

# Documentazione di progetto

Nicola Panizzolo, Tommaso Soncin, Michael Sarto

CT006 2023/2024

## Contents

<b>1</b>	<b>Introduzione</b>	<b>3</b>
1.1	Tema del progetto . . . . .	3
1.2	Descrizione . . . . .	3
<b>2</b>	<b>Funzionalità principali</b>	<b>4</b>
2.1	Gestione degli account . . . . .	4
2.1.1	Registrazione . . . . .	4
2.1.2	Login . . . . .	4
2.2	Gestione dei progetti . . . . .	4
2.2.1	Caricamento di un progetto . . . . .	4
2.2.2	Creazione degli stati per la valutazione . . . . .	4
2.2.3	Valutazione dei progetti . . . . .	5
2.2.4	Chat . . . . .	5
2.3	Amministrazione . . . . .	6
2.3.1	Ruoli . . . . .	6
2.3.2	Admin Dashboard . . . . .	6
<b>3</b>	<b>Progettazione concettuale e logica della base di dati</b>	<b>6</b>
<b>4</b>	<b>Query principali</b>	<b>6</b>
4.1	Metodologia . . . . .	6
4.1.1	Nessuna Necessità di Richieste Particolari al Database . . . . .	6
4.1.2	Struttura Ottimale del Database . . . . .	6
4.2	Discussione . . . . .	7
4.2.1	Riduzione del Carico di Lavoro del Database . . . . .	7
4.2.2	Facilità di Manutenzione . . . . .	7
4.3	Query Principali . . . . .	7
4.3.1	Creazione di un progetto . . . . .	7
4.3.2	Elenco dei progetti da valutare . . . . .	8
<b>5</b>	<b>Principali scelte progettuali</b>	<b>8</b>
5.1	Trigger . . . . .	8
5.2	Sicurezza . . . . .	9
5.3	Politiche di autorizzazione . . . . .	9
5.4	Performance . . . . .	10

<b>6</b>	<b>Ulteriori informazioni</b>	<b>10</b>
6.1	Backend . . . . .	10
6.1.1	Flask . . . . .	10
6.1.2	Flask Login . . . . .	11
6.1.3	SQLAlchemy . . . . .	11
6.2	Frontend . . . . .	11
6.2.1	Bootstrap 5 . . . . .	11
6.2.2	Jquery . . . . .	11
6.2.3	Dropzone . . . . .	11
6.2.4	TinyMCE . . . . .	11
6.2.5	Socket.IO . . . . .	11
<b>7</b>	<b>Contributo al progetto</b>	<b>12</b>

# 1 Introduzione

## 1.1 Tema del progetto

Viene richiesto di curare il design e l'implementazione di una web application per la valutazione interna dei progetti di ricerca da sottoporre per finanziamento all'Unione Europea. I ricercatori devono poter creare nuovi progetti da sottoporre a valutazione, ciascuno dei quali comprende una descrizione testuale ed uno o più documenti in formato PDF da valutare, possibilmente di tipo diverso (data management plan, ethics deliverable, ecc.). Ciascun progetto ha uno stato, fra cui:

- approvato: i valutatori hanno espresso parere favorevole e non sono più possibili modifiche;
- sottomesso per valutazione: sottoposto per la prossima finestra di valutazione, ma non ancora valutato;
- richiede modifiche: i valutatori hanno richiesto modifiche in vista di un'ulteriore valutazione;
- non approvato: i valutatori hanno espresso parere contrario e non sono più possibili modifiche;

I valutatori possono accedere ai diversi progetti da valutare nella prossima finestra di valutazione, scaricare i documenti da valutare e creare un report di valutazione per ognuno di essi, oltre ad aggiornare lo stato del progetto, per esempio ad "approvato". I ricercatori hanno accesso ai report di valutazione e possono rivedere i loro progetti alla luce degli stessi, finché non vengono approvati o rifiutati.

Vengono forniti alcuni spunti possibili per arricchire lo scenario, senza pretesa di esaustività:

- Ogni progetto ha uno storico di versione per tenere traccia delle diverse iterazioni del processo di valutazione: in particolare, per ogni documento del progetto, i ricercatori ed i valutatori devono poter accedere a tutte le versioni sottomesse precedentemente ed al relativo report di valutazione. Le versioni possono includere informazioni ausiliarie, che dettagliano i cambiamenti rispetto alle versioni precedenti.
- Ogni progetto include una componente di messaggistica, tramite la quale i ricercatori ed i valutatori possono interagire. Per esempio i ricercatori possono chiedere ulteriori chiarimenti riguardo ad una valutazione ricevuta, a cui i valutatori possono rispondere in maniera anonima.
- I valutatori possono inserire commenti puntuali direttamente all'interno dei PDF sottomessi per valutazione, per esempio in forma di note. In questo modo i report di valutazione possono fare riferimento anche a tali commenti puntuali (es. "si veda la nota a pagina 4").

## 1.2 Descrizione

L'applicazione è una piattaforma completa per la gestione della valutazione dei progetti di ricerca. L'obiettivo principale di questa piattaforma è fornire ai ricercatori uno strumento efficiente per la valutazione dei progetti.

Una delle caratteristiche chiave dell'applicazione è la possibilità di inserire i propri progetti all'interno del sistema, consentendo ad altri utenti autorizzati di eseguire la valutazione.

Ogni progetto ha più stati visibili all'autore e a tutti gli utenti autorizzati, in modo da tenere traccia dell'avanzamento della valutazione.

La piattaforma offre una serie di funzionalità per facilitare il processo di valutazione, come la possibilità di scrivere note e caricare correzioni.

L'obiettivo finale dell'applicazione è migliorare l'efficienza nel processo di valutazione dei progetti di ricerca, fornendo una soluzione completa e intuitiva per gli utenti.

## 2 Funzionalità principali

Di seguito forniremo una breve descrizione delle principali funzionalità implementate nell'applicazione.

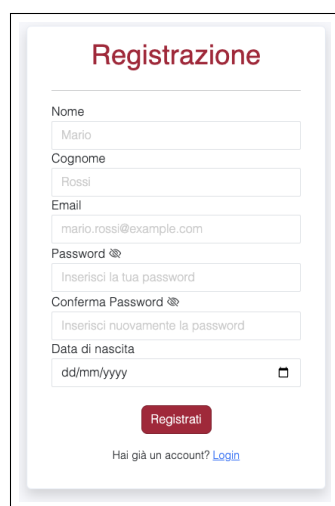
### 2.1 Gestione degli account

#### 2.1.1 Registrazione

L'utente ha la possibilità di registrarsi autonomamente tramite un form in cui inserire i propri dati ed effettuare la creazione della password, inizialmente tutti gli utenti sono registrati come ricercatori, se dovesse essere necessaria la modifica di un ruolo sarà compito dell'amministratore;

#### 2.1.2 Login

Dopo aver effettuato la registrazione rispettando i vari criteri per mail e password l'utente potrà effettuare il login all'interno dell'applicazione.





**Registrazione**


Nome  
Mario

Cognome  
Rossi

Email  
mario.rossi@example.com

Password   
Inserisci la tua password

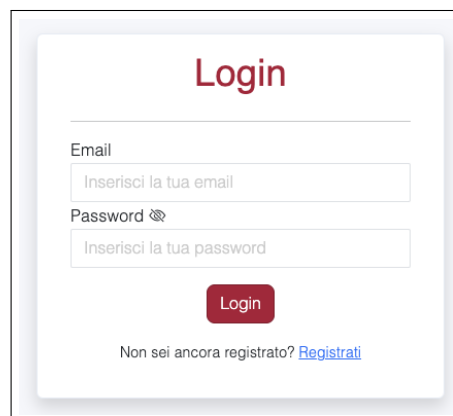
Conferma Password   
Inserisci nuovamente la password

Data di nascita  
dd/mm/yyyy 

**Registrati**


Hai già un account? [Login](#)

Figure 1: Screenshot registrazione



**Login**

Email  
Inserisci la tua email

Password   
Inserisci la tua password

**Login**

Non sei ancora registrato? [Registrati](#)

Figure 2: Screenshot login

### 2.2 Gestione dei progetti

#### 2.2.1 Caricamento di un progetto

L'utente ha la possibilità di caricare un progetto allegando multipli file *.pdf*, per concludere l'operazione è necessario selezionare il tipo di progetto da un menù a tendina, inserire una breve descrizione e facoltativamente scrivere una nota a riguardo.

#### 2.2.2 Creazione degli stati per la valutazione

I possibili stati di un progetto sono rimasti gli stessi descritti nel tema del progetto (Submitted, Require Changes, Approved, Not Approved), nella visualizzazione di un progetto l'utente può osservare ogni stato di tale progetto, al quale vengono affiancate 3 informazioni: i file allegati, le note del valutatore e la data.

## 2.2.3 Valutazione dei progetti

Un utente valutatore può cambiare lo stato dei progetti aggiungendone uno nuovo, per farlo deve caricare uno o più file *.pdf*, così da poter mantenere il file originale e magari aggiungerne un altro con le eventuali correzioni, dopo aver caricato i file può selezionare il nuovo stato da un menù a tendina e aggiungere delle eventuali note, come indicato dal tema del progetto.

## 2.2.4 Chat

In ogni progetto è presente una chat sulla quale possono scrivere il creatore del progetto e gli eventuali valutatori.

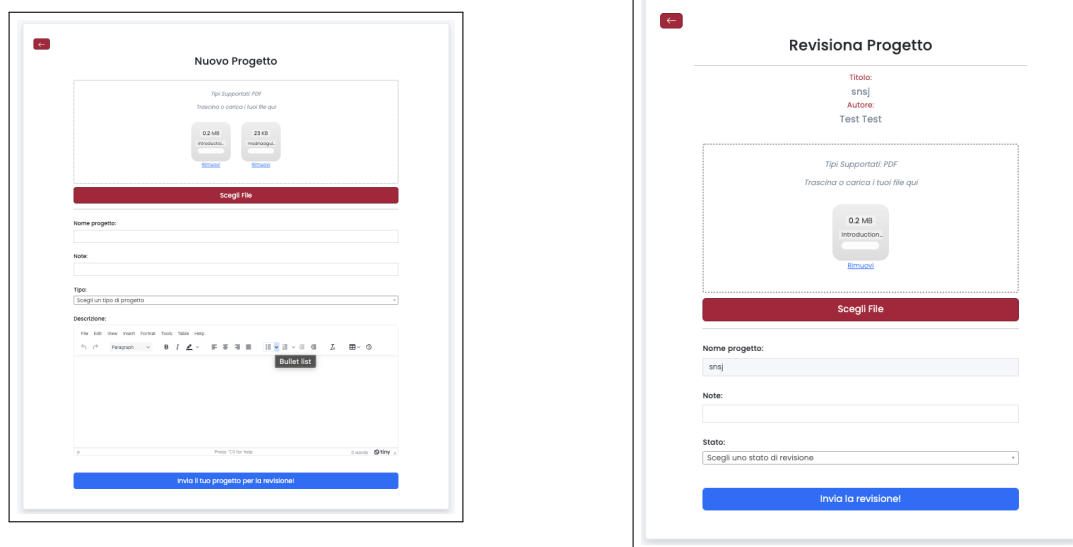


Figure 3: Pagine per la creazione e la valutazione di un progetto

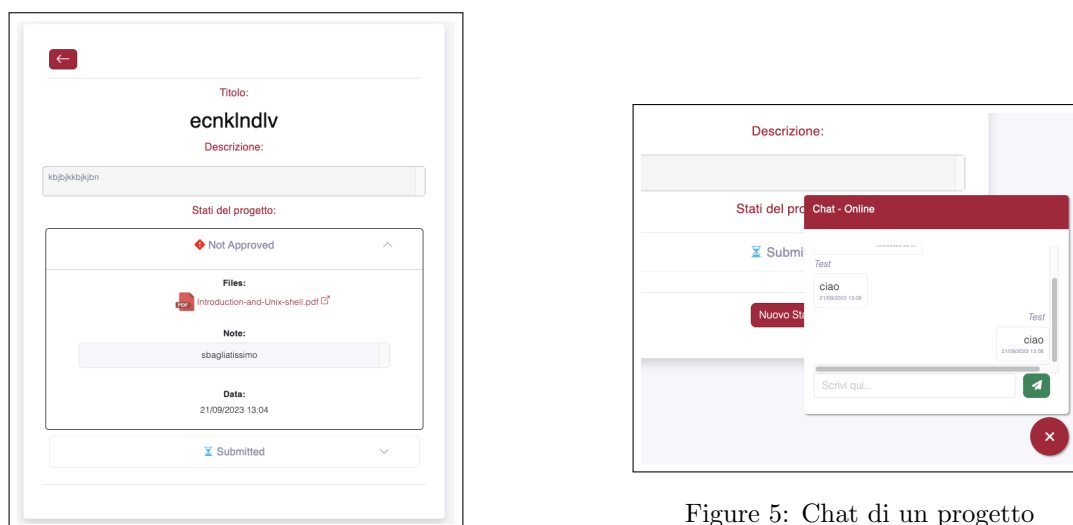


Figure 5: Chat di un progetto

Figure 4: Esempio di visualizzazione degli stati di un progetto

## 2.3 Amministrazione

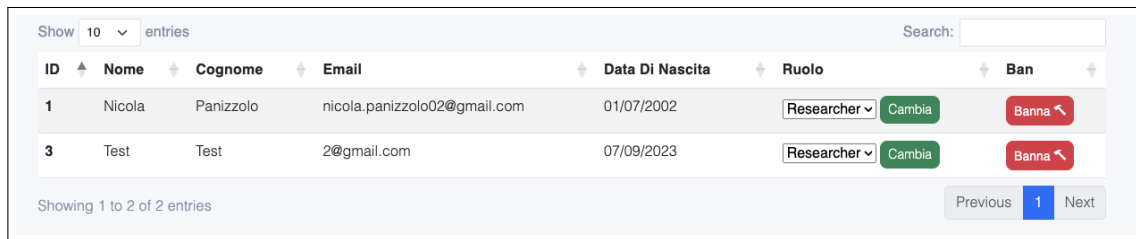
### 2.3.1 Ruoli

L'assegnazione dei privilegi agli utenti dell'applicazione si basa su 3 ruoli:

- Researcher: è il ruolo con meno privilegi, può solo creare i propri progetti senza valutarne altri;
- Reviewer: è un Researcher che ha anche la possibilità di valutare i progetti degli altri utenti;
- Admin: può accedere alla Admin Dashboard e può anche valutare i progetti, questo ruolo viene assegnato dall'amministratore di sistema.

### 2.3.2 Admin Dashboard

Questa pagina consiste in una tabella interattiva con la quale l'utente admin può avere accesso ai dati anagrafici dei singoli utenti, modificarne il ruolo ed eventualmente rimuovere gli utenti dal sistema.



ID	Nome	Cognome	Email	Data Di Nascita	Ruolo	Ban
1	Nicola	Panizzolo	nicola.panizzolo02@gmail.com	01/07/2002	Researcher	Banna
3	Test	Test	2@gmail.com	07/09/2023	Researcher	Banna

Figure 6: Admin Dashboard

## 3 Progettazione concettuale e logica della base di dati

## 4 Query principali

### 4.1 Metodologia

Nel corso del nostro progetto, abbiamo scelto di utilizzare query molto semplici per interagire con il database. Questa scelta è stata guidata da due considerazioni principali.

#### 4.1.1 Nessuna Necessità di Richieste Particolari al Database

Inizialmente, abbiamo valutato le nostre esigenze di accesso ai dati e abbiamo constatato che non vi era alcuna necessità di richieste particolari o complesse. Le informazioni di cui avevamo bisogno potevano essere recuperate in modo efficace attraverso query dirette e semplici. L'assenza di complessità nelle nostre richieste ha reso inutile l'uso di query più intricate.

#### 4.1.2 Struttura Ottimale del Database

Un elemento chiave che ha influenzato la nostra scelta è stata l'ottima struttura del database. Le tabelle erano organizzate in modo chiaro e logico, con attributi ben definiti e relazioni ben gestite tra di loro. Questa struttura ha semplificato notevolmente l'interrogazione del database.

## 4.2 Discussione

Grazie all'adozione di query semplici, siamo stati in grado di ottenere risultati soddisfacenti nel nostro progetto. La struttura ben progettata delle tabelle del database è stata fondamentale in questo processo. Inoltre l'astrazione delle tabelle tramite i Model di Flask ha permesso di ridurre l'utilizzo dei join solo nel momento in cui strettamente necessario. Di seguito, discutiamo alcune delle principali ragioni per cui abbiamo adottato questo approccio.

### 4.2.1 Riduzione del Carico di Lavoro del Database

L'uso di query semplici ha comportato un carico di lavoro minore sul database, riducendo il tempo di esecuzione delle operazioni, in quanto vengono sfruttati gli indici generati tramite le foreign keys.

### 4.2.2 Facilità di Manutenzione

La gestione di query semplici è stata vantaggiosa anche dal punto di vista della manutenzione. Le query erano facilmente comprensibili e modificabili, semplificando il processo di adattamento alle nuove esigenze del progetto.

## 4.3 Query Principali

### 4.3.1 Creazione di un progetto

La creazione di un progetto risulta essere una sequenza di query che vanno ad effettuare l'insert in più tabelle.

1. Utilizziamo la query seguente per produrre il menù a tendina dove l'utente può selezionare il tipo del progetto che sta andando a creare.

```
SELECT id, name FROM types
```

Con la prossima query avviene l'effettiva creazione del progetto.

2. 

```
INSERT INTO projects( id_user, id_type, name, description)
VALUES (1,1, 'TEST', 'TEXT')
```

Per ottenere l'id\_user abbiamo utilizzato l'oggetto current\_user della classe flask\_login istanziato sulla base del risultato della query di login:

```
SELECT * FROM users WHERE email='test@test.it'
```

3. Tramite flask abbiamo quindi la possibilità di ottenere l'id della row appena creata nella tabella projects.
4. Otteniamo l'id stato "Submitted" tramite una query (ci servirà come valore di default).

```
SELECT id FROM states WHERE name='Submitted'
```

5. Andiamo a creare lo stato (history) per questo nuovo progetto. Ci permetterà di avere uno storico di tutti gli stati del progetto.

```
INSERT INTO project_history(id_project, id_state, id_user_reviewer)
VALUES (1,2,1)
```

6. Tramite flask abbiamo quindi la possibilità di ottenere l'id della row appena creata nella tabella projects.history.
7. Andiamo infine a definire il path di caricamento dei file per questo progetto in questo preciso stato.

```
INSERT INTO project_files(path, id_project_history) VALUES
('/app/db_files/1/1/1/TEST1.pdf',1),
('/app/db_files/1/1/1/TEST2.pdf',1),
('/app/db_files/1/1/1/TEST3.pdf',1)
```

### 4.3.2 Elenco dei progetti da valutare

Abbiamo avuto l'esigenza di dover mostrare ai reviewer solo i progetti che non sono stati chiusi affinché possano avere la possibilità di valutarli.

1. 

```
SELECT * FROM users WHERE id=1 AND id_role IN
(SELECT id FROM roles WHERE is_reviewer=TRUE)
```

Se questa query (dove l'id corrisponde all'id dell'utente che ha effettuato il login) non ritorna alcuna riga blocchiamo l'accesso alla pagina in quanto l'utente non è autorizzato

2. 

```
SELECT *
FROM projects
WHERE id IN
(SELECT id_project
FROM project_history
WHERE id IN
(SELECT MAX(id)
FROM project_history
WHERE id_project NOT IN
(SELECT id FROM projects WHERE id_user = 1)
GROUP BY id_project)
AND id_state IN
(SELECT id
FROM states
WHERE is_closed = FALSE))
```

Questa query ci permette di avere l'elenco di tutti i progetti, senza duplicati, che non sono stati creati dall'utente loggato (non vogliamo che un reviewer si possa approvare un progetto in modo autonomo) e che possiedono come ultimo stato attribuito (history) uno stato con flag "non chiuso".

3. Andiamo quindi a prendere i tipi di progetto disponibili nel software, questo permetterà al backend in python di raggruppare gli oggetti Project in sezioni del dizionario Types divise per Project.id\_type ai fini una stampa ordinata per tipo:

```
SELECT id, name FROM types
```

## 5 Principali scelte progettuali

### 5.1 Trigger

Il trigger **is\_reviewer\_trigger** implementa un rigoroso controllo di validazione per garantire che gli utenti designati come revisori siano effettivamente qualificati come tali prima di consentire loro di apportare inserimenti o aggiornamenti alla tabella project\_history. Inoltre estende la possibilità di inserimento per un dato progetto agli utenti che non sono qualificati come revisori ma sono autori dello stesso. Questa funzione rappresenta una soluzione efficace per garantire l'integrità dei dati e la conformità alle regole imposte dal sistema di valutazione.

```
CREATE OR REPLACE FUNCTION is_reviewer()
RETURNS TRIGGER AS
$$
BEGIN
IF NEW.id_user_reviewer IS NOT NULL THEN
IF (NEW.id_user_reviewer NOT IN
```



```

(SELECT u.id
FROM users u JOIN roles r ON u.id_role=r.id
WHERE r.is_reviewer AND u.id=NEW.id_user_reviewer) AND
(NEW.id_user_reviewer != (SELECT id_user FROM projects WHERE id=NEW.id_project))
) THEN
RETURN NULL;
END IF;
END IF;
RETURN NEW;
END $$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER is_reviewer_trigger
BEFORE INSERT OR UPDATE ON project_history
FOR EACH ROW EXECUTE FUNCTION is_reviewer()

```

## 5.2 Sicurezza

È stata adottata una soluzione avanzata basata su Docker per garantire un elevato grado di isolamento e controllo dell'accesso al database e ogni dato viene analizzato prima di essere inserito nella base di dati. In particolare il progetto, sotto il punto di vista della sicurezza, tocca i seguenti punti:

- Container Docker Isolato: Per garantire un ambiente altamente sicuro, è stato creato un contenitore Docker isolato dal resto della rete. L'isolamento è fondamentale per proteggere il database da accessi non autorizzati in particolare:
  - Accesso Limitato agli Utenti: Solo due utenti hanno accesso a questo database, il superutente (root) e un utente specifico denominato "progetto";
  - Network Isolata: Il container Docker è configurato in modo da essere completamente isolato dalla rete esterna. Questo significa che il database non è accessibile dalla rete pubblica o da altre risorse di rete, garantendo una protezione aggiuntiva contro attacchi esterni;
  - Controllo Preciso dell'Accesso: limitare l'accesso al database solo al container Python impedisce ad altri servizi o applicazioni non autorizzati di interagire direttamente con il database.
- Sanitizzazione delle Query: tutte le query eseguite nel database sono sanificate attraverso SQLAlchemy ORM:
  - Prevenzione degli Attacchi SQL Injection: L'ORM SQLAlchemy è progettato per prevenire gli attacchi di SQL injection, in quanto sanitizza automaticamente le query, impedendo l'inserimento di codice SQL dannoso;
  - Prevenzione degli Attacchi XSS: dove l'ORM non può arrivare siamo intervenuti manualmente a sanificare i dati in modo che i testi stampati nel DOM (descrizione dei progetti, note e messaggi della chat) fossero sanificati anche da iniezioni di codice JavaScript.

## 5.3 Politiche di autorizzazione

La tabella "Roles" è progettata per contenere informazioni sulle tipologie di ruoli assegnabili agli utenti del sistema. Ogni utente è associato a un ruolo specifico attraverso un campo "id\_role". La gestione delle politiche di autorizzazione per la valutazione dei progetti è implementata attraverso la tabella "Roles" in questo modo:

- Flag per Valutazione dei Progetti: all'interno della tabella "Roles", è presente un flag che indica se gli utenti con quel ruolo specifico hanno il permesso di valutare i progetti;
- Ruolo dell'Amministratore(id\_role=1): L'utente con "id\_role=1", ovvero l'amministratore, è l'unico che può:

- Cambiare i ruoli degli altri utenti: questo privilegio consente all'amministratore di assegnare o revocare il diritto di valutare i progetti ad altri utenti;
- Bannare: in caso di comportamenti ritenuti non corretti ha la possibilità di bloccare l'accesso agli utenti incriminati.
- Privilegi Amministrativi: La gestione dei privilegi amministrativi è stata configurata in modo da impedire che un utente possa promuovere(se stesso o altri) a ruolo di amministratore. Per farlo è necessario un intervento diretto da parte degli sviluppatori del sistema all'interno del database, previa esplicita richiesta.

## 5.4 Performance

Nella progettazione e nell'utilizzo del database, abbiamo adottato una strategia di ottimizzazione basata sull'utilizzo di indici generati dalle foreign key delle tabelle del database e sfruttato il "lazy loading" implicito offerto da SQLAlchemy per massimizzare le performance.

In modo specifico, gli indici Foreign Key consentono quanto segue:

- Join Efficienti: Quando si eseguono operazioni di join tra tabelle che condividono chiavi esterne, la presenza di indici FK riduce notevolmente il tempo necessario per effettuare join;
- Mantenimento dell'Integrità dei Dati: Gli indici FK impediscono l'inserimento di dati non validi nelle tabelle. Ciò contribuisce a garantire che i dati siano coesi e che rispettino le relazioni previste.

Mentre il Lazy Loading Implicito di SQLAlchemy offre i seguenti vantaggi:

- Accesso On-Demand: Quando si accede a un oggetto collegato tramite una relazione, SQLAlchemy carica solo quei dati dal database che sono necessari in quel momento, evitando il caricamento eccessivo di dati inutilizzati;
- Riduzione del Carico di Lavoro del Database: L'uso del lazy loading impedisce al database di eseguire query pesanti in modo prematuro, contribuendo a ridurre il carico di lavoro del database e a migliorare le prestazioni complessive del sistema.

## 6 Ulteriori informazioni

Per la realizzazione dell'applicazione sono state utilizzate diverse librerie e framework, alcuni per lo sviluppo del backend e alcuni di frontend.

### 6.1 Backend

#### 6.1.1 Flask

Flask è un micro-framework web leggero e flessibile per la creazione di applicazioni web in Python. È progettato per essere semplice da imparare e utilizzare, consentendo agli sviluppatori di creare rapidamente applicazioni web con un minimo sforzo. Flask offre funzionalità di base per la gestione delle richieste HTTP, la creazione di route, la gestione dei template HTML e l'integrazione di database, ma è altamente estendibile grazie all'ampia disponibilità di estensioni e plug-in. Flask è particolarmente adatto per progetti più piccoli o applicazioni web leggere, ed è ampiamente utilizzato per la creazione di servizi web, API e prototipi.

### 6.1.2 Flask Login

Flask-Login è un'estensione per il framework web Flask in Python che semplifica l'implementazione dell'autenticazione utente in un'applicazione web. Questa estensione fornisce funzionalità per la gestione delle sessioni utente, la verifica delle credenziali, il riconoscimento degli utenti autenticati e la protezione delle route o delle risorse dell'applicazione che richiedono l'autenticazione.

### 6.1.3 SQLAlchemy

SQLAlchemy è una libreria Python molto popolare per la gestione delle operazioni di accesso e manipolazione dei database relazionali. Essa fornisce un'interfaccia ad alto livello per comunicare con database SQL, rendendo più semplice e astratta l'interazione con i database da parte degli sviluppatori.

## 6.2 Frontend

### 6.2.1 Bootstrap 5

Bootstrap 5 è una delle versioni più recenti del framework di front-end open-source Bootstrap, utilizzato per la progettazione e lo sviluppo di siti web e applicazioni web. In sintesi Bootstrap 5 fornisce un set di strumenti e componenti per la creazione di siti web e applicazioni web moderni, reattivi e personalizzabili. Le sue caratteristiche di personalizzazione, design responsive e utilizzo semplice lo hanno reso la chiave per lo sviluppo front-end dell'applicazione.

### 6.2.2 JQuery

Jquery è una libreria JavaScript open-source ampiamente utilizzata per semplificare la manipolazione del DOM (Document Object Model) e l'interazione con gli elementi HTML all'interno di pagine web, nel progetto è stata fondamentale per semplificare lo sviluppo di alcune funzioni secondarie.

### 6.2.3 Dropzone

Nelle pagine dove gli utenti caricano i progetti viene utilizzata Dropzone.js: è una libreria JavaScript open-source che semplifica l'implementazione del caricamento di file da parte degli utenti in un'applicazione web.

### 6.2.4 TinyMCE

TinyMCE è un editor di testo WYSIWYG (What You See Is What You Get) open-source ampiamente utilizzato per l'integrazione in applicazioni web e siti web. Il nome "TinyMCE" è l'acronimo di "Tiny Moxiecode Content Editor", è utilizzato da qualsiasi sito web o applicazione che richiede un editor di testo WYSIWYG per consentire agli utenti di creare e modificare contenuti in modo intuitivo, nel nostro caso viene utilizzato per la descrizione dei progetti.

### 6.2.5 Socket.IO

Socket.IO è una libreria JavaScript open-source che consente la comunicazione bidirezionale in tempo reale tra un server web e un client web. È spesso utilizzato per implementare applicazioni web interattive, chat in tempo reale, giochi online e altre applicazioni che richiedono una comunicazione istantanea tra il server e il client. Nel nostro scenario, Socket.IO offre un modo efficiente per consentire la comunicazione in tempo reale tra i partecipanti ai progetti di ricerca all'interno dell'applicazione. Consentendo ai proprietari dei progetti e ai valutatori di scambiare informazioni istantaneamente, migliora l'efficienza e la collaborazione all'interno del sistema di valutazione dei progetti.



Figure 7: Dropzone per il caricamento dei progetti

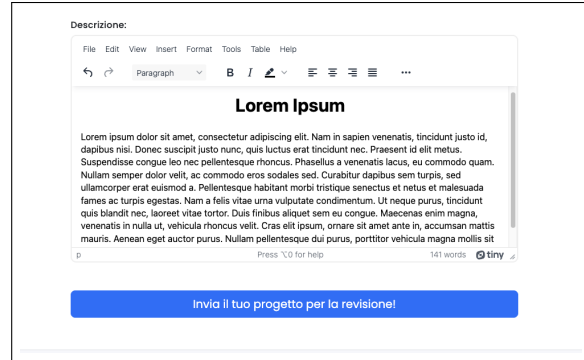


Figure 8: TinyMCE per la descrizione dei progetti

## 7 Contributo al progetto