



SAPIENZA
UNIVERSITÀ DI ROMA

Progettazione e sviluppo di piattaforma per la gestione di risorse linguistiche

Facoltà di Ingegneria dell'informazione, informatica e statistica
Corso di Laurea in Informatica

Candidato

Andrea Gasparini
Matricola 1813486

Relatore

Prof. Roberto Navigli

Anno Accademico 2019/2020

Tesi discussa il 14 Dicembre 2020
di fronte a una commissione esaminatrice composta da:
Prof. Roberto Navigli (presidente)
Prof. Emiliano Casalicchio
Prof. Pietro Cenciarelli
Prof. Luigi Cinque
Prof. Daniele Gorla
Prof. Walter Quattrocioni
Prof. Adolfo Piperno

Progettazione e sviluppo di piattaforma per la gestione di risorse linguistiche
Relazione di Tirocinio. Sapienza – Università di Roma

© 2020 Andrea Gasparini. Tutti i diritti riservati

Questa relazione è stata composta con L^AT_EX e la classe Sapthesis.

Versione: 24 novembre 2022

Email dell'autore: gasparini.1813486@studenti.uniroma1.it

Sommario

L'oggetto di questo tirocinio è la progettazione e lo sviluppo di una piattaforma web per migliorare la gestione delle risorse linguistiche di **SapienzaNLP**, il gruppo di ricerca del dipartimento di Informatica che si occupa di multilingual Natural Language Understanding.

La relazione è divisa in 4 parti fondamentali: l'introduzione allo **scenario di riferimento** e alla necessità del lavoro svolto; l'**analisi dei requisiti**, volta a migliorare la modellazione delle funzionalità della piattaforma; la **progettazione del sistema** e della sua architettura; i dettagli dello sviluppo e delle **fasi di implementazione**.

Indice

| | | |
|----------|--|-----------|
| 1 | Introduzione | 1 |
| 1.1 | Le risorse linguistiche nell’NLP | 1 |
| 1.2 | Le licenze software | 2 |
| 1.3 | La necessità della piattaforma | 4 |
| 2 | Analisi dei requisiti | 5 |
| 2.1 | Requisiti della piattaforma | 5 |
| 2.2 | Raffinamento e funzionalità aggiuntive | 6 |
| 2.2.1 | Utenti | 6 |
| 2.2.2 | Autenticazione | 6 |
| 2.2.3 | Gestione utenti | 7 |
| 2.2.4 | Gestione risorse | 7 |
| 3 | Progettazione | 9 |
| 3.1 | Architettura | 9 |
| 3.2 | Sessione utente | 10 |
| 3.3 | Generazione dei moduli pdf | 11 |
| 3.4 | Diagramma ER | 11 |
| 4 | Implementazione | 13 |
| 4.1 | Data tier | 13 |
| 4.2 | Application tier | 14 |
| 4.2.1 | Il framework Spring | 15 |
| 4.3 | Presentation tier | 17 |
| 4.3.1 | Approccio component based | 17 |
| 4.3.2 | Richieste AJAX | 18 |
| 4.3.3 | Le pagine realizzate | 19 |
| 5 | Conclusioni | 27 |
| 5.1 | Sviluppi futuri | 28 |
| | Bibliografia | 31 |

Capitolo 1

Introduzione

1.1 Le risorse linguistiche nell’NLP

Il Natural Language Processing (NLP) è un settore dell’intelligenza artificiale che studia i problemi legati a comprensione, analisi e generazione automatica del linguaggio naturale (come ad esempio un testo scritto o una conversazione orale). Il **Natural Language Understanding** (NLU) [6] è un argomento specifico dell’NLP che riguarda in particolare la capacità delle macchine di leggere e comprendere il significato non solo di un testo, ma anche più in generale del linguaggio naturale. È facile rendersi conto di quanto sia complesso l’obiettivo che l’NLU si pone, per via della grande ambiguità che può avere il linguaggio umano, basti pensare ai diversi significati che una stessa frase può assumere in base al contesto del discorso. Nell’affrontare questa tematica è quindi importante prendere in considerazione la struttura sintattica e l’analisi semantica, ovvero rispettivamente la disposizione delle parole in una frase secondo un criterio ben preciso e del significato che sta dietro alle parole anche nel contesto della frase.

Cos’è una risorsa linguistica? Per far sì che un algoritmo di NLP possa elaborare un testo basandosi su sintassi e semantica in maniera automatica è necessario che possa "imparare" a partire da dei dati linguistici strutturati. Una risorsa contenente questa tipologia di dati e che ne consenta facilmente il trattamento e l’analisi da parte di una macchina viene detta **risorsa linguistica**. Da non confondere con i dizionari informatizzati, il cui scopo è limitato a fornire informazioni linguistiche sul lessico di una lingua e che sono costruiti per essere consultati solamente da un utente umano.

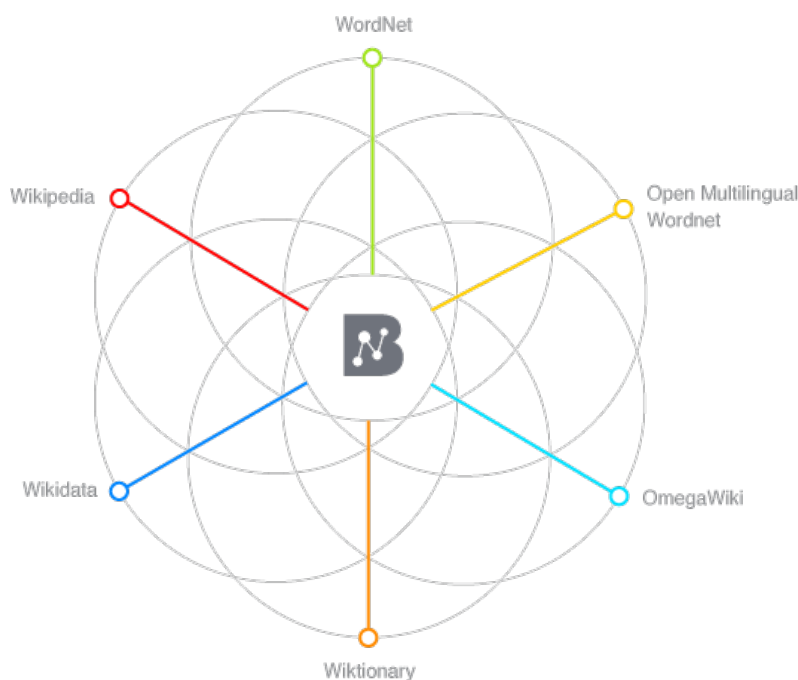
Inizialmente la costruzione di risorse linguistiche avveniva manualmente, come ad esempio per WordNet e FrameNet, ma questo richiedeva un grande impegno in termini di tempi e costi. Solo negli ultimi anni si sono sperimentati altri approcci per la creazione delle risorse, da quelli automatici o semi-automatici, in cui parte del lavoro viene svolto da una macchina, a quelli collaborativi che sarebbero impensabili senza l’ausilio di internet.

WordNet [4] WordNet è una delle più importanti e storiche risorse linguistiche per l’inglese. Il progetto, nato nel 1985 e ancora oggi molto utilizzato, si basa

sull'idea di creare un dizionario in cui la ricerca non avviene alfabeticamente ma concettualmente. All'interno di WordNet le parole sono organizzate in gruppi di sinonimi detti *synset*, dall'inglese synonyms set, ciascuno dei quali rappresenta un concetto lessicale. La struttura è quella di un grafo, in cui i nodi rappresentano i synset e gli archi le relazioni lessicali e semantico-concettuali presenti tra loro.

BabelNet [7] Una delle principali risorse linguistiche pubblicate da SapienzaNLP è BabelNet, un dizionario enciclopedico multilingue consultabile online¹, creato integrando automaticamente risorse come Wikipedia, WordNet, Wiktionary, Wikidata e molte altre. BabelNet è allo stesso tempo una rete semantica che collega concetti, lemmi ed entità nominate tramite delle relazioni semantiche chiamate **Babel synset**. Ogni Babel synset comprende diverse lingue e rappresenta uno specifico significato raggruppando tutti i sinonimi che lo esprimono, con relative definizioni ed esempi.

Attualmente BabelNet supporta 284 lingue diverse, per un totale di 15.780.364 Babel synsets, 808.974.108 sensi, 277.036.611 relazioni semantico-lessicali.



1.2 Le licenze software

Come in generale per qualunque prodotto software, anche l'utilizzo delle risorse pubblicate da SapienzaNLP è sempre vincolato da delle licenze d'uso, ovvero una serie di autorizzazioni che vengono concesse all'utente per quanto riguarda l'utilizzo, la modifica e l'eventuale redistribuzione.

Le tipologie di licenze esistenti nell'ambito software sono molteplici e possono variare da un utilizzo a fini commerciali, come ad esempio **OEM** che consente l'utilizzo del software preinstallato sull'hardware che si acquista ma vincolandolo solo

¹<https://babelnet.org>

ad esso, fino ad un utilizzo completamente libero, come per **GNU General Public License** la quale permette di accedere al codice sorgente, modificarlo e ridistribuirne delle copie – anche a pagamento – fintanto che questo avvenga utilizzando la stessa licenza. Un'esigenza molto comune è quella di avere delle licenze personalizzate ad hoc per casi specifici, ma nel farlo è importante tenere in considerazione molteplici aspetti e questioni giuridiche per poter distribuire tranquillamente il software.

Le licenze Creative Commons (CC) [2] Creative Commons è un'organizzazione no-profit nata con lo scopo di semplificare e standardizzare le modalità per ottenere una licenza personalizzata in base alla combinazione di 4 semplici clausole:

- **Attribution (BY)**, che impone di dover sempre indicare l'autore
- **Non Commercial (NC)**, che impedisce l'utilizzo per fini commerciali
- **No Derivative Works (ND)**, che impedisce rielaborazioni e ridistribuzioni
- **Share Alike (SA)**, che consente di modificare e ridistribuire fintanto che venga utilizzata la stessa licenza

Oltre a rendere disponibile una semplice pagina web² che in base alle proprie preferenze consiglia la licenza più adatta, al fine di rendere le licenze più accessibili possibile, vengono rese disponibili in 3 diversi "livelli":

- **Legal Code** che consiste nella licenza vera e propria, composta di termini tecnici e giuridici
- **Common Deed** – anche detto "Human Readable" – che contiene un breve riepilogo delle clausole presenti, chiarendo cosa è permesso e cosa no
- **Machine Readable** che rappresenta la licenza in un formato strutturato di facile interpretazione all'interno di un motore di ricerca o di un software

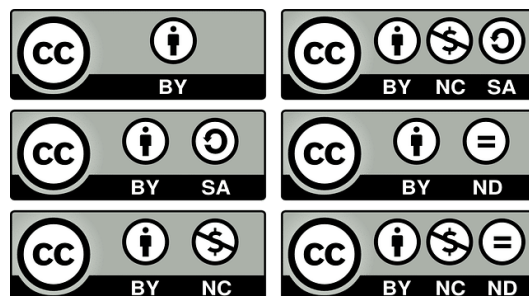


Figura 1.1. Le possibili combinazioni per licenze Creative Commons

²<https://creativecommons.org/choose/>

1.3 La necessità della piattaforma

Attualmente le modalità di accesso e di approvazione per le risorse pubblicate da SapienzaNLP sono molteplici e diverse tra loro: per alcune è necessario effettuare una richiesta tramite email, per altre è disponibile il download diretto tramite la repository su GitHub, ma è anche possibile che sia necessario registrarsi con i propri dati sulla pagina web relativa alla risorsa e successivamente attendere che venga approvata la richiesta. La frammentarietà di questo approccio consente di evidenziare in particolare tre problemi:

- non tutte le modalità di accesso verificano che l'utente approvi esplicitamente le licenze d'uso con cui vengono fornite le risorse
- l'utente può trovarsi a dover gestire più credenziali d'accesso (username e password) per le pagine web da cui scaricare le relative risorse
- la gestione e l'approvazione degli accessi risulta essere complessa

Lo scopo è quindi quello di avere un unico portale per richiedere l'accesso alle risorse, gestire lo stato di approvazione delle richieste e far approvare esplicitamente le modalità con cui è concesso l'utilizzo dei dati.

La piattaforma sviluppata rende possibile visualizzare tutte le risorse disponibili per il download e selezionarne una o più per richiederne l'accesso. Fornisce inoltre un modulo pdf, generato dinamicamente, contenente le clausole specifiche da dover firmare per poter effettuare la richiesta. La possibilità di effettuare il download può essere poi concessa da un amministratore, che revisiona i dati dell'utente e il modulo di richiesta, e può garantire l'accesso alle risorse richieste (o a parte di esse) generando dei link di download utilizzabili solamente dall'utente richiedente.

Capitolo 2

Analisi dei requisiti

2.1 Requisiti della piattaforma

Inizialmente sono stati raccolti ed analizzati i requisiti principali della piattaforma, così da poterne definire ulteriori funzionalità e guidare le successive fasi dello sviluppo. Gli attori¹ del sistema vengono suddivisi nelle seguenti tipologie:

- **Utente web:** usufruisce delle risorse di SapienzaNLP e utilizza la piattaforma per richiederne l'accesso e per scaricarle
- **Utente gestore:** gestisce le richieste effettuate, ne può cambiare lo stato (ad esempio approvandole o rifiutandole) e può generare dei link di download tramite cui scaricare le risorse associate

Selezione delle risorse Gli utenti web possono selezionare una o più risorse tra quelle disponibili nella piattaforma per cui richiedere l'accesso, aggiungendo una descrizione obbligatoria per indicare l'utilizzo che se ne vuole fare. A questa azione segue la generazione di un pdf contenente un resoconto e le informazioni legali che l'utente dovrà obbligatoriamente approvare per poter richiedere l'accesso alle risorse selezionate.

Richiesta di accesso Gli utenti web possono sottoporre una richiesta di accesso alle risorse, inserendo i propri dati e il pdf precedentemente generato, firmato dove richiesto. I dati richiesti all'utente sono i seguenti: nome, cognome, username, email, azienda/organizzazione e paese di provenienza.

Autenticazione amministrazione Gli utenti gestori, possono accedere ad un pannello di amministrazione effettuando l'autenticazione tramite username e password.

Cronologia delle richieste Gli utenti gestori, tramite il pannello di amministrazione, possono accedere allo storico di tutte le richieste di accesso alle risorse effettuate, ordinate cronologicamente.

¹Coloro che interagiscono con il sistema

Dettaglio e accettazione/rifiuto richiesta Gli utenti gestori, tramite il pannello di amministrazione, possono accedere ai dettagli delle richieste effettuate, visualizzarne il modulo pdf associato e cambiare lo stato della richiesta, approvandola o rifiutandola. La cronologia dei cambiamenti di stato, e l'utente che li ha effettuati, viene registrata nel sistema e mostrata insieme ai dettagli della singola richiesta.

Generazione di link per il download Gli utenti gestori, tramite il pannello di amministrazione, possono generare dei link di download per una o più risorse relativamente ad una richiesta di accesso. I link vengono inviati automaticamente all'indirizzo email che l'utente ha specificato in fase di sottomissione della richiesta, tramite cui potrà scaricare le risorse.

2.2 Raffinamento e funzionalità aggiuntive

I requisiti appena elencati rappresentano le funzionalità essenziali per permettere accesso e gestione delle risorse, ovvero lo scopo principale della piattaforma. È però necessario tenere anche conto dei dati relativi ad utenti gestori, alle risorse e alle loro versioni; è quindi seguita una fase di affinamento dei requisiti e delle funzionalità.

2.2.1 Utenti

Viene aggiunta una terza tipologia di utente, l'**Utente amministratore** che può accedere al pannello di amministrazione tramite autenticazione e, oltre alle richieste effettuate, può visualizzare e gestire gli utenti, le risorse e le relative versioni. Può accedere a tutte le funzionalità della piattaforma. Le tipologie presenti nella piattaforma sono quindi le seguenti:

- Utente web
- Utente gestore
- Utente amministratore

2.2.2 Autenticazione

Per semplificare il processo di inserimento dei dati da parte di un utente web che richiede l'accesso alle risorse (v. 2.1) ed evitare che per ogni nuova richiesta debbano essere re-inseriti, l'accesso alla piattaforma viene gestito da una procedura di autenticazione per tutte le tipologie di utenti. In questo modo se uno stesso utente effettua più volte delle richieste, i dati necessari saranno già presenti nel sistema e sarà quindi necessario aggiungervi solamente il modulo pdf firmato.

Registrazione Viene quindi aggiunta la possibilità di registrarsi alla piattaforma inserendo i dati che erano previsti per le richieste di accesso alle risorse. Questa procedura permette di creare un nuovo utente web, che quindi non avrà accesso al pannello di amministrazione.

Recupero password Per tutte le tipologie di utente viene data la possibilità di generare una nuova password, inviata tramite email, che diventa permanente nel momento in cui viene utilizzata per effettuare l'accesso.

Autenticazione Ad esclusione di **Registrazione** e **Recupero password**, per utilizzare le funzionalità della piattaforma relative alla propria tipologia di utente, deve essere prima effettuata l'autenticazione inserendo username e password.

2.2.3 Gestione utenti

Per consentire un controllo completo della piattaforma, vengono aggiunte anche le funzionalità relative alla gestione degli utenti, rese disponibili ai soli utenti amministratori.

Lista utenti esistenti È possibile visualizzare una lista di tutti gli utenti registrati nella piattaforma e avere un resoconto delle relative informazioni principali, come username e nome completo.

Dettaglio utente esistente È possibile visualizzare e modificare i dati degli utenti, ad eccezione della password, e visualizzare lo storico delle attività, nello specifico:

- delle richieste effettuate, per gli utenti web
- dei cambi di stato delle richieste, per gli utenti gestori o amministratori

Disabilitazione utente esistente È possibile disabilitare un utente, negandogli la possibilità di autenticarsi e, di conseguenza, di utilizzare qualunque funzionalità della piattaforma. Questa azione non elimina l'utente ed è reversibile, così che venga sempre mantenuto lo storico delle richieste e dei cambi di stato effettuati.

Creazione utente gestore È possibile creare un nuovo utente gestore inserendo gli stessi dati necessari per la registrazione spontanea di un utente web. Il nuovo utente gestore sarà abilitato ad accedere al pannello di amministrazione, con le sole funzionalità previste di gestione delle richieste.

2.2.4 Gestione risorse

Anche per quanto riguarda le risorse è importante consentire le funzionalità per la loro gestione, rendendole disponibili ai soli utenti amministratori.

Lista risorse esistenti È possibile visualizzare una lista di tutte le risorse disponibili nella piattaforma e avere una panoramica sulle relative informazioni principali.

Dettaglio risorsa esistente È possibile visualizzare in dettaglio e modificare i dati delle risorse, inoltre visualizzare le versioni disponibili associate ad essa.

Eliminazione risorsa esistente È possibile eliminare definitivamente dalla piattaforma una risorsa e con essa tutte le versioni associate.

Creazione nuova risorsa È possibile aggiungere una nuova risorsa, specificando un nome, una descrizione e la licenza con cui viene fornita.

Creazione e modifica versione risorsa È possibile aggiungere e modificare una versione di una risorsa, specificandone il nome, la data di rilascio e il file a cui fa riferimento per il download.

Capitolo 3

Progettazione

3.1 Architettura

Il progetto è basato su una architettura 3-tier [12], ovvero è suddiviso in 3 diversi moduli sviluppati separatamente e dedicati alla gestione dei dati (**Data tier**), alla logica funzionale (Logic o **Application tier**) e all'interfaccia utente (**Presentation tier**); questi comunicano e scambiano dati tra loro con un approccio client-server.

Con questa architettura è possibile raggiungere un buon livello di scalabilità e modularità, evitando di includere la logica dell'applicazione sia a livello del server che lato client (modello 2-tier).

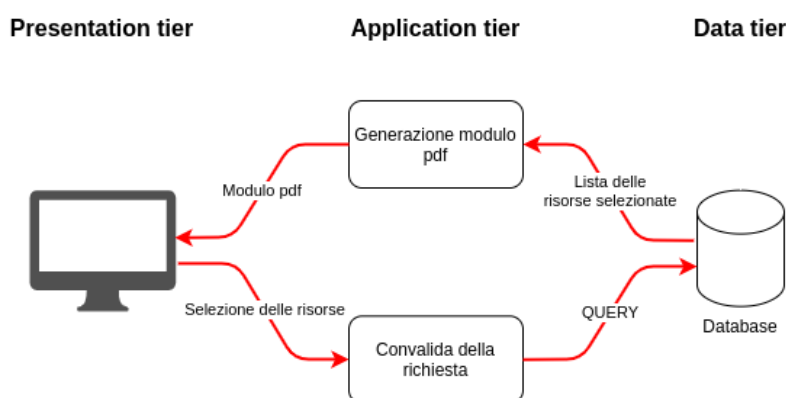


Figura 3.1. Selezione delle risorse con l'architettura 3-tier

Il terzo livello software inserito nel mezzo contiene la maggior parte della logica e si posiziona ad un livello intermedio in modo da essere disponibile ad ogni servizio del sistema. Nel momento in cui è necessario cambiare un componente di logica dell'applicazione è sufficiente modificare solo un "mattoncino" specifico del livello intermedio. Ognuno dei tre livelli architetturali è associato ad una specifica categoria logica:

- Il Presentation tier sarà la UI (User Interface) dei servizi per l'utente, garantendogli la possibilità di interagire con gli altri livelli tramite un'interfaccia grafica, in maniera trasparente

- L'Application tier si occupa della parte applicativa e logica, include l'esecuzione di applicazioni, la logica che si occupa della convalida dei dati e delle richieste/risposte nella comunicazione fra i processi degli altri livelli
- Il Data tier include i servizi che gestiscono il database; che forniscono i dati necessari al completamento delle richieste del livello Logico. In questo caso la logica del server interagisce con la gestione, l'accesso e la sicurezza dei dati

Tra i vantaggi di questa architettura possiamo includere la distribuzione e la riusabilità del codice, in quanto tutte le logiche che vengono utilizzate dagli utenti finali sono nel server intermedio. Un'altra caratteristica è che questa separazione tra l'interfaccia utente, la logica ed i dati permette una maggiore flessibilità nello sviluppo di nuove funzionalità; ad esempio partendo dal presupposto che le componenti server e logiche sono già testate, lo sviluppo di un nuovo componente nella UI sarà più rapido e con una fase di test più breve. In pratica un aggiornamento, il fix di un bug o l'implementazione di una nuova funzionalità di un singolo livello applicativo ha un impatto minimo sugli altri.

L'approccio si riflette anche nella gerarchia delle competenze, in quanto gli eventuali team di sviluppo si possono separare in diverse tipologie per specializzarsi negli ambiti relativi ai 3 livelli dell'architettura: il front-end, il back-end ed il data-end. La scalabilità è un aspetto fondamentale di questa architettura in quanto separando i diversi livelli ognuno può scalare in termini di capacità e potenza indipendentemente dagli altri. Per esempio se stiamo ricevendo un numero elevato di richieste web che non si stanno traducendo in un carico direttamente proporzionale in richieste computazionali che caricano le applicazioni, possiamo scalare solo questo livello.

3.2 Sessione utente

Trattandosi di una piattaforma web le cui funzionalità sono pensate per essere accessibili solo a seguito di autenticazione, la gestione della sessione utente ne è una parte molto importante. Una sessione rappresenta infatti la possibilità che un utente possa usufruire di un servizio in maniera continua, senza doversi autenticare ad ogni operazione che necessita di verificare che si sia in possesso di un account valido o che si abbiano permessi sufficienti.

Approccio basato su token L'approccio utilizzato è quello basato su **token**, per cui a seguito di ogni autenticazione deve essere generato un codice univoco che viene associato all'utente, volto a verificare la sessione. Questo codice, detto *token*, è definito come valido per un breve periodo di tempo, ma viene "rinnovato" ad ogni richiesta effettuata dal client verso l'Application tier, così da garantire la continuità della sessione, fino alla scadenza o al logout esplicito da parte dell'utente. Per ogni richiesta effettuata viene quindi verificato che l'utente sia autenticato e che sia autorizzato ad effettuare eventuali operazioni riservate (ad esempio accedere al pannello di amministrazione).

3.3 Generazione dei moduli pdf

Per quanto riguarda il modulo pdf che l'utente deve firmare per richiedere l'accesso alle risorse, è stato scelto di effettuare una generazione dinamica del contenuto di modo da essere già compilato piuttosto che utilizzare un template con campi vuoti. In questo modo il modulo conterrà le licenze specifiche per le risorse richieste e un resoconto con i dati dell'utente, in quanto già disponibili nel sistema vista la registrazione obbligatoria. Vengono inoltre aggiunti dei metadati relativi alle risorse selezionate e alla descrizione inserita rispetto all'utilizzo previsto.

L'aggiunta dei metadati garantisce di poter effettuare un controllo automatico sulla validità del documento che viene caricato, infatti se questi non sono presenti o risultano in un formato errato, sicuramente il modulo non sarà stato generato dal sistema e di conseguenza si potrà ignorare la richiesta. Salvare i dati inseriti dall'utente nei metadati del pdf permette inoltre di poterne effettuare l'inserimento nel database solo nel momento in cui viene effettuato l'upload e la validazione del documento, che conterrà quindi dati corretti.

3.4 Diagramma ER

Il diagramma in Figura 3.3 mostra una visione generale dei dati gestiti dal sistema e delle relazioni tra di essi. È stato realizzato con **MySQL WorkBench** [8] ed utilizza la notazione *crow's foot* [11], più orientata all'implementazione, in cui le relazioni sono rappresentate da linee che collegano le entità e la cui cardinalità è espressa tramite i simboli presenti alle estremità (v. Figura 3.2). La creazione di questo diagramma ha permesso di avere sempre chiara la struttura che il database stava prendendo e che avrebbe avuto in fase di implementazione, ma soprattutto ha il vantaggio di poter essere esportato direttamente in codice SQL, contenente la creazione delle tabelle e dei vincoli principali.

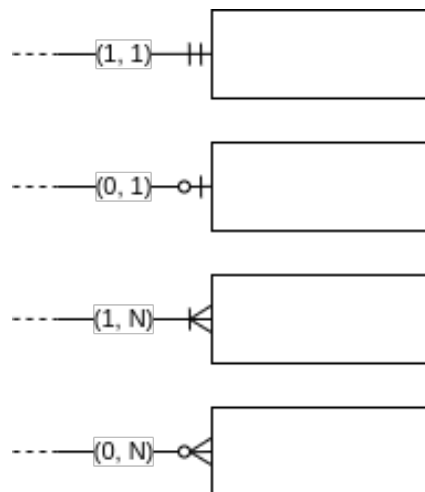


Figura 3.2. notazione *crow's foot*

Non essendo supportati i concetti di IS-A e Generalizzazione, è bene specificare che le tipologie di utenti sono state unificate nella tabella *user* e che la suddivisione è comunque gestita tramite la tabella *user_type*. Le relazioni di *user* con *submission_change* e *download_link* rappresentano le operazioni effettuate dagli utenti gestori/amministratori per quanto riguarda il cambio stato e la generazione di link per scaricare le risorse.

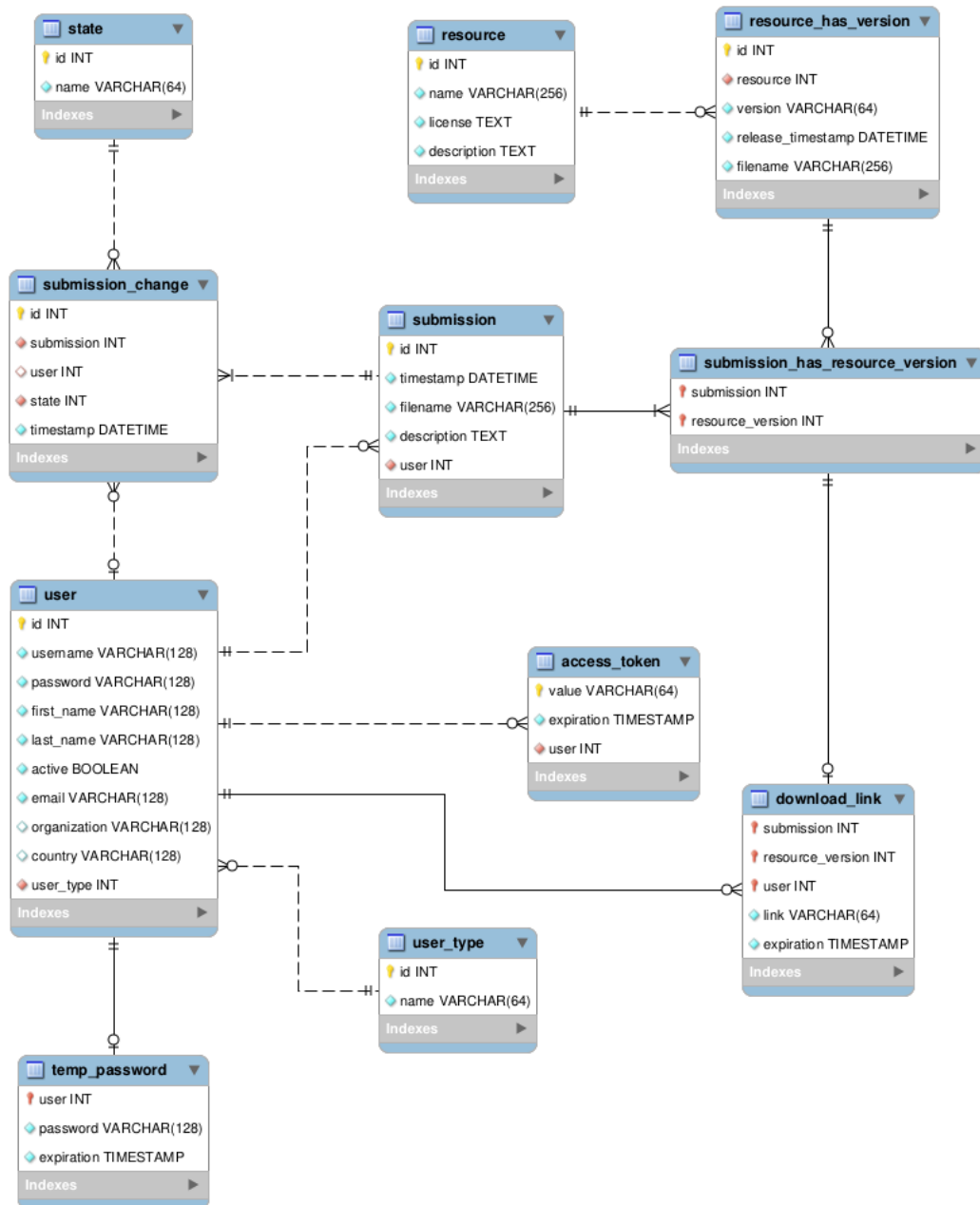


Figura 3.3. Diagramma ER

Capitolo 4

Implementazione

4.1 Data tier

Il database scelto per questo progetto è **MariaDB**, una soluzione open source completa e sviluppata dagli autori di MySQL. Ha la versatilità per supportare carichi di lavoro transazionali, analitici e ibridi, nonché modelli di dati relazionali e JSON. Ha molte delle caratteristiche dei database di fascia alta, come ad esempio la scalabilità per crescere orizzontalmente e la possibilità di svilupparsi in un ambiente completamente distribuito per eseguire milioni di transazioni al secondo ed analisi interattive ad hoc. Il suo design è ideale per la gestione di dati tipici della maggior parte delle applicazioni di database web.

Sono state inoltre sviluppate delle API che consentono di stabilire facilmente una connessione al fine di effettuare modifiche ed interrogazioni sulla base dati, mappando quelle che si definiscono operazioni **CRUD** (Create, Retrive, Update e Delete).

Pattern DAO Per l'implementazione si è utilizzato **Java** come linguaggio di programmazione e il **Design Pattern DAO** (Data Access Object) [13]. Per ogni entità del database sono state quindi definite:

- Una classe **DTO** (Data Transfer Object), che rappresenta la relativa entità e le cui istanze vengono utilizzate per lo scambio dei dati con il database
- Un'interfaccia **DAO** che definisce i metodi che si possono utilizzare per interagire con la relativa entità del database, tra cui le operazioni CRUD necessarie
- Almeno una classe che implementi il DAO, responsabile dell'effettiva connessione con il database e della logica con cui i metodi accedono ai dati

Questa struttura migliora ulteriormente la modularità e la flessibilità del progetto (v. sezione 3.1), in quanto l'Application tier può fare affidamento solamente sui metodi esposti dall'interfaccia DAO, ignorando quale sia la specifica implementazione. In questo modo se ad esempio fosse necessario utilizzare una diversa tecnologia per il database sarebbe sufficiente una nuova implementazione del DAO, che manterrebbe gli stessi metodi e non necessiterebbe quindi di effettuare ulteriori modifiche di adattamento.

```
1 public interface UserDao
2 {
3     void insertUser(User user);
4
5     void updateUser(User user);
6     void updateUserDetails(User user);
7     void updateUserPassword(int id, String password);
8
9     void deleteUser(int id);
10
11     Optional<User> getUser(int id);
12     Optional<User> getUser(String username, String password);
13     Optional<User> getUserByEmail(String email);
14
15     List<User> getUsers();
16 }
```

Snippet 4.1. Interfaccia DAO relativa all'entità *user*

Project Lombok Una libreria open source che è stata ampiamente utilizzata per ridurre i tempi di scrittura e migliorare la leggibilità del codice è **Project Lombok**; un *Annotation processor* che, tramite delle annotazioni nel codice, sostituisce la scrittura di metodi molto comuni e simili tra loro (come *setter* o *getter*). Nelle classi che rappresentano dei dati, come le classi DTO, questo è stato essenziale per poter scrivere in poche righe un codice chiaro e funzionale; ad esempio, la classe **User** (v. 4.2), grazie alle annotazioni **@Getter**, **@Builder** e **@Setter**, dispone di metodi *get* per ogni campo, di un'implementazione del pattern Builder e di un metodo *set* per il campo *id*.

```
1 @Getter
2 @Builder
3 public class User
4 {
5     @Setter
6     private Integer id;
7     private String username;
8     private String password;
9     private String firstName;
10    private String lastName;
11    private String email;
12    private String organization;
13    private String country;
14    private int type;
15    private boolean active;
16 }
```

Snippet 4.2. Classe DTO relativa all'entità *user*

4.2 Application tier

In questo modulo viene gestita la parte logica ed applicativa della piattaforma, tra cui il controllo della sessione utente e i vincoli derivanti dai requisiti; come ad

esempio la possibilità di accedere ad alcune funzionalità solo per certe tipologie di utenti o il controllo per cui i link di download siano validi solo per le richieste approvate e solo per l'utente che ha effettuato la richiesta.

4.2.1 Il framework Spring

Il linguaggio utilizzato è **Java** accoppiato a **Spring**, un framework per lo sviluppo di applicazioni web, nello specifico nella versione **Boot** [10], che ne semplifica la configurazione e garantisce la possibilità di avere un web server *stand-alone* direttamente eseguendo la classe principale del progetto o effettuandone la build, che produrrà un file JAR eseguibile con la stessa funzionalità.

Services Per la gestione dell'invio delle email, della generazione del modulo pdf e dell'interazione con il file system per salvare i moduli inviati e permettere il download delle risorse, sono state definite delle classi annotate come **@Service**, ovvero relative solo a parte della *business logic* dell'applicazione. I Service vengono poi utilizzati nel resto del progetto tramite la **Dependency Injection** di Spring, che si occupa di fornire la stessa istanza a tutti i componenti che ne richiedono l'utilizzo. La forma di injection applicata è quella tramite il costruttore della classe (v. 4.3) che viene parametrizzato automaticamente dal framework all'avvio dell'applicazione.

```
1 @RestController
2 public class AuthenticationController
3 {
4     // [...]
5
6     private final EmailService emailService;
7
8     @Autowired
9     public AuthenticationController(EmailService emailService)
10    {
11        this.emailService = emailService;
12    }
13
14    // [...]
15 }
```

Snippet 4.3. Esempio di Constructor Injection per EmailService

Controllers Tra i componenti principali del framework troviamo le classi Controller, definite tramite l'annotazione **@RestController** e adibite ad eseguire le funzionalità nel rispetto dei vincoli dell'applicazione. I metodi di queste classi vengono mappati a degli endpoint del web server che permettono la loro invocazione tramite richieste HTTP, creando così delle API accessibili tramite l'interfaccia utente.

Ad esempio nella classe **AuthenticationController** è stato implementato un metodo **signUp** (v. 4.4) che permette di effettuare la registrazione di un nuovo utente. Questo metodo, tramite l'annotazione **@PostMapping**, è accessibile effettuando una richiesta POST all'endpoint */signup*.

Nel dettaglio, effettua dei controlli sulla validità dei parametri e sull'unicità di username e indirizzo email, per poi istanziare un nuovo oggetto **User**, inviare

un'email di conferma tramite il Service (v. sezione 4.2.1) che gestisce le email, inserire l'utente nel database tramite un oggetto DAO e infine associare un nuovo `AccessToken` all'utente per garantirgli la sessione.

```

1 @PostMapping(path = "/signup")
2 public ResponseEntity signUp(@RequestParam String username,
   @RequestParam String email, @RequestParam String firstName,
   @RequestParam String lastName, @RequestParam String organization,
   @RequestParam String country)
3 {
4     if (StringUtils.isEmpty(username, email, firstName, lastName,
       organization, country))
5         return RestMessage.badRequest()
6             .body("All fields are mandatory.")
7             .toResponseEntity();
8
9     UserDao userDao = new UserSQL();
10    AccessTokenDAO tokenDAO = new AccessTokenSQL();
11
12    Optional<RestMessage> constraintViolations =
13        getConstraintsViolations(userDao, username, email);
14    if (constraintViolations.isPresent())
15        return constraintViolations.get()
16            .accessToken(accessToken)
17            .toResponseEntity();
18
19    User user = User.builder()
20        .username(username)
21        .email(emailService.checkEmailFormat(email))
22        .firstName(firstName)
23        .lastName(lastName)
24        .password(StringUtils.generatePassword())
25        .organization(organization)
26        .country(country)
27        .type(UserType.DEFAULT.getId())
28        .active(true)
29        .build();
30
31    emailService.sendNewUserEmail(user);
32
33    userDao.insertUser(user);
34
35    Timestamp newExp = TokenUtils.getNewTimestamp(millisRefresh);
36    AccessToken newToken = AccessToken.random(newExp, user.getId());
37
38    tokenDAO.insertAccessToken(newToken);
39
40    return RestMessage.ok()
41        .body(user)
42        .accessToken(newToken)
43        .toResponseEntity();

```

Snippet 4.4. Metodo `signUp` della classe `AuthenticationController`

Filters I filtri possono essere utilizzati per introdurre funzionalità extra prima dell'esecuzione dei metodi di un controller, nel caso in cui vengano invocati tramite

richiesta HTTP. In questo modo è possibile ad esempio implementare meccanismi di logging, autenticare e autorizzare gli utenti o gestire le eccezioni.

La validità delle sessioni utente e l'accesso alle funzionalità riservate per utenti amministratori o gestori, sono state gestite tramite dei pattern negli endpoint; utilizzando ad esempio `/api/admin` come prefisso per tutti quelli su cui era necessario verificare che l'utente fosse un amministratore (v. 4.5).

Definendo delle classi che estendono `OncePerRequestFilter` è stato possibile implementare dei controlli eseguiti prima di ogni richiesta che rispetti il pattern specificato, ritornando un codice di errore nel caso le condizioni non vengano soddisfatte o permettendo l'esecuzione della richiesta altrimenti.

```

1 @Component
2 public class AdminFilter extends OncePerRequestFilter
3 {
4     private final Pattern pattern = Pattern.compile("/api/admin/.*");
5
6     @Override
7     protected void doFilterInternal(HttpServletRequest request,
8         HttpServletResponse response, FilterChain chain)
9     {
10         AccessToken accessToken = TokenUtils.getAttribute(request);
11
12         User user = new UserSQL().getUser(accessToken.getUser()).get();
13         int userType = user.getType();
14         int adminUserType = UserType.ADMIN.getId();
15
16         if (userType == adminUserType) chain.doFilter(request, response);
17         else response.setStatus(HttpServletResponse.SC_FORBIDDEN);
18     }
19
20     @Override
21     protected boolean shouldNotFilter(HttpServletRequest request)
22     {
23         return !pattern.matcher(request.getServletPath()).matches();
24     }
25 }
```

Snippet 4.5. Filtro per verificare che l'utente possa accedere ad un endpoint riservato

4.3 Presentation tier

In questo modulo è stata implementata la **UI** (User Interface), ovvero il mezzo di comunicazione tra l'utente finale e i moduli precedenti. In questo caso si tratta di pagine web che presentano le informazioni e guidano l'utente nell'utilizzo delle funzionalità della piattaforma; che equivalgono a delle richieste HTTP inviate all'Application tier e a delle conseguenti risposte di conferma o di errore.

4.3.1 Approccio component based

Per lo sviluppo si è scelta una logica basata su componenti applicata al paradigma object-oriented, implementata utilizzando **TypeScript**. L'approccio utilizzato è quindi quello di sviluppare delle classi che rappresentino un elemento dinamico

dell'interfaccia, a cui è delegata la generazione della propria struttura HTML, la definizione dello stile grafico CSS e della logica che permette di svolgere una specifica funzionalità.

È stata quindi definita una classe astratta **AbstractWidget** per rappresentare i comportamenti comuni di ogni componente della UI, come ad esempio il modo in cui vengono mostrati eventuali messaggi di errore o in cui viene gestito lo stato di attesa di dati da parte del server.

TypeScript La scelta di TypeScript [3] è ormai quasi uno standard nell'ambito dello sviluppo web. Si tratta di un linguaggio di programmazione open-source che estende la sintassi di JavaScript, rendendo compatibile il codice scritto con quest'ultimo per il compilatore TypeScript. La compilazione in questo caso (detta anche "transpilazione") produce infatti codice JavaScript. Il motivo per cui viene utilizzato questo linguaggio è dovuto alla caratteristica di JavaScript per cui non vengono effettuati controlli statici sui tipi di dato, che vengono convertiti implicitamente. Anche se questa caratteristica è stata uno dei motivi del suo successo, la mancanza di controlli all'atto della compilazione rende più difficile trovare degli errori, soprattutto quando si sviluppano applicazioni di grandi dimensioni o complessità. Tra i motivi principali per utilizzare TypeScript troviamo quindi:

- Possibilità di annotare i tipi di dato con controllo in fase di compilazione
- Inferenza di tipo (type-inference)
- Aggiunta delle interfacce
- Aggiunta degli enum
- Aggiunta dei tipi generici

4.3.2 Richieste AJAX

Per effettuare lo scambio di dati con l'Application tier, e quindi le richieste alle API HTTP, è stato adottato un approccio asincrono tramite richieste **AJAX** (Asynchronous JavaScript and XML). Questo significa che la richiesta non sarà bloccante per l'interfaccia e che i dati saranno aggiornati nel momento in cui arriverà la risposta dal server, senza esplicito ricaricamento della pagina web. Nello specifico ogni componente gestisce e mostra un proprio stato di attesa, così che al completamento delle richieste effettuate si aggiorni dinamicamente in base al responso del server.

Oltre che per semplificare la creazione e la manipolazione degli elementi HTML all'interno dei componenti, è stata utilizzata la libreria jQuery [9] anche per effettuare le richieste AJAX, garantendo la compatibilità cross-browser.

Le Promise in JavaScript Per gestire le molteplici richieste da dover effettuare, è stata definita una funzione generica `apiRequest`, che prende come parametri l'url dell'endpoint a cui effettuare la richiesta e un oggetto facoltativo per specificare eventuali parametri richiesti dalle API. Ad ogni richiesta viene aggiunto il token relativo alla sessione come header di autorizzazione e aggiornato con quello che si

riceve nella risposta del server. Questa funzione ritorna una **Promise** [5], ovvero un oggetto che rappresenta un'operazione che non è stata ancora completata, ma lo sarà in futuro, utile per gestire computazioni asincrone. La richiesta AJAX all'interno del costruttore viene eseguita contemporaneamente alla creazione della Promise, associando il valore ritornato dal server in caso di successo alla funzione `resolve` o l'eventuale errore alla funzione `reject`. In questo modo i widget che necessitano di effettuare delle richieste API, possono utilizzare una funzione specifica che si occupa di effettuare la richiesta AJAX, mentre per gestirne il risultato sarà sufficiente utilizzare i metodi `.then()` e `.catch()` della Promise ritornata, rispettivamente per il caso di successo e il caso di errore. Entrambi i metodi prendono come parametro una funzione che verrà eseguita nel momento in cui l'operazione sarà completata, in questo caso a seguito della risposta del server.

```
1 function apiRequest<T>(url : string, data : Object = {}) : Promise<T>
2 {
3     return new Promise((resolve, reject) => {
4         $.ajax({
5             url,
6             method: "post",
7             data,
8             headers: { Authorization: localStorage.getTokenValue() },
9             success: (res : RestResponse<T>) =>
10                {
11                    localStorage.setToken(res.accessToken);
12                    resolve(res.body);
13                },
14             error: (jqXHR, status, err) =>
15                {
16                    reject(Utils.parseResponse(jqXHR, status, err));
17                }
18            });
19    });
20 }
```

Snippet 4.6. Funzione `apiRequest` per effettuare le richieste AJAX

4.3.3 Le pagine realizzate

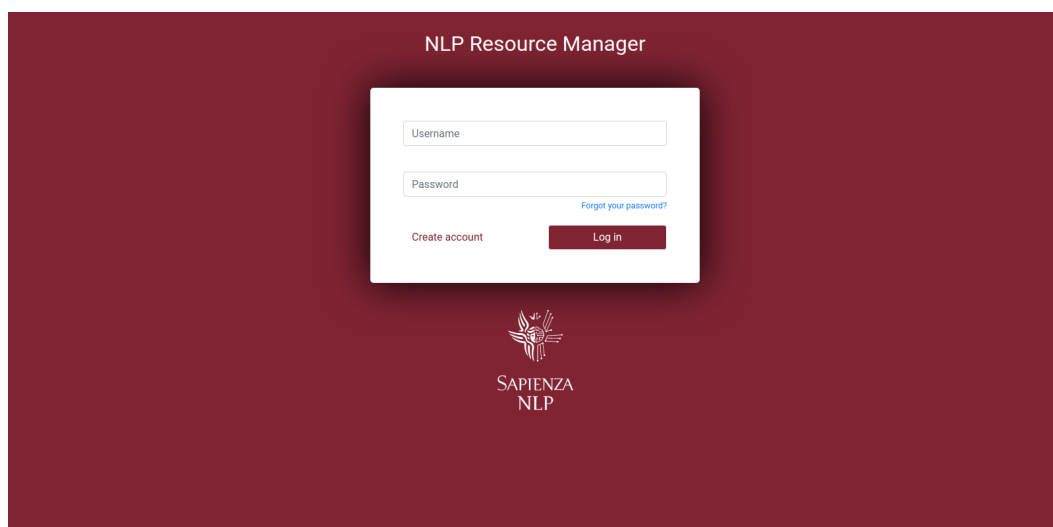
Per quanto riguarda lo stile delle pagine e dei widget è stato utilizzato Bootstrap principalmente per garantire un design responsive tramite il grid-system [1] e per alcune classi CSS relative agli elementi più semplici dell'interfaccia. Inoltre, ad ogni widget è stato associato un file SASS, ovvero un'estensione del classico CSS, per definire gli stili più specifici.

Autenticazione e registrazione

Le schermate e i widget per l'autenticazione e la registrazione implementano un form e la relativa richiesta AJAX. La struttura dei due widget è condivisa, di modo che il bottone per passare da una vista all'altra sia sempre la label sulla sinistra, mentre sulla destra ci sia sempre quello per effettuare l'azione.

Una volta autenticati, si viene reindirizzati alla successiva schermata in base alla tipologia di utente, così che un utente web possa direttamente selezionare le risorse per

cui richiedere accesso (v. sezione 4.3.3), mentre un utente gestore o amministratore possa avere da subito una panoramica sulle richieste (v. sezione 4.3.3).



The screenshot shows the login interface of the NLP Resource Manager. It features a dark red background with the title "NLP Resource Manager" at the top. In the center, there is a white login box containing two input fields for "Username" and "Password". Below the password field is a blue link that says "Forgot your password?". At the bottom of the box are two buttons: "Create account" and "Log in". Below the login box, the Sapienza NLP logo is displayed, consisting of a stylized emblem above the text "SAPIENZA NLP".

Figura 4.1. Autenticazione



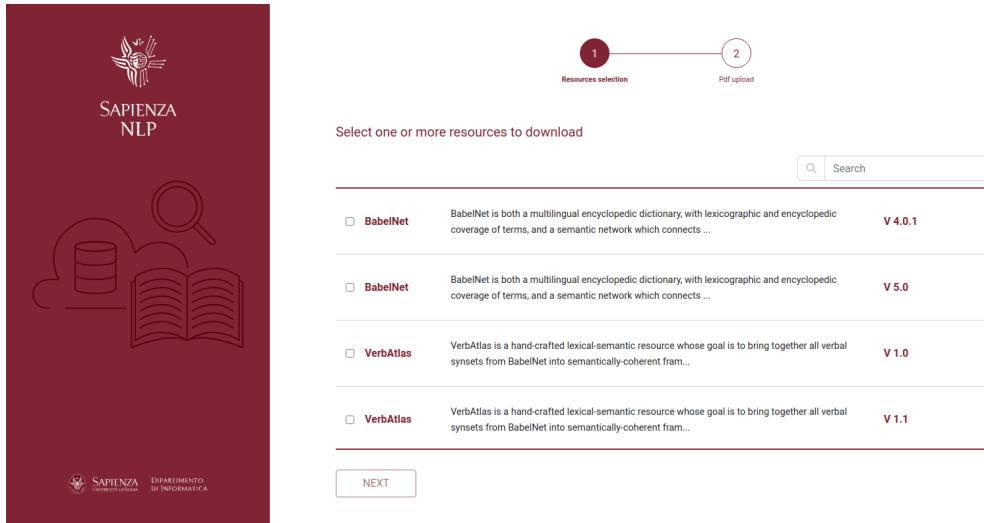
The screenshot shows the registration interface of the NLP Resource Manager. It features a dark red background with the title "NLP Resource Manager" at the top. In the center, there is a white registration box containing several input fields: "First name*", "Last name*", "Username*", "Email address*", "Organization* (e.g. University)", and "Country* (e.g. Italy)". At the bottom of the box are two buttons: "Log in instead" and "Create account". Below the registration box, the Sapienza NLP logo is displayed, consisting of a stylized emblem above the text "SAPIENZA NLP".

Figura 4.2. Registrazione

Selezione delle risorse

Come in tutte le altre schermate della home, l'interfaccia è divisa in due parti: sulla sinistra una sidebar placeholder con i riferimenti Sapienza e sulla destra il widget principale. In questa schermata il widget si occupa di richiedere e mostrare la lista di tutte le risorse disponibili, oltre che di gestirne la selezione multipla e il filtro per parola chiave tramite la barra di ricerca. L'inserimento della descrizione relativa all'utilizzo previsto è gestito dallo stesso, con una transizione delle informazioni mostrate e dei bottoni disponibili.

Una volta confermata l'operazione, viene effettuata una richiesta che, in caso di conferma, ritorna il modulo pdf da dover firmare, scaricato automaticamente dal client. L'upload del pdf firmato sarà poi possibile dalla schermata successiva (v. sezione 4.3.3), accessibile dal menù di navigazione in alto.



SAPIENZA NLP

1 Resources selection 2 Pdf upload

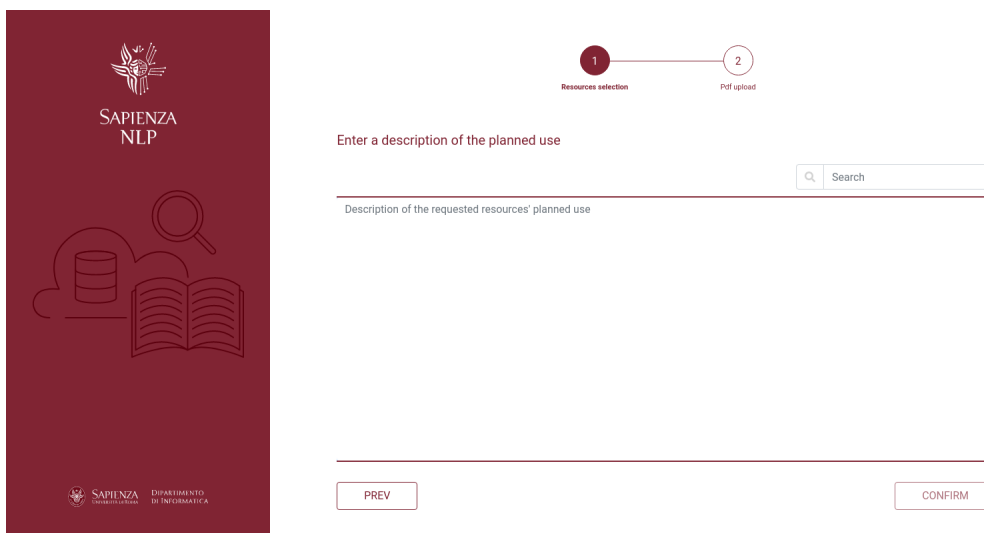
Select one or more resources to download

Search

| | | | |
|--------------------------|-------------------|--|----------------|
| <input type="checkbox"/> | BabelNet | BabelNet is both a multilingual encyclopedic dictionary, with lexicographic and encyclopedic coverage of terms, and a semantic network which connects ... | V 4.0.1 |
| <input type="checkbox"/> | BabelNet | BabelNet is both a multilingual encyclopedic dictionary, with lexicographic and encyclopedic coverage of terms, and a semantic network which connects ... | V 5.0 |
| <input type="checkbox"/> | VerbaAtlas | VerbaAtlas is a hand-crafted lexical-semantic resource whose goal is to bring together all verbal synsets from BabelNet into semantically-coherent fram... | V 1.0 |
| <input type="checkbox"/> | VerbaAtlas | VerbaAtlas is a hand-crafted lexical-semantic resource whose goal is to bring together all verbal synsets from BabelNet into semantically-coherent fram... | V 1.1 |

NEXT

Figura 4.3. Selezione delle risorse



SAPIENZA NLP

1 Resources selection 2 Pdf upload

Enter a description of the planned use

Search

Description of the requested resources' planned use

PREV CONFIRM

Figura 4.4. Inserimento descrizione dell'utilizzo

Upload del modulo pdf firmato

L'ultimo passo per richiedere l'accesso alle risorse è quello di effettuare l'upload del modulo pdf firmato. Questa operazione è stata implementata con un widget per gestire anche il drag and drop e i controlli sulla tipologia e sulla dimensione del file. Una volta effettuato il *submit*, il server si occupa di controllare che il file sia in un

formato valido e che contenga dei metadati di validazione e, in caso di successo, il client mostra una schermata di conferma (v. Figura 4.6).

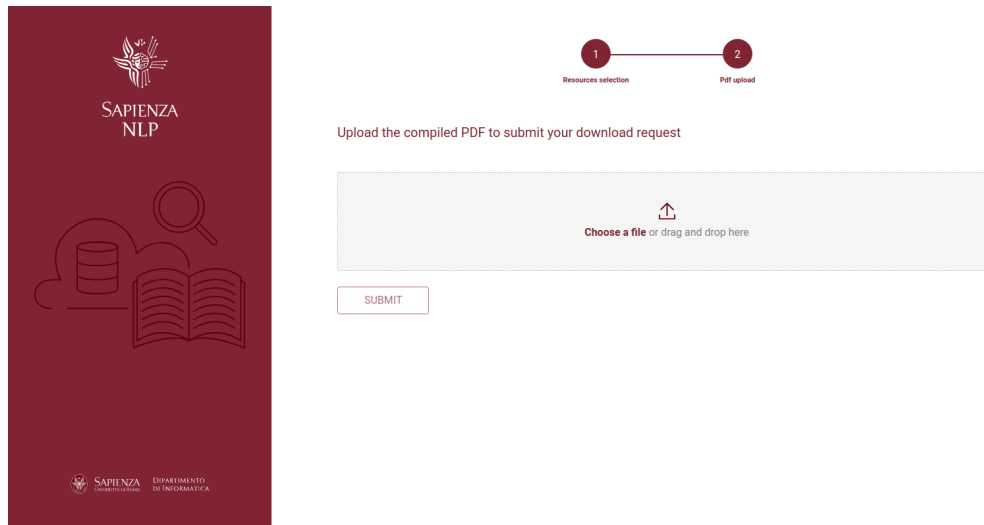


Figura 4.5. Upload modulo pdf firmato

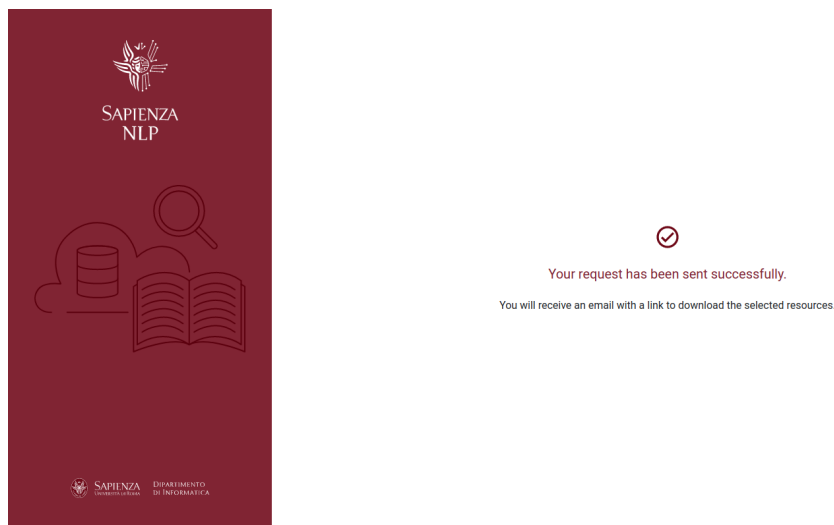


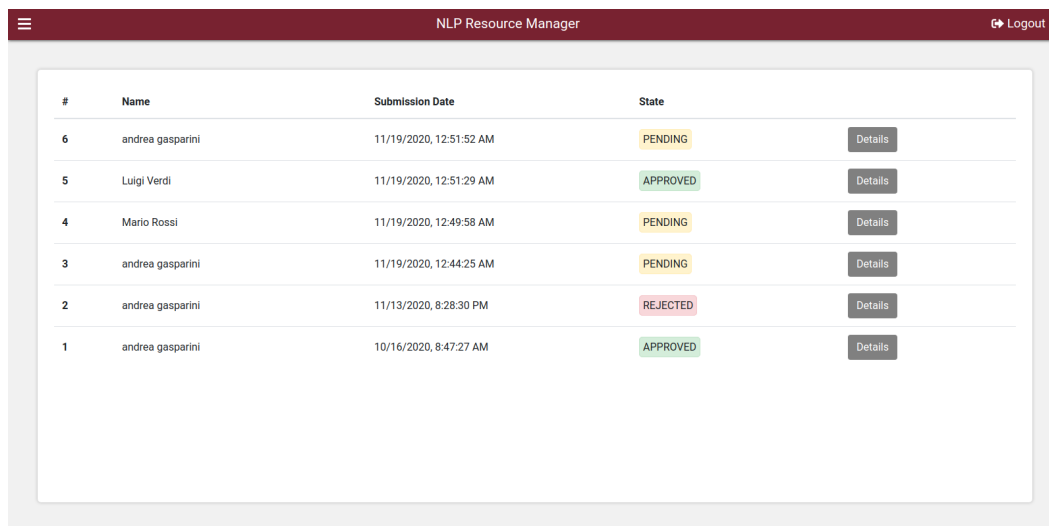
Figura 4.6. Conferma completamento richiesta

Gestione delle richieste effettuate

Nelle schermate di amministrazione l'interfaccia perde la sidebar presente nella home in favore di una barra di navigazione che permette di accedere alle varie sezioni del pannello. L'accesso è limitato a soli utenti autorizzati e in caso il server notifichi una violazione si viene reindirizzati alla schermata di autenticazione. In questo caso sono presenti più widget, ognuno assolve una diversa funzionalità, per cui è possibile:

- Visualizzare tutte le richieste effettuate, con la data di sottomissione lo stato attuale e un bottone per aprirne il dettaglio (v. Figura 4.7)

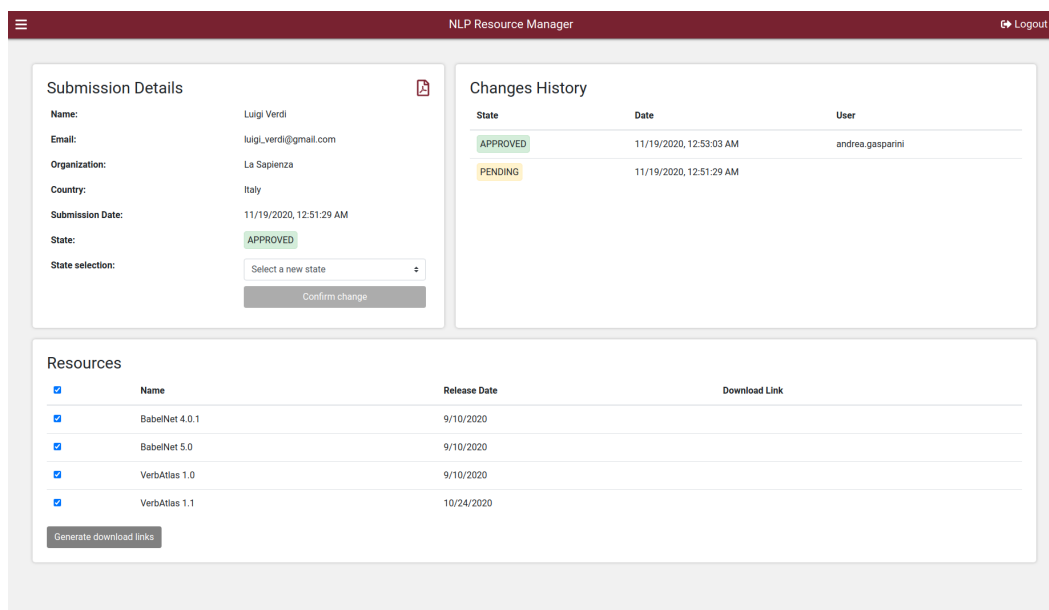
- Visualizzare i dettagli della richiesta, con la possibilità di modificarne lo stato e di visualizzare il modulo pdf caricato dall'utente (v. Figura 4.8)
- Visualizzare lo storico dei cambiamenti di stato, con il relativo utente che ha effettuato l'operazione (v. Figura 4.8)
- Visualizzare la lista delle risorse associate alla richiesta, con la possibilità di selezionare per quali generare i link di download (v. Figura 4.8)



The screenshot shows the 'NLP Resource Manager' interface. At the top, there is a header bar with a menu icon, the title 'NLP Resource Manager', and a 'Logout' button. Below the header is a table listing submissions. The table has columns for '#', 'Name', 'Submission Date', 'State', and a 'Details' button for each row. The submissions are numbered 1 to 6. The states are: 1 (APPROVED), 2 (REJECTED), 3 (PENDING), 4 (PENDING), 5 (APPROVED), and 6 (PENDING).

| # | Name | Submission Date | State | Details |
|---|------------------|-------------------------|----------|---------|
| 6 | andrea gasparini | 11/19/2020, 12:51:52 AM | PENDING | Details |
| 5 | Luigi Verdi | 11/19/2020, 12:51:29 AM | APPROVED | Details |
| 4 | Mario Rossi | 11/19/2020, 12:49:58 AM | PENDING | Details |
| 3 | andrea gasparini | 11/19/2020, 12:44:25 AM | PENDING | Details |
| 2 | andrea gasparini | 11/13/2020, 8:28:30 PM | REJECTED | Details |
| 1 | andrea gasparini | 10/16/2020, 8:47:27 AM | APPROVED | Details |

Figura 4.7. Resoconto di tutte le richieste effettuate



The screenshot shows the 'NLP Resource Manager' interface with the details of a submission. The interface is divided into three main sections: 'Submission Details', 'Changes History', and 'Resources'.

Submission Details: This section contains a form with the following fields:

- Name: Luigi Verdi
- Email: luigi_verdi@gmail.com
- Organization: La Sapienza
- Country: Italy
- Submission Date: 11/19/2020, 12:51:29 AM
- State: APPROVED
- State selection: A dropdown menu with the text 'Select a new state' and a plus icon.
- A 'Confirm change' button.

Changes History: This section shows a table of state changes:

| State | Date | User |
|----------|-------------------------|------------------|
| APPROVED | 11/19/2020, 12:53:03 AM | andrea.gasparini |
| PENDING | 11/19/2020, 12:51:29 AM | |

Resources: This section shows a table of resources:

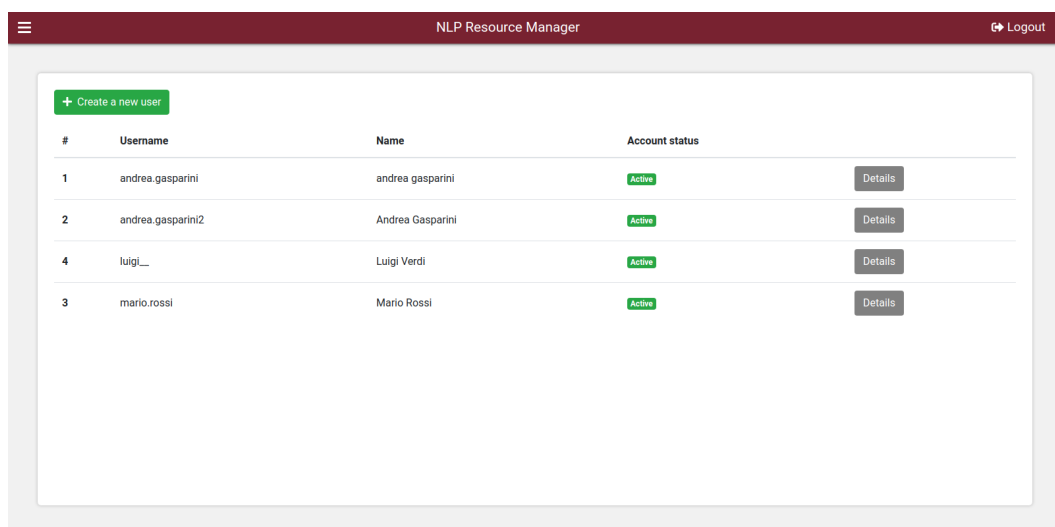
| <input checked="" type="checkbox"/> | Name | Release Date | Download Link |
|-------------------------------------|----------------|--------------|---------------|
| <input checked="" type="checkbox"/> | BabelNet 4.0.1 | 9/10/2020 | |
| <input checked="" type="checkbox"/> | BabelNet 5.0 | 9/10/2020 | |
| <input checked="" type="checkbox"/> | VerbAtlas 1.0 | 9/10/2020 | |
| <input checked="" type="checkbox"/> | VerbAtlas 1.1 | 10/24/2020 | |

Below the table is a 'Generate download links' button.

Figura 4.8. Dettaglio di una richiesta effettuata

Gestione degli utenti

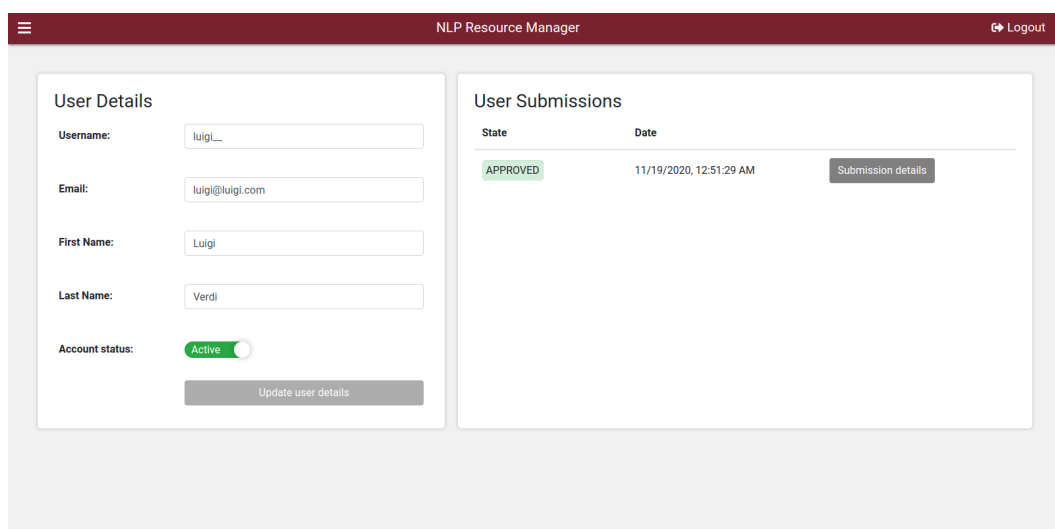
In queste schermate di amministrazione è possibile visualizzare un resoconto di tutti gli utenti registrati e di aprirne un dettaglio, in cui è possibile modificarne i dati e visualizzare la cronologia delle azioni effettuate (richieste sottomesse o cambi di stato in base alla tipologia di utente). Per il widget relativo alla cronologia è stato riutilizzato quello già presente nel dettaglio delle richieste grazie all'approccio modulare basato su componenti che favorisce il riuso del codice. Allo stesso modo, per creare un nuovo utente viene utilizzato lo stesso widget che in Figura 4.10 permette la modifica dei dati di uno già esistente.



The screenshot shows the 'NLP Resource Manager' interface. At the top, there is a header bar with a menu icon, the title 'NLP Resource Manager', and a 'Logout' button. Below the header, there is a green button labeled '+ Create a new user'. The main content area displays a table of users with the following columns: '#', 'Username', 'Name', 'Account status', and a 'Details' button for each row.

| # | Username | Name | Account status | |
|---|-------------------|------------------|----------------|---------|
| 1 | andrea.gasparini | andrea gasparini | Active | Details |
| 2 | andrea.gasparini2 | Andrea Gasparini | Active | Details |
| 4 | luigi_ | Luigi Verdi | Active | Details |
| 3 | mario.rossi | Mario Rossi | Active | Details |

Figura 4.9. Resoconto degli utenti registrati



The screenshot shows the 'NLP Resource Manager' interface with the 'User Details' and 'User Submissions' sections. The 'User Details' section on the left contains form fields for 'Username', 'Email', 'First Name', and 'Last Name', along with an 'Account status' toggle switch and an 'Update user details' button. The 'User Submissions' section on the right displays a table of user submissions with columns for 'State' and 'Date', and a 'Submission details' button for each row.

| State | Date | |
|----------|-------------------------|--------------------|
| APPROVED | 11/19/2020, 12:51:29 AM | Submission details |

Figura 4.10. Dettaglio di un utente

Gestione delle risorse

In queste schermate è stata implementata la gestione dell'elemento cardine della piattaforma, ovvero le risorse. Nel resoconto generale vengono mostrate le risorse registrate nel database con nome e descrizione, oltre alla possibilità di crearne di nuove. La selezione di una di esse permette di accedere al dettaglio, in cui è possibile modificarne i dati, eliminarla definitivamente e aggiungervi una nuova versione o modificare quelle disponibili.

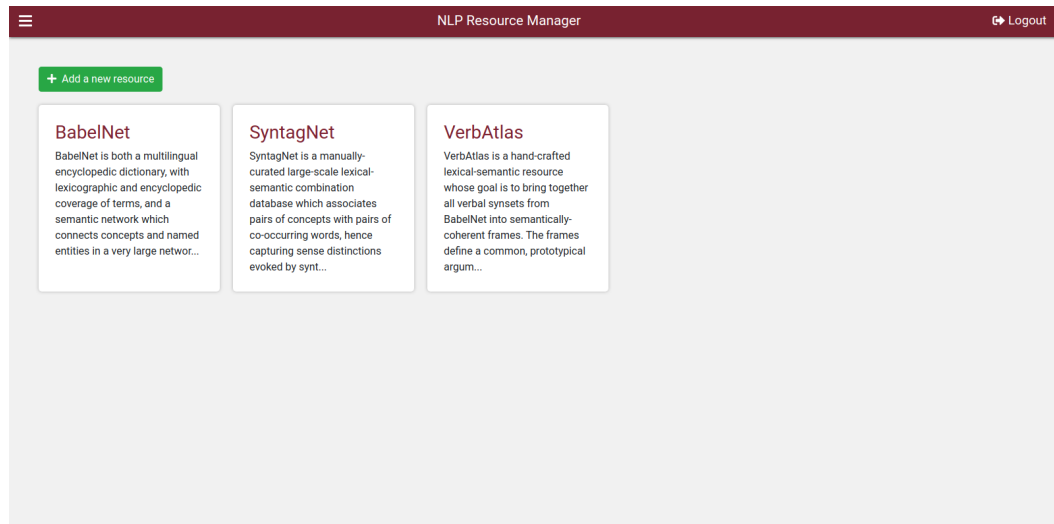


Figura 4.11. Resoconto delle risorse disponibili

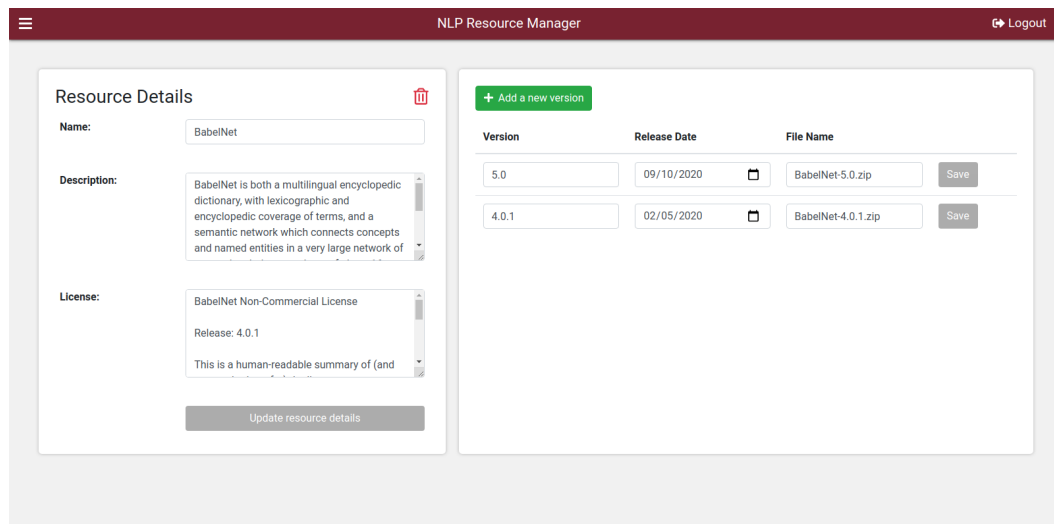


Figura 4.12. Dettaglio di una risorsa

Capitolo 5

Conclusioni

In questa relazione è stato descritto il lavoro svolto durante l'attività di tirocinio, inizialmente presentando la necessità di una piattaforma web che permetta di raccogliere e rendere disponibili in un unico canale le risorse linguistiche di SapienzaNLP, per poi ripercorrere le principali fasi e scelte affrontate.

Prima dello sviluppo di questo tirocinio era possibile che le modalità di accesso alle risorse fossero molto diverse tra loro e questo comportava ovviamente delle difficoltà anche da un punto di vista di gestione. L'utilizzo di queste è inoltre sempre vincolato dal rispetto delle relative licenze, che solitamente venivano semplicemente fornite allegate alle risorse scaricate.

Tra gli aspetti che sono stati migliorati troviamo quindi la possibilità di accedere, tramite un'unica coppia di username e password, a tutte le risorse pubblicate. La stessa piattaforma e la stessa modalità di autenticazione include inoltre anche la parte di gestione di quelle disponibili e di approvazione delle richieste effettuate dagli utenti. In questo modo è stata eliminata la dispersività che era presente in entrambi i casi ed è stato possibile definire delle modalità di accesso condivise per tutte le risorse, così da garantire anche una maggiore flessibilità in caso di eventuale necessità di modifiche.

Un altro miglioramento rispetto alla precedente organizzazione è relativo alla generazione dinamica di un modulo pdf per la richiesta di accesso alle risorse. Da un punto di vista di fruibilità si ha il vantaggio di avere i dati dell'utente inseriti automaticamente, a differenza del classico approccio in cui il modulo è anonimo e richiede la compilazione nonostante il sistema che lo fornisce conosca già i dati necessari. Il modulo risolve inoltre la gestione delle licenze, in quanto quelle relative alle risorse richieste saranno incluse nel pdf, con relativa clausola specifica per approvarle esplicitamente firmando.

L'aver realizzato la piattaforma con un approccio modulare ed estendibile è infine da considerare come il principale vantaggio tecnico. È stata infatti ridotta al minimo la dipendenza tra i vari livelli dell'architettura, così da rendere possibile facilmente future modifiche e nuove implementazioni. Il fatto che i dati vengano messi a disposizione tramite chiamate API lascia massima libertà sulla scelta delle tecnologie per lo sviluppo dell'interfaccia utente, consentendo ad esempio la creazione di un'equivalente applicazione mobile con le stesse funzionalità della versione web. Allo stesso modo la scelta di una nuova tecnologia per il database o per la gestione

della logica applicativa comporterà sempre la sola implementazione del nuovo modulo.

Durante il tirocinio ho avuto modo di applicare alcuni dei concetti appresi nei vari corsi di studio, dalla gestione delle varie fasi di progettazione all'applicazione di buone tecniche e paradigmi di programmazione. Mi ha dato inoltre l'opportunità di imparare e approfondire alcune tecnologie e approcci, migliorando la mia conoscenza ed esperienza di progettazione e sviluppo per il web.

5.1 Sviluppi futuri

Essendosi conclusa insieme al tirocinio anche la fase più sostanziosa della realizzazione della piattaforma, è importante valutare a questo punto quelli che potrebbero essere degli sviluppi futuri per migliorare la qualità e l'utilità del lavoro svolto.

Integrare un'area personale per gli utenti La modalità di accesso alle risorse definita attualmente per la piattaforma prevede che l'utente riceva tramite email i link per il download, limitando l'utilizzo dell'interfaccia alla sola fase di richiesta. Implementando un'area personale, si potrebbe mostrare uno storico delle richieste effettuate, e grazie ad esso risolvere l'eventualità in cui se ne effettua più di una per le stesse risorse, ad esempio nel caso in cui non si è più in possesso dei link di download ricevuti tramite email. Inoltre potrebbe permettere la modifica dei dati inseriti in fase di registrazione, attualmente possibile solo tramite il pannello di amministrazione.

Anonimizzare gli utenti disabilitati La scelta di disabilitare gli utenti piuttosto che eliminarli è stata fatta per garantire consistenza nei dati presenti nel database e mantenere tutte le attività svolte dall'utente nonostante non possa più utilizzare la piattaforma. Questo però preclude la possibilità di cancellare i dati dell'utente (a meno di azione diretta sul database) nel caso si renda necessario ad esempio per esplicita richiesta. Si potrebbe pensare ad un sistema di anonimizzazione dell'utente da cancellare, applicabile anche automaticamente dopo un lasso di tempo definito dalla disabilitazione dell'account.

Possibilità di apporre Firma Digitale per i moduli pdf Una funzionalità che potrebbe migliorare la fruibilità della procedura di richiesta per l'accesso alle risorse e allo stesso tempo far ottenere maggiori garanzie sul modulo firmato è l'integrazione della doppia possibilità di firmare per via autografa o per via di Firma Digitale.

La firma elettronica qualificata (FEQ) – o digitale – sostituisce una tradizionale firma apposta su carta, garantendo anche autenticità, integrità e validità legale dei documenti, poiché un titolare di FEQ può firmare digitalmente i propri documenti solo tramite un certificato digitale di sottoscrizione, rilasciato da enti autorizzati.

Possibilità di autenticarsi tramite Identità Digitale Un'altra integrazione da valutare è la possibilità di effettuare l'autenticazione utilizzando un sistema di identificazione digitale verificato. Prendiamo come esempio **SPID**¹, il Sistema

¹<https://www.spid.gov.it>

Pubblico di Identità Digitale implementato in Italia, che consente di accedere ai servizi della pubblica amministrazione e dei privati aderenti tramite un'identità digitale unica.

La possibilità di autenticarsi utilizzando SPID come sistema di riconoscimento non avrebbe l'obiettivo di eliminare tutta la fase di registrazione (v. sottosezione 2.2.2), ma di offrire un'alternativa all'utente che permetta l'autocompilazione dei dati anagrafici e l'accettazione dei termini di licenza per le risorse richieste senza necessità di firmare un modulo pdf.

Valutare tecnologie alternative Essendo stato lo sviluppo e il testing della piattaforma svolto solamente in locale, in seguito al rilascio ufficiale si potrebbero effettuare delle nuove valutazioni sulle tecnologie utilizzate e sulle scelte effettuate. Ad esempio sulla tipologia di database, piuttosto che sul framework per lo sviluppo dell'Application tier. Ci possono essere molteplici motivi per cui un cambiamento di questo tipo potrebbe rendersi necessario, ad esempio per un discorso di performance o di consumi. Importante è quindi aver progettato il sistema per adattarsi facilmente anche a modifiche che comporterebbero la sostituzione di un intero livello architetturale (v. sezione 3.1).

Bibliografia

- [1] BOOTSTRAP TEAM. *Bootstrap Grid System*. Available from: <https://getbootstrap.com/docs/4.0/layout/grid/>.
- [2] CREATIVE COMMONS. About The Licenses – Creative Commons. Available from: <https://creativecommons.org/licenses/?lang=en>.
- [3] MICROSOFT. *TypeScript: Typed JavaScript at Any Scale*. Available from: <https://www.typescriptlang.org/>.
- [4] MILLER, G. A., BECKWITH, R., FELLBAUM, C., GROSS, D., AND MILLER, K. J. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, **3** (1990), 235. Available from: <http://wordnetcode.princeton.edu/5papers.pdf>.
- [5] MOZILLA AND INDIVIDUAL CONTRIBUTORS. *Promise – JavaScript | MDN web docs*. Available from: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise.
- [6] NAVIGLI, R. Natural language understanding: instructions for (present and future) use. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 5697–5702 (2018).
- [7] NAVIGLI, R. AND PONZETTO, S. P. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, **193** (2012), 217.
- [8] ORACLE CORPORATION. *Database Design and Modeling*. Available from: <https://dev.mysql.com/doc/workbench/en/wb-data-modeling.html>.
- [9] THE JQUERY FOUNDATION. *jQuery API Documentation*. Available from: <https://api.jquery.com/>.
- [10] VMWARE INC. *Spring Boot*. Available from: <https://spring.io/projects/spring-boot>.
- [11] WIKIPEDIA CONTRIBUTORS. Entity–relationship model – Crow’s foot notation — Wikipedia, the free encyclopedia (2020). [Online; accessed 30-October-2020]. Available from: https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model#Crow’s_foot_notation.

- [12] WIKIPEDIA CONTRIBUTORS. Multitier architecture – Three-tier architecture — Wikipedia, the free encyclopedia (2020). [Online; accessed 05-November-2020]. Available from: https://en.wikipedia.org/wiki/Multitier_architecture#Three-tier_architecture.
- [13] WILLIAM CRAWFORD, J. K. *J2EE Design Patterns*. O'Reilly Media, 1 edn. (2003). ISBN 9780596004279,0596004273.