Progetto di Basi di Dati 2022/2023

Università Cà Foscari, Venezia

Gestione di Esami Universitari

Biscotti Luca 886744

Tonello Alessia 887443

INDICE

1. INTRODUZIONE

1.1 Overview del progetto

Il progetto consiste nello sviluppo di un web application mirata alla gestione di esami universitari da parte del docente, che mira a semplificare il processo di creazione, gestione e registrazione degli esami all'interno di un contesto accademico.
Il docente avrà la possibilità, una volta effettuato il login, di creare nuovi esami con le relative prove associate, visualizzare gli studenti iscritti a tali prove e verbalizzarne il voto.

Nel corso di questo documento, saranno presentati i requisiti funzionali e non funzionali dell'applicativo, l'architettura di sistema proposta e le modalità di implementazione. Saranno inoltre fornite istruzioni dettagliate per l'utilizzo dell'applicazione da parte dei docenti. Infine, saranno discussi i possibili miglioramenti futuri e le considerazioni per l'espansione del sistema.

2. FUNZIONALITÀ PRINCIPALI

2.1. Login

Per poter aver accesso a tutte le funzionalità, il docente dovrà autenticarsi effettuando il login con le proprie credenziali. Non è necessaria una fase di registrazione, dato che ogni docente sarà già in possesso delle credenziali universitarie per poter accedere al portale, pertanto i docenti dovranno fornire email universitaria con annessa password per accedere al sistema.

2.2. Creazione esami

Dopo aver effettuato il login, l’utente verrà reindirizzato verso la home page, dove verranno visualizzati gli esami creati dal docente e anche gli esami di cui è membro. Da qui i docenti possono creare nuovi esami o modificare quelli già presenti. Ogni esame avrà un ID e un nome per identificarlo in modo univoco.

2.3. Visualizzazione studenti iscritti a un esame

È possibile visualizzare tutti gli studenti che si siano iscritti almeno ad una delle prove di un determinato esame.
Da qui sarà possibile verbalizzare il voto per quell'esame.
Saranno anche visibili varie informazioni riguardanti lo studente iscritto, in particolar modo la matricola, nome, cognome e mail universitaria, e inoltre sarà possibile vedere a quali delle varie prove dell'esame lo studente si sia iscritto, e in caso di superamento, di poter far vedere il voto sostenuto in tale prova.

2.4. Creazione prove

Ogni esame è formato da una o più prove intermedie.
Ciascuna di queste prove avrà un ID e un nome che le identifica, il tipo (orale, scritto, progetto), la data di svolgimento, il tipo di ricaduta che avrà sull'esame, quanto influisce sul voto complessivo e altre informazioni inerenti allo svolgimento della prova.

2.5. Visualizzazione studenti iscritti a una prova

È possibile visualizzare gli studenti che si sono iscritti alle varie prove.
Da qui il docente registrerà il voto. Inoltre troverà le loro informazioni tra cui il numero di matricola (che li identifica), nome e cognome, email, la data di scadenza entro cui lo studente può accettare o meno il voto in caso fosse positivo e la conferma dell'accettazione.

2.6. Modifica prova

In questa sezione sarà possibile effettuare delle modifiche ad alcuni dettagli di una prova.
È possibile modificare il titolo della prova, la sua durata, la data in cui sarà sostenuta e l'aula nella quale si terrà.
Solamente il docente che ha creato la prova sarà in grado di modificarne i contenuti.

2.7. Modifica e visualizzazione membri d'esame

Ogni docente sarà in grado di aggiungere o rimuovere altri docenti dalla lista dei membri appartenenti a un esame.
Il ruolo di "Presidente" viene assunto dal docente che ha creato l'esame, e non potrà essere rimosso dalla partecipazione.

2.8 Visualizzazione degli studenti

Il sistema consente di visualizzare lo stato degli studenti, mostrando lo storico degli esami e delle prove sostenute.
In una prima pagina saranno presenti tutte le informazioni riguardanti lo studente, consecutivamente, sarà possibile fare un focus specifico su un singolo studente, andando a visualizzare tutti gli esami al quale si è iscritto, con annesso stato di verbalizzazione; inoltre, sarà possibile visualizzare ancora più specificatamente a quali prove di quell'esame lo studente si sia iscritto, mostrando anche eventuali voti.

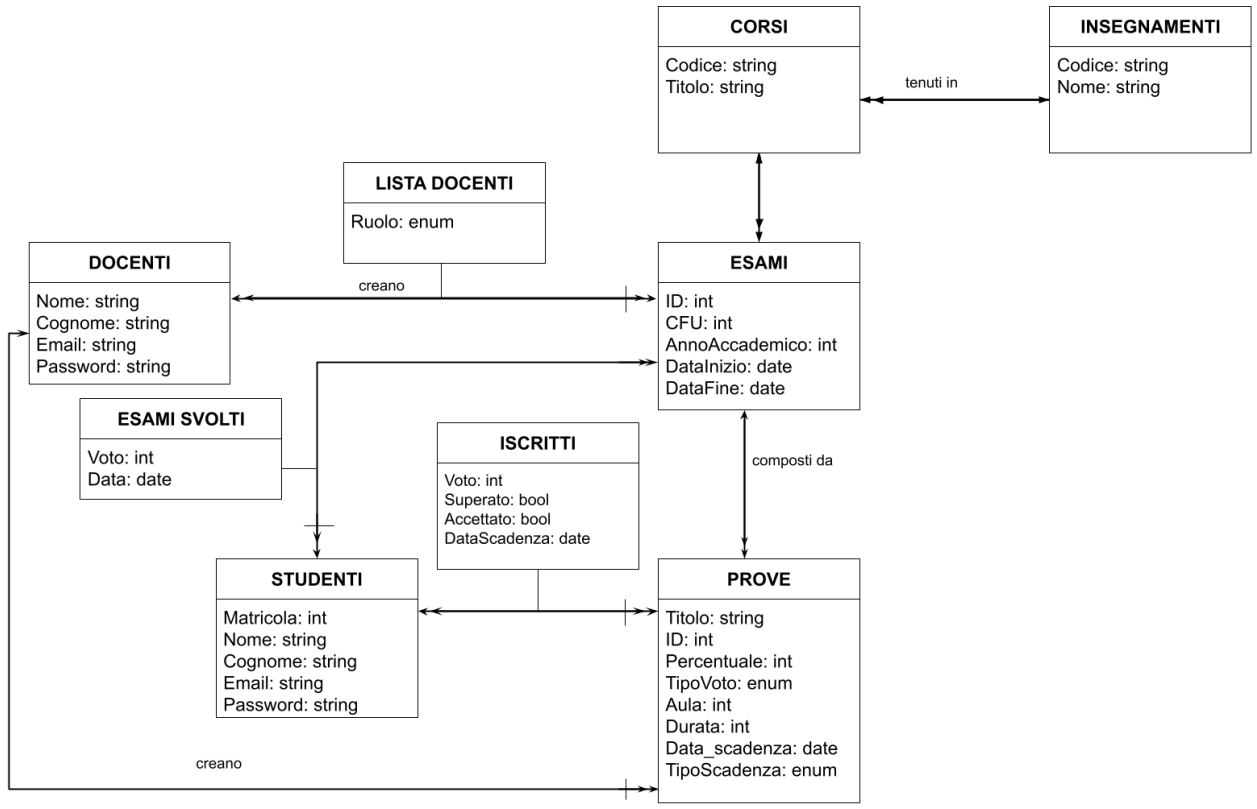
2.9. Logout

Il docente esce dal portale, effettuando la chiusura della sessione aperta per entrare.

3. PROGETTAZIONE

3.1 Progettazione concettuale

Lo schema concettuale progettato è il seguente:



DOCENTI: ogni docente possiede un account personale con l'email dell'istituto e una password. Ognuno di essi può creare uno o più esami, i quali saranno composti da almeno una prova, anch'essa creata dal docente.

STUDENTI: ogni esame e prova avrà una lista degli studenti iscritti, composti da numero di matricola, nome e cognome, email dell'istituto.

ESAME: ogni esame può essere gestito da uno o più docenti, ed è composto da un ID univoco, il numero di CFU, l'anno accademico, la data di inizio e fine.

PROVE: ogni esame può essere composto da una o più prove, ma che possono essere gestite da uno solo dei docenti che tengono quell'esame. Ogni prova possiede un ID univoco, un titolo, la percentuale che rappresenta il peso che ha sul voto dell'esame intero, il tipo di voto (idoneità, bonus, punteggio), in che aula si svolge e la sua durata, la data di svolgimento e la sua data di scadenza per la validità del voto.

CORSI: in un corso possono esserci più esami da svolgere, e sono caratterizzati da un codice univoco ed il titolo.

INSEGNAMENTI: gli insegnamenti hanno più corsi, e sono formati da un codice che li caratterizza e un nome.

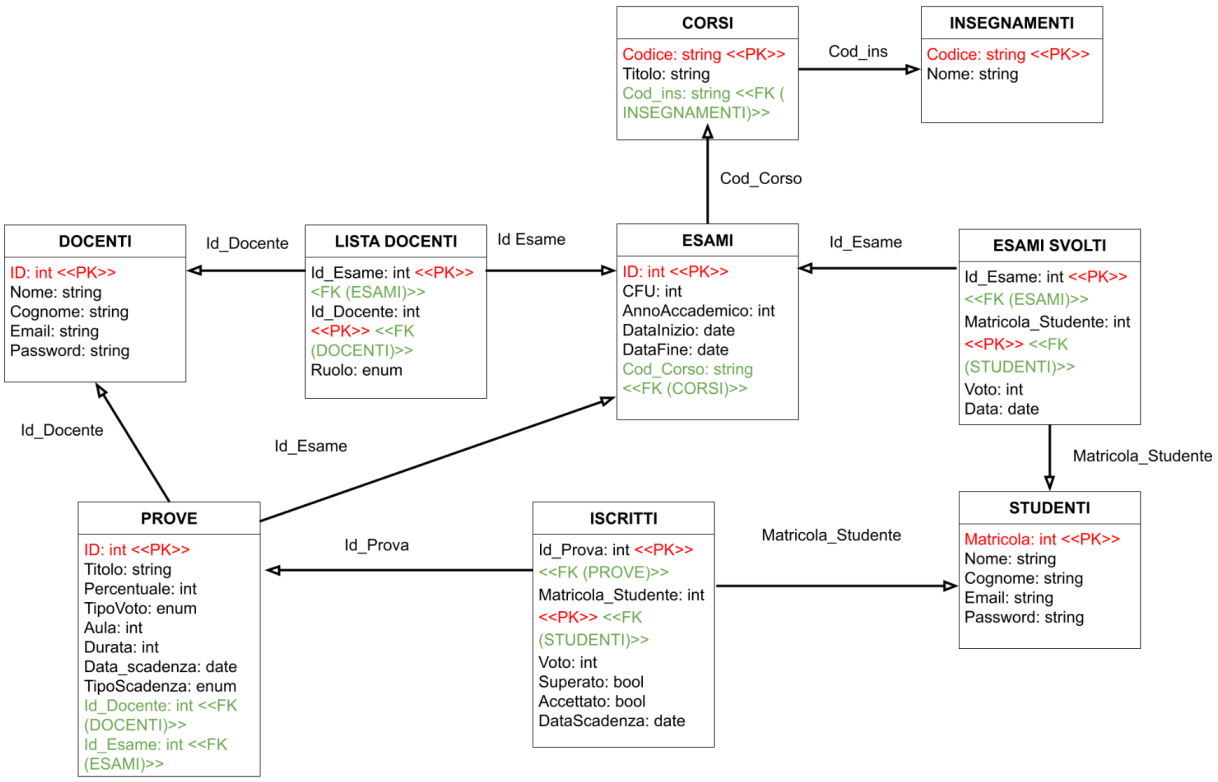
Tra le tabelle di congiunzione troviamo:
LISTA DOCENTI: essendo che in un esame è possibile che ci siano più docenti che hanno tenuto il corso relativo, questa tabella è necessaria per descrivere il ruolo che hanno i vari docenti all'interno dell'esame.

ESAMI SVOLTI: uno studente può svolgere uno o più esami e viceversa. In questa tabella verrà descritto il voto di uno specifico esame svolto da uno studente.

ISCRITTI: simile alla tabella precedente, questa comprende anche la data di scadenza del voto e la possibilità di scegliere allo studente di accettare il voto o meno.

3.2. Progettazione logica

La progettazione logica è la seguente, dove le chiavi primarie sono evidenziate in rosso mentre quelle esterne in verde:



4. QUERY PRINCIPALI

Fra le varie query utilizzate all'interno del progetto, ce ne sono state alcune più complesse da realizzare e perfezionare fra quelle più importanti per la base del progetto.

Fra queste vi è per esempio la query utilizzata per mostrare la lista degli studenti a tutte le prove di un determinato esame, implementando tutte le informazioni utili, come i dati personali e gli eventuali voti.
Pur sembrando inizialmente una task facile, ha richiesto una certa complessità al livello logico e strutturale.

Qui di seguito il codice preso in esempio:

```python
esame = Esami.query.get(id)
prove = Prove.query.filter(Prove.id_esame == id).all()
lista = Studenti.query.join(ListaIscritti, Studenti.matricola ==
ListaIscritti.matricola_studente).join(Prove, ListaIscritti.id_prova ==
Prove.id).join(EsamiSvolti, isouter=True).filter(Prove.id_esame ==
esame.id).add_columns(Prove.titolo,ListaIscritti.id_prova,EsamiSvolti.voto.l
abel("totale"), ListaIscritti.voto)
```

Nel codice riportato qui sopra, vengono eseguite diverse JOIN fra quattro tabelle, per ottenere una lista degli studenti che si sono iscritti a una prova appartenente a un determinato esame, riportando tutti i dati necessari di tali prove, e in caso esistesse, anche il voto verbalizzato dell'esame stesso.

L'ultima query scritta in raw SQL è la seguente:

```unset
SELECT studenti.*, prove.titolo, lista_iscritti.id_prova, esami_svolti.voto
AS totale, lista_iscritti.voto
FROM studenti
JOIN lista_iscritti ON studenti.matricola =
lista_iscritti.matricola_studente
JOIN prove ON lista_iscritti.id_prova = prove.id
LEFT JOIN esami_svolti ON lista_iscritti.id_prova = esami_svolti.id_prova
AND esami_svolti.matricola_studente = studenti.matricola
WHERE prove.id_esame = <id> AND esami_svolti.id_esame IS NULL;
```

Successivamente, per far si che il risultato occorresse in una maniera leggibile all'utente, c'è stata la necessità di raggruppare ogni riga del risultato della query precedente, e fare in modo che per ogni studente si potesse sapere tutte le informazioni riguardanti un determinato esame in maniera chiara e concisa.
Per fare questo, è stato utilizzato un group by, filtrato per matricola, poi successivamente è stata creata una lista in python per memorizzare per ogni singolo studente, ogni esito per ogni prova presente in quel determinato esame.

Qui di seguito il codice preso in discussione:

```python
studenti = []
	lista_ordinata = sorted(lista, key=lambda x: x.Studenti.matricola)  # Ordina per matricola
	for matricola, group in groupby(lista_ordinata, key=lambda x: x.Studenti.matricola):  # Raggruppa per matricola
		voto = [0] * len(prove)
		s = [0] * len(prove)  # Crea una lista di zeri per le prove
		for row in group:
			for i, prova in enumerate(prove):
			test = 0
			if row.id_prova == prova.id:
				s[i] = 1
				test = 1
			if row.voto is not None and test == 1:
				voto[i] = row.voto
			else:
				if(voto[i] == 0):
					voto[i] = "Non sostenuto"

		studenti.append({
			'studente': row.Studenti,
			'totale': row.totale,
			'fatto': s,
			'voto': voto
		})
```

La soluzione appena mostrata è stata complessa nella sua costruzione, dato che è stata anche una delle prime sfide riscontrate al livello di codice all'interno del progetto.

Un altra query che abbiamo definito interessante è la seguente:

```python
studente = Studenti.query.get(id)

   esami_sostenuti = Esami.query \
       .join(EsamiSvolti) \
       .filter(EsamiSvolti.registrato == studente) \
```

```
        .order_by(asc(Esami.codice_corso)) \
        .add_columns(EsamiSvolti.data, EsamiSvolti.voto) \
        .all()

    esami_iscritto = Esami.query \
        .join(Prove, ListaIscritti) \
        .filter(Prove.id_esame == Esami.id, ListaIscritti.iscritto ==
studente, ListaIscritti.id_prova == Prove.id,
Esami.id.notin_([esame.Esami.id for esame in esami_sostenuti])) \
        .all()
```

Il codice appena illustrato, rappresenta le query utilizzate per recuperare dal database tutti gli esami al quale un determinato studente abbia preso iscrizione; ciò significa, non solo gli esami verbalizzati, ma anche quelli senza ancora un voto, ma al quale lo studente abbia preso iscrizione ad almeno una delle prove appartenenti ad esso.

Essendo che nella tabella ‘EsamiSvolti’ un record viene creato solamente dopo la verbalizzazione effettiva del voto d’esame, è stato leggermente più difficoltoso, riuscire a recuperare i dati desiderati al fine del task.

Qui di seguito ci sono le query in raw sql:

```
Unset
SELECT esami.*, esami_svolti.data, esami_svolti.voto
FROM esami
JOIN esami_svolti ON esami.id = esami_svolti.id_esame
WHERE esami_svolti.matricola_studente = <id>
ORDER BY esami.codice_corso ASC;
```

Questa prima query si occupa di prendere tutti gli esami verbalizzati, perciò tutti gli esami che lo studente ha superato, ordinati in ordine crescente.

```
Unset
SELECT esami.*
FROM esami
JOIN prove ON prove.id_esame = esami.id
```

```
JOIN lista_iscritti ON lista_iscritti.id_prova = prove.id
WHERE lista_iscritti.matricola_studente = <id>
AND esami.id NOT IN (SELECT esami_svolti.id_esame FROM
esami_svolti WHERE esami_svolti.matricola_studente = <id>)
ORDER BY esami.codice_corso ASC;
```

In quest'altra query, vengono selezionati invece tutti gli esami che abbiano almeno una delle sue prove con uno studente specifico, che però non sia presente nella lista degli esami già verbalizzati (perché di questo se ne occupa già la query precedente).

Come si è potuto notare, durante lo svolgimento di questo progetto, non è stato necessario fare utilizzo di query altamente complesse, dato che le interrogazioni richieste non risultavano generalmente complicate, ma nonostante questo, c'è stata l'occasione di strutturare del codice allettante riguardo le interazioni col database, purché non eccessivamente difficoltose, ma bensì stimolanti.

5. SCELTE PROGETTUALI

5.1. Politiche di Integrità e Controllo degli Accessi

Nel progetto, abbiamo adottato una serie di politiche di integrità dei dati e politiche di controllo degli accessi per garantire la correttezza delle informazioni e la sicurezza dell'applicazione.

5.2. Integrità dei dati

Abbiamo implementato una serie di controlli per garantire l'integrità dei dati nel nostro database. Ad esempio, prima di consentire la registrazione di un nuovo esame o prova, verifichiamo che i dati inseriti siano validi e rispettino i requisiti previsti. Questo ci ha permesso di prevenire l'inserimento di dati errati o inconsistenti nel database.

5.3. Controllo degli accessi e delle politiche di autorizzazione

Abbiamo definito un sistema di controllo degli accessi basato su ruoli e autorizzazioni per garantire che solo gli utenti autorizzati possano accedere alle diverse parti dell'applicazione. Ad esempio, solo i docenti che sono responsabili di un esame possono accedere e modificare le informazioni relative a quell'esame. Questo è stato realizzato attraverso la funzione `permission` che controlla se l'utente corrente ha il permesso di accedere a una determinata funzionalità.

5.4. Uso dei trigger e indici

Contrariamente ad altre metodologie di progettazione, in questo progetto non abbiamo utilizzato trigger o indici. Le nostre scelte progettuali si sono concentrate su altre tecniche per garantire l'integrità dei dati e il controllo degli accessi.

5.5. Conclusioni

Pur non utilizzando trigger o indici in questo progetto, abbiamo adottato politiche di integrità dei dati rigorose e un sistema di controllo degli accessi basato su ruoli per garantire che l'applicazione sia affidabile e sicura. Queste scelte progettuali sono state fondamentali per creare un'esperienza utente coerente e una gestione efficiente delle informazioni.

6. ULTERIORI INFORMAZIONI

6.1. Sviluppo Front-End

Per la realizzazione della parte front-end del progetto, ci siamo avvalsi prevalentemente del linguaggio CSS e, seppur in quantità minore, anche del codice JavaScript. L'utilizzo di tali tecnologie ci ha permesso di definire lo stile e la presentazione degli elementi dell'interfaccia grafica, consentendo una migliore esperienza visiva e di navigazione da parte dell’utente.

Oltre al codice scritto a mano ci siamo avvalsi della libreria Bootstrap, una libreria open-source di front-end che offre una vasta collezione di strumenti e componenti CSS, HTML e JavaScript per la creazione rapida e semplificata di interfacce grafiche.
Grazie ad essa siamo stati in grado di utilizzare vari strumenti messi a disposizione come i modali e la barra di navigazione, garantendoci un minor tempo di sviluppo e maggior qualità grafica.

6.2. Sviluppo Back-End

Nel corso dello sviluppo del backend del nostro progetto, abbiamo fatto ampio uso del framework web Flask insieme al potente ORM SQLAlchemy. Questa scelta ci ha consentito di creare una solida applicazione web che è sia scalabile che facilmente manutenibile, con una particolare attenzione alla gestione efficiente del database.

Abbiamo organizzato il nostro codice utilizzando il concetto di Blueprint in Flask. Questo ci ha permesso di strutturare il nostro backend in moduli autonomi, migliorando la chiarezza del codice e consentendo una migliore gestione delle varie funzionalità dell'applicazione. Ogni Blueprint rappresenta un insieme di funzionalità correlate, come la gestione degli esami, degli studenti o delle prove. Questa struttura modulare ha semplificato la gestione e l'espansione dell'applicazione nel tempo.

Per la gestione del database, abbiamo scelto di utilizzare SQLAlchemy come ORM. Questo framework ci ha permesso di definire i modelli delle nostre tabelle di database come classi Python, rendendo il processo di creazione e gestione dei dati molto più intuitivo. Abbiamo sfruttato le relazioni definite nei modelli per stabilire collegamenti tra le tabelle e ottenere i dati desiderati con facilità.

Come sistema di gestione del database, abbiamo optato per SQLite. Questo DBMS leggero e incorporato è stato perfetto per le esigenze del nostro progetto, consentendo un facile setup e una gestione agevole dei dati.

La libreria flask-login è stata utilizzata per gestire l'autenticazione degli utenti e il controllo degli accessi alle diverse parti dell'applicazione. Abbiamo creato una funzione `load_user` che recupera un oggetto `Docenti` dal database in base all'ID dell'utente memorizzato nella sessione. Questo ci ha consentito di tenere traccia dell'utente attualmente autenticato e di garantire l'accesso solo a chi è autorizzato.

Abbiamo definito una serie di endpoint attraverso i moduli Blueprint di Flask per gestire diverse funzionalità dell'applicazione:

- **Login e Logout**: Abbiamo creato un endpoint per la pagina di login che gestisce sia richieste GET che POST. L'utente viene reindirizzato alla pagina principale se è già autenticato. In caso contrario, verifichiamo le credenziali tramite un modulo di login personalizzato e autentichiamo l'utente se le credenziali sono valide.

- **Creazione di Esami e Prove**: Abbiamo implementato endpoint dedicati alla creazione di nuovi esami e prove. Gli utenti autorizzati possono inserire i dettagli richiesti e i record corrispondenti vengono creati nelle tabelle associate.

- **Visualizzazione Dettagli**: Abbiamo creato endpoint per visualizzare i dettagli di corsi, esami, studenti e prove. Utilizziamo le relazioni definite nei modelli SQLAlchemy per ottenere i dati necessari e restituirli all'utente.

- **Gestione Studenti e Prove**: Abbiamo creato endpoint che consentono agli utenti autorizzati di visualizzare e gestire le informazioni sugli studenti e sulle prove. Questo include funzionalità come la registrazione dei risultati delle prove e la modifica dei dettagli delle prove.

L'uso di Flask e SQLAlchemy, insieme alla gestione dei ruoli e degli accessi, ha permesso di sviluppare un backend solido e sicuro. L'organizzazione tramite Blueprint ha semplificato la struttura del codice, rendendola più chiara e manutenibile. L'utilizzo di SQLite come DBMS si è dimostrato efficace per le esigenze del progetto. Complessivamente, queste scelte progettuali hanno contribuito a creare un backend efficace, consentendo all'applicazione di gestire in modo fluido la complessità delle operazioni richieste.

7. CONTRIBUTO AL PROGETTO

7.1. Divisione del lavoro

Entrambi i membri del nostro team hanno collaborato attivamente nel corso di tutto il progetto. Sin dall'inizio, abbiamo lavorato insieme per sviluppare l'idea iniziale, mettendo a fuoco la progettazione concettuale e logica. Abbiamo condiviso idee, discusso e cercato il punto d'incontro che ci ha portato alla visione finale dell'applicazione.

Nella fase di codifica, ci siamo organizzati in base alle nostre abilità e preferenze per affrontare le parti front-end e back-end dell'applicazione. Nel front-end, ci siamo concentrati sull'interfaccia utente e sul design grafico, lavorando insieme per creare un'esperienza coinvolgente e user-friendly.

Parallelamente, nella parte back-end, abbiamo collaborato strettamente per utilizzare al meglio le tecnologie a disposizione, come Flask e SQLAlchemy. Abbiamo strutturato il database e implementato la logica dell'applicazione, lavorando a stretto contatto per garantire coerenza e affidabilità.

In sintesi, la nostra collaborazione ha consentito di sfruttare al meglio le nostre competenze individuali e di combinare le diverse prospettive in un risultato finale solido e ben integrato. Questo approccio ci ha permesso di affrontare sfide complesse e di realizzare con successo l'obiettivo del progetto.

7.2. Tecnologie utilizzate

Per tenere traccia degli aggiornamenti riguardanti la parte di coding, si è utilizzata una repository su GitHub. Ogni membro del gruppo ha potuto creare un proprio branch personale, lavorando in modo indipendente e proponendo le proprie modifiche. Successivamente, le modifiche sono state revisionate, discusse e integrate nel branch principale (master). Questo ci ha permesso di evitare sovrascritture accidentali e conflitti tra le modifiche e ci ha garantito un approccio collaborativo e strutturato nella gestione del codice.