Università degli Studi di Trieste - Dipartimento di Ingegneria e Architettura

Progetto – Basi di Dati
"Sistema di prenotazione per le aule universitarie"

Giovanni de Maria

Matricola: IN0500952

Anno Accademico 2022-2023

`https://github.com/giovannidemaria/DBProjectUNITS`

Indice

1 Descrizione del progetto e scopi d'uso

Il progetto si propone di sviluppare un sistema di prenotazione di aule universitarie che permetta la gestione simultanea di più università. Sarà possibile gestire informazioni sui corsi e sugli eventi che si svolgono all'interno delle aule e di assegnare una persona responsabile per ogni prenotazione o evento.

Il sistema sarà gestito dal personale amministrativo di ogni università che avrà accesso alle funzionalità di aggiunta, modifica e cancellazione di informazioni sulle aule, sui corsi, sugli eventi e sulle persone che prenotano. Gli utenti potranno accedere al sistema attraverso un sito web senza autenticazione che consentirà loro di visualizzare le informazioni sulle aule disponibili, le informazioni sui corsi e sugli eventi in corso o programmati.

2 Entità e relazioni

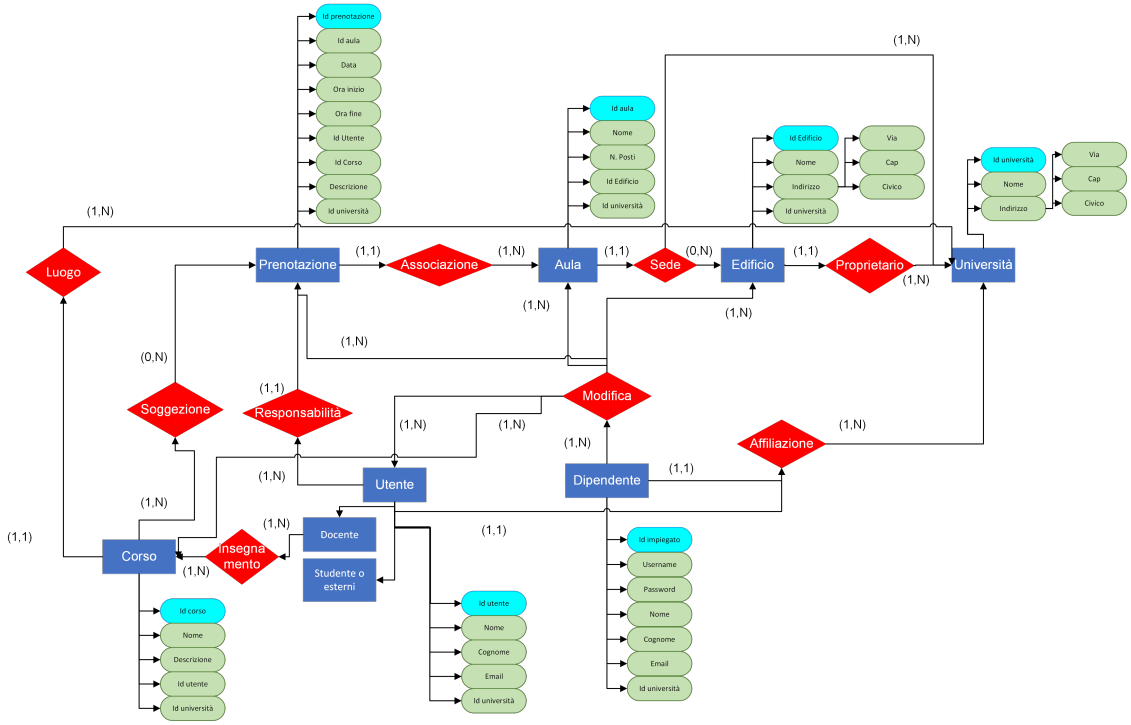

Figura 1: Schema entity - relationship
Gli identificatori sono stati evidenziati in ciano.

Entità	**Descrizione**	**Attributi**	**Identificatore**
Università	Rappresenta un'Università	id_università, nome, indirizzo	id_università
Edificio	Rappresenta un edificio dell'Università	id_edificio, nome, indirizzo, id_università	id_edificio
Utente	Rappresenta l'utente che si prende la responsabilità delle singole prenotazioni. Può essere un docente, uno studente o un esterno.	id_utente, nome, cognome, email, id_università	id_utente
Corso	Rappresenta un corso dell'Università	id_corso, nome, descrizione, id_utente, id_università	id_corso
Aula	Rappresenta una classe all'interno di un Edificio	id_aula, nome, capacità, id_edificio,id_edificio, id_università	id_aula
Prenotazione	Rappresenta la prenotazione di un'Aula	id_prenotazione, id_aula, data, ora_inizio, ora_fine, id_corso, descrizione, id_università	id_prenotazione
Dipendente	Rappresenta un dipendente dell'università (inteso come persona impiegata in mansioni di segreteria)	id_dipendente, username, password, nome, cognome, email, id_università	id_dipendente

Tabella 1: Entità

Relazione	**Descrizione**	**Attributi**	**Identificatore**
Soggezione	Un corso può essere soggetto a una prenotazione	Corso, Prenotazione	/
Associazione	Una prenotazione è associata a un'aula	Prenotazione, Aula	/
Sede	Un'aula si trova in un edificio	Aula, Edificio	/
Proprietario	Un'Università è proprietaria di uno o più edifici	Università, Edificio	/
Insegnamento	Un docente insegna un corso	Utente, Corso	/
Responsabilità	Un utente ha la responsabilità di una prenotazione	Utente, Prenotazione	/
Affiliazione	Un utente o un dipendente è affiliato a un'Università	Dipendente, Utente, Università	/
Luogo	Un corso è insegnato in un'Università	Corso, Università	/
Modifica	Un dipendente modifica una prenotazione, un'aula, un utente, un corso o un edificio	dipendente, Prenotazione, Aula, Utente, Edificio	/

Tabella 2: Relationships

3 Gruppi di frasi omogenee

1. **Frasi sulle aule:**
   - Delle aule si vuole tener nota del loro ID identificativo, del nome, della capacità e dell'ID identificativo dell'edificio di cui fanno parte.
   - Ogni aula ha una capacità massima di studenti.
   - Le aule possono essere prenotate per le lezioni, gli esami, seminari o altri eventi accademici.
   - Le aule possono essere assegnate a un determinato utente (che può essere un docente, uno studente o una persona esterna).
   - È necessario tenere traccia delle date e degli orari delle prenotazioni per ogni aula.
2. **Frasi sui corsi:**
   - Dei corsi si vuole tener nota del nome, dell'ID identificativo e della descrizione.
   - Ogni corso è associato a un docente responsabile e può essere tenuto in una o più aule.
3. **Frasi sugli edifici:**
   - Degli edifici si vuole tener nota del loro ID identificativo, del nome, dell'indirizzo e dell'università di cui fanno parte.
   - La tabella 'edifici' contiene le informazioni sugli edifici all'interno del campus universitario.
4. **Frasi sugli utenti:**
   - Degli utenti si vuole conoscere l'ID identificativo, il nome, il cognome, l'indirizzo email e l'eventuale numero di matricola.
   - Ogni utente, se è un docente, può insegnare uno o più corsi. Se l'utente non è un docente, allora non può insegnare alcun corso.
   - I dettagli di una prenotazione includono la data e l'ora di inizio e fine della prenotazione.
   - Ogni prenotazione è associata a un determinato dipendente o utente tramite un campo di riferimento all'ID del dipendente o utente.

5. **Frasi sulle prenotazioni:**
    - La tabella 'prenotazioni' registra le prenotazioni effettuate per l'uso delle aule universitarie.
    - Ogni prenotazione è identificata da un ID univoco nella tabella 'prenotazioni'.
    - I dettagli di una prenotazione includono la data e l'ora di inizio e fine della prenotazione.
    - Ogni prenotazione è associata a un determinato utente responsabile tramite un campo di riferimento all'ID dell'utente.
    - La tabella 'prenotazioni' potrebbe anche includere informazioni aggiuntive, come lo scopo della prenotazione, il corso a cui è associata o eventuali requisiti specifici per l'aula.
    - Le prenotazioni possono sovrapporsi solo se fanno riferimento a aule diverse. Per la stessa aula, le prenotazioni devono essere pianificate in modo non conflittuale.
    - È possibile effettuare modifiche o cancellazioni alle prenotazioni esistenti tramite opportune operazioni di aggiornamento o eliminazione dei record corrispondenti.
6. **Frasi sui dipendenti:**
    - La tabella 'dipendenti' è creata per memorizzare informazioni sui dipendenti dell'università, inclusi il loro ID, nome utente di accesso, password, nome, cognome, email e ID dell'università di appartenenza.
7. **Frasi sulle università:**
    - La tabella 'università' è creata per memorizzare informazioni sulle università, inclusi il loro ID, nome e l'indirizzo della sede principale.

4 Vincoli non esprimibili

- Un utente non può prenotare più di un'aula contemporaneamente.
- Un'aula non può essere prenotata per periodi di tempo sovrapposti.
- Un edificio non può essere associato a più di un proprietario università contemporaneamente.
- Un'aula non può essere in più di un edificio contemporaneamente.
- Un dipendente della segreteria può soltanto modificare i dati relativi alla sua università di appartenenza.
- Una prenotazione deve essere associata a un utente affiliato alla stessa università in cui si trova l'aula.

5 Tabella dei volumi

- Ogni università ha 15 edifici.
- Ogni edificio ha 10 aule.
- Ogni università ha 1200 docenti. A questi si aggiungono circa 500 utenti non docenti che hanno effettuato almeno una prenotazione (Si precisa che gli studenti sono caricati sul database soltanto se hanno effettuato almeno una prenotazione).
- Quasi ogni docente insegna almeno un corso.
- Ogni università ha 50 dipendenti della segreteria autorizzati a effettuare modifiche. Vengono aggiunte 4.000.000 (Prenotazioni)+600 (Aule)+5.000 (Corsi)+60 (Edifici)+5.300 (Utenti) = 4.010.960. Inoltre si stimano circa altre 2.000 modifiche e eliminazioni.

Concetto	Volume
Università	4
Edificio	60
Utente	5.300
Dipendente	200
Corso	5.000
Aula	600
Prenotazione	4.000.000

Tabella 3: Volumi di tipo Entity

Concetto	Volume
È oggetto	3.200.000
Insegna	5.000
È insegnato in	5.000
Responsabilità	4.000.000
Associata a	4.000.000
Modifica	4.012.960
In	600
Proprietario	60

Tabella 4: Volumi di tipo Relationship

6 Valutazione delle rindondanze

- La relazione tra aula e edificio potrebbe essere rindondante poiché ogni edificio appartiene a un'università. Quindi, si potrebbe sostituire con la relazione Aula e Università. Tuttavia, non è una buona idea eliminare la relazione tra Aula ed Edificio e sostituirla con una relazione tra Aula e Università perché ciò potrebbe portare a una perdita di informazioni importanti. La relazione tra Aula ed Edificio può rappresentare un'informazione utile per generare statistiche e report sui dati relativi alle prenotazioni delle aule e alla loro utilizzazione all'interno dell'edificio.
- L'identificatore id_università è necessario in entità come Utenti, Dipendenti ed Edifici perché questi oggetti sono strettamente legati all'Università a cui appartengono.

  Nel caso delle aule, dei corsi e delle prenotazioni, invece, l'informazione sull'Università può essere ottenuta attraverso delle relazioni:

  1. Edificio a cui appartiene l'Aula. Ad esempio, l'Università associata a un'aula può essere recuperata attraverso la relazione tra Aula ed Edificio, che a sua volta ha un identificatore di Università associato ad esso.
  2. Università d'appartenenza di un utente. Ad esempio, l'identificatore "id_università" nella tabella "prenotazioni" può essere ottenuto attraverso la relazione tra la Prenotazione e l'Utente associato, che a sua volta appartiene a una determinata Università.
  3. Lo stesso discorso si applica anche ai Corsi. Tuttavia, non si tratta di una ridondanza "dannosa" in quanto può risultare comoda per le query e le operazioni di join sul database, consentendo di estrarre le informazioni relative all'università di appartenenza senza dover effettuare una serie di join tra le tabelle. Però, per evitare stati indesiderati, si preferisce sopprimere il campo "id_università" nelle tabelle corsi e prenotazioni (si può quindi sopprimere anche la relationship "Luogo").

7 Normalizzazione

Le tabelle edifici e università non sono in prima forma normale in quanto l'attributo "indirizzo" contiene informazioni composte, come cap, via e civico. Per raggiungere la prima forma normale, è necessario scomporre l'attributo "indirizzo" in attributi separati, come cap, via e civico, in modo che ogni attributo contenga solo un valore atomico.

8 Eliminazione delle generalizzazioni

Si ha una sola generalizzazione con entità padre Utente ed entità figlie rispettivamente "Docente" e "Studente o esterni".

Possiamo osservare che nessuna delle entità figlie presenta attributi diversi. Inoltre, si ritiene che non si avrà interesse ad accedere a tali entità in maniera distinta e specifica.

L'unica cosa che si nota è che soltanto i docenti hanno una relazione in più ("insegna"), ma questa differenza può essere espressa semplicemente aggiungendo un attributo di tipo `enum('Docente', 'Altro')` nell'entità padre.

Questa scelta implica la necessità di andare ad inserire un vincolo (Trigger) che permetta la relazione "insegna" solo tra entità in cui l'attributo "tipo" corrisponde con "Docente".

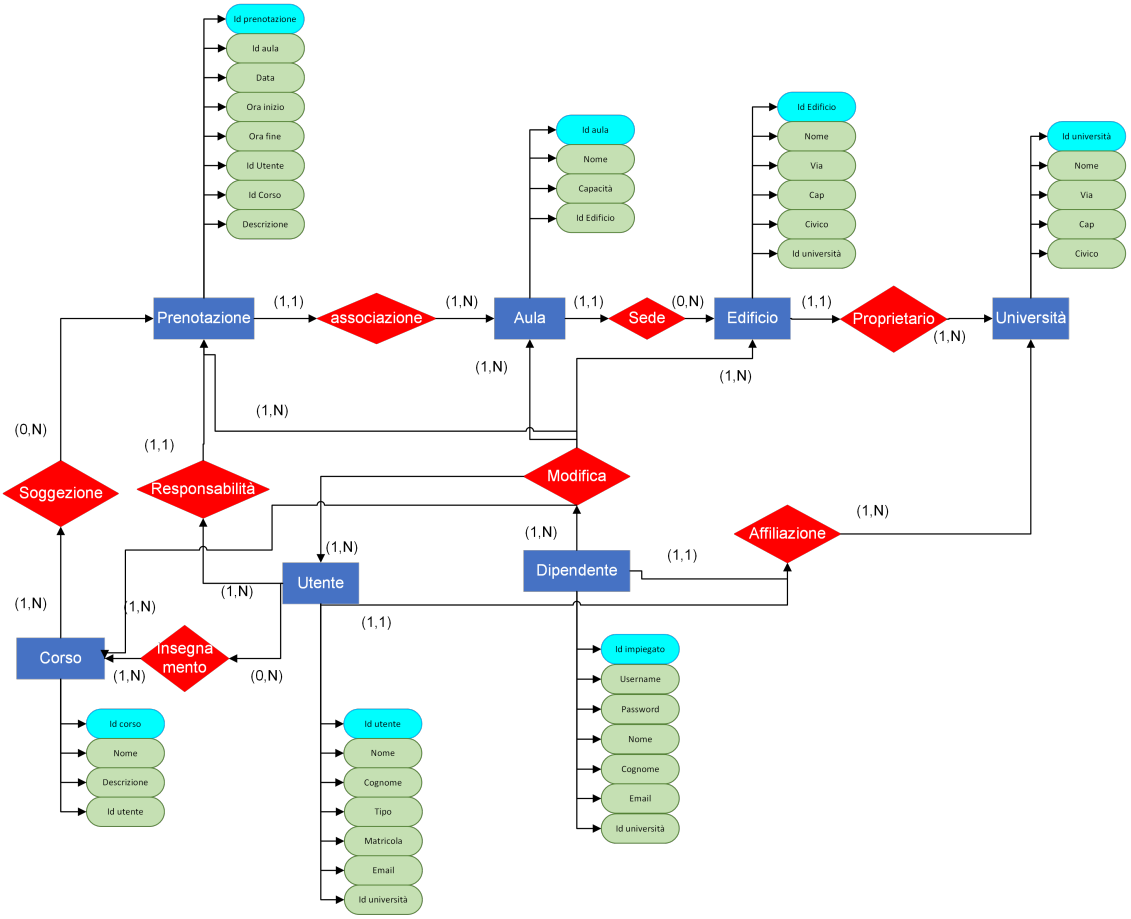

Figura 2: Schema entity - relationship ristrutturato
Gli identificatori sono stati evidenziati in ciano.

9 Passaggio al modello razione e schema logico

- Aula(**id_aula**, nome, capacità, id_edificio)
- Corso(**id_corso**, nome, descrizione, id_utente)
- Dipendente(**id_dipendente**, username_login, password, nome, cognome, email, id_università)
- Edificio(**id_edificio**, nome, cap, via, civico, id_università)
- Prenotazione(**id_prenotazione**, id_aula, data, ora_inizio, ora_fine, descrizione, id_utente, id_corso)
- Università(**id_università**, nome, cap, via, civico)
- Utente(**id_utente**, nome, cognome, email, tipo, matricola, id_università)

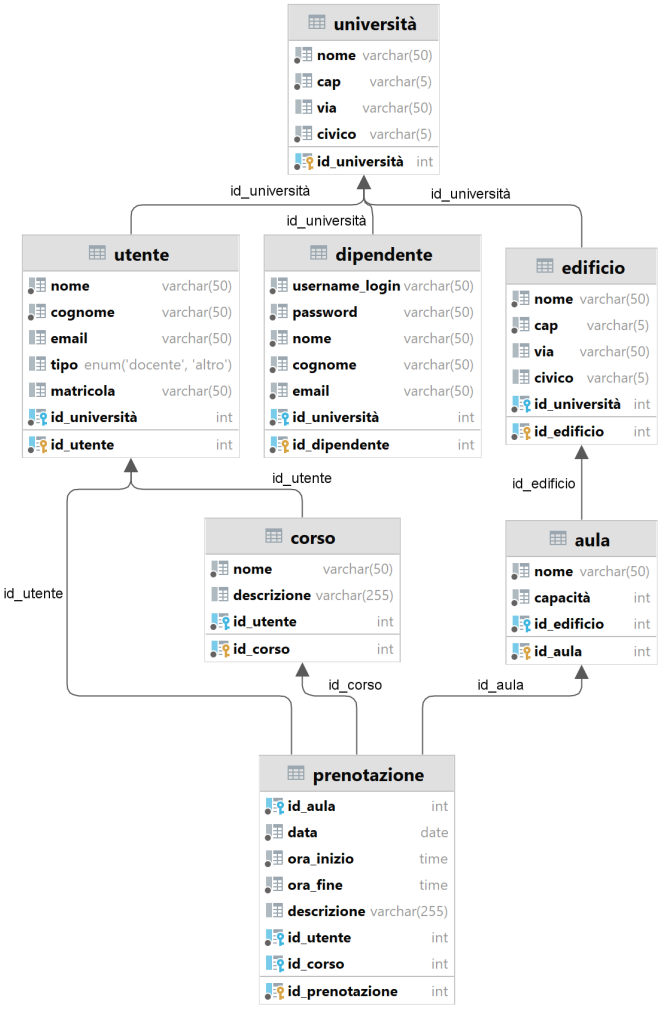

Figura 3: Schema logico

10 Realizzazione delle operazioni

In questa sezione verranno descritti i Trigger, le Stored Procedures e viste usati nel progetto. Il loro codice è indicato in un file a parte.

10.1 Trigger

`controllo_associazione_corsi_docenti`

Questo trigger è progettato per essere eseguito prima dell'inserimento di una riga nella tabella "Corsi". Il suo scopo è verificare se l'utente associato al corso ha il ruolo di "Docente" nel sistema. Se l'utente non ha il ruolo corretto, viene generata un'eccezione con un messaggio di errore specifico, impedendo l'inserimento della riga nella tabella "Corsi". Questo trigger assicura che solo gli utenti con il ruolo corretto possano essere associati a un corso, garantendo la correttezza dei dati nel database.

`controllo_piu_prenotazioni_per_utente`

Questo trigger viene eseguito prima dell'inserimento di una nuova riga nella tabella "Prenotazioni". Controlla se l'utente ha già effettuato una prenotazione per lo stesso giorno e se l'intervallo di ore della nuova prenotazione si sovrappone con altre prenotazioni attive dell'utente. Se viene rilevata una sovrapposizione, viene generata un'eccezione.

`controllo_piu_università_per_edificio`

Questo trigger viene eseguito prima dell'inserimento di una nuova riga nella tabella "Edifici". Controlla se l'edificio è già associato a un'altra università. Se viene rilevata un'associazione esistente, viene generata un'eccezione.

`aula_edificio`

Questo trigger viene eseguito prima dell'inserimento di una nuova riga nella tabella "Aule". Controlla se l'aula è già associata a un altro edificio. Se viene rilevata un'associazione esistente, viene generata un'eccezione.

`controllo_universita_utente_aula`

Questo trigger viene eseguito prima di inserire una nuova riga nella tabella "Prenotazioni". Recupera l'id_universita dell'aula associata alla prenotazione (attraverso l'associazione tra le tabelle Aula, Edificio e Prenotazioni) e l'id_universita dell'utente associato alla prenotazione. Se i due id_universita non combaciano, viene generato un errore con il messaggio specificato. In caso contrario, l'inserimento della riga nella tabella "Prenotazioni" procede normalmente.

Considerazione sul vincolo delle prenotazioni sovrapposte

Si potrebbe implementare un trigger per controllare la disponibilità dell'aula prima di inserire la prenotazione nella tabella delle prenotazioni. Tuttavia ciò non serve perché questo controllo è già stato implementato nella stored procedure che si occupa di registrare la prenotazione nella base di dati.

10.2 Stored Procedures

Si precisa che sotto ogni descrizione si indicano anche i nomi dei parametri per invocare queste Stored Procedures attraverso l'applicazione indicata nel paragrafo "Demo del progetto" del presente documento.

`AuleDisponibili()`

La stored procedure "`AuleDisponibili()`" restituisce l'elenco delle aule disponibili in una data ora e giorno, filtrate in base alla loro capacità, fornendo una soluzione efficiente e automatizzata per la pianificazione delle attività scolastiche e universitarie.

Nomi dei parametri

`nomeUniversita, capacita, data, oraInizio, oraFine`

`ElencoPrenotazioniAula()`

La stored procedure "`ElencoPrenotazioniAula()`" restituisce un elenco dettagliato delle prenotazioni di un'aula in un determinato intervallo di tempo. La procedura richiede come input la data di inizio e la data di fine dell'intervallo temporale, insieme all'ID dell'aula di interesse.

Nomi dei parametri

`dataInizio, dataFine, aulaID`

`ElencoPrenotazioniUtente()`

La stored procedure "`ElencoPrenotazioniUtente()`" prende in input l'ID di un utente (indifferente se docente o altro) e due date, e restituisce tutte le prenotazioni di aule effettuate nell'intervallo di tempo specificato.

Nomi dei parametri

`dataInizio, dataFine, utenteID`

`AnnullaPrenotazione()`

La stored procedure "`AnnullaPrenotazione()`" consente di annullare una prenotazione esistente nel sistema. Prende in input l'ID della prenotazione da annullare e esegue le seguenti operazioni:

1. Verifica se la prenotazione esiste nel database.
2. Se la prenotazione viene trovata, elimina la corrispondente riga dalla tabella delle prenotazioni.
3. Restituisce un messaggio di conferma indicando che la prenotazione è stata annullata con successo.

La stored procedure "`AnnullaPrenotazione()`" è utile per gestire situazioni in cui un'aula non è più disponibile per una prenotazione effettuata in precedenza, consentendo agli utenti di liberare l'aula e rendendola nuovamente disponibile per altre prenotazioni.

Nomi dei parametri

`prenotazioneID`

`PrenotaAula()`

La stored procedure "`PrenotaAula()`" consente di effettuare una prenotazione di un'aula per un determinato corso presso un'istituzione accademica. Accetta come parametri di input l'ID dell'aula, la data, l'ora di inizio e di fine della prenotazione, la descrizione del corso, l'ID del corso e l'ID dell'utente che effettua la prenotazione.

All'interno della procedura, vengono eseguiti controlli per verificare la disponibilità dell'aula nel periodo specificato. Se l'aula è disponibile, viene creata una nuova riga nella tabella delle prenotazioni con i dettagli forniti. In caso di conflitto con altre prenotazioni esistenti, la procedura restituisce un messaggio di errore appropriato.

Inoltre, viene controllata l'esistenza delle righe corrispondenti ai valori specificati di AulaID, UtenteID e CorsoID nelle tabelle "aule", "utenti" e "corsi" rispettivamente, prima di eseguire l'inserimento di una nuova riga nella tabella "prenotazioni". Ciò è importante per garantire la

consistenza dei dati nel database. Senza questa verifica, potrebbero verificarsi problemi di consistenza se si inseriscono righe nella tabella "prenotazioni" che fanno riferimento a valori di AulaID, UtenteID o CorsoID non esistenti nelle tabelle corrispondenti.

La stored procedure "`PrenotaAule()`" contribuisce a semplificare il processo di prenotazione delle aule, garantendo che le prenotazioni siano gestite correttamente e prevenendo sovrapposizioni o doppie prenotazioni.

Nomi dei parametri

`aulaID, data, oraInizio, oraFine, descr, corsoID, utenteID`

10.3 Viste

`ElencoPrenotazioni`

La vista "`ElencoPrenotazioni`" fornisce una visualizzazione dei dettagli delle prenotazioni effettuate per le aule.

La vista è costruita utilizzando diverse operazioni di join tra le tabelle. Questo consente di ottenere un'unica tabella che combina le informazioni necessarie per visualizzare in modo chiaro e dettagliato le prenotazioni delle aule, inclusi i relativi contesti (aula, edificio, università), il corso associato (se presente) e il responsabile della prenotazione.

`ElencoAule, ElencoEdifici e ElencoUniversità`

La vista "`ElencoAule`" fornisce un elenco delle aule disponibili, insieme alle informazioni sull'edificio e sull'università a cui appartengono. Ogni riga della vista rappresenta un'aula e include il nome dell'aula. il nome dell'edificio in cui si trova l'aula, il nome dell'università a cui l'edificio e l'aula appartengono, l'indirizzo completo dell'edificio, ottenuto concatenando via, civico e cap.

Semplificando la vista "`ElencoAule`", è possibile anche ottenere le viste "`ElencoEdifici`" e "`ElencoUniversità`". Nella vista "`ElencoEdifici`" si visualizza il nome dell'edificio, l'università d'appartenenza e l'indirizzo. Invece nel caso "`ElencoUniversità`" si visualizza il nome dell'Università e l'indirizzo della sede principale.

Questa vista fornisce un modo pratico per ottenere un elenco completo delle aule disponibili nel sistema, insieme alle informazioni sull'edificio e sull'università di appartenenza. È particolarmente utile per scopi di visualizzazione e reportistica, consentendo agli utenti di ottenere rapidamente una panoramica delle aule e delle relative posizioni.

`ElencoDocenti e ElencoNonDocenti`

La vista "`ElencoDocenti`" fornisce un elenco dei docenti presenti nel sistema, includendo i loro nomi, cognomi, indirizzi email, matricole e l'università a cui sono affiliati. Questa vista permette di ottenere un rapido accesso alle informazioni essenziali sui docenti presenti nel sistema.

In maniera del tutto analoga è stata creata la vista "`ElencoNonDocenti`" che invece fornisce un elenco degli utenti con una tipologia "Altro".

`ElencoCorsi`

Questa vista elenca i corsi disponibili nel sistema, includendo il nome del corso, la sua descrizione, il nome completo del docente che lo tiene (concatenando nome e cognome) e il nome dell'università in cui il corso è tenuto. È possibile utilizzare questa vista per ottenere una visione completa dei corsi offerti, comprensiva di informazioni sul docente e sull'università associata.

`ElencoDipendenti`

Questa vista fornisce un elenco dei dipendenti dell'università, includendo il nome, il cognome, l'email e il nome dell'università di appartenenza. Il campo password è escluso dalla vista per proteggere la privacy dei dipendenti.

11 Demo del progetto

È disponibile una demo con una app Python (Flask) che serve una semplice applicazione web che realizza alcune funzioni sql descritte nel progetto. Viene utilizzato un container Docker e, per semplicità, non è stata implementato il Backend per la segreteria.

La decisione di non implementare il backend per la segreteria nella demo è principalmente motivata dalla complessità aggiuntiva che comporta la gestione dei dati sensibili e la sicurezza del sistema. L'implementazione di un sistema di crittografia adeguato per i campi sensibili del database, come password o informazioni personali, richiede una conoscenza avanzata della crittografia e delle best practice di sicurezza.

In questa applicazione si evita totalmente con le Stored Procedures di fare concatenazione di stringhe perché potrebbe comportare a delle SQL injections. Per quanto riguarda invece le viste, viene effettuato un controllo sull'input.

Vengono utilizzate le transazioni. Sono importanti soprattutto per le stored procedure che comportano un cambiamento dello stato del database (`PrenotaAula()` e `AnnullaPrenotazione()`). Le transazioni sono un meccanismo fondamentale per garantire l'integrità e la coerenza dei dati all'interno di un database. Si ricorda in particolare che sono fondamentali per garantire che le modifiche siano eseguite in modo atomico, coerente, isolato e durevole, seguendo il principio ACID.

Si specifica infine che in realtà il database ha una piccola differenza rispetto a quello descritto nel documento: per motivi legati alla codifica si sceglie di usare parole senza la a accentata (`à`). Viene sostituita con la a normale senza l'accento (`a`).

11.1 Funzionamento

L'applicazione sviluppata ha lo scopo di gestire richieste POST (nel caso delle stored procedure) oppure GET (nel caso delle viste) e restituire una risposta JSON in base al percorso URL specificato.

Se il percorso URL inizia con `/sp/` seguito dal nome di una stored procedure, l'applicazione eseguirà la stored procedure corrispondente utilizzando i parametri forniti nel corpo della richiesta in formato JSON (i nomi dei parametri da indicare sono nel paragrafo "Stored Procedures"). I risultati della stored procedure verranno quindi restituiti come una risposta JSON.

In maniera analoga, se il percorso URL inizia con `/v/` seguito dal nome di una vista, l'applicazione eseguirà una query di selezione `SELECT * FROM nomeVista` per ottenere i dati dalla vista specificata. I risultati della query verranno restituiti come una risposta JSON.

11.2 Esempi d'uso

Per usare questa applicazione si può utilizzare l'applicativo cURL (negli esempi è usata la versione per Windows, la sintassi della versione per sistemi operativi basati su kernel GNU/Linux è leggermente diversa).

Esempio 1

```
curl -X POST -H "Content-Type: application/json" -d "{\"dataInizio\":\"
    2010-05-16\",\"dataFine\":\"2023-05-18\",\"aulaID\":2}" http://localhost
    :5000/sp/ElencoPrenotazioniAula
```

Esempio 2

```
curl http://localhost:5000/v/ElencoDocenti
```

12 Dichiarazione finale

Il progetto (inteso come questo documento e tutti i file nel repository github) è stato svolto da me in completa autonomia.

Giovanni de Maria
IN0500952
Trieste, 28 maggio 2023