'aggiornamento dell'ontologia. Nel caso i corsi o le risorse vengono richieste, `Recommender` interagisce con `GraphDatabase`, in questo caso `StardogDatabase`, per importare la versione più recente della conoscenza e per effettuare le interrogazioni.

Siccome il sistema gestisce diversi fonti di informazioni, in diversi formati, `Pathadora` utilizza diversi parse per convertire l'input e l'output in una versione uniforme e strutturata. `DataToOwl` permette di convertire le informazioni in un formato owl, leggibile dall'ontologia. `OutputToJson` permette di parsare il risultato ottenuto dalla gestione delle richieste, in un formato `JSON`, utilizzabile dal lato server.

`CourseScaper` è uno script progetto per estrarre le informazioni che riguardano i corsi dell'Università di Bologna. Tale componente funziona in modalità semi-automatica, siccome unibo non utilizza lo stesso formato di pagina web per rappresentare i corsi di diverse facoltà. La correttezza dell'informazione estratta viene verificata manualmente step-by-step durante l'esecuzione del processo.

Il sistema memorizzerà localmente una versione base dell'ontologia che servirà per modellare la conoscenza, senza inferire o dedurre informazioni. Inizialmente l'ontologia è stata modellata utilizzando tool esterne e poi è stata inserita come risorsa locale del sistema. La versione completa che rappresenta il knowledge graph viene gestita dal sistema di persistenza dei grafi, che permette la manipolazione e l'interrogazione.

5.3 Progettazione del knowledge graph

Per poter organizzare ed esprimere in modo logico le relazioni semantiche tra gli elementi del dominio, è stato progettato il grafo della conoscenza tramite la modellazione semantica delle informazioni. Le informazioni sono state memorizzate in un database a grafo e visualizzate come una struttura a grafi. La modellazione dell'ontologia Pathadora definisce un formato di rappresentatine delle istanze del knowledge graph.

La conoscenza viene rappresentata da un knowledge framework multidimensionale. Ogni concetto del dominio è stato modellato tramite le classi e le sottoclassi per le informazioni più dettagliate. La modellazione multidimensionale permette di aggregare le informazioni a diversi livelli di dettaglio.

5.3.1 L'università

L'università come istituzione è stata modellata tramite una gerarchia a tre livelli. In cima di questa gerarchia sono le scuole, per poi scendere fino ai corsi. Utilizzando la gerarchia e una organizzazione a livelli, è più facile mappare le preferenze di uno studente con determinato academic path. In questo modo, se si ottene una scuola che soddisfa alcune caratteristiche dello studente, allora si assicura che scendendo sono questo ramo, i risultati ottenuti continuano a soddisfare le sue preferenze.

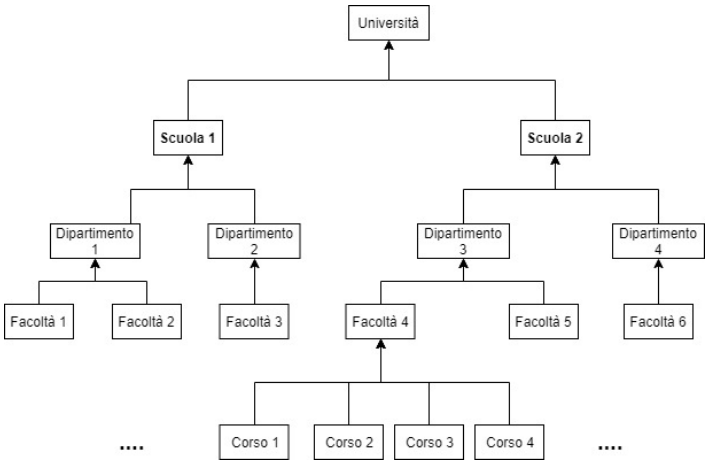

Figura 5.9: Organizzazione dell'università

Basandosi sull'organizzazione ufficiale dell'Università di Bologna, nell'ontologia sono state modellate 11 diversi tipi di scuole, specificando i dipartimenti per ogni tipo di scuola. Ad ogni dipartimento è stato specificato l'area scientifica a cui appartiene, siccome è una informazione importante per la raccomandazione delle facoltà. Le facoltà sono organizzate come individui di sottoclassi in base al tipo di diploma da conferire. Il diploma, come menzionato prima, è un paramento fondamentale per il sistema di raccomandazione e l'ontologia modella nei minimi dettagli tale elemento. Ad esempio `Bachelor DISTAL` rappresenta tutte le facoltà triennali del dipartimento `Agricultural and Food Sciences DISTAL`. Tale livello di dettaglio semplifica l'aggregazione dei percorsi da raccomandare. Per ogni facoltà vengono modellate informazioni che riguardano le caratteristiche specifiche come la lingua, tipo di facoltà, etc.

La modellazione semantica di un corso consiste nell'inserimento di proprietà che rappresentano informazioni dettagliate sul corso. Per ogni corso vengono considerate le informazioni che riguardano l'anno del corso, il periodo dello svolgimento, il numero dei crediti, il tipo di corso, obbligatorio oppure opzionale, l'area scientifica, etc. I corsi vengono specificati come individui di una classe che rappresenta una facoltà, permettendo di scendere la gerarchia sotto lo stesso ramo. Le associazioni tra dipartimenti, facoltà e corsi vengono modellate tramite le proprietà.

L'ontologia, oltre alla organizzazione strutturale dell'università, modella anche informazione che riguardano il mondo accademico. `CourseType` è una classe che modella i tipi dei corsi di una certa facoltà, mentre `Area` modella l'area a cui appartiene una scuola. `CourseSSD` modella il settore scientifico di un corso.

Nel caso i dati da inserire non sono completati, il sistema Pathadora inserirà l'informazione utilizzando valori di default definiti a priori.

5.3.2 Lo studente

La modellazione delle informazioni dello studente si basa su una knowledge graph. Per rappresentare le relazioni tra i diversi concetti in base al livello di dettaglio, le informazioni dell'utente sono state divise in tre classi:

- *General*: La rappresentazione delle informazioni generali si focalizza sulla modellazione di concetti che riguardano le passioni, le lingue e i dati anagrafici, i quali servono solo a livello di annotazione dell'entità studente. `Passion` è la classe che modella le passioni, avendo come sottoclassi un elenco di possibili valori associati a tale classe. `Language` modella un elenco di lingue che uno studente potrebbe parlare, specificando il livello di conoscenza tramite gli individui della classe `VocabolarySkills`.

- *Personal*: Le informazioni personali descrivono informazioni che riguardano le disabilità, gli obbiettivi e lo stile d'apprendimento dello studente. La classe `Disabilities` rappresenta tre diversi tipi di disabilità che uno studente potrebbe specificare: `Intellectual`, `Physical` e `Sensory`. La classe `Goals` modella diversi tipi di obbiettivi, che verranno utilizzato per produrre una percorso di apprendimento adeguato per lo studente.

- *Accademic*: Le informazioni accademiche rappresentano i risultati ottenuti a livello accademico da parte dello studente. `DegreeType` è una classe che modella i tipi di diplomi conferite dallo studente.

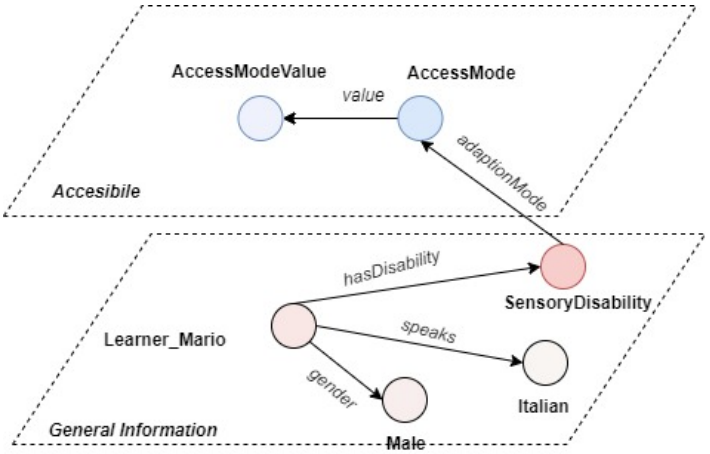

Figura 5.10: Knowledge graph sulle disabilità dello studente

5.3.3 Le risorse

La rappresentazione delle risorse consiste nella modellazione della classe `LearningObject`. Per ogni risorsa vengono specificate informazioni che descrivono il livello d'accessibilità di tale risorsa. Le informazioni consistono nel specificare *il font size*, *check ratio* oppure *reading ease*. Tutti questi tre parametri vengono aggregati tramite l'esecuzione di script, che prende in input una risorsa e calcola un coefficiente che rappresenta il valore di quel determinato parametro.

Le risorse vengono inserite nell'ontologia a richiesta di un docente e sarà lui a specificare le caratteristiche associate a tale risorsa. Oltre alle caratteristiche che riguardano l'accessibilità di una risorsa, verranno specificate proprietà che mettono il relazione le risorse con i corsi. In questo modo avremo una versione della conoscenza corretta, dove i componenti sono tutti connessi e in relazione.

5.3.4 L'accessibilità

Il concetto di accessibilità viene gestito focalizzandosi sull'ontologia importata nel sistema `accessible.owl`. L'ontologia permette la modellazione del livello di accessibilità di una risorsa digitale, basandosi sul tipo della risorsa e i possibili formati di conversione.

Durante la registrazione lo studente avrà già specificato le disabilità che potrebbe avere. Ad ogni disabilità viene definito un livello di disabilità, che servirebbe per mappare la disabilità con un coefficiente numerico. Come già spiegato nella sezione precedente, la stessa strategia viene effettuata anche sulle risorse. In questo modo si potrebbe definire una relazione tra le disabilità dello studente e l'accessibilità della risorsa.

Tenendo in considerazione le relazioni tra accessibilità e disabilità, si tende a consigliare ad allo studente una lista di risorse che potranno essere convertite in altri formati, avendo un altro livello di accessibilità. Le classi `AccessModeValue` e `AccessModeRequired` rappresentano le modalità di accesso ad una risorsa. I diversi tipi di conversione di una risorsa vengono modellati tramite la classe `AdaptationDetailValue`.

Capitolo 6

Implementazione

In questo capitolo verrà spiegato la fase d'implementazione del progetto e le problematiche incontrate durante lo sviluppo. Il capitolo inizierà con le fasi di sviluppo, poi successivamente si focalizzerà sulla descrizione delle tecnologie e tool utilizzate.

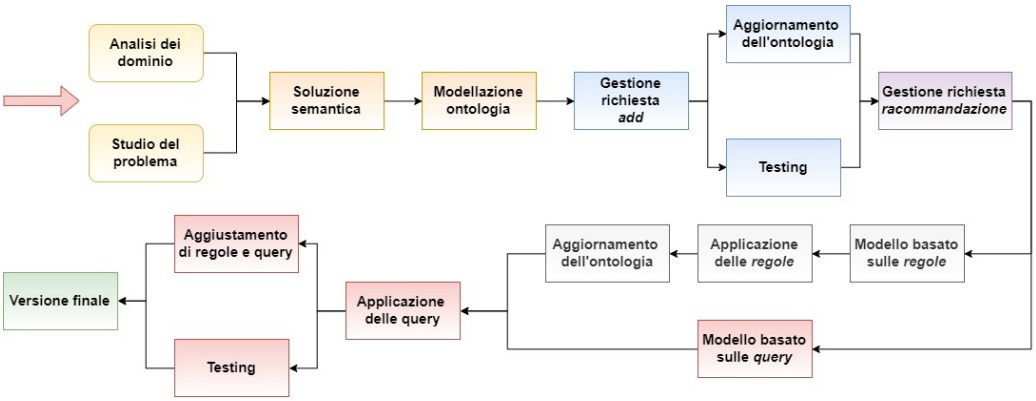

Figura 6.1: Road map delle fasi di sviluppo del progetto

6.1 Fase di sviluppo

Il progetto è iniziato con lo studio dettagliato dei requisiti funzionali e non funzionali del sistema, introducendo schemi e diagrammi da considerare durante la progettazione. All'inizio è stata effettuata un analisi dettagliata del dominio e del problema che il progetto mirava di risolvere.

Lo step successivo consisteva nella modellazione dell'ontologia Pathadora. Localmente sì è stato utilizzato il software *Protegè* per modellare e definire

l'ontologia nel formato `owl`. Questo file sarebbe la versione base dell'ontologia che andrebbe inserito manualmente nel sistema durante gli step successivi. L'aggiornamento dell'ontologia è stato continuo durante l'intera fase di sviluppo del progetto.

Inizialmente la modellazione dell'università non consisteva in una gerarchia dove in cima si trovarono le scuole e in fondo i corsi, ma tramite classi che si mettevano in relazione tramite la definizione delle proprietà. Gestendo l'organizzazione delle università tramite la gerarchia, permette di inferire nuove informazioni e facilitare la ricerca dei livelli sottostanti.

La modellazione degli studenti è evoluta e rifattorizzata, con la rimodellazione della struttura dell'università. La raccomandazione delle facoltà non sfruttava le informazione che riguardavano le scuole e i dipartimenti a cui appartivano, producendo un risultato non corretto. Per aiutare con la raccomandazione dei corsi e delle risorse, sono stati modellati nell'ontologia i concetti che riguardavano le passioni e lo stile d'apprendimento per lo studente.

Le risorse sono state gli ultimi componenti modellati tramite owl. Per effettuare la loro rappresentazione sono state importate diverse ontologie che sincronizzate con pathadora, permettevano di descrivere le risorse focalizzandosi sull'accessibilità.

Uno step importantissimo è stato l'inserimento dei individui che rappresentavano la conoscenza del sistema. La maggior parte delle entità sono state inserite manualmente, mentre i corsi sono stati inseriti utilizzando un metodo semi-automatico. Un web scraper è stato implementato per aiutare all'estrazione dei corsi dal sito unibo, verificando la correttezza dell'informazione localmente.

Dopo aver implementato una versione funzionante dell'ontologia, il processo di sviluppo si è focalizzato sull'implementazione del componente principale del sistema, ovvero `PathadoraRecommender`. La versione iniziale dell'engine riusciva a importare l'ontologia e interrogarla in modo da estrarre gli individui della conoscenza. Successivamente è stato progettato e sviluppato `PathadoraManager` in grado di gestire la prima richiesta d'inserimento. Per il suo sviluppo è stato utilizzato la libreria `OWL-API`, che offre meccanismi di manipolazione dell'ontologia nel linguaggio Java.

Successivamente alla fase di verifica e di testing della prima tipologia di richiesta, il processo di sviluppo si è concentrato sulle richieste di raccomandazione. La gestione delle raccomandazioni è stata effettuata in due fasi principali:

- sviluppo del modello a regole: Tramite la libreria `SWRL-API` è stato possibile creare e applicare regole SWRL sull'ontologia modellata. La libreria offre metodi per istanziare un reasoner e inferire conoscenza dall'applicazione delle regole. Tale strumento è stato utilizzato per applicare le regole che riguardavano le facoltà e aggregare il risultato interrogando la conoscenza dell'ontologia.

- applicazione delle interrogazioni: Dopo aver applicato le regole, il sistema aveva una versione aggiornata dell'ontologia che poi veniva importata nel database, Stardog. Stardog, un sistema di persistenza dei grafi di conoscenza, permette l'applicazione delle query SPARQL, utilizzando la sua libreria nel linguaggio Java. L'interrogazione del sistema di persistenza è servito per produrre il risultato desiderato per quanto riguarda i corsi e le risorse da consigliare.

Il processo di sviluppo si conclude controllando e testando integramente il sistema e aggiustando le regole e le interrogazioni in base ai risultati ottenuti.

6.2 Il sistema Pathadora

In questa sezione verranno spiegati in modo dettagliato i componenti più importanti del sistema e il loro processo di implementazione.

6.2.1 Ontologie

Nella sezione di Progettazione (vedi sec. 5.3) abbiamo spiegato la strategia della progettazione del knowledge graph, ma in seguito verranno spiegate dettagliatamente le ontologie implementate e importate che svolgono un ruolo importante del sistema. L'ontologia `pathadora-ontology` è stata implementata, mentre `LOM` e `AccessibleOCW` sono state importate.

In seguito verranno spiegate le ontologie `LOM` e `AccessibleOCW` nel loro formato ufficiale. Durante la spiegazione del `pathadora-ontology` verranno spiegati gli interventi e le modifiche fatte per sincronizzare insieme le ontologie.

6.2.1.1 LOM

LOM (*Learning Object Metadata*) è un ontologia che modella i metadati per descrivere le risorse digitali e non digitali, facilitando l'interoperabilità e il riutilizzo tra diversi framework di apprendimento. Tale ontologia descrive semanticamente i concetti dei metadati che rappresentano il significato delle risorse tramite meccanismi comprensibili dai componenti software del sistema. IEEE ha progettato e sviluppato lo schema di dati concettuale per definire la struttura dei metadati delle istanze di `LearningObject` (risorse).

Un LO (*Learning Object*) per l'ontologia è qualsiasi risorsa didattica digitale o non digitale che potrebbe essere utilizzata per l'apprendimento. Gli standard LOM possono essere applicati a qualsiasi risorsa, indipendentemente dal formato. Tramite l'ontologia si potrebbe specificare la struttura della risorsa, la granularità, il tipo d'interattività, etc.

Gli elementi dell'ontologia che descrivono una risorsa secondo IEEE, sono raggruppati in diverse categorie:

- *generali*: raggruppa le informazioni generali che descrivono la risorsa didattica nel suo insieme;
- *ciclo di vita*: raggruppa le caratteristiche relative alla storia e allo stato attuale della risorsa e tiene traccia degli aggiornamenti fati;
- *Meta-Metadata*: raggruppa le informazione sull'istanza dei metadati stessa (meta-modellazione);
- *tecniche*: raggruppa i requisiti tecnici e le caratteristiche tecniche della risorsa;
- *educative*: raggruppa le caratteristiche educative della risorsa;
- *i diritti*: raggruppa i diritti di proprietà intellettuale e le condizioni d'uso della risorsa;
- *relazioni*: raggruppa le caratteristiche che definiscono la relazione tra diverse risorse;
- *annotazioni*: offre commenti sull'utilizzo della risorsa;
- *classificazione*: descrive la risorsa in relazione con un determinato sistema di classificazione.

Le categorie sopraelencate raggruppano gli elementi dell'ontologia. L'ontologia LOM, definisce i seguenti componenti per modellare le informazioni sulle risorse:

- *name*: il nome con cui si fa riferimento all'elemento dati;
- *explanation*: la definizione dell'elemento;
- *size*: in numero dei valori permessi;
- *order*: l'ordine dei valori;
- *value* space: l'insieme dei valori consentiti per l'elemento, tipicamente un vocabolario o un riferimento a un altro standard;
- *datatype*: indica il tipo dei valori.

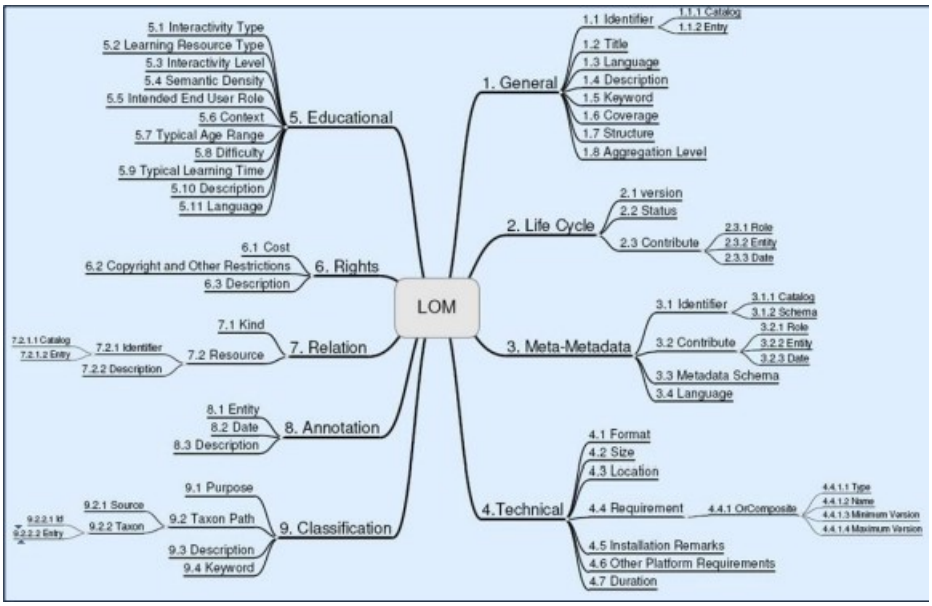

Figura 6.2: Gerarchia degli elementi di LOM

.

6.2.1.2 AccessibleOCW

AccessibleOCW è un ontologia sviluppata da OCW (*OpenCourseWare*) per rappresentare l'accessibilità e gli adattamenti delle risorse digitali nell'ambito di e-learning.

L'ontologia modella il concetto di `Learner`, che nel sistema Pathadora, rappresenta lo studente. La modellazione del learner si basa su due aspetti importanti: le proprietà che riguardano le disabilità e le esigenze d'accessibilità per una determinata risorsa digitale. Ogni tipo di disabilità ha le sue caratteristiche specifiche. Il concetto di `DigitalResource` viene modellato nell'ontologia per rappresentare qualsiasi tipo di risorsa didattica (ad esempio testo, immagine, video, audio, presentazione, etc).

L'ontologia ha specificato un insieme di classi per la modellazione dell'accessibilità e l'adattamento del formato delle risorse, tra le quali:

- `AdaptationDetailRequired`: il tipo di adattamento obbligatorio dei dettagli di una risorsa;
- `AdaptationDetailValue`: diversi tipi di adattamento dei dettagli di una risorsa;
- `AdaptationMediaRequired`: il tipi adattamento obbligatorio del formato audio-visivo della risorsa;
- `AdaptationMediaTypeValue`: diversi tipi di adattamento del formato audio-visivo della risorsa;
- `AdaptationTypeValue`: diversi tipi di adattamento di una risorsa (ad esempio testo alternativo per le immagini, descrizione testuale per le risorse audio, captions per le risorse video, etc);
- `ControlFlexibilityValue`: rappresenta la flessibilità con il controllo dei dispositivi elettronici (ad esempio keyboard, mouse, etc);
- `DisplayTransformabilityValue`: rappresenta le trasformazioni che si possono effettuare ad una risorsa in modo da aumentare il suo livello d' accessibilità (ad esempio colore del background, font size, font weight, etc);
- `EducationalComplexityValue`: modella il livello di complessità di una risorsa;

La connessione dei vari concetti dell'ontologia viene effettuate tramite le proprietà dell'ontologia. Le più importanti utilizzate dal sistema Pathadora sono: `accessMode` che definisce la modalità d'accesso ad una risorsa, `adaptionType` che specifica la tipologia d'adattamento che aumenterebbe il livello d'accessibilità e `adaptionMediaReq` che specifica la tipologia del formato d'adattamento che deve essere applicato. L'ontologia specifica delle sotto proprietà in modo rappresentare l'informazione più dettagliatamente.

`AccessibleOCW` tramite le proprietà dei dati, modella il formato dei valori associato ai dati. Possiamo menzionare `hasAdaption` che prende una `DigitalResource` nel dominio e come range un valore URI. Le annotazioni vengono importante dall'ontologia DC (*Dublin Core*).

L'ontologia internamente ha definito un insieme d'individui che sono istanze delle sue classi. Tali individui vengono utilizzati successivamente dall'ontologia `pathadora-ontology` come riferimento durante la definizione della conoscenza del sistema.

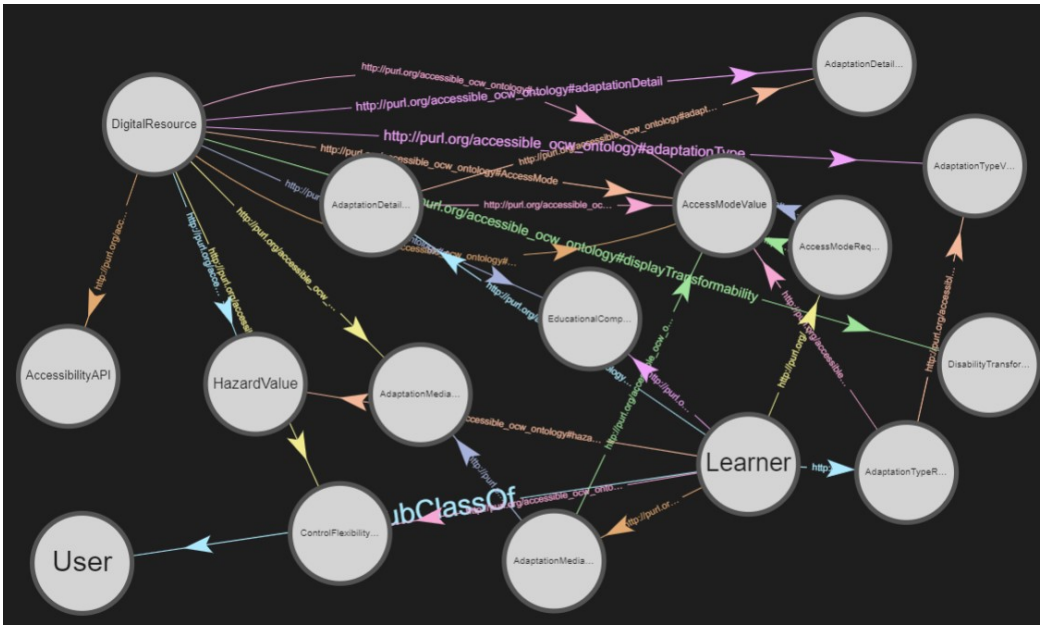

Figura 6.3: Visualizzazione a grafo degli elementi principali dell'ontologia `AccessibleOCW`

6.2.1.3 Pathadora

L'ontologia `pathadora-ontology` racchiude tutti i concetti del sistema, importando altre ontologie pubbliche come `LOM` e `AccessibleOCW`, in modo da offrire un modello completo per la rappresentazione della conoscenza.

6.2.1.3.1 Learner

L'entità `Learner` modellato in Pathadora è lo stesso concetto rappresentato nell'ontologia `AccessibleOCW`. A tale entità vengono aggiunte altre proprietà per definire semanticamente le sue caratteristiche. Tramite la proprietà `hasDegree`, l'ontologia permette di modellare i diplomi conferiti dallo studente. La proprietà `futureDegree` mette in relazione lo studente con il tipo di diploma che vorrebbe conferire nel futuro. Tramite `hasGoals`, l'ontologia modella gli obbiettivi dello studente e fornisce anche delle sottoproprietà basandosi sul tipo dell'obbiettivo (ad esempio `isPathDriven`, `isScoreDriven`, `isTimeDriven`, etc). Lo stile d'apprendimento di uno studente viene specificato tramite la proprietà `learningStyle`. Come nel caso degli obbiettivi, anche le passioni vengono modellate tramite la proprietà `passionateOf` e le sottoproprietà in base alla tipo della passione. Le informazioni che riguardano le lingue che uno studente potrebbe parare vengono inserite tramite la proprietà `speaks`. La proprietà `hasDisability` rappresenta le informazioni che riguardano le disabilità dello studente e come range di valori aspetta un istanza della classe `Disabilities`.

6.2.1.3.1 University

Come spiegato nella sezione Università (vedi sec. 5.3.1) l'organizzazione di tale istituzione viene rappresentata tramite una gerarchia di classi. I dipartimenti associati ad una scuola vengono messi in relazione tramite la proprietà `departmentOf`, mentre le facoltà di un dipartimento vengono specificate tramite `facultyOfDepartment`. Seguendo la stessa logica, i corsi vengono associati ad una facoltà tramite la proprietà `courseOfFaculty`. La proprietà `courseLanguage` permette di specificare in quale lingue verranno tenute le lezione di un corso. Un determinato corso viene dichiarato come obbligatorio oppure opzionale tramite la proprietà `isCourseObligatory` avendo come range un tipo di dato `boolean`. Le proprietà `courseSSD` e `courseType` mettono in relazione rispettivamente un corso con il suo tipo e il suo settore scientifico. Il range di queste due proprietà deve essere un

istanza delle classi `CourseSSD` e `CourseType`.

Per le istanze che rappresentano i corsi, sono state definite un insieme di proprietà dei dati tra le quali:

- `coursePeriod`: specifica il periodo del corso;
- `courseCFU`: specifica i CFU del corso;
- `courseYear`: specifica l'anno del corso;

6.2.1.3.1 Learning Object

Nella spiegazione delle ontologie `LOM` (vedi sec. 6.2.1.1) e `AccessibleOCW` (vedi sec. 6.2.1.2), abbiamo visto la modellazione delle risorsa da due punti di vista diversi basandosi sui metadati e sull'accessibilità. Pathadora tende di sincronizzare questi due punti di vista, aggiungendo nuovi concetti nella rappresentazione delle informazioni che riguardano le risorse.

Alla modellazioni `AccessibleOCW` vengono inserite delle proprietà che descrivono il livello dell'accessibilità, mappando la modalità d'accesso ad un risorsa con in tipo di dato numerico. Le tre proprietà da prendere in considerazione sono:

- `resourceCheckRatio`: coefficiente che rappresenta il contrasto visivo della risorsa in un range da $\{1, ..., 100\}$;
- `resourceReadingEase`: coefficiente che specifica la leggibilità di una risorsa in base al numero di parole in una frase, la lunghezza del testo, in numero di sillabi, etc. Il range del coefficiente va da $\{1, ..., 100\}$;
- `resourceFontSize`: specifica la dimensione del font del testo e lo mappa in un coefficiente d'accessibilità;

L'ontologia tramite la proprietà `resourceType` associa ad una risorsa un il tipo che potrebbe essere testuale, video, audio, immagine, etc. Tutti gli elementi sviluppati nelle ontologie `LOM` e `AccessibleOCW` vengono utilizzati per specificare le caratteristiche delle risorse.

6.2.1.3.1 Disabilities

Pathadora si basa principalmente sull'ontologia `AccessibleOCW` per modellare le disabilità di uno studente, però aggiunge delle proprietà non gestite da tale ontologia. Pathadora definisce un insieme di proprietà dei dati per rappresentare il livello d'accessibilità di uno studente, ad esempio `austismSpectrumLevel`, `blindnessLevel`, `hearingLossLevel`, etc). Durante la registrazione l'utente dovrebbe specificare questi livelli, anche stimarli in modo non preciso, per permettere al sistema di interpretali e mapparli con i coefficienti d'accessibilità delle risorse.

Come menzionata nella sezione Learner (vedi sec. 6.2.1.3), la proprietà `hasDisabilites` mette in relazione uno studente con un tipo di disabilità. Pathadora raggruppa le disabilità in diverse categorie che saranno istanze delle classi `Intellectual`, `Physical` e `Sensory`.

6.2.2 Le richieste dal client

```
{
  "action": "add",
  "type": "resource",
  "class": "LearningObject",
  "annotation_properties": {
    "id": "1_Res_Mat",
    "resourceTopic": "Number_Operations_in_Base_Ten"
  },
  "object_properties": {
    "resourceOfCourse": "Course_MATEMATICA",
    "adaptionType": ["auditory", "description"],
    "AccessMode": ["itemsize", "visual","textual"],
    "resourceType": "Acc_ResourceType_Textual"
  },
  "data_properties": {
    "resourceFontSize": "12",
    "resoruceExtension": "pdf",
    "resoruceReadingEase": "33",
    "resourceCheckRatio": "15"
  }
}
```

Listing 6.1: Esempio di richiesta d'inserimento di una risorsa

```
{
  "action": "add",
  "type": "learner",
  "class": "Learner",
  "annotation_properties": {
    "id": "Learner_Rossi",
    "hasName": "Rossi",
    "hasSurname": "Matteo",
    "hasEducation": "yes",
    "birthdate": "2021-11-26"
  },
  "object_properties": {
    "gender": "Gender_Male",
    "speaks": ["Language_English"],
    "hasGoals": "Goal_ScoreDriven",
    "degree": "Degree_Master",
    "future_degree": "Degree_Master",
    "passionateOf": ["Passion_Animals"],
    "hasDisability": ["Disability_FASD"]
  },
  "data_properties": {
    "autismSpectrumLevel": 10,
  }
}
```

Listing 6.2: Esempio di richiesta d'inserimento di un learner

```
{
    "action": "faculties_generation",
    "degree": "Degree_Bachelor",
    "learner": "Learner_Rossi"
 }
```

Listing 6.3: Esempio di richiesta di generazione delle facoltà

```
{
  "action": "add",
  "type": "course",
  "class": "Algoritmi_di_Ottimizzazione",
  "annotation_properties": {
    "id": "1_algo_ot"
  },
  "object_properties": {
    "isCourseObligatory": "no",
    "courseType": "Course_Type_D",
    "courseSSD": "Course_SSD_AGR"
  },
  "data_properties": {
    "courseYear": 2,
    "coursePeriod": 2,
    "courseCFU": 4
  }
}
```

Listing 6.4: Esempio di richiesta d'inserimento di un corso

```
{
  "action": "course_generation",
  "faculty": "Facolta_Medicina",
  "degree": "Degree_Bachelor",
  "year": 1,
  "learner": "Learner_Rossi"
}
```

Listing 6.5: Esempio di richiesta di generazione dei corsi

```
{
  "action": "resource_generation",
  "course": "Corso_Chirurgia",
  "learner": "Learner_Rossi"
}
```

Listing 6.6: Esempio di richiesta di generazione delle risorse del corso

6.2.3 Inserimento delle entità

La manipolazione dell'ontologia viene effettuata tramite gli strumenti offerti dall'`OWL-API`, una libreria scritta in linguaggio Java. Tale libreria viene inclusa nel package `SWRL-API`, che servirà successivamente per l'applicazione delle regole.

`PathadoraManager` nel suo stato dichiara un istanza di `OWLOntology` e `OWLOntologyManager` che implementano metodi per il caricamento dell'ontologia, la sua interrogazione e l'inserimento di nuove istanze. La funzione `loadOntologyFromOntologyDocument(PATH)` viene utilizzata per caricare l'ontologia da un file salvato localmente. Per caricare le ontologie importate si applica il metodo `getOWLImportsDeclaration(IRI)` della classe `OWLDataFactory`.

Per aggiungere un assioma dell'ontologia, la classe `OWLOntologyManager` implementa il metodo `addAxiom` per dichiarare nuove proprietà e relazioni.

```
void addProperty(String key, String value,
   OWLNamedIndividual ind) {
  OWLDataFactory df = getManager().getOWLDataFactory();

  OWLNamedIndividual val = ontEntityBy(INDIVIDUAL, value);
  OWLObjectProperty prop = ontEntityBy(OBJECT_PROP, key);
  OWLObjectPropertyAssertionAxiom propertyAssertion =
      df.getOWLObjectPropertyAssertionAxiom(prop, ind, val);

  getManager().addAxiom(pathadora, propertyAssertion);
}
```

Listing 6.7: Metodo per aggiungere un proprietà nell'ontologia.

Prima di concludere l'operazione, il manager effettua l'aggiornamento dell'ontologia e del file che lo memorizza localmente tramite la chiamate del metodo `saveOntology`. Dopo l'aggiornamento dell'ontologia, il manager risponde alla richiesta d'inserimento con un codice `OK` se l'operazione è andata a buon fine, altrimenti con `ERROR`.

6.2.4 Modello di regole

Il modello basato sulle regole viene implementato utilizzando la libreria `SWRL-API` che offre strumenti per manipolare le ontologie tramite l'applicazione delle regole.

La libreria `SWRL-API` è una libreria sviluppata tramite il linguaggio Java e include anche la libreria `OWL-API`. Tale libreria include strumenti per la modifica e l'esecuzione di regole e query.

```
public void applyRule(String ruleName, String rule){
      SWRLRuleEngine engine = initializeRuleEngine();
      engine.createSWRLRule(ruleName, rule);
      engine.infer();
}
```

Listing 6.8: Metodo abbreviato per l'applicazione di una regola tramite `SWRL`.

```
public SWRLRuleEngine initializeRuleEngine(
    PathadoraManager manager) {
   return SWRLAPIFactory
          .createSWRLRuleEngine(manger.pathadoraOnt());
}
```

Listing 6.9: Metodo abbreviato per l'inizializzazione dell'engine tramite `SWRL`.

Le regole SWRL vengono applicate per la raccomandazione delle facoltà. `Recommender` inizialmente applicata le regole che associano allo studente le scuole e i dipartimenti.

```
public void initializeOntologyWithRules(RuleModel model){
   model.applyRule(SCHOOL, Rules.schoolRule());
   model.applyRule(DEPARTMENT, Rules.departmentRule());
}
```

Listing 6.10: Metodo abbreviato per l'inizializzazione dell'ontologia applicando le regole iniziali

```
accessible_ocw_ontology:Learner(?learner) ^
pathadora:passionateOf(?learner,?passion) ^
pathadora:passionArea(?passion, ?sArea) ^
pathadora:schoolArea(?school, ?sArea)
->
pathadora-ontology:recommendedSchool(?learner,?school)
```

Listing 6.11: Semplice esempio della raccomandazione delle scuole basandosi sull'area di una scuola e sulle passioni specificate dallo studente

```
pathadora:Departments(?depart) ^
pathadora:recommendedDepartment('learner', ?depart) ^
pathadora:facultyOfDepartment(?fac, ?depart) ^
pathadora:degree(?fac, 'degree')
->
pathadora:recommendedFaculty('learner', ?fac)
```

Listing 6.12: Semplice esempio abbreviato della raccomandazione di facoltà basandosi sul tipo di diploma specificato e l'elenco dei dipartimenti consigliati

Alla fine di questa richiesta, `PathadoraManager` applicherà la funzione `facultiesJsonResponse(output)` della classe `OutputToJson` per parsare il risultato nel formato `JSON`.

6.2.5 Modello di interrogazioni

Stardog è lo strumento utilizzato per la gestione e l'interrogazione del grafo di conoscenza. La libreria `stardog-spring` offre vari meccanismi per interagire con il database tramite il linguaggio Java. La libreria permette di inizializzare il database e stabilire la connessione con `PathadoraManager`.

Inizialmente si inizializza una connessione, configurando le impostazioni autenticativi per il database. Tale componente viene passato come argomento d'input per l'istanziamento del pool di connessioni. Il pool viene configurando specificando i suoi parametri.

```
Connection initializeConnection(){
   checkDuplicatedDatabases();

   ConnectionConfiguration connectionConfig =
```

```
    ConnectionConfiguration
        .to(Provider.database)
        .server(Provider.url)
        .reasoning(false)
        .credentials(Provider.username, Provider.password);

    ConnectionPool connectionPool =
       createConnectionPool(connectionConfig);

    Connection connection = connectionPool.obtain();
    importPrefixes(connection);

    return connection;
}
```

Listing 6.13: Metodo per inizializzare e connettere con il database. Il metodo `checkDuplicatedDatabases()` controlla in sistema di persistenza se il database è già presente per rimuoverlo. Mentre il metodo `importPrefixes(connection)` serve per importare il namespace dell'ontologia al grafo di conoscienza. Dopo aver specificare il namespace, il database sarebbe pronto ad effettuare le interrogazioni.

```
ConnectionPool conPool(ConnectionConfiguration config) {
    ConnectionPoolConfig poolConfig = ConnectionPoolConfig
       .using(config)
       .minPool(MIN_POOL)
       .maxPool(MAX_POOL)
       .expiration(1, TimeUnit.HOURS)
       .blockAtCapacity(1, TimeUnit.MINUTES);

    return poolConfig.create();
    }
}
```

Listing 6.14: Metodo per configurare un istanza di `ConnectionPool` che servirebbe come paramento d'input per l'inizializzazione del database.

La libreria `stardog-spring` permette d'importare l'ontologia nel database, aprendo una connessione e poi specificando il formato della modellazione della conoscenza. Alla fine,il metodo `commit()` effettua l'operazione e chiude la connessione con il database.

```
public void importData(String path) {
   connection.begin();
   connection.add().io()
         .format(RDFXML)
         .stream(new FileInputStream(path));
   connection.commit();
}
```

Listing 6.15: Metodo per importare l'ontologia nel database

La parte più importante del modello basato sulle interrogazioni sono i meccanismi che applicano le query. La libreria importata permettere di specificare un istanza di `SelectQuery` che rappresenta l'interrogazione da effettuare. Successivamente, la libreria mette a disposizione la classe `SelectQueryResult` per aggregare il risultato della query.

```
List<Map<String, String>> queryDatabase(String queryStr) {
   SelectQuery query = connection.select(queryStr);
   try(SelectQueryResult aResult = query.execute()) {
      List<Map<String, String>> output = new ArrayList<>();
      while (aResult.hasNext()) {
         Map<String, String> mapInfo = new HashMap<>();
         aResult.next()
               .stream()
               .map(Object::toString)
               .forEach(e -> mapInfo.put(extractKey(e),
                  extractValue(e)));
         output.add(mapInfo);
      }
      return output;
   }
}
```

Listing 6.16: Metodo per aggregare il risultato di una query e convertirlo in una lista di mappe. Ogni mappa rappresenta un n-upla, in base alle variabili di selezione definite nella query.

6.2.5.1 Interrogazione corsi

L'applicazione delle query serve per aggregare una lista di corsi e di risorse da raccomandare allo studente, basandosi sulla conoscenza inferita dall'applicazione delle regole sulle facoltà.

```
SELECT ?course, ?year, ?cfu, ?ssd, ?type, ?period,
      ?isObligatory, ?language
WHERE {
  {
    SELECT ?department ?faculty ?course_degree
    WHERE {
      ?learner a accessible:Learner .
      ?learner pathadora:id 'learner' .
      ?school a pathadora:School .
      ?learner pathadora:recommendedSchool ?school .
      ?department a pathadora:Departments .
      ?school pathadora:schoolHasDepartment ?department .
      ?faculty pathadora:facultyOfDepartment ?department .
      ?faculty a ?course_degree .
      FILTER (?course_degree, 'degree') .
    }

    ?course a ?course_degree .
    ?course pathadora:isCourseObligatory ?isObligatory .
    ?course pathadora:courseYear 'year' .
    ?course pathadora:courseCFU ?cfu .
    ?course pathadora:courseSSD ?ssd .
    ?course pathadora:courseType ?type .
    ?course pathadora:coursePeriod ?period .
    ?course pathadora:isCourseObligatory ?isObligatory .
    ?course pathadora:courseLanguage ?language
  }
}
GROUP BY ?course ?language
```

Listing 6.17: Esempio di una query per aggregare un lista di corsi basandosi sui paramentri d'input dell'interrogazione (*learner, year, degree)*

6.2.5.2 Interrogazione risorse

L'interrogazione delle risorse basandosi su una query SPARQL.

```
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
.
```

Listing 6.18: Add caption here

6.3 Strumenti software utilizzati

In questa sezione verranno spiegati gli strumenti più importanti utilizzati durante lo sviluppo del progetto.

6.3.1 Protégé

Protégé è software open source che offre un interfaccia per creare e manipolare le ontologie. Protégé è stato implementato utilizzando il linguaggio Java e l'interfaccia utilizzando la libreria `Swing`.

Protégé è stato sviluppato presso la Stanford University ed è reso disponibile sotto la licenza BSD 2. Le versioni precedenti sono state sviluppate in collaborazione con l'Università di Manchester.

L'interfaccia del software offre la sezione `Entities` che contiene i tool per manipolare l'ontologia. In questa sezione possiamo navigare le classi, le proprietà e gli individui dell'ontologia. La pagina è costruita da mini sezioni che possono esser ridimensionate e offrono diverse funzionalità come visualizzazione delle annotazioni, rappresentazione gerarchia delle classi, espandono le informazioni di una classe, etc.

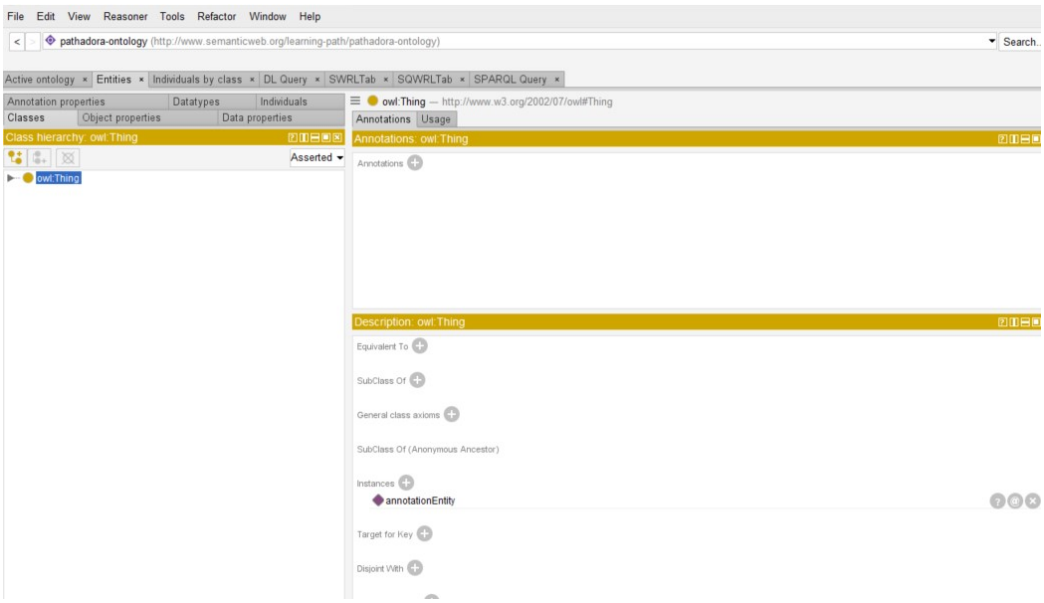

Figura 6.4: Interfaccia della sezione `Entities` di Protégé

Protégé fornisce un reasoner che viene chiamato `HermiT` per effettuare il ragionamento sull'ontologia. Dopo l'esecuzione del processo di reasoning, l'interfaccia permette di visualizzare le informazioni inferite da tale processo.

Protégé permette di installare dei plugin per importare nuove funzionalità per l'editor. Aggiungendo il plugin `SWRL Tab`, lo strumento permette di applicare le regole e manipolare l'ontologia. Allo stesso modo aggiungendo il plugin `SPARQL Query` è possibile interrogare la conoscenza dell'ontologia modellata. Tutti i plugin che vengono installai, aggiungono una sezione nell'interfaccia di Protégé.

6.3.2 Stardog

Stardog è un database di grafi di conoscenza implementato utilizzando il linguaggio Java. Fornisce meccanismi di supporto per RDF e OWL, offrendo ampia capacità di ragionamento e utilizza SPARQL come linguaggio per le interrogazioni.

6.3.2.1 Amministrazione di database

Lo scopo principale di Stardog è la gestione dei database e le funzionalità offerte per interagire con tali componenti. Stardog supporta l'inserimento dei dati in diversi formati seguendo diverse politiche d'inserimento. Le informazioni potrebbero essere aggiunte trami CLI, l'interfaccia web oppure utilizzando le librerie apposite. Stardog garantisce un alto livello di fault-tolerance in caso di scenari sbagliati, offrendo la possibilità di backup e restore dei database. Per ottenere le prestazioni migliori le informazioni memorizzate nei database potranno essere ottimizzate e compresse.

Stardog offre una varietà di parametri da impostare per configurare i database, dove le opzioni più importanti sono:

- `database.name`: specifica il nome del database tramite una stringa;
- `database.namespaces`: specifica il namespace da utilizzare per le interrogazioni;
- `database.archetypes`: specifica il nome di uno o più archetypes di database, utilizzati per associare ontologie e vincoli a nuovi database;
- `database.connection.timeout`: specifica la quantità di tempo in cui una connessione al database può essere aperta, ma inattiva, prima di essere chiusa automaticamente per recuperare le risorse;

- `graph.aliases`: opzione di configurazione per abilitare o disabilitare il supporto per gli alias di grafi denominati;
- `progress.monitor.enabled`: se abilitato, l'avanzamento delle varie attività verrà stampato nel registro del server;
- `strict.parsing`: controlla se Stardog analizza RDF rigorosamente o in modo approssimativo.

Stardog offre alcune funzioni memorizzate in modo da essere riutilizzate, evitando la duplicazione e garantendo la coerenza. Le funzioni memorizzate vengono trattate in modo simile alle funzioni integrate e definite dall'utente. Tale funzioni possono essere utilizzate nei vincoli `FILTER` e nelle assegnazioni `BIND` nelle interrogazioni SPARQL, nelle query di percorso e nelle regole.

Stardog fornisce strumenti per gestire le interrogazioni in esecuzione rispettando la configurazione impostante in fase d'esecuzione. Tali strumenti permettono di listare le query, di cancellarle, di visualizzare il loro stato, di interrompere la loro esecuzione e di loggare infiorazioni utili per l'analisi.

Nell'amministrazione dei database, Stardog garantisce la semantica transazionale supportando transazioni **ACID**:

- *atomicità*: II database garantiscono l'atomicità dove le azioni sono irriducibili e indivisibili.
- *consistenza*: I dati memorizzati devono essere validi in base al modello di dati e alle garanzie offerte dal database, garantendo la non violazione dei vincoli d'integrità. La consistenza viene fornita interrompendo una transazione che porterebbe il database in uno stato inconsistente o non valido.
- *isolamento*: Basandosi sulla configurazione in fase d'inizializzazione del parametro `transaction.isolation`, una connessione Stardog viene eseguirà isolata a livello di `SNAPSHOT`, dove il risultato dell'operazione sarebbe visibile dopo l'esecuzione del `commit`.
- *durabilità*: Di default, le transazioni di scrittura sono durevoli e non sono necessarie altre azioni.

6.3.2.2 Amministrazione di server

Stardog utilizza il valore dell'argomento del JVM `java.io.tmpdir` per scrivere file temporanei per molte operazioni diverse, e per cambiare lo

spazio temporaneo per sfruttare un particolare volume del disco o partizione, basta cambiare il valore di tale opzione.

Il file `stardog.properties` contiene le proprietà da configurare sul comportamento globale di Stardog e deve essere localizzato sotto la cartella $ `STARDOG_HOME` .

Le proprietà più importanti che Stardog permette di settare per la configurazione del server sono:

- `query.timeout` : specifica il limite superiore per il tempo di esecuzione della query ereditato da tutti i database, a meno che non venga sovrascritto in modo esplicito da un valore di timeout specifico del database nelle opzioni del database;
- `database.connection.timeout` : specifica la quantità di tempo in cui una connessione al database può essere aperta, ma inattiva, prima di essere chiusa automaticamente per recuperare le risorse;
- `backup.dir` : specifica la directory che deve essere utilizzata da Stardog per l'archiviazione dei backup;
- `backup.keep.last.number.backups` : controlla quanti backup del server verranno conservati da Stardog;
- `memory.management` : disabilità la gestione della memoria personalizzata e fa si che Stardog esegua tutte le query sul heap;
- `memory.mode` : definisce tre modalità standard per la configurazione del consumo di memoria in base allo scenario di utilizzo: `bulk_load` , `read_optimized` , `write_optimized` .
- `password.length.min/max` : la lunghezza minima e massima della password.

Il modo più semplice per avviare e stoppare il server, eseguendolo sulla porta predefinita, è:

```
$ stardog-admin server start
...
$ stardog-admin server stop
```

Lo Stardog backup effettuerà il backup dell'intero server Stardog, di tutti i database e dei metadati associati. A differenza dei backup del database, che eseguono un backup completo del database ogni volta che viene eseguito, il backup del server esegue un backup incrementale dei dati. In questo modo, ogni volta che viene eseguito il comando, è necessario salvare solo gli aggiornamenti dei database dall'ultimo backup. Tramite il comando `server restore` è possibile ripristinare i backup del server creati dal backup del server.

```
$ stardog-admin server backup
// Specify backup location
$ stardog-admin server backup /path/to/backup/location
```

Stardog utilizza `stardog.log` per tenere traccia dei log quando il server è in esecuzione. Per abilitare i log di tipo `access` bisogna impostare l'opzione `logging.access.enabled` a `true`.

6.3.2.3 Sicurezza

Il modello di sicurezza di Stardog gestisce il controllo degli accessi in base ai ruoli standard. Stardog utilizza Apache Shiro per l'autenticazione, autorizzazione e la gestione delle sessioni.

I permessi di un ruolo possono essere definiti sul soggetto, sull'azione e sulla risorsa. Le azioni possono essere `read`, `write`, `create`, `delete`, etc. Una risorsa è un'entità o un servizio Stardog a cui è controllato l'accesso. Le risorse sono identificate dal loro tipo e dal loro nome. Una particolare risorsa è indicata come `type_prefix:name` e potrebbe essere: `user`, `named-graph`, `data-source`, `metadata`, etc.

Tutto il traffico di rete tra client e Stardog può essere eseguito su protocolli `HTTP` o `HTTPS`. Per garantire la riservatezza delle credenziali di autenticazione dell'utente si dovrebbe configurare il server Stardog in modo che accetti solo connessioni crittografate con `SSL`. L'operazione di configurazione deve effettuare i seguenti step:

1. creare o acquisire un certificato;
2. configurare il server Stardog;
3. configurare il client Stardog;

4. abilitare `SSL` all'avvio del server;

5. testare la connessione client e server Stardog;

6.3.2.4 Stardog Studio

Stardog Studio è un IDE sviluppata per eseguire le funzionalità di Stardog, utilizzando un interfaccia. A parte all'amministrazione dei cluster Stardog, in Studio sono disponibili quasi tutte le funzionalità esistenti tramite la CLI e altri endpoint.

Tramite Stardog Studio è possibile:

- ottenere una panoramica di alto livello del grafo della conoscenza, visualizzando le connessioni trai i sorgenti di dati;
- esplorare i dati, effettuando la ricerca delle classi e delle proprietà;
- scrivere e eseguire interrogazioni in `SPARQL` e `GraphQL`;
- scrivere e editare file OWL e RDFS;
- configurare grafi virtuali e sorgenti di dati;
- gestire tutti i processi che riguardano i database.

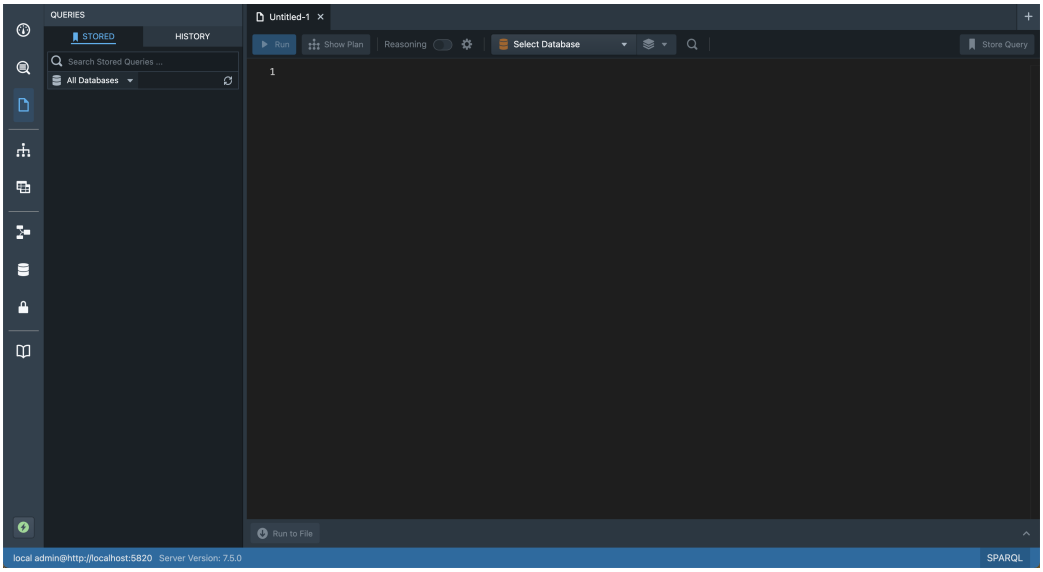

Figura 6.5: Interfaccio di Stardog Studio

Capitolo 7

Risultati

Capitolo 8

Conclusioni

Conclusioni to be completed

References

[1] A learning path recommendation model based on a multidimensional knowledge graph framework for e-learning, https://www.sciencedirect.com/science/article/pii/S095070512030085X?via%3Dihub

[2] <Name of the reference here>, <urlhere>

[3] <Name of the reference here>, <urlhere>

@inproceedingsinproceedings, author = Kalou, A.K. and Koutsomitropoulos, Dimitrios and Solomou, Georgia and Botsios, Sotirios, year = 2015, month = 09, pages = 171-182, title = Metadata Interoperability and Ingestion of Learning Resources into a Modern LMS, isbn = 978-3-319-24128-9, doi = 10.1007/978-3-319-24129-6$_1$5

Elenco delle figure

Listings