Alma Mater Studiorum
Università di Bologna
Campus di Cesena

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

Titolo della tesi

Relazione finale in

Web Semantico

Relatore:
Prof.ssa
Antonella Carbonaro

Presentata da:
Andrea Betti

Sessione III
Anno Accademico 2020-2021

Introduzione

Il progetto vede come oggetto lo sviluppo di un'applicazione web che raccolga informazioni relative a studenti universitari per sottoporle a un sistema di raccomandazione di corsi accademici e relative risorse didattiche associate.

L'applicazione permetterà inoltre ai docenti universitari di associare nuove risorse ai corsi di cui sono titolari, su cui verranno generati automaticamente metadati relativi alle proprietà del file e al suo contenuto.

Verrano inoltre analizzate varie tecniche di generazione automatica di metadati.

Indice

Introduzione **i**

1 Stato dell'arte **1**

1.0.1 Metodi di generazione automatica 2

1.1 Ricerca sul tema 11

1.1.1 Ricerca sperimentale 11

1.1.2 Ricerca applicativa 12

2 Contesto del progetto **17**

3 Tecnologie utilizzate **19**

3.0.1 Angular 19

3.0.2 Bootstrap 21

3.0.3 Node.js 22

3.0.4 Npm 24

3.0.5 Express.js 24

3.0.6 MongoDB 25

3.0.7 Mongoose 26

3.0.8 Textract 28

4 Implementazione **29**

4.1 Frontend 29

4.1.1 Servizio di autenticazione 30

4.1.2 Classi di utilità 31

4.2 Backend 32
4.3 Estrazione e generazione dei metadati 33
4.3.1 Estrazione dei metadati intrinseci del file 33
4.3.2 Grado di leggibilità di Flesh-Kincaid 33
4.3.3 Accessibilità delle immagini in un documento 34

Conclusioni **35**

Bibliografia **37**

Ringraziamenti **38**

Elenco delle figure

1.1 Schema di Kea 16

3.1 Struttura di Angular 20
3.2 Rappresentazione visuale delle transizioni delle fasi dell'event loop. 23
3.3 Mappatura di oggetti tra Node e MongoDB tramite Mongoose. 27

Elenco delle tabelle

Capitolo 1

Stato dell'arte

I *metadati* sono informazioni associate ad una pagina web o ad una porzione di essa che ne descrivono il contenuto, permettendo che i risultati delle ricerche degli utenti abbiano una maggiore efficacia.

Generare metadati di buona qualità in modo efficiente è essenziale per organizzare e rendere accessibile la grande quantità di risorse presente sul web. Il successo di biblioteche digitali, il sostenimento dell'interoperabilità e l'evoluzione del web semantico dipendono da una generazione di metadati efficiente[1].

L'utilizzo di strumenti automatici di generazione di metadati può facilitare la gestione di una quantità sempre crescente di informazioni. Idealmente, questi strumenti sono capaci di estrarre informazioni da risorse strutturate e non strutturate, creando metadati di qualità che possano permettere interoperabilità semantica.

Tra i benifici delle generazione automatica di metadati vi è la scalabilità, per coprire la metadatazione di grandi quantità di informazioni che richiederebbe molto tempo se venisse gestita tutta manualmente, e la consistenza nella generazione dei dati.

1.0.1 Metodi di generazione automatica

A differenza della generazione manuale dei metadati, la generazione automatica dei metadati si basa su metodi automatici per assistere o completare il processo di creazione dei metadati. Greenberg ha distinto due metodi di generazione automatica dei metadati: l'estrazione dei metadati (*metadata extraction*) e la raccolta dei metadati (*metadata harvesting*). L'estrazione dei metadati utilizza l'indicizzazione automatica e le tecniche di recupero delle informazioni per generare metadati strutturati utilizzando il contenuto originale delle risorse. La raccolta di metadati consiste invece nel raccogliere in modo automatico metadati già prodotti e archiviati da diversi repository.

All'interno di questa dicotomia di metodi di estrazione, vi sono molte tecniche più specifiche che i ricercatori hanno sviluppato per la generazione automatica di metadati. Polfreman et al. ha identificato sei tecniche che sono state sviluppate nel corso degli anni: raccolta di meta-tag, estrazione di contenuti, indicizzazione automatica, text e data mining, autogenerazione di dati estrinseci e social tagging. Sebbene l'ultima tecnica non sia propriamente una tecnica di generazione automatica di metadati, poiché viene utilizzata per generare metadati con un intervento minimo richiesto dai professionisti dei metadati, può essere vista come una possibile modalità per semplificare il processo di creazione dei metadati[2].

1.0.1.1 Estrazione di meta-tag

L'**estrazione di meta-tag** è un processo dove vengono identificati valori per campi di metadati e popolati attraverso l'osservazione dei tag associati ai documenti, di fatto convertendo i metadati raccolti da più repository in formati diversi.

La più grande debolezza di questo tipo di tool è il fatto che la sua efficacia dipenda dalla qualità dei metadati raccolti. Tra i tool che sfruttano questa tecnica vi sono MarcEdit, che mediante appositi harvester raccoglie metadati da record conformi al protocollo **Open Archives Initiative Protocol for**

Metadata Harvesting (**OAI-PMH**) e che può convertire in diversi formati (MARC, MARC XML, MODS, EAD), e tool come Editor-Converter Dublin Core Metadata e Firefox Dublin Core Viewer Extension che convertono informazioni trovate nei meta-tags dei file HTML trovati sul web in elementi Dublin Core.

L'OAI-PMH fornisce un framework di interoperabilità, indipendente dall'applicazione, basata sul raccoglimento dei metadati. Il framework di OAI-PMH comprende i service provider, che eseguono richieste per effettuare la raccolta dei metadati, e i data provider, che espongono i metadati. OAI-PMH è largamente utilizzato da biblioteche digitale, repository istituzionali e archivi digitali[3]. Alcuni esempi di archivi che implementano OAI-PMH sono:

- Digital Public Library of America (DPLA), aggrega metadati relativi a enti istituzionali americani
- National Science Digital Library (NSDL), è un libreria digitale ad accesso libero contenente collezioni di metadati inerenti a scienze, tecnologia, ingegneria e matematica
- OAIster, un catalogo di milioni di record relativi a risorse di fonti eterogenee
- Open Language Archives Community (OLAC), una libreria virtuale di risorse linguistiche

1.0.1.2 Estrazione di contenuti

L'**estrazione di contenuti** è una forma di estrazione di metadati che sfrutta algoritmi di machine learning per associare parole chiave al documento sulla base del suo contenuto. Il vantaggio di questo approccio è che il risultato è indipendente dalla qualità dei metadati associati alla risorsa. Esempi di applicazioni che impiegano questa tecnica sono Kea, che utilizza le tecniche TF-IDF e first-occurence per identificare e assegnare frasi chiave da risorse

testuali, e Open Text Summarizer che esegue l'estrazione automatica di un riassunto del testo passato in input e delle relative parole chiave.

I metodi di estrazione di parole chiave possono essere suddivisi in tre categorie: metodi statistici, metodi basati sui grafi e metodi basati sul deep learning[4].

1.0.1.2.1 Metodi statistici

I **metodi statistici** sono i più semplici. Calcolano le statistiche per le parole chiave e utilizzano tali statistiche per valutarle. Tra questi metodi rientrano il calcolo la frequenza delle parole, la collocazione linguistica, la co-occorrenza e tecniche più sofisticate come TF-IDF e YAKE!.

TF-IDF (Term Frequency–Inverse Document Frequency) stima l'importanza della parola nel documento rispetto all'intero corpus (insieme di più documenti). Calcola la frequenza di ogni termine nel documento e la pondera con l'inverso della frequenza del termine nell'intero corpus. Le parole chiave verranno selezionate tra i termini con i punteggi più alti. Questo algoritmo favorisce i termini frequenti nel documento di testo e non frequenti negli altri documenti. Il vantaggio di TF-IDF è che è veloce e lo svantaggio è che necessita di un corpus di almeno qualche dozzina di documenti. TF-IDF è indipendente dalla lingua.

YAKE! (Yet Another Keyword Extractor) è un metodo di estrazione di parole chiave che utilizza funzionalità statistiche ricavate da un singolo documento per estrarre le parole chiave. Estrae le parole chiave in cinque passaggi:

1. **Pre-elaborazione e identificazione del termine candidato**: il testo è suddiviso in frasi, blocchi (parte della frase separata da punteggiatura) e token. Il testo viene pulito, etichettato e vengono identificate le stop words.

2. **Estrazione delle caratteristiche**: calcola le seguenti cinque caratteristiche statistiche per i termini (parole) nel documento:

- Casing: conta il numero di volte (in proporzionale al numero complessivo di volte in cui compare) in cui il termine appare in maiuscolo o come acronimo nel testo. Un termine significativo di solito appare più spesso in maiuscolo.
- Posizione del termine: posizione mediana della frase del termine nel testo. I termini più vicini all'inizio tendono a essere i più significativi.
- Normalizzazione della frequenza dei termini: misura la frequenza equilibrata dei termini nel documento.
- Relazione del termine al contesto: misura con quanti termini diversi il termine candidato coesiste. Termini più significativi coesistono con termini meno diversi.
- Termine in frasi diverse: misura quante volte i termini compaiono in frasi diverse. Un punteggio più alto indica un termine più significativo.

3. **Calcolo del punteggio del termine**: le funzioni del passaggio precedente vengono combinate in un unico punteggio con l'equazione creata dall'uomo.

4. **Generazione di n-grammi e calcolo dei punteggi delle parole chiave**: l'algoritmo identifica tutti gli n-grammi validi. Le parole negli n-grammi devono appartenere allo stesso blocco e non devono iniziare o terminare con una stopword. Successivamente, ogni n-grammo viene valutato moltiplicando i punteggi dei suoi membri e normalizzato per ridurre l'impatto della lunghezza di n-gram. Le stopword vengono trattate in modo diverso per ridurre al minimo il loro impatto.

5. **Deduplicazione e ranking dei dati**: vengono rimosse le parole chiavi simili tra loro.

1.0.1.2.2 Metodi basati sui grafi

I metodi basati su grafi generano un grafo dei termini correlati presenti nei documenti. Un grafo, ad esempio, può collegare i termini che si verificano contemporaneamente nel testo. I metodi basati su grafi utilizzano metodi di classificazione che considerano la struttura del grafo per valutare l'importanza del vertice. Uno dei metodi di questa categoria più conosciuti è TextRank.

TextRank è un metodo di classificazione basato su grafi utilizzato per estrarre frasi pertinenti o trovare parole chiave. L'estrazione delle parole chiave avviene in cinque passaggi:

- **Tokenizzazione e annotazione del testo** con tag di Part of Speech (PoS) (nomi, articoli, verbi...)
- **Costruzione del grafo di co-occorrenza di parole**: i vertici nel grafo sono parole con tag PoS selezionati. Due vertici sono collegati con un arco se compaiono all'interno della finestra di N parole nel testo. Il grafo è non orientato e non ponderato.
- **Graph ranking**: partendo con il punteggio di ciascun vertice inizializzato a 1, viene applicato sul grafo l'algoritmo di ranking utilizzando la seguente equazione:

$$S(Vi) = (1-d) + d\left(\sum_{j \in In(Vi)} \frac{S(Vj)}{|Out(Vj)|}\right) \tag{1.1}$$

Il peso *S(Vi)* di un vertice *Vi* è calcolato considerando i pesi dei vertici connessi al nodo *Vi*. Nell'equazione, d è il fattore di smorzamento impostato su 0,85. Gli *In(Vi)* corrispondono ai collegamenti in entrata al vertice *Vi* e gli *Out(Vj)* ai collegamenti in uscita dal vertice *Vj*. Poiché stiamo considerando grafi non orientati, un collegamenti in entrata verso un nodo è anche un collegamento in uscita dallo stesso nodo. L'algoritmo viene eseguito su ciascun nodo in diverse iterazioni

fino a quando i pesi sui nodi convergono: la variazione tra le iterazioni è inferiore a 0,0001.

- **Selezione delle parole con il punteggio più alto**: le parole (quindi i vertici) sono ordinate per punteggio, dal più alto al più basso. Viene poi selezionato il primo terzo delle parole.
- **Estrazione delle parole chiave**: in questo passaggio, le parole selezionate nella fase precedente vengono unite in parole chiave composte da più parole se compaiono in una stessa frase. Il punteggio delle parole composte corrisponde alla somma dei punteggi delle parole singole che le compongono.

L'algoritmo viene eseguito su ciascun documento separatamente e non necessita di un corpus di documenti per eseguire l'estrazione delle parole chiave. TextRank è indipendente dalla lingua.

RAKE (Rapid Automatic Keyword Extraction) è un altro algoritmo di estrazione delle parole chiave basato su grafi. L'algoritmo si basa sull'osservazione che le parole chiave sono spesso composte da più parole e di solito non includono stop-word o punteggiatura.

Include i seguenti passaggi:

- **Estrazione delle parole chiave candidate**: il testo viene suddiviso in base alle parole chiave candidate in base alle parole di arresto e al delimitatore di frase. Una parola chiave candidata è una frase che si trova tra due stop-word o delimitatori di frase. I delimitatori di frase sono, ad esempio, i caratteri di punteggiatura.
- **Costruzione del grafo di co-occorrenza delle parole chiave**: i vertici nel grafo corrispondono a parole. Sono collegati se compaiono insieme nelle parole chiave candidate. Il grafo è ponderato: il peso corrisponde al numero di volte in cui le parole collegate appaiono insieme nelle parole chiave candidate. Un vertice può essere collegato a sé stesso nel caso corrisponda a una parola che appare in una parola chiave candidata insieme a sé stessa.

- **Punteggio delle parole**: ogni parola nel grafo è valutata con uno dei seguenti punteggi:
  - *Grado della parola* (`deg(w)`): numero di parole con cui la parola w co-occorre (somma dei pesi degli archi). Il grado favorisce le parole che compaiono più spesso e in parole chiave più lunghe.
  - *Frequenza della parola* (`freq(w)`): numero di volte che la parola appare in qualsiasi parola chiave candidata. Questo criterio favorisce le parole che appaiono più frequentemente.
  - *Rapporto tra grado e frequenza* (`deg(w)/freq(w)`): favorisce le parole che ricorrono principalmente nelle parole chiave candidate più lunghe. Il grado favorirà le parole chiave più brevi.
- **Punteggio delle parole chiave candidate**: il punteggio di ciascuna parola chiave candidata è la somma dei punteggi delle parole che la compongono.
- **Parole chiave adiacenti**: le parole chiave candidate non includono stopword. Poiché a volte le stopword possono essere parte della parola chiave, vengono aggiunte in questo passaggio. L'algoritmo individua le coppie di parole chiave di cui fa parte almeno una stopword e, nel caso quest'ultima sia presente almeno due volte nel testo, le aggiunge all'insieme delle stopword esistenti. Il punteggio della nuova parola chiave è la somma dei punteggi delle parole chiave che la compongono.
- **Estrazione delle parole chiave**: viene infine estratto il terzo delle parole chiave con il punteggio migliore.

La differenza principale tra RAKE e TextRank è che RAKE considera le ricorrenze all'interno delle parole chiave candidate invece di una finestra fissa. Utilizza una procedura di punteggio più semplice e statistica: non include l'ottimizzazione. L'algoritmo viene eseguito su ciascun documento separatamente, quindi non necessita di un corpus di documenti.

1.0.1.2.3 Metodi basati sul deep learning

La comparsa del deep learning ha consentito metodi basati sull'embedding, un insieme di tecniche di modellazione in cui le parole vengono mappate in vettori di numeri reali.

Questi metodi trovano principalmente un elenco di parole chiave candidate (ad esempio, Bennani et al. considerano solo parole chiave costituite da nomi e aggettivi). Incorporano il documento e le parole chiave candidate nello stesso spazio di inclusione e misurano la similarità (ad es. similarità del coseno) tra i valori del documento e delle parole chiave. Selezionano le parole chiave più simili al testo del documento in base alla misura di similarità.

1.0.1.3 Indicizzazione automatica

Così come la content extraction, l'**indicizzazione automatica** prevede l'utilizzo di machine learning e algoritmi rule-based per estrarre metadati dalle risorse. La differenza sta nel fatto che i termini estratti vengono successivamente mappati in vocabolari controllati come Library of Congress Subject Headings (LCSH), Getty Thesaurus of Geographic Names (TGN), Library of Congress Name Authority (LCNA) o ontologie relative a un dominio specifico.

1.0.1.4 Text e data mining

A causa della complessità delle tecniche utilizzate per l'estrazione di metadati, la maggior parte degli usi di queste tecniche sono stati sviluppati per risolvere i problemi della generazione automatica di metadati nel contesto di specifici progetti di ricerca. I principali motivi sono il fatto che l'efficacia di queste tecniche dipende dalla qualità e dalla quantità dei dati di training, e che i vocabolari controllati come LCSH sono troppo complicati e diversificati per essere applicati con mezzi automatici.

1.0.1.5 Autogenerazione di dati estrinseci

La generazione automatica di dati estrinseci è il processo di estrazione dei metadati su una risorsa informativa che non è contenuta all'interno della risorsa stessa. La generazione automatica di dati estrinseci può includere l'estrazione di metadati tecnici come il formato e le dimensioni del file, ma può essere anche l'estrazione di funzionalità più complesse come il livello di classe scolastica a cui è destinata una risorsa educativa o il tipo di lettori per cui è destinato un documento. Processi di questo tipo sono difficili da implementare a causa dell'assenza di informazioni sul contesto di una risorsa contenute nella risorsa stessa. Tra i tool che implementano questa tecnica di estrazione di metadati vi sono Dspace e JHove.

1.0.1.6 Social Tagging

Il social tagging è una forma di generazione di metadati, sebbene non propriamente automatica, che permette agli utenti di creare e assegnare tag alle risorse di un sito web. A causa del costo relativamente basso della generazione e del mantenimento dei metadati attraverso il social tagging e della sua attuale popolarità diffusa, alcuni progetti hanno tentato di utilizzare questi dati per migliorare i repository di archiviazione di metadati. Ad esempio, Linstaedt et al. usa sofisticati programmi per computer per analizzare le immagini statiche trovate all'interno di Flickr e utilizzare questi risultati d'analisi per propagare tag utente pertinenti a nuove immagini elaborate.

Un esempio di utilizzo del social tagging più complesso può consistere nell'utilizzare tecniche di apprendimento automatico, tenendo traccia dei cambiamenti applicati sui metadati associati a una risorsa da parte degli utenti e migliorare quindi i meccanismi di learning del database sulla base di queste modifiche. Tra i tool che supportano il social tagging vi sono Dspace e Omeka.

Sebbene gli strumenti di generazione automatica di metadati offrano molti vantaggi, soprattutto per quanto riguarda la semplificazione del processo di creazione dei metadati, esistono barriere significative all'adozione e all'im-

plementazione diffusa di questi strumenti. Un problema incontrato con gli strumenti di generazione automatica di metadati è che molti di essi sono sviluppati localmente per soddisfare le esigenze specifiche di un determinato progetto o come parte della ricerca accademica.

Diversi tool sono focalizzati alla risoluzione di pochi problemi specifici legati alla generazione di metadati, come possono essere l'estrazione di parole chiave di Kea piuttosto che l'estrazione di tag HTML e la successiva conversione in elementi Dublin Core di Editor Converter Dublin Core, portando quindi a sforzi significativi per coordinare le applicazioni selezionate e ottenere l'output desiderato. È inoltre necessaria una buona conoscenza tecnica di linguaggi di programmazione per implementare questi sistemi propriamente.

Un altro problema è la discontinuità delle applicazioni, che possono presto risultare obsolete e perdere supporto tecnico.

1.1 Ricerca sul tema

Gli sforzi di ricerca per la generazione automatica di metadati possono essere classificati in due aree: **ricerca sperimentale**, focalizzata sulle tecniche di recupero delle informazioni e sui contenuti delle risorse digitali, e **ricerca applicativa**, riguardante lo sviluppo di software per la creazione di contenuti e strumenti di generazione di metadati utilizzati nel contesto operativo[5].

1.1.1 Ricerca sperimentale

La crescita dei repository di risorse digitali fornisce una grande quantità di collezioni per lo studio della generazione automatica di metadati.

I ricercatori che gestiscono il contenuto delle risorse digitali per la generazione di metadati hanno sperimentato principalmente con la **struttura dei documenti** e i **sistemi di rappresentazione della conoscenza**.

1.1.1.1 Struttura del documento

I ricercatori hanno identificato relazioni tra genere, contenuto e struttura del documento. Ad esempio, il genere del documento può dare informazioni sulla densità testuale, che può essere utilizzata per prevedere le prestazioni dell'algoritmo di estrazione dei metadati per determinati tipi di documenti.

Il genere del documento inoltre determina una struttura prevedibile che facilita l'estrazione automatica di metadati. Ad esempio, i documenti di ricerca includono informazioni standard come "titolo", "autore" e "affiliazione dell'autore del documento. Gli esperimenti che sfruttano la struttura del documento in questa seconda sede utilizzando un algoritmo Support Vector Machine (SVM) (ad esempio, Han et al., 2003) e Variable Hidden Markov Model (DVHMM) (Takasu, 2003) hanno avuto un discreto successo per la generazione di metadati.

1.1.1.2 Sistemi di rappresentazione della conoscenza

La memorizzazione di metadati può essere eseguita attraverso l'utilizzo di ontologie, thesauri, sistemi classificatori, file di autorità e altri strumenti di rappresentazione della conoscenza. Attraverso l'utilizzo di regole di inferenza, è possibile generare ulteriori informazioni relative a un oggetto sulla base dei valori delle sue proprietà.

1.1.2 Ricerca applicativa

Tra gli strumenti in grado di generare automaticamente metadati è possibile distinguere i software per la creazione di contenuti da tool più specializzati conosciuti come applicazioni per la generazione di metadati.

1.1.2.1 Software per la creazione di contenuti

Esempi di software per la creazione di contenuti sono Microsoft Word, Adobe Acrobat e Nullsoft Winamp, essenzialmente qualsiasi software che può essere utilizzato per creare contenuti digitali, testuali o multimediali.

Nel contesto del web, il software per la creazione di contenuti viene utilizzato per creare una risorsa digitale a cui è possibile accedere tramite un browser e il software associato. Anche un surrogato bibliografico può essere considerato contenuto.

I software per la creazione di contenuti possono supportare la generazione di metadati tramite mezzi automatici, semiautomatici e umani. Le tecniche automatiche sono spesso impiegate per produrre metadati tecnici come la data di creazione, la data dell'ultima modifica, la dimensione in byte e il formato del file.

Alcuni software per la creazione di contenuti estraggono i metadati dal contenuto del documento nel tentativo di fornire rappresentazioni descrittive (ad esempio, Word assegna automaticamente un titolo in base alla prima riga di un documento). Alcuni software per la creazione di contenuti includono un modello per facilitare l'immissione di metadati umani. Le tecniche automatiche possono quindi essere impiegate dall'applicazione per convertire i metadati immessi in un linguaggio di codifica specificato, incorporarli in un'intestazione di risorsa o inserirli in un database di metadati.

1.1.2.2 Applicazioni e generazione automatica di metadati

Le applicazioni per la generazione di metadati differiscono dal software per la creazione di contenuti in quanto sono progettate specificamente e solo per produrre record di metadati. La quantità di elaborazione automatica e umana richiesta per produrre metadati distingue i generatori, applicazioni che si basano principalmente su tecniche automatiche, dagli editor, che integrano l'elaborazione automatica con quella umana.

1.1.2.2.1 Data Fountains

Data Fountains è uno strumento utilizzabile non solo per descrivere le risorse Internet, ma in primo luogo per scoprirle. Ha tre funzionalità principali:

- generazione di metadati per una determinata pagina web dato il suo URL
- generazione di metadati per le risorse inerenti a un argomenti particolare
- esecuzione del drill-down dei collegamenti a partire da un URL iniziale al fine di generare record di metadati ed estrarre le parti significative del testo analizzato

1.1.2.2.2 JHOVE

JHOVE (JSTOR/Harvard Object Validation Environment) è un framework utilizzabile per eseguire l'identificazione del formato, la validazione e la caratterizzazione di oggetti digitali:

- L'identificazione del formato è il processo atto a determinare il formato a cui è conforme un oggetto digitale
- La convalida del formato è il processo di determinazione del livello di conformità di un oggetto digitale alla specifica per il suo presunto formato
- La caratterizzazione del formato è il processo di determinazione delle proprietà significative specifiche del formato di un oggetto

Queste azioni sono spesso necessarie durante il funzionamento di routine degli archivi digitali e per le attività di conservazione digitale.

JHOVE utilizza un'architettura plug-in estensibile; può essere configurato al momento della sua chiamata per includere qualsiasi modulo di formato specifico e gestori di output desiderati. La versione iniziale di JHOVE include moduli per flussi di byte arbitrari, testo codificato ASCII e UTF-8, audio AIFF e WAV, GIF, JPEG, JPEG 2000, TIFF e PDF; e gestori di output di testo e XML.

1.1.2.2.3 Kea

Kea estrae dai documenti testuali n-grammi con lunghezza predefinita che non iniziano o terminano con una stop word. Nell'indicizzazione controllata, raccoglie solo gli n-grammi che corrispondono ai termini presenti nel thesaurus (dizionario dei sinonimi). Se il thesaurus definisce relazioni tra termini non consentiti (non descrittori) e termini consentiti (descrittori), sostituisce ciascun descrittore con un non descrittore equivalente. Per ciascuna frase candidata, Kea computa quattro feature:

- **TF-IDF**, misura l'importanza di un termine rispetto a una collezione di documenti. Aumenta al numero di volte che il termine è contenuto nel documento, e diminuisce con la frequenza del termine nella collezione proporzionalmente
- **First occurrence**, è computata come la percentuale della quantità di testo che precede la prima occorrenza del termine nel documento. La probabilità che i termini che compaiono all'inizio dei documenti siano elementi chiave è maggiore rispetto a quella degli altri
- **Lunghezza della locuzione**, è il numero delle parole che la compongono
- **Grado del nodo**, è il numero di frasi nel set candidato collegati semanticamente a una specifica frase candidata. A un grado del nodo alto aumenta la probabilità che sia una espressione chiave.

Prima di poter estrarre frasi chiave da nuovi documenti, Kea deve prima creare un modello che apprenda la strategia di estrazione dai documenti indicizzati manualmente. Ciò significa che per ciascun documento nella directory di input deve essere presente un file con estensione ".key" e lo stesso nome del documento corrispondente. Questo file dovrebbe contenere frasi chiave assegnate manualmente, una per riga. Dato l'elenco delle frasi candidate, Kea contrassegna quelle che sono state assegnate manualmente come esempi positivo e tutte le altre come esempi negativi. Analizzando i valori delle

feature per le frasi candidate positive e negative, viene calcolato un modello che riflette la distribuzione dei valori delle feature per ciascuna frase.

Una volta terminata la sessione di training, Kea utilizza il modello per calcolare per ogni frase candidata la probabilità di essere una frase chiave considerando le feature calcolate. Le frasi con le probabilità più alte vengono selezionate nella serie finale di frasi chiave. L'utente può specificare il numero di frasi chiave che devono essere selezionate[6].

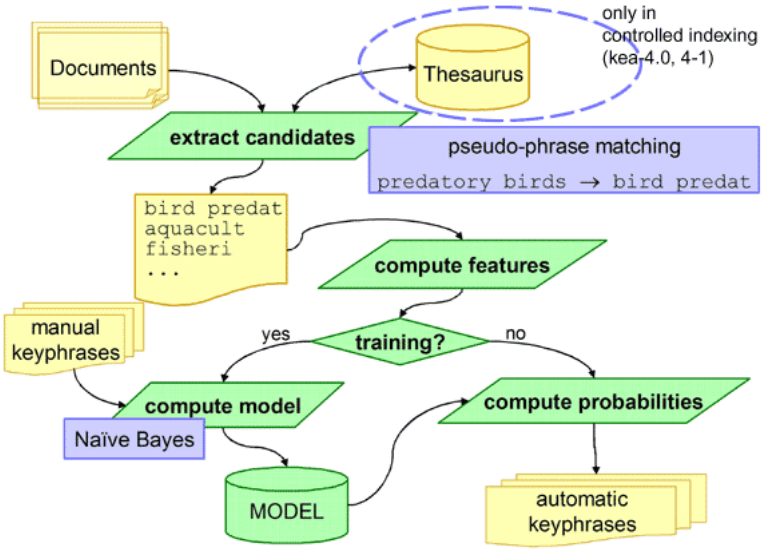

Figura 1.1: Schema di Kea

Capitolo 2

Contesto del progetto

Capitolo 3

Tecnologie utilizzate

3.0.1 Angular

Angular è un framework open source per lo sviluppo di applicazioni web scritto interamente in TypeScript.

Un elemento fondamentale per la costruzione di applicazioni Angular è il componente, al quale viene affidato il controllo di una porzione dello schermo, acquisendo la responsabilità di visualizzare i dati ricevuti e gestirne l'interazione con l'utente secondo il concetto di view.

Una moderna applicazione Angular è sintetizzabile in un insieme di componenti ordinate in modo gerarchico con le quali vengono fornite le funzionalità con cui far interagire l'utente.

Tra i principali motivi che rendono vantaggioso questo modello di programmazione vi è il fatto di poter assemblare applicazioni diverse utilizzando un insieme comune di componenti, favorendo quindi il riutilizzo del codice; al contempo la struttura definita dell'applicazione ne favorisce la suddivisione in moduli e la leggibilità del codice scritto.

Ogni componente ha un modello HTML che dichiara come quel componente viene visualizzato.

Angular estende l'HTML con una sintassi aggiuntiva che permette di inserire valori dinamici inizializzati nel componente. Angular aggiorna automati-

camente il DOM renderizzato a ogni cambiamento dello stato del componente e seguendo le istruzioni fornite dalle cosiddette direttive. Questo modello di progettazione consente di scrivere codice più flessibile e testabile.

Angular inoltre fa uso del pattern dependency injection, che consente di dichiarare le dipendenze delle classi TypeScript lasciando gestire l'istanziazione interamente al framework. Quando Angular crea un componente, chiede all'injector i servizi richiesti. Un injector mantiene un container delle istanze dei servizi che sono stati precedentemente creati. Se un'istanza di un servizio richiesto non è nel container, l'injector ne crea uno e lo aggiunge al container.

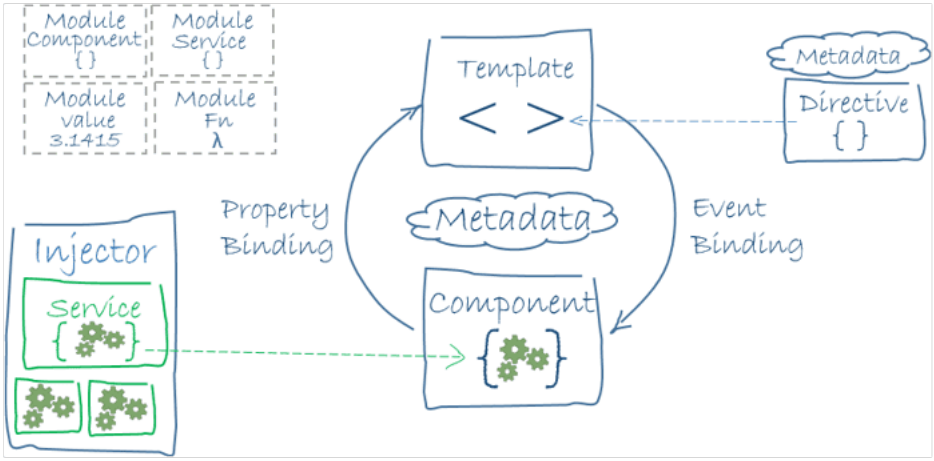

Figura 3.1: Struttura di Angular

3.0.1.1 Vantaggi e svantaggi

Essendo basato su TypeScript, un linguaggio di programmazione che differisce da JavaScript e che offre numerose funzionalità aggiuntive come ad esempio la presenza di tipi statici, possono essere implementati numerosi tool per supportare meglio i programmatori e ridurre i bug nelle applicazioni. Altra differenza importante è la presenza di numerose funzionalità aggiuntive come RxJS o la presenza stessa di un modello dei dati consistente integrato

grazie al linguaggio Typescript, elemento assente ad esempio nel framework rivale React.

Al contempo l'onere di avere molte funzionalità integrate comporta un peso maggiore in termini di spazio di archiviazione in seguito alla prima installazione, mentre in altri concorrenti il peso effettivo del framework è decisamente inferiore, risultando migliori per applicazioni meno complesse o che necessitano di meno funzionalità di quelle proposte da Angular.

Infine va rilevata una ridotta flessibilità del framework di Google rispetto ai rivali, con tutti i vantaggi e gli svantaggi derivanti da questa scelta soprattutto a livello progettuale. Un programmatore che sceglie Angular desidera una progettazione meno impegnativa e approva tutte le scelte che il framework ha preso al posto suo; un programmatore di un altro framework desidera invece fare delle scelte personalizzate, dettate dalle proprie necessità progettuali, assumendosi poi però la responsabilità di mantenere aggiornate tutte le dipendenze inserite all'interno della propria applicazione.

3.0.2 Bootstrap

Bootstrap è un framework CSS gratuito e open source per la creazione di siti e applicazioni per il web. Contiene modelli di design basati su CSS e estensioni opzionali di JavaScript per tipografia, moduli, pulsanti, navigazione e altri componenti dell'interfaccia.

Tra le principali caratteristiche di questa libreria troviamo la piena compatibilità con tutte le ultime versioni di tutti i principali browser utilizzati, il supporto al responsive web design, l'adozione di una filosofia mobile-first per quanto riguarda il design oltre che un'ottima documentazione a corredo della libreria per rendere più facile la sua diffusione.

Gli sviluppatori possono sfruttare le classi CSS definite in Bootstrap per personalizzare ulteriormente l'aspetto dei loro contenuti.[7]

3.0.3 Node.js

Node.js è un runtime JavaScript basato sugli eventi, non bloccante e asincrono costruito sul motore JavaScript V8 di Chrome e sulla libreria libuv. È usato per sviluppare applicazioni che fanno un ampio utilizzo di JavaScript sia lato client che lato server, e che quindi beneficiano della riciclabilità del codice e dell'assenza di commutazione di contesto.

Se un task entra in fase di stallo o pausa per l'esecuzione di un operazione I/O, può essere avviato un altro task. Ciò garantisce un alto tasso di efficienza, in quanto non è necessario attendere l'esecuzione di un singolo task per proseguire l'esecuzione dell'intero programma.

Per facilitare lo sviluppo di codice JavaScript complesso, Node.js supporta lo standard CommonJS che permette uno sviluppo modularizzato e la distribuzione di software in pacchetti attraverso Npm.[8]

Node.js usa un `[event loop][]` come costrutto di runtime anziché una libreria come in altri linguaggi. In altri sistemi, c'è sempre una chiamata bloccante per avviare l'event-loop. In genere il comportamento è definito tramite callback all'inizio di uno script e alla fine avvia un server attraverso una chiamata bloccante come EventMachine::run(). In Node.js non esiste alcuna chiamata per avviare il ciclo. Node.js entra semplicemente nel ciclo degli eventi dopo aver eseguito lo script di input, e ne esce quando non ci sono più callback da eseguire.[9]

Quando Node.js viene avviato, inizializza l'event loop, processa gli script di input forniti che potrebbero effettuare chiamate API asincrone, programmare timer o chiamare `process.nextTick()`, per poi processare l'event loop.[10]

Le operazioni dell'event loop possono essere suddivise in fasi; ciascuna di esse ha una coda FIFO di callback da eseguire. Quando l'event loop entra in una fase, eseguirà qualsiasi operazione a essa relativa, per poi eseguire callback nella coda di quella fase finché non viene svuotata o quando è stato raggiunto il numero massimo consentito di callback eseguite, e quindi entrare nella fase successiva.[10]

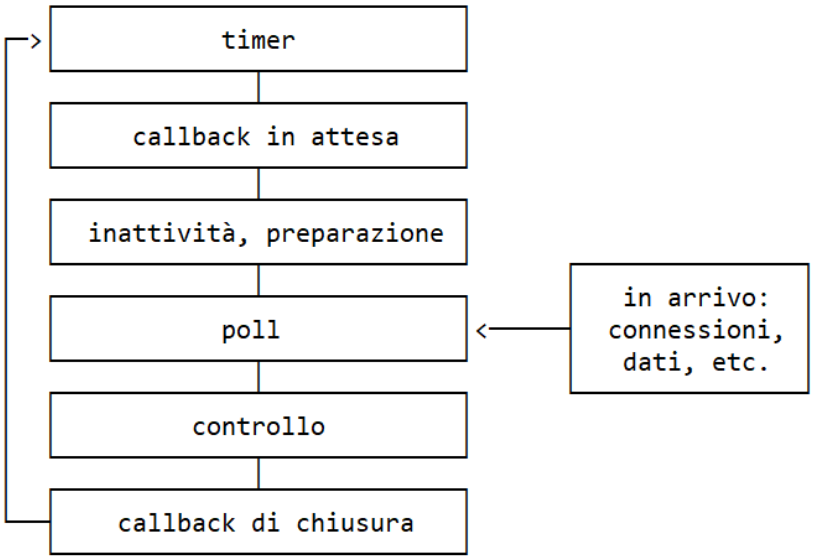

Figura 3.2: Rappresentazione visuale delle transizioni delle fasi dell'event loop.[10]

Le fasi dell'event loop sono le seguenti:

- **timer**: questa fase esegue le callback programmate da `setTimeout()` e `setInterval()`.
- **callback in attesa**: esegue le callback di input/output prelevate dalla coda di attesa relative a operazioni di sistema, come ad esempio errori TCP.
- **inattività, preperazione**: operazioni interne.
- **poll**: vengono prelevati e inseriti nella coda nuovi eventi collegati al completamento di operazioni di I/O. Se la coda non è vuota, verranno processate le diverse callback in essa presenti. In caso contrario, se sono state registrate delle callback con la funzione `setImmediate()`, verrà terminata questa fase e si passerà alla fase successiva per processarle. Nel caso in cui non siano state registrate callback con la funzione `setImmediate()`, l'event loop resta in attesa che vengano aggiunte

nuove callback alla coda. Mentre è in questa fase, l'event loop controlla se sono presenti callback che devono essere eseguite nella coda dei timer e, se lo sono, ritorna nella prima fase ed esegue le callback presenti in quella coda.[11]

- **controllo**: vengono invocate le callback `setImmediate()`.
- **callback di chiusura**: vengono gestiti gli eventi di chiusura, per pulire lo stato dell'applicazione.

La funzione `process.nextTick()` viene eseguita subito dopo il completamento dell'operazione corrente, indipendentemente dalla fase in cui l'event loop si trova. È principalmente utilizzata per gestire errori, risorse non necessarie, ritentare richieste non andate a buon fine, oppure per effettuare operazioni dopo l'esaurimento dello stack delle chiamate ma prima che l'event loop continui.[10]

3.0.4 Npm

Il package manager **npm** è una raccolta di pacchetti gratuiti o a pagamento utilizzabili sul proprio progetto di sito web. Utilizzabile al seguito dell'installazione di Node.js viene messo a disposizione come strumento da linea di comando con il quale scaricare moduli JavaScript poi disponibili all'interno del proprio progetto. Viene data la possibilità allo sviluppatore di caricare i propri moduli per metterli a disposizione della community.

3.0.5 Express.js

Express.js è un framework per applicazioni web Node.js flessibile e leggero che fornisce una serie di funzioni avanzate per applicazioni web e dispositivi mobili.

Permette di gestire le richieste HTTP tramite middleware, funzioni con accesso all'oggetto richiesta `req`, all'oggetto risposta `res` e alla successiva funzione middleware nel ciclo richiesta-risposta dell'applicazione.

Quando il server riceve una richiesta HTTP la racchiude all'interno di un oggetto ServerRequest. Questo oggetto, insieme all'oggetto ServerResponse, viene passato al primo middleware che ne può modificare il contenuto o aggiungere proprietà.

3.0.6 MongoDB

MongoDB è un DBMS non relazionale open source che utilizza un modello orientato ai documenti, il quale supporta diverse forme di dati. È una delle numerose tecnologie di database non relazionali nate a metà degli anni 2000 sotto il nome di NoSQL per l'utilizzo in applicazioni per big data e altre operazioni di elaborazione di dati che non beneficiano dell'utilizzo di modelli relazionali rigidi. L'architettura MongoDB è composta da collezioni e documenti, piuttosto che da tabelle e righe.

In MongoDB, un record è un documento, che consiste in una struttura dati composta da coppie di campo e valore. I documenti MongoDB usano una variante degli oggetti JSON chiamata Binary JSON (BSON), che può contenere ulteriori tipologie di dati. I campi dei documenti sono analoghi alle colonne dei database relazionali, e i valori in essi contenuti possono essere una varietà di tipi di dati, tra cui altri documenti, array e array di documenti.

I documenti, forniti di una chiave primaria come identificatore univoco, sono l'unità di base dei dati in MongoDB. Le collezioni contengono insiemi di documenti, e sono analoghe alle tabelle nei database relazionali. Le collezioni possono contenere qualsiasi tipo di dato, ma come restrizione i dati in una collezione non possono essere sparsi in altri database.

La mongo shell è un'interfaccia JavaScript a MongoDB interattiva che permette di effettuare interrogazioni e aggiornamenti sui dati, oltre ad eseguire operazioni di amministrazione. La shell è un componente standard delle distribuzioni open source di MongoDB. Una volta installato MongoDB, gli utenti connettono la mongo shell alle istanze MongoDB in esecuzione.

Il formato BSON con cui sono conservati i documenti permette di rappresentare i dati in forma binaria; ciò rende possibile lo sharding automatico,

una proprietà chiave di MongoDB che permette alle collezioni di essere distribuite su molteplici sistemi a seguito della crescita del volume dei dati da conservare.

A differenza di altri database NoSQL, MongoDB non richiede l'utilizzo di schemi di database (strutture logiche dei dati contenuti nel database) predefiniti e memorizza qualsiasi tipo di dato. Ciò dà agli utenti la flessibilità di creare un qualsiasi numeri di campi in un documento, facilitando la scalabilità dei database MongoDB rispetto a quelli relazionali.

Il fatto di poter disporre di un oggetto JSON-like permette una rapida manipolazione delle informazioni, senza necessitare di operazioni di join e quindi ridurre il costo delle operazioni.[12]

3.0.7 Mongoose

Mongoose è una libreria di Object Data Modelling (ODM) per MongoDB e Node.js. Gestisce le relazioni tra i dati, permette la definizione di schemi, ed è utilizzata come convertitore tra oggetti nel codice e la loro rappresentazione in MongoDB.

La chiamata `require("mongoose")` restituisce un oggetto Singleton, così come avviene per qualsiasi modulo importato in ES6. Ciò significa che alla prima chiamata verrà creata e restituita un'istanza della classe `Mongoose`, e alle chiamate successive verrà restituita la stessa istanza creata precedentemente e restituita la prima volta.[13]

Uno schema descrive il costrutto dei dati di un documento, ovvero definisce il nome e il tipo di ciascun elemento. È ragionevole associare uno schema a ciascuna collezione presente nel database. Un esempio di schema in Mongoose è il seguente:

```
var userSchema = new mongoose.Schema({
 name: String,
 email: String,
 createdOn: Date,
 verified: Boolean
});
```

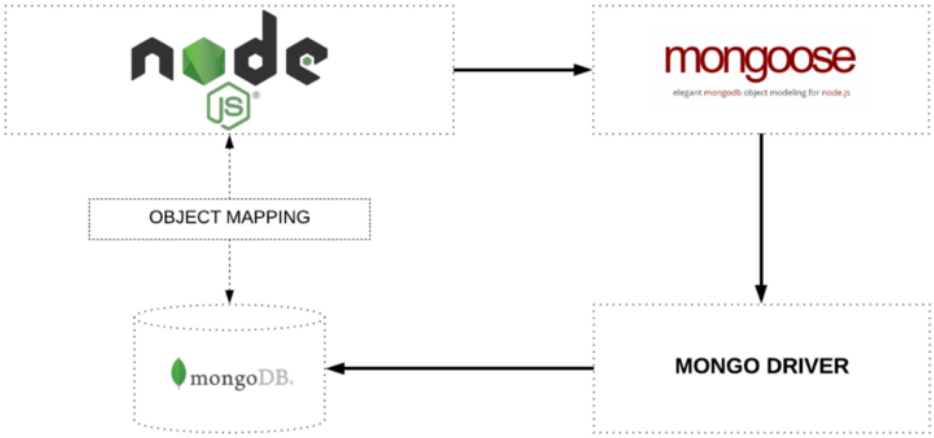

Figura 3.3: Mappatura di oggetti tra Node e MongoDB tramite Mongoose.[13]

Un altro elemento chiave di Mongoose è il cosiddetto modello. Esso è la versione compilata dello schema; un'istanza del modello verrà mappata a un documento nel database. Si occupa di gestire la lettura, la creazione, l'aggiornamento e la cancellazione dei documenti.[14]

```
var User = mongoose.model("User", userSchema);
```

Le specifiche di connessione tra Mongoose e il database sono contenute nella cosiddetta stringa di connessione, espressa nel seguente formato:[15]

```
mongodb://[username:password@]host1[:port1][,...hostN[:portN
    ]][/[database][?opzioni]]
```

Vi sono due metodi per effettuare la connessione al database con Mongoose: `mongoose.connect` e `createConnection`.[16]

Il primo imposta la connessione di default, la quale sarà accessibile da qualsiasi punto dell'applicazione se impostata correttamente.

```
var dbURI = "mongodb://localhost/mydatabase";
```

```
mongoose.connect(dbURI);
```

Il secondo viene utilizzato nel caso sia necessario stabilire più di una connessione, che sia allo stesso database che a un altro.

```
var dbURI = "mongodb://localhost/myadmindatabase";
var adminConnection = mongoose.createConnection(dbURI);
```

3.0.8 Textract

Textract è una libreria Python atta all'estrazione del testo da un file. Supporta l'estrazione da file di diversi formati comuni appoggiandosi a parser dedicati.

Una volta invocato, il metodo `texttract.process('path/to/file.extension')` instrada il nome del file al parser appropriato e restituisce il testo estratto come stringa di byte codificata con la codifica specificata.[17]

Capitolo 4

Implementazione

4.1 Frontend

Il frontend è stato sviluppato utilizzando Angular 8, utilizzando Angular CLI per installare librerie ed eseguire l'applicazione.

L'applicazione è suddivisa in sottocartelle per ciascun componente:

- `home`: componente il cui contenuto è accessibile solo a seguito dell'autenticazione. Nel caso l'utente autenticato sia uno studente, verrà renderizzato il componente `dashboard-user`.
- `login`: permette l'accesso dell'utente registrato nel sistema, sarà accessibile a chi non ha eseguito l'accesso.
- `register`: permette la registrazione di un utente. È utilizzato per due route diverse, relative alla registrazione dello studente e alla registrazione del docente, per cui verranno richiesti campi diversi.
- `dashboard-user`: componente attraverso il quale vengono sottomesse al sistema di raccomandazione le diverse richieste per ottenere il learning path utilizzando come dati di input le informazioni associate all'utente autenticato. L'invio delle richieste avviene attraverso tre form corrispondenti all'invio di informazioni per ciascun step della produzione del learning path:

  - *richiesta dei dipartimenti e delle facoltà consigliate*: vengono utilizzate come input le informazioni dello studente associate al momento della registrazione
  - *richiesta dei corsi consigliati*: una volta ottenuti i dipartimenti e le facoltà consigliate, l'utente può inviare la richiesta per visualizzare i corsi consigliati dal sistema di raccomandazione selezionando il dipartimento, la facoltà e l'anno di interesse
  - *richiesta delle risorse consigliate*: una volta selezionato il corso di interesse tra quelli consigliati, verranno visualizzate le risorse suggerite sulla base dei dati dello studente

- `profile-teacher`: pannello che permette al docente di associare nuovi corsi al suo profilo tra quelli memorizzati nel sistema
- `dashboard-courses`: vengono visualizzate le informazioni dei corsi. Gli amministratori possono modificarle.
- `dashboard-resources`: pannello visualizzabile in home dai docenti e dagli amministratori, permette il caricamento di nuove risorse didattiche ai corsi di cui sono titolari.

La gestione delle richieste al server backend è contenuta nei metodi di servizi dedicati.

4.1.1 Servizio di autenticazione

Il servizio di autenticazione contiene la logica delle funzioni di login, logout e registrazione, utilizzabili dai componenti del sistema.

I subject e gli observable della libreria RxJS sono usati per tenere traccia dell'utente attualmente loggato e notificare del login e del logout, mediante il metodo `this.currentUserSubject.next()`, i componenti che eseguono `subscribe()` sull'oggetto `currentUser`.

Il metodo `login()` invia al server una richiesta POST per l'autenticazione utilizzando le credenziali dell'utente. Se l'utente è presente nel database, le

sue informazioni verranno memorizzate nel local storage per tenere traccia del login in tutte le pagine.

Il metodo `login()` rimuove dal local storage le informazioni relative all'utente loggato e imposta `currentUserSubject` a `null`, notificando i subscriber del logout.

Il metodo `register()` invia al server una richiesta POST per la registrazione dell'utente, per poi proseguire con la chiamata di `login()`.

4.1.2 Classi di utilità

L'**Auth Guard** è un Route Guard usato per limitare l'accesso a determinati route a utenti autorizzati, passato come parametro `canActivate` ai percorsi di route interessati. Il metodo `canActivate()`, invocato ogni volta che qualcuno tenterà di accedere al route associato, restituisce un valore booleano che indica se consentire o meno la navigazione. Se l'utente non è autenticato, verrà reindirizzato alla route di login.

L'**Error Interceptor** intercetta le risposte delle richieste API e provvede a eseguire il logout dall'applicazione in caso di errori. Implementa la classe HttpInterceptor della libreria `@angular/common` ed è aggiunto tra i provider dell'applicazione. Grazie alle informazioni presenti nell'oggetto Provider, il Root Injector sa come creare e recuperare un'istanza dell'ErrorInterceptor quando serve e fornirla a qualsiasi componente, direttiva o altro servizio che ne faccia richiesta.

Il **JWT Interceptor** intercetta le richieste invocate per aggiungere un token JWT di autenticazione nell'header qualora l'utente dovesse essere loggato.

4.2 Backend

Il server viene avviato lanciando il file `server.js`, attraverso il quale viene creata un'istanza Express, viene stabilita una connessione col database MongoDB e vengono definite le diverse route utilizzate dall'applicazione.

Le route sono suddivise nei seguenti moduli, ciascuno dei quali fa riferimento a uno schema Mongoose utilizzato nel database:

- `users`
  - `/register` (post): carica nel database un nuovo utente utilizzando le informazioni del corpo dell'oggetto richiesta e effettua il login di quest'ultimo assegnandogli un token
  - `/` (get): restituisce i dati di tutti gli utenti registrati nel database
  - `/me` (get): restituisce i dati dell'utente loggato
  - `/courses` (get): restituisce i dati dei corsi associati all'utente autenticato, nel caso quest'ultimo abbia il ruolo `teacher`
  - `/courses` (post): riceve come input un array di id di corsi e li associa al docente autenticato
  - `/courses/:id` (delete): rimuove il corso con id specificato dall'elenco dei corsi associati al docente autenticato autenticato
- `auth`
  - `/` (get): restituisce le informazioni relative all'utente loggato
  - `/` (post): effettua il login dell'utente con email e password sottomesse generando un token
- `courses`
  - `/` (get): restituisce i dati di tutti i corsi registrati nel database
  - `/:id` (get): restituisce le informazioni associate al corso con id specificato

- `/` (post): se l'utente autenticato ha il ruolo `admin`, registra un nuovo corso
- `/:id` (post): se l'utente autenticato ha il ruolo `admin`, aggiorna il corso con id specificato
- `/:id` (delete): se l'utente autenticato ha il ruolo `admin`, rimuove il corso con id specificato
- `/resource/:course_id` (post): se l'utente autenticato ha il ruolo `teacher`, associa una risorsa al corso con id associato

Per le route che richiedono la verifica dell'identità dell'utente loggato è utilizzata una funzione middleware che accede al token associato all'header della richiesta, decodificandolo con JWT per ottenere come output le informazioni dell'utente che potranno poi essere utilizzate dall'handler associato alla route.

4.3 Estrazione e generazione dei metadati

Al caricamento di una risorsa associata a un corso, vengono calcolati una serie di metadati relativi alle proprietà della risorsa e del suo contenuto di cui il modello di profilazione farà uso nel calcolo del suggerimento delle risorse consigliate.

4.3.1 Estrazione dei metadati intrinseci del file

Al caricamento di una risorsa associata a un corso, i metadati a essa associati vengono estratti mediante la libreria `metadata-extract`, che a sua volta fa uso di diversi estrattori per più estensioni.

4.3.2 Grado di leggibilità di Flesh-Kincaid

Viene inoltre calcolato attraverso uno script Python un metadato relativo al grado di leggibilità del testo contenuto nel file, utilizzando la formula di

Flesh-Kincaid. Più il valore è alto, più il testo risulta essere semplice da leggere.

$$F = 206,835 - (84,6 * S) - (1,015 * P)$$

La difficoltà di lettura dipende da S e da P, rispettivamente il numero medio di sillabe contenute in una parola e il numero medio di parole contenute in una frase.

La scelta dei coefficienti è una conseguenza di un processo di affinamento del grado di istruzione di una persona in grado di leggere e comprendere la lingua inglese, ottimizzati in modo che venga restituito un valore compreso tra 0 e 100. Viene data più importanza al valore di S piuttosto che a P.[18]

4.3.3 Accessibilità delle immagini in un documento

È stato prodotto uno script attraverso in cui viene restituito un indicatore di accessibilità del testo contenuto nelle immagini presenti in un documento PDF.

Per ciascuna immagine presente nel documento, vengono catturate le aree che contengono testo ed estratto il colore dei caratteri (prelevando il colore più rilevante nella textbox analizzata). Viene poi calcolato il rapporto di contrasto tra questo colore e il colore di sfondo dell'immagine, seguendo i criteri di WCAG 2 dove il range di differenza di luminosità tra i due colori varia da 1 (ad esempio bianco su bianco) a 21 (ad esempio nero su bianco).

Viene poi calcolata la media dei rapporti relativi alle immagini di ciascun file, e il risultato finale corrisponderà alla media di questi risultati per ogni file.

Conclusioni

Bibliografia

[1] Jane Greenberg. Metadata generation: Processes, people and tools.

[2] Jung ran Park and Andrew Brenza. Evaluation of semi-automatic metadata generation tools: A survey of the current state of the art.

[3] https://guides.lib.utexas.edu/metadata-basics/harvesting.

[4] https://towardsdatascience.com/keyword-extraction-methods-the-overview-355573

[5] Kristina Spurgin Jane Greenberg and Abe Crystal. Final report for the amega (automatic metadata generation applications) project.

[6] http://community.nzdl.org/kea/description.html.

[7] https://getbootstrap.com/docs/4.1/getting-started/introduction/.

[8] About node.js. https://stackoverflow.com/tags/node.js/info.

[9] Informazioni su node.js. https://nodejs.org/it/about/.

[10] The node.js event loop, timers, and process.nexttick(). https://nodejs.org/it/docs/guides/event-loop-timers-and-nexttick/.

[11] Claudio Marotta. Node.js event loop: cos'è e come funziona. https://www.mrwebmaster.it/javascript/node-js-event-loop-come-funziona_12434.html, dicembre 2017.

[12] Margaret Rouse. Mongodb. https://searchdatamanagement.techtarget.com/definition/MongoDB.

[13] Nick Karnik. Introduction to mongoose for mongodb. https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/, febbraio 2018.

[14] Simon Holmes. *Mongoose for Application Development*, chapter The cornerstones of Mongoose. Packt Publishing, agosto 2013.

[15] Connection string uri format. https://docs.mongodb.com/manual/reference/connection-string/.

[16] Simon Holmes. *Mongoose for Application Development*, chapter Establishing a Database Connection. Packt Publishing, agosto 2013.

[17] https://readthedocs.org/projects/textract/downloads/pdf/stable/.

[18] Eugen-Gabriel Garais. Web applications readibility.

[19] Latex. latex.com.

[20] https://monkeylearn.com/blog/named-entity-recognition/.

[21] Malcolm Polfreman and Shrija Rajbhandari. Metatools - investigating metadata generation tools.

Ringraziamenti

Ringrazio la relatrice Catia Prandi e la correlatrice Chiara Ceccarini per la disponibilità dimostrata e i suggerimenti proposti durante lo svolgimento del lavoro di tesi.

Ringrazio infine tutti coloro che mi hanno sostenuto durante questo percorso universitario e che hanno creduto nel raggiungimento di questo mio importante traguardo.