Text

Description automatically generated

GRUPPO

NOME MATRICOLA

Jinpeng Zhang 886854

Rebecca Frisoni 885771

Martina Ragusa 885113

A.A. 2022/2023

WebApp per la creazione e gestione di prove ed esami

Graphical user interface, application, icon

Description automatically generated

Table of Contents

[Introduzione 3](#_Toc141262446)

[Contesto dell’applicazione 3](#_Toc141262447)

[Funzionalità principali 3](#_Toc141262448)

[Progettazione della base di dati 4](#_Toc141262449)

[Progettazione concettuale 4](#_Toc141262450)

[Progettazione logica 6](#_Toc141262451)

[Progettazione fisica 7](#_Toc141262452)

[Query principali 12](#_Toc141262453)

[Principali scelte progettuali 13](#_Toc141262454)

[Struttura del back-end 14](#_Toc141262455)

[Struttura front-end 15](#_Toc141262456)

[Design finale 15](#_Toc141262457)

[Contributo al progetto 18](#_Toc141262458)

# 

# Introduzione

Il gruppo ha scelto di sviluppare il progetto di una WebApp per la gestione degli esami universitari. In questo documento verrà illustrata la struttura del progetto e come è stato sviluppato. Inizialmente verranno descritte le funzionalità principali fornite dall’applicazione. Successivamente verrà spiegato come è stata effettuata la modellazione concettuale e logica della base di dati e verrà illustrata una selezione di query importanti del progetto. Per finire, verranno specificate le scelte progettuali che sono state utilizzate.

# Contesto dell’applicazione

L'applicativo che abbiamo sviluppato consente la gestione degli esami universitari: esso è uno strumento avanzato e intuitivo pensato per agevolare docenti e studenti nell'organizzazione e nella registrazione dei risultati accademici. Questo sistema è stato progettato per semplificare il processo di gestione degli esami, permettendo ai docenti di creare nuovi esami in modo facile e veloce, e di visionare la situazione riguardante un determinato esame di ciascuno studente.

Con la nostra applicazione di gestione degli esami universitari, aspiriamo a fornire un ambiente efficiente e trasparente per facilitare il percorso degli studenti verso il successo accademico e supportare i docenti nella gestione di un processo essenziale per il mondo universitario.

# Funzionalità principali

L’applicativo sviluppato permette di gestire le funzioni principali riguardanti gli esami universitari. In particolare, di seguito vengono elencate tali funzionalità:

1. Creazione di esami: i docenti hanno il potere di creare nuovi esami, specificando il numero di prove richieste per ciascun esame e altre informazioni rilevanti.
2. Gestione delle prove: Gli studenti possono sostenere una delle prove disponibili in uno degli appelli messi a disposizione. A seguito del superamento di tale prova verranno registrate le informazioni necessarie, tra cui la data di superamento e il voto conseguito.
3. Scadenza delle prove: Ogni prova ha una data di scadenza, oltre la quale la votazione ottenuta dallo studente non sarà più valida.
4. Visualizzazione dello stato degli studenti: Il sistema offre una panoramica dello stato di ciascuno studente, mostrando le prove valide sostenute, lo storico di tutte le prove sostenute e quelle che devono ancora essere superate per registrare l’esame.
5. Visualizzazione degli appelli: È possibile ottenere l’elenco degli appelli e degli studenti che hanno superato le diverse prove.
6. Verifica dell’idoneità all’esame: Gli studenti possono verificare se sono in condizione di registrare l’esame, visualizzando le prove valide già sostenute e quelle ancora necessarie per completare il percorso accademico.

# 

# Progettazione della base di dati

La progettazione della base di dati per l'applicazione di gestione degli esami universitari è stata svolta con attenzione per garantire l'efficienza, l'integrità e la coerenza dei dati all'interno del sistema. La base di dati gioca un ruolo fondamentale nell'archiviazione e nella gestione dei dati relativi agli esami, alle prove, agli studenti e agli appelli. Di seguito sono presentati i principi guida e gli elementi chiave della progettazione della base di dati:

1. **Analisi dei requisiti**

La progettazione della base di dati è stata avviata da un'analisi approfondita dei requisiti dell'applicazione. Sono stati identificati i principali attori coinvolti, come docenti e studenti, nonché le entità principali, come Docente, Studente, Esame e Prova. Inoltre, sono state individuate le relazioni tra le diverse entità, come la creazione degli esami da parte dei Docenti.

1. **Normalizzazione dei dati**

Per garantire la ridondanza dei dati e l'integrità referenziale, sono stati applicati i principi di normalizzazione delle tabelle. Le tabelle sono state organizzate fino alla forma normale desiderata, evitando dipendenze funzionali ridondanti e assicurando che ciascun attributo sia strettamente legato alla chiave primaria della tabella.

1. **Definizione delle tabelle e degli attributi**

Le tabelle sono state definite in modo coerente e conforme alla struttura delle entità coinvolte nell'applicazione. Ogni tabella ha una chiave primaria univoca per identificare in modo inequivocabile ogni record. Sono stati inoltre definiti gli attributi necessari per ciascuna entità, garantendo la registrazione accurata delle informazioni.

# Progettazione concettuale

Abbiamo utilizzato il modello concettuale per modellare i dati della nostra base di dati. Come primo step abbiamo individuato le entità principali coinvolte nel dominio di interesse: Studenti, Esami, Prove e Docenti. La classe Studenti presenta tre attributi: il nome (di tipo stringa), il cognome (stringa) e il numero di matricola (int), che costituisce la chiave primaria. La classe Esami contiene le informazioni relative agli esami specificando tre attributi: il nome dell’esame (stringa), il codice identificativo (stringa), che costituisce la chiave primaria, il numero dei cfu assegnati (di tipo intero) e l’anno accademico a cui appartiene. La classe Docenti contiene le informazioni relative ai docenti, specificate dagli attributi: nome (stringa), cognome (stringa), indirizzo e-mail (stringa), password (stringa) e l’ID (chiave primaria, di tipo intero). Successivamente, vi è la classe Prove, che contiene, appunto, informazioni riguardanti la prova stessa, quali l’identificativo (chiave primaria, di tipo stringa), la tipologia (stringa), il nome (stringa), il tipo di voto (stringa), la percentuale (int), la data (date) e l’ora (string) in cui viene svolta, e la data di scadenza (date).

La classe Studenti è messa in relazione con la classe Esami attraverso una relazione ‘molti a molti’ “Registrazione\_esame”, la quale contiene informazioni sul voto (di tipo int) e la data di superamento (Date). Studenti possiede un’altra relazione ‘molti a molti’ “Appelli” che contiene informazioni riguardanti un appello dell’esame. Essa ha come attributi: il voto (int), la data superamento (Date). La classe Esami è, a sua volta, messa in relazione con la classe Docenti attraverso la relazione ‘molti a molti’ denominata “Creazione\_esame”, la quale possiede l’attributo “ruolo\_docente”. Essa è inoltre messa in relazione con la classe Prove con una ‘uno a molti’, denominata “haProve”. La classe Prove è, a sua volta, messa in relazione con la classe Docenti con una relazione ‘molti a uno’ denominata “Crea”.

A picture containing diagram, text, plan, technical drawing

Description automatically generated

# Progettazione logica

Successivamente, partendo dalla progettazione concettuale, abbiamo proseguito con la progettazione logica.

Abbiamo trasformato le relazioni ‘molti a uno’ e ‘uno a molti’ aggiungendo le chiavi esterne necessarie. In Prove è stata aggiunta la chiave esterna IdE di tipo stringa, che punta ad Esami. Per la relazione Creazione\_esame, invece, è stata creata un’ulteriore tabella Creazione\_esame, contenente le due chiavi esterne IdD per Docenti (di tipo int) e IdE per Esami (di tipo stringa). Successivamente sono state trasformate le relazioni molti a molti. All’interno di Registrazione\_esame sono state aggiunte le due chiavi esterne IdS e IdE, entrambe di tipo stringa, rispettivamente per la tabella Studenti e per la tabella Esami. Similmente, all’interno di Appelli sono state aggiunte IdS e IdP, chiavi esterne per Studenti, di tipo stringa, e Prove, di tipo int.

A picture containing text, diagram, screenshot, number

Description automatically generated

# Progettazione fisica

Una volta completate con successo sia la progettazione concettuale che la progettazione logica del nostro database, siamo stati in grado di ottenere una visione chiara e completa del sistema così da procedere con la progettazione fisica del database.

Inizialmente, abbiamo scelto di utilizzare il DBMS SQLite per diversi motivi che si sono dimostrati molto validi per le nostre esigenze specifiche. In primo luogo, l'aspetto della leggerezza ha giocato un ruolo fondamentale nella decisione. SQLite offre un database senza server, incorporato all'interno delle nostre applicazioni, il che ha comportato un notevole vantaggio in termini di efficienza e risorse di sistema. Le prestazioni di questo DBMS sono state sorprendenti, grazie alla sua natura leggera, alla mancanza di configurazioni complesse e alle transazioni locali che hanno reso le operazioni di lettura e scrittura molto rapide ed efficienti. Questo ci ha consentito di ottenere elevate prestazioni del database anche in situazioni in cui era richiesta una rapida elaborazione dei dati.

Un altro aspetto cruciale nella nostra scelta è stato il supporto multipiattaforma offerto da SQLite. La possibilità di utilizzare il database su diverse piattaforme ci ha garantito una maggiore flessibilità nell'implementazione delle nostre applicazioni, permettendo le componenti del gruppo di lavorare facilmente sull’app data la presenza di sistemi operativi diversi.

Tuttavia, verso la fase finale della progettazione, abbiamo iniziato a notare alcune limitazioni e carenze associate all'utilizzo di SQLite come DBMS principale. In particolare, abbiamo riconosciuto che mancavano alcune delle funzionalità più avanzate necessarie per un'amministrazione dei dati sofisticata e per gestire progetti di maggiore complessità. La scalabilità di SQLite si è rivelata una sfida, poiché iniziavamo a lavorare con un sistema più grande e complesso, si è sentita la necessità di un DBMS in grado di affrontare carichi di lavoro più impegnativi e supportare un numero maggiore di utenti concorrenti.

Dopo un'attenta valutazione delle nostre esigenze, abbiamo deciso di migrare il nostro database e tutte le informazioni ad esso associate a PostgreSQL, un sistema di gestione di database relazionale (RDBMS) che ci ha fornito una vasta gamma di funzionalità avanzate. Questa decisione è stata guidata dal desiderio di sfruttare al meglio le potenzialità offerte da PostgreSQL, tra cui l'utilizzo di trigger per gestire automaticamente gli eventi di database, le funzioni definite dall'utente per personalizzare l'elaborazione dei dati e la scalabilità per gestire progetti complessi con un numero crescente di utenti e dati.

La migrazione al nuovo RDBMS ha rappresentato una sfida tecnica, ma ci ha fornito la soluzione più adatta alle nostre esigenze e ci ha permesso di raggiungere una maggiore efficienza e flessibilità nella gestione del database.

Di seguito vengono descritti alcuni dei vincoli implementati, i cui costrutti per la creazione si basano sul linguaggio PostgreSQL.

Trigger sulla tabella “registrazione\_esame”:

A screenshot of a computer program

Description automatically generated

Il trigger soprastante Passed è stato creato per verificare lo stato superamento di un determinato esame. La funzione Is\_Passed() viene chiamata prima dell’inserimento e dell’aggiornamento della tabella “registrazione\_esame”. Esso permette di garantire che l’esame venga registrato solamente quando la prova corrispondente è stata superata. In caso contrario viene generato un errore e l’operazione di inserimento o aggiornamento viene interrotta. //Forse è solo inserimento.

Trigger sulla tabella “Appelli”:

A screenshot of a computer program

Description automatically generated

Il trigger Only\_One soprastante viene utilizzato per controllare che uno studente sia iscritto solamente ad un appello di un esame. La funzione Secure\_Only\_One() associata viene eseguita prima dell’operazione di inserimento nella tabella Appelli. Nel caso in cui lo studente sia già iscritto all’appello viene generato un errore, e l’operazione viene interrotta.

A computer screen shot of a program

Description automatically generated

Il trigger Substitution è stato sviluppato per garantire che, dopo il sostenimento di una prova per il quale lo studente presentava già un voto, venga aggiornato lo stato della prova stessa. La funzione Is\_Replaced() viene eseguita dopo l’operazione di inserimento. Se la condizione è soddisfatta, ovvero se lo studente aveva già sostenuto la prova ad un appello precedente, i dati della tabella vengono modificati seguendo il seguente criterio:

* Se la votazione ricevuta all’appello più recente (NEW.voto) è sufficiente, il voto già registrato viene sostituito da quello nuovo ricevuto;
* Se la votazione ricevuta (NEW.voto) risulta insufficiente, la prova sostenuta in precedenza viene invalidata, con conseguente registrazione della nuova (NEW.stato\_superamento) a fallimento.

A screenshot of a computer program

Description automatically generated

Il trigger Double\_Date\_Test è stato sviluppato per garantire che uno studente non si iscriva a più esami lo stesso giorno. La funzione Deny\_Insert\_Double\_Date\_Test() viene eseguita prima dell’operazione di inserimento nella tabella Prova. Nel caso in cui i dati che si stanno cercando di inserire presentino la stessa data di quella già presente nel sistema, ma l’identificativo della prova non coincida, viene generato un errore e l’operazione di inserimento viene interrotta.

Ruoli

Per quanto riguarda la gestione dei ruoli e le politiche di amministrazione per il nostro progetto, al fianco del ruolo postgres, già presente di default all’interno del database, sono stati definiti due ruoli: Docente e Studente.

Il ruolo “postgres” costituirà il superutente, ovvero colui che avrà il compito di amministrare il sistema. Avrà quindi accesso a tutte le tabelle, su cui potrà effettuare tutte le operazioni disponibili.

Avremo poi una distinzione tra il ruolo di Docente e Studente.

A close up of a password

Description automatically generated

Il ruolo docente, come suggerisce dal nome, identificherà i docenti. Esso consente l’accesso alle tabelle relative a tutto ciò che riguarda gli esami, ovvero Esame, Prova, Appelli, Registrazione\_esame e Creazione\_esame. Su queste tabelle, i docenti potranno effettuare operazioni di SELECT, INSERT e DELETE, che gli consentiranno dunque di aggiungere esami, modificare e registrare i voti.

A close-up of a computer code

Description automatically generated

L’altro ruolo che è stato definito al fianco di Docente è il ruolo Studente. Esso permette di accedere alle informazioni relative agli esami, senza però consentire la modifica di quanto è contenuto all’interno di tali tabelle.

# 

# Query principali

Di seguito verranno commentate alcune delle query principali, in particolare quella per visualizzare la lista degli studenti che hanno superato un esame in particolare e quella per visualizzare la lista degli esami che sono stati superati da uno studente.

Per quanto riguarda la prima query, necessaria per avere una visione di tutti gli studenti che hanno superato un determinato esame, la tabella degli Studenti viene messa in relazione con la tabella Appelli utilizzando l’attributo idS di entrambe le tabelle. Vengono poi effettuate due ulteriori operazioni di join con le tabelle Prova ed Esame, utilizzando, rispettivamente, idP e idE. Il risultato delle operazioni di giunzione viene filtrato selezionando solamente gli esami il cui idE della tabella Studente coincida con l’omonimo attributo della tabella Esame. Per selezionare solamente gli esami che sono stati effettivamente superati, si verifica che lo “stato\_superamento” sia “True”.

È stato utilizzata la clausola GROUP BY(Esame.idE, Studente.idS) per effettuare l’operazione di raggruppamento e dunque partizionare le righe del risultato utilizzando i due attributi idE e idS. Infine, viene utilizzata la clausola HAVING per mostrare solamente gli studenti che hanno superato le prove che ammontano a 100 con la percentuale di superamento.

A picture containing text, screenshot, font

Description automatically generated

La query seguente, invece, viene utilizzata per visualizzare la lista di tutti gli esami che sono stati superati da uno studente, ovvero deve essere verificata la condizione per cui tutte le prove presentino come stato di superamento “true”. Per realizzare questa query, dunque, vengono effettuate due operazioni di join: la prima mette in relazione la tabella Esame con la tabella Prova, confrontando l’attributo idE presente in entrambe le tabelle; la seconda viene utilizzata per legare la tabella Appelli. Il risultato delle operazioni di giunzione viene filtrato verificando che idS della tabella Appelli coincida con idS della tabella Esame e che idP della tabella Appelli abbia lo stesso valore di idP di Prova. Si selezionano poi solamente gli esami che sono stati superati utilizzando “Appelli.stato\_superamento == True”.

Infine, vengono utilizzate le clausole GROUP BY e HAVING. La prima viene adoperata per eseguire l’operazione di raggruppamento sull’attributo idE della tabella Esame, mentre la seconda, come nella query descritta in precedenza, permette di selezionare solamente le prove che ammontano a 100 con la percentuale di superamento.

A picture containing text, font, screenshot

Description automatically generated

# Principali scelte progettuali

In questa sezione vengono descritte le principali scelte progettuali che sono state prese dal gruppo, distinguendo tra i linguaggi di programmazione utilizzati e i frameworks adoperati.

Per quanto riguarda i linguaggi di programmazione sono stati utilizzati Python, JavaScript, HTML, CSS, SCSS, SQL.

Inoltre, sono stati utilizzati diversi framework, tra i quali Flask, Bootstrap e Ajax.

Nella fase di progettazione del progetto, sono stati selezionati, sulla base di ciò che ci era stato presentato dal professore, i linguaggi di programmazione che avremmo utilizzato. Abbiamo utilizzato JavaScript, linguaggio di scripting, per il lato client per sviluppare le funzionalità interattive e dinamiche.

Per il lato server, invece, abbiamo utilizzato Python, linguaggio ad oggetti, sempre sulla base delle informazioni che ci erano state fornite dal professore. Esso è infatti di facile lettura e comprensione, che ci ha permesso una maggiore facilità nell’utilizzo. Inoltre, offre un'ampia varietà di framework e librerie per facilitare lo sviluppo web, come Flask, che abbiamo deciso di utilizzare.

Flask è un micro-framework basato sul linguaggio Python che, nonostante presenti solo le funzionalità di base per lo sviluppo web, per il nostro progetto, si è rivelato perfetto per sviluppare le funzionalità server-side in modo efficiente.

Per quanto riguarda il front-end abbiamo utilizzato una combinazione di HTML, CSS e JavaScript. HTML è stato adoperato per la struttura principale del contenuto; per progettare e definire l’aspetto e lo stile delle pagine web abbiamo utilizzato CSS; infine, per fornire interattività e dinamicità all’interfaccia utente è stato utilizzato JavaScript.

A fianco di CSS è stato utilizzato anche SCSS (Sassy CSS), un’estensione di sintassi per CSS che ci ha offerto funzionalità più avanzate.

In combinazione ai tre linguaggi nominati sopra, è stato utilizzato anche il framework Bootstrap che ci ha messo a disposizione varie componenti predefinite, facilitando il processo di creazione di una migliore interfaccia utente.

Al fianco di JavaScript, affinché l’interfaccia fosse più interattiva, abbiamo utilizzato Ajax (Asynchronous JavaScript And XML.). Ajax permette alle pagine web di essere aggiornate in modo asincrono, scambiando dati con il web server. Questo consente di modificare in parte una pagina web senza che essa venga ricaricata nella sua interezza e dunque permettendo all’interazione dell’utente di non essere interrotta.

Per la gestione del database, è stato utilizzato il linguaggio SQL (Structured Query Language) per definire e manipolare i dati. L’abbiamo impiegato per creare e manipolare le tabelle del nostro database, definendo la struttura dei dati e gestendo le operazioni di lettura, scrittura, modifica e cancellazione.

A sua volta, il supporto di Flask per l’integrazione con un database relazionale ci ha consentito di creare e gestire le tabelle, eseguire query e garantire l’integrità e la coerenza dei dati.

# Struttura del back-end

La figura sottostante mostra le principali parti che costituiscono il back-end, in cui sono presenti i moduli e i file che permettono al progetto di funzionare in modo ottimale.

A screenshot of a computer screen

Description automatically generated

Segue una breve descrizione dei file e delle directories più importanti:

**\_\_init\_\_.py**: è il file che permette di inizializzare l’applicazione e configurarla prima dell’avvio del server.

**routes.py**: contiene le routes, ovvero le URL dell'applicazione web che gli utenti possono visitare per accedere a diverse pagine del progetto. In particolare, esso contiene le funzioni associate alla gestione delle pagine e le funzionalità principali dell’applicazione, tra le quali possiamo trovare la visualizzazione degli studenti iscritti ad un determinato esame o gli esami creati da un docente.

**static/**: contiene i file statici dell’applicazione, come fogli di stile CSS, script JavaScript e risorse, tra cui tutte le immagini utilizzate.

**templates/**: contiene i template HTML utilizzati per generare le pagine dinamiche dell’applicazione.

# Struttura front-end

Sicuramente, nell’applicativo per la gestione degli esami, un ruolo fondamentale lo ricopre il front-end. Di seguito verranno elencate le tecnologie che sono state utilizzate per realizzarlo, che hanno giocato un ruolo fondamentale nello sviluppo di una buona interfaccia utente e di facile utilizzo. La nostra idea è stata quella di rendere l’esperienza utente per i docenti che intendono gestire gli esami universitari il quanto più semplice e intuitiva possibile.

Come già citato nella sezione “principali scelte progettuali”, sono stati utilizzati HTML, CSS, SCSS, JavaScript e Ajax.

* HTML (HyperText Markup Language) ha svolto un ruolo fondamentale nel definire la struttura principale del contenuto. Siamo infatti stati in grado di organizzarlo in modo coerente e gerarchico, agevolando così la creazione di pagine web con una struttura ben organizzata e facilmente navigabile.
* CSS (Cascading Style Sheets) e SCSS (Sassy CSS) sono stati utilizzati per progettare e definire l’aspetto e lo stile delle pagine web.
* JavaScript è stato utilizzato per fornire interattività e dinamicità all’interfaccia utente. Al suo fianco è stato utilizzato anche l’interfaccia Ajax, che ci ha permesso di elevare ulteriormente il grado di interattività del progetto

# Design finale

Di seguito vengono proposte alcune immagini che permettono di dare uno sguardo generale al design finale dell’applicativo.

A screen shot of a login

Description automatically generated

Figura - Pagina di login

A screenshot of a computer

Description automatically generated

Figura - Pagina che permette di visualizzare gli studenti data la matricola

A screenshot of a computer

Description automatically generated

Figura - Pagina di creazione di un esame

A screenshot of a computer

Description automatically generated

Figura - Pagina di visualizzazione degli esami già presenti

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedFigura - Pagina riguardante un esame specifico

A screenshot of a computer

Description automatically generated

Figura - Pagina per aggiungere una nuova prova

# Contributo al progetto

Lo sviluppo del progetto è stato portato avanti da tutti i componenti del gruppo. In particolare, dopo una prima definizione della struttura del progetto nella sua interezza, il gruppo ha proseguito con la definizione dello schema concettuale e la successiva traduzione in schema relazionale. In seguito, il lavoro è stato diviso in tre nel seguente modo: back-end e front-end Jinpeng Zhang , front-end Rebecca Frisoni, stesura documentazione Martina Ragusa.