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 2](#_Toc141109388)

[Contesto dell’applicazione 2](#_Toc141109389)

[Funzionalità principali 3](#_Toc141109390)

[Progettazione della base di dati 3](#_Toc141109391)

[Progettazione concettuale 4](#_Toc141109392)

[Progettazione logica 5](#_Toc141109393)

[Progettazione fisica 5](#_Toc141109394)

[Query principali 6](#_Toc141109395)

[Principali scelte progettuali 7](#_Toc141109396)

[Struttura del back-end 8](#_Toc141109397)

[Contributo al progetto 9](#_Toc141109398)

# Introduzione

Il gruppo ha scelto di sviluppare il progetto 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 eseguite la progettazione concettuale e la progettazione logica, avevamo una visione d’insieme del database che ci ha permesso di proseguire con la progettazione fisica del database stesso.

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

//descrizione trigger

# 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.

models/database.py: Questo modulo definisce le classi di modello che rappresentano le tabelle del database. Utilizziamo SQLAlchemy per semplificare l’interazione con il database PostgreSQL.

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 Java-Script e risorse, tra cui tutte le immagini utilizzate.

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

# 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.