Relazione Progetto Basi di Dati

Joshua Michelett | Mat: 283057

Indice:

1 - Raccolta / Analisi dei Requisiti:

1.1 - Documento di specifica:

Si vuole realizzare una base di dati per la gestione di una biblioteca domestica contenente circa 1000 libri. Per ogni libro nella biblioteca, si conoscono genere, titolo, autore, casa editrice, anno e luogo nella casa. Possono esistere più copie di uno stesso libro. Ogni utente che ha accesso alla biblioteca è definito da un nome utente ed una password. Ogni utente può prendere in prestito un libro dalla biblioteca e, una volta letto e riposto nello stesso posto in cui è stato preso, può lasciare una recensione sul libro ritornato. Una recensione è definita da un voto (da 0 a 5) e da un commento (facoltativo).

1.2 - Analisi del documento di specifica:

Da questo documento possiamo identificare tre componenti principali:

- **Frasi relative ai libri**: biblioteca domestica contenente circa 1000 libri. Per ogni libro nella biblioteca, si conoscono genere, titolo, autore, casa editrice, anno e luogo nella casa. Possono esistere più copie di uno stesso libro.

- **Frasi relative agli utenti**: Ogni utente che ha accesso alla biblioteca è definito da un nome utente ed una password. Ogni utente può prendere in prestito un libro dalla biblioteca e, una volta letto e riposto nello stesso posto in cui è stato preso, può lasciare una recensione sul libro ritornato.

- **Frasi relative alle recensioni**: Ogni utente può lasciare una recensione sul libro ritornato. Una recensione è definita da un voto (da 0 a 5) e da un commento (facoltativo).

1.3 - Glossario dei termini:

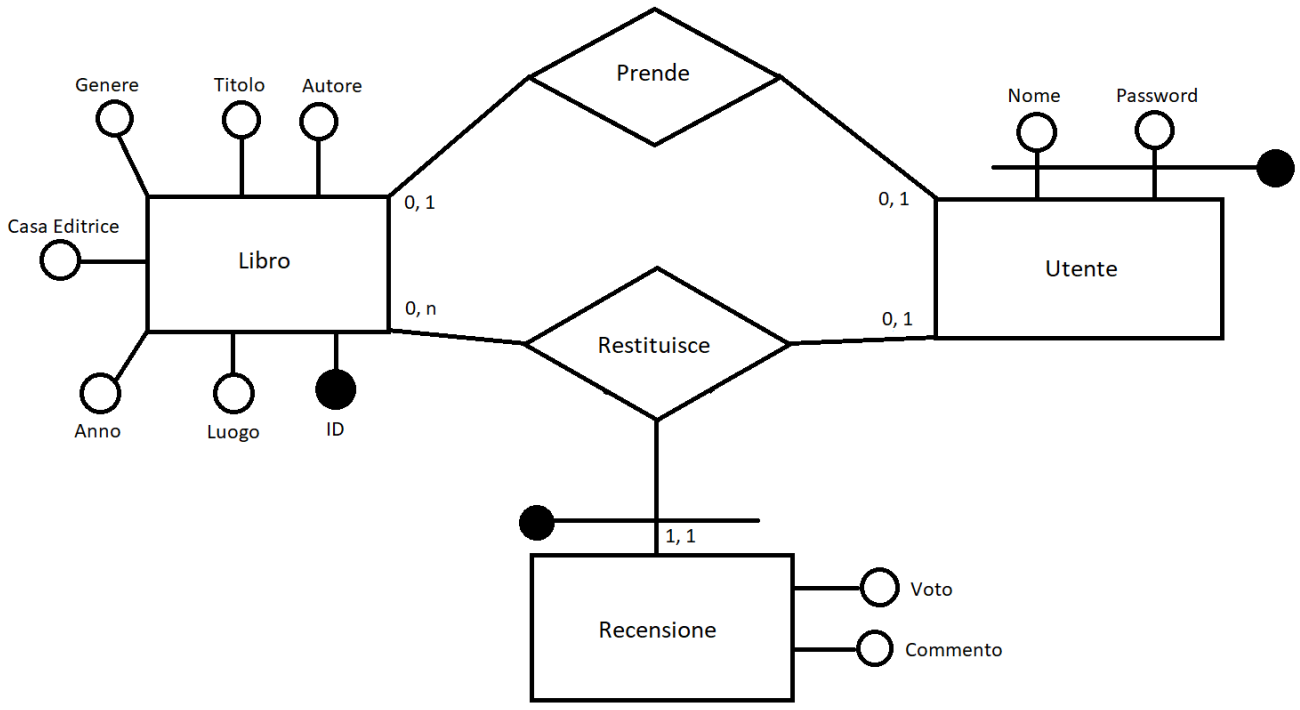
Termine	Descrizione	Collegamenti
Libro	Oggetto definito dai suoi parametri, può essere preso in prestito, in tal caso, non è disponibile per altri utenti.	Utente
Utente	Utente che può prendere in prestito e restituire libri e lasciare recensioni su di essi	Libro, Recensione
Recensione	Voto e commento rilasciato dall’utente al ritorno di un libro	Utente, Libro

1.4 - Operazioni sui dati:

- Inserire un nuovo libro
- Rimuovere un libro già esistente
- Modificare le informazioni di un libro
- Inserire un nuovo utente
- Rimuovere un utente esistente
- Prendere in prestito un libro
- Ritornare un libro preso in prestito
- Lasciare una recensione su un libro ritornato
- Visualizzare le informazioni dei libri
- Visualizzare le recensioni dei libri
- Visualizzare la media dei voti sui libri
- Ordinare e filtrare i libri per le loro caratteristiche

2 - Progettazione Concettuale

2.1 - Diagramma E-R:



Lo sviluppo del diagramma E-R è stato fatto tramite **strategia mista,** individuando i concetti principali e definendoli come entità scollegate. Successivamente viene fatto uno schema scheletro che mano a mano viene raffinato fino a raggiungere un diagramma che soddisfa tutte le caratteristiche descritte nel documento di specifica.

2.2 - Dizionario dei Dati:

Entità:

Entità	Descrizione	Attributi	Identificatore
Libro	Oggetto della libreria che viene preso in prestito e restituito con una recensione	Autore, Titolo, Genere, Casa Editrice, Anno, Luogo, ID	ID
Utente	Utente che utilizza la base di dati per prendere in prestito e restituire libri, lasciando una recensione	Nome, Password	Nome, Password

Relazioni:

Relazione	Descrizione	Componenti	Attributi
Prende	Presa in prestito di un libro da parte di un utente	Libro, Utente	
Restituisce	Restituzione di un libro da parte di un utente, con rilascio di una recensione	Libro, Utente	Voto, commento

2.3 - Business Rules:

Regole di vincolo
L’utente che restituisce un libro deve restituire il libro che ha preso in prestito
In caso un utente prenda in prestito e successivamente restituisca un libro su cui ha già lasciato una recensione, la nuova recensione sostituirà quella precedente
Il voto per ogni recensione deve essere compreso tra 0 e 5
Il libro restituito deve essere restituito nella stessa posizione iniziale

Regole di derivazione
La valutazione di ogni libro viene calcolata con la media dei voti delle recensioni riferite a quel libro

2.4 - Analisi di Prestazione:

2.4.1 - Tavola dei volumi:

Concetto	Tipo	Volume
Libro	Entità	1000
Utente	Entità	5
Prende	Relazione	4
Recensione	Entità	200

Assumendo un numero medio di utenti uguale a 5 (in quanto si tratta di una biblioteca domestica), possiamo dire che la relazione “Prende” non supererà il numero di utenti, in quanto ogni utente può prendere al massimo un solo libro prima di restituirlo.

La relazione “Restituisce” avrà un volume maggiore in quanto dovremo tener conto delle recensioni rilasciate dagli utenti.

2.4.2 - Tavola delle operazioni:

Operazione	Tipo	Frequenza	Accessi	Costo
Inserire un libro	Interattiva	1 volta/settimana	1 Scrittura	0.142
Rimuovere un libro	Interattiva	1 volta/mese	1 Scrittura	0.033
Modificare un libro	Interattiva	1 volta/mese	1 Scrittura	0.033
Inserire un nuovo utente	Interattiva	1 volta/6 mesi	1 Scrittura	0.005
Rimuovere un utente	Interattiva	1 volta/6 mesi	1 Scrittura	0.005
Prendere in prestito un libro	Interattiva	1 volta/settimana	2 Scrittura	0.286
Restituire un libro	Interattiva	1 volta/settimana	3 Scrittura	0.429
Visualizzare libri	Interattiva	2 volte/giorno	1000 Lettura	1000
Visualizzare recensioni libro	Interattiva	10 volte/giorno	201 Lettura	1005
Visualizzare media voti libro	Interattiva	10 volte/giorno	201 Lettura	1005

Costo = Frequenza * Peso * (α * Accessi-scrittura + Accessi-lettura)

Peso: peso operazioni interattive / batch: 0.5
α: coefficiente operazioni di scrittura: 2.0

Costo totale: 3010.93

2.5 - Occupazione Volume:

Libro (1000):

Attributo	Tipo	Dimensione
Genere	Stringa (max 256 caratteri)	256B
Titolo	Stringa (max 256 caratteri)	256B
Autore	Stringa (max 256 caratteri)	256B
Casa Editrice	Stringa (max 256 caratteri)	256B
Anno	Numero	4B
Luogo	Stringa (max 256 caratteri)	256B
ID	Numero	4B

Memoria occupata = 1000 * (256B + 256B + 256B + 256B + 4B + 256B + 4B) = **1.288MB**

Utente (5):

Attributo	Tipo	Dimensione
Nome	Stringa (max 256 caratteri)	256B
Password	Stringa (max 256 caratteri)	256B

Memoria occupata = 5 * (256B + 256B) = **2.56KB**

Prende (4):

Attributo	Tipo	Dimensione
Nome	Stringa (max 256 caratteri)	256B
Password	Stringa (max 256 caratteri)	256B
ID	Numero	4B

Memoria Occupata = 4 * (256B + 256B + 4B) = **2.06KB**

(in fasi successive, aggregando l'ID del libro preso in prestito all'utente corrispondente, è possibile risparmiare memoria senza ripetere le informazioni di Nome e Password)

Recensione (200):

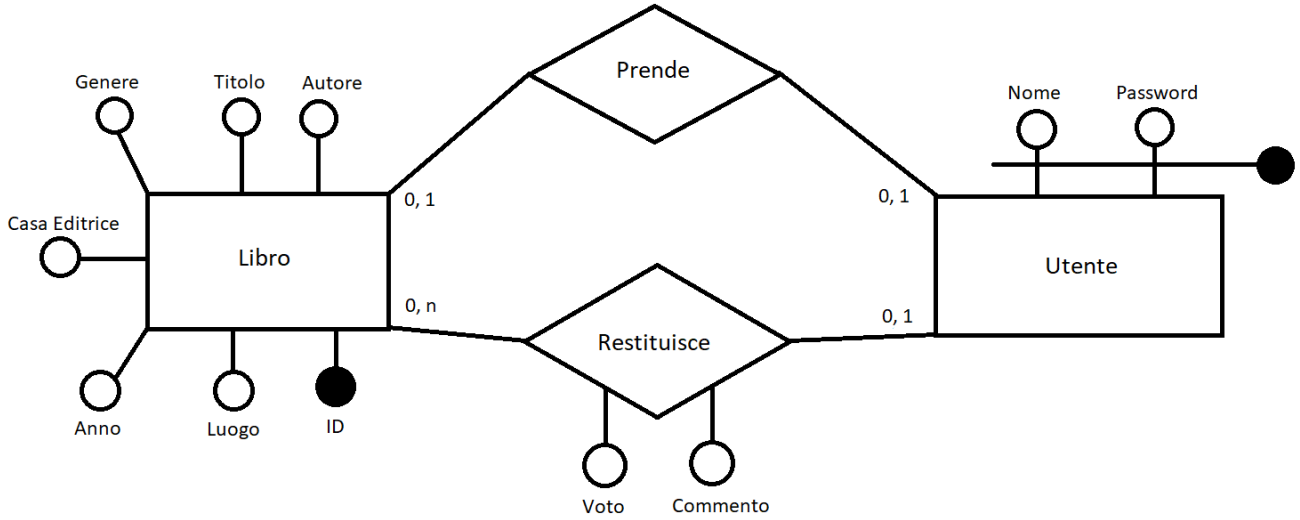
Attributo	Tipo	Dimensione
Nome	Stringa (max 256 caratteri)	256B
Password	Stringa (max 256 caratteri)	256B
ID	Numero	4B
Voto	Numero	4B
Commento	Stringa (max 1024 caratteri)	1024B

Memoria occupata = 200 * (256B + 256B + 4B + 4B + 1024B) = **308.8KB**

Memoria totale occupata = 1.288MB + 2.56KB + 2.06KB + 308.8KB = **1.6MB**

3 - Progettazione Logica

3.1 - Ristrutturazione del modello concettuale



Il modello E-R è stato ristrutturato eliminando l'entità Recensione accorpandola alla relazione Restituisce, in questo modo possiamo creare una tabella in cui salviamo il Voto e Commento di ogni Nome e Password per il rispettivo ID del libro restituito.

3.2 - Traduzione nel modello logico:

LIBRO (ID, Genere, Titolo, Autore, CasaEditrice, Anno, Luogo)

UTENTE (Nome, Password, IDLibro)

RESTITUZIONE (NomeUtente, PasswordUtente, IDLibro, Voto, Commento)

Con questo progetto logico, accorpiamo la relazione "Prende" nella tabella "Utente". Questo perché ogni utente può prendere in prestito solo un libro, quindi risulta più efficiente piuttosto che avere una tabella per tenere traccia di tutti i libri presi in prestito.

I vincoli di integrità di questo modello logico sono:

UTENTE (IDLibro) REFERENCES **LIBRO** (ID)
RESTITUZIONE (IDLibro) REFERENCES **LIBRO** (ID)
RESTITUZIONE (Nome, Password) REFERENCES **UTENTE** (Nome, Password)

Nel modello logico non sono presenti dipendenze funzionali non banali all'infuori delle chiavi della tavola, ogni dipendenza funzionale contiene a sinistra la chiave della tavola a cui appartiene:

LIBRO (ID, Genere, Titolo, Autore, CasaEditrice, Anno, Luogo)

F = {ID → Genere, Titolo, Autore, CasaEditrice, Anno, Luogo}

UTENTE (Nome, Password, IDLibro)

F = {Nome, Password → IDLibro}

RESTITUZIONE (NomeUtente, PasswordUtente, IDLibro, Voto, Commento)

F = {NomeUtente, PasswordUtente, IDLibro → Voto, Commento}

4 - Progettazione Fisica

La base di dati viene implementata utilizzando il DBMS MySQL ed utilizzando l'engine di storage InnoDB.

5 - Implementazione MySQL
Analisi dello script SQL che implementa il database

5.1 - Creazione del database:

```
DROP DATABASE IF EXISTS Biblioteca;
CREATE DATABASE Biblioteca;
USE Biblioteca;
```

In questa parte di codice viene creato il database Biblioteca dove successivamente verranno create le tavole secondo la progettazione logica.

5.2 - Creazione delle tavole:

5.2.1 - Libri:

```
DROP TABLE IF EXISTS libri;

CREATE TABLE libri (
      ID int NOT NULL AUTO_INCREMENT,
      Genere varchar(255),
      Titolo varchar(255),
      Autore varchar(255),
      CasaEditrice varchar(255),
      Anno int,
      Luogo varchar(255),
      PRIMARY KEY (ID)
);
```

La tavola libri viene creata con una chiave primaria con auto incremento (ID) di tipo intero per poter identificare univocamente tutti gli elementi della biblioteca.

Questa tavola conterrà le informazioni di tutti i libri disponibili forniti dal file BIBLIOTECA.csv.

```
LOAD DATA INFILE '/var/lib/mysql-files/BIBLIOTECA.csv'
INTO TABLE libri
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 2 ROWS
(@col1, @col2, @col3, @col4, @col5, @col6)
SET Genere = @col1, Titolo = @col2, Autore = @col3, CasaEditrice = @col4,
Anno = @col5, Luogo = @col6;
```

Conoscendo la formattazione del file BIBLIOTECA.csv, possiamo caricare le informazioni sui libri presenti nella biblioteca nella tavola libri tramite il comando LOAD DATA

5.2.2 - Utenti:

```
DROP TABLE IF EXISTS utenti;

CREATE TABLE utenti (
    Nome varchar(255) NOT NULL,
    Password varchar(255) NOT NULL,
    IDLibro int,
    PRIMARY KEY (Nome, Password),
    FOREIGN KEY (IDLibro) REFERENCES libri(ID) ON DELETE SET DEFAULT
);
```

La tavola utenti contiene informazioni sul nome utente, password e ID del libro attualmente posseduto.

Ogni elemento della tavola è identificato dalla chiave primaria composta dai parametri Nome e Password.

L'attributo IDLibro è una chiave esterna rispetto all'attributo ID della tavola libri. Nel caso in cui il libro a cui fa riferimento viene rimosso dalla libreria, il valore dell'attributo IDLibro viene impostato a NULL.

5.2.3 - Restituzioni:

```
DROP TABLE IF EXISTS restituzioni;

CREATE TABLE restituzioni (
    Nome varchar(255) NOT NULL,
    Password varchar(255) NOT NULL,
    IDLibro int NOT NULL,
    Voto int,
    Commento varchar(1023),
    CHECK (Voto >= 0 AND Voto <= 5),
    PRIMARY KEY (Nome, Password, IDLibro),
    FOREIGN KEY (Nome, Password) REFERENCES utenti(Nome, Password) ON
DELETE CASCADE,
    FOREIGN KEY (IDLibro) REFERENCES libri(ID) ON DELETE CASCADE
);
```

Tavola contenente le informazioni relative alla restituzione di libri da parte degli utenti, registrata da Nome e Password dell'utente, IDLibro del libro restituito, un Voto (compreso tra 0 e 5) e Commento rilasciato dall'utente al momento della restituzione.
Nome, Password e IDLibro identificano ogni elemento della tavola (chiave primaria), ma sono anche chiavi esterne (Nome e Password sono chiavi esterne per la tavola utenti e IDLibro è chiave esterna per la tavola libri).

La rimozione del libro referenziato o dell'utente a cui appartiene la restituzione causa una rimozione della recensione.

5.3 - Procedure definite sulla base di dati:

Alcune di queste procedure servono per semplificare la gestione di informazioni dentro il database, altre incapsulano comportamenti più complessi.

Per delimitare le procedure utilizzando un carattere diverso da ";" viene usato il carattere "//".

```
DELIMITER //
```

Molte di queste procedure sono molto semplici ed abbastanza autoesplicative (aggiungiUtente, rimuoviUtente, rimuoviLibro. AggiungiLibro).

5.3.1 - Prendi Libro:

```
CREATE PROCEDURE prendiLibro(IN inID int,
                             IN inNome varchar(255),
                             IN inPassword varchar(255))
BEGIN
      IF (SELECT EXISTS(
            SELECT *
            FROM utenti
            WHERE Nome = inNome
            AND Password = inPassword
            AND IDLibro IS NULL
      )) THEN
            IF (SELECT NOT EXISTS(
                  SELECT *
                  FROM utenti
                  WHERE IDLibro = inID
            )) THEN
                  UPDATE utenti
                  SET IDLibro = inID
                  WHERE Nome = inNome AND Password = inPassword;
            END IF;
      END IF;
END //
```

Controlla se l'utente selezionato non possiede nessun libro e che il libro selezionato non sia già posseduto da un altro utente. Se questo è il caso, aggiorna l'attributo IDLibro dell'utente selezionato con l'ID selezionato.

5.3.2 - Restituisci Libro:

```
CREATE PROCEDURE restituisciLibro(IN inID int,
                                  IN inNome varchar(255),
                                  IN inPassword varchar(255),
                                  IN inVoto int,
                                  IN inCommento varchar(255))
BEGIN
     IF (SELECT EXISTS(
          SELECT *
          FROM utenti
          WHERE Nome = inNome
          AND Password = inPassword
          AND IDLibro = inID
     )) THEN
          IF (SELECT EXISTS(
               SELECT *
               FROM restituzioni
               WHERE Nome = inNome
               AND Password = inPassword
               AND IDLibro = inID
          )) THEN
               UPDATE restituzioni
               SET Voto = inVoto, Commento = inCommento
               WHERE Nome = inNome
               AND Password = inPassword
               AND IDLibro = inID;

          ELSE
               INSERT INTO restituzioni(Nome,
                                        Password,
                                        IDLibro,
                                        Voto,
                                        Commento)
               VALUES (inNome, inPassword, inID, inVoto, inCommento);
          END IF;

          UPDATE utenti
          SET IDLibro = NULL
          WHERE Nome = inNome AND Password = inPassword;

     END IF;
END //
```

Questa procedura controlla che l'utente possiede il libro che si vuole restituire e a seconda del caso in cui esista già o no una recensione lasciata dall'utente selezionato sul libro selezionato, verrà creata modificata la recensione esistente o creata una nuova.
Successivamente viene rimosso il libro posseduto dall'utente.

5.3.3 - Voto Medio:

```
CREATE PROCEDURE votoMedio(IN inID int)
BEGIN
    SELECT AVG(Voto)
    FROM restituzioni
    WHERE IDLibro = inID;
END //
```

Calcola la media dei voti delle recensioni di un libro.

5.4 Accesso alla base dati:

- Indirizzo: solidgallium.ddns.net
- Porta: 3306
- Utente: josh
- Password: password
- Database: Biblioteca

6 - Client

Il client è un programma scritto in Python con l'utilizzo di tre librerie principali:

- **mysql-connector-python:** modulo per collegare il programma al database, inviare query e ottenere i risultati di tali query.
- **tkinter:** modulo per la creazione di un interfaccia grafica.
- **ttkthemes**: modulo per l'utilizzo di temi e stili grafici per l'interfaccia grafica.

Inizialmente, il client si collega al database tramite mysql-connector-python:

globalVars.py:

```
# object to store the connection to the database
mydb = None

try:
    # MySQL database connection
    mydb = mysql.connector.connect(
                host = "solidgallium.ddns.net", # database IP (:3306)
                user = "josh",                  # username
                password = "password",          # password
                database = "Biblioteca"         # database to use
            )

    # cursor to issue commands to the database
    cursor = mydb.cursor(buffered = True)

# catch any exception in case the database is not reachable
except Exception as e:
    print(e)
```

Una volta stabilita la connessione al database, il client manda messaggi SQL alla base dati tramite cursore (`cursor`) per ottenere informazioni sui libri, utenti e restituzioni, per fare ricerca di dati, modificare i dati (libri, utenti) e gestire la presa in prestito e restituzione dei libri da parte degli utenti.

Codice per inviare query al database:

globalVars.py:

```
# function to send commands to the database
def sendMySQL(command):
    global mydb

    # variable for storing the result of the query
    result = None

    try:
        # send the MySQL command
        cursor.execute(command)
        print(cursor)
        # store the result
        result = cursor.fetchall()
        # commit the changes to the database
        mydb.commit()

    # catch any exception
    except Exception as e:
        print(e)

    finally:
        # return the result of the command
        return(result)
```

Le query SQL inviate dal client spesso coincidono con le procedure definite nella base dati, ma vengono eseguite direttamente piuttosto che chiamando procedure per poter notificare all'utente l'incorrettezza della richiesta in caso di comportamento anomalo:

6.1 - Codice per restituire un libro e lasciare una recensione:

widgets.py (424-494):

```
# hash the password to check the matching on the database
hashedPassword = sha256(getPassword().encode('utf-8')).hexdigest()

# get the ID of the book owned by the current user
ownedBook = sendMySQL("SELECT IDLibro " +
                      "FROM utenti " +
                      "WHERE Nome = '" + getUser() + "' "
                      "AND Password = '" + hashedPassword + "';")

# if the value is None (NULL)
if ownedBook[0][0] is None:
      # notify the user that they don't have any book owned
      messagebox.showerror("Errore Restituzione",
                           "Non possiedi alcun libro da restituire")
      return()

# if the length of the comment is too large
if len(widgets["commentTextBox"].get(1.0, "end-1c")) > 1023:
      # notify the user
      messagebox.showerror("Errore di commento",
                           "Il commento non può superare i 1023 caratteri")
      return()

# get the previous review left by the current user on the current book
previousReview = sendMySQL("SELECT * " +
                           "FROM restituzioni " +
                           "WHERE Nome = '" + getUser() + "' " +
                           "AND Password = '" + hashedPassword + "' " +
                           "AND IDLibro = '" + str(ownedBook[0][0]) + "';")

# if there is no previous review left, create a new one
if len(previousReview) == 0:
      # default construction of the INSERT query to return the book
      insertSQL = "INSERT INTO restituzioni(Nome, Password, IDLibro, Voto"
      valuesSQL = "VALUES ('" + getUser() + "', '" + hashedPassword + "', '"
                           + str(ownedBook[0][0]) + "', '" +
                           getStrings()["scoreDropdown"].get() + "'"
```

```
        # if the user left a comment, add it in the query
        if widgets["commentTextBox"].get(1.0, "end-1c") != "":
            insertSQL += ", Commento) "
            valuesSQL += ", '"+widgets["commentTextBox"].get(1.0, "end-1c")
                                                              +"');"
        # otherwise terminate the query as it is
        else:
            insertSQL += ") "
            valuesSQL += ");"

        # send the query to return the book in the "restituzioni" table
        sendMySQL(insertSQL + valuesSQL)

    # otherwise if a review already exists, update the existing one
    else:
        # default construction of the UPDATE query to leave a review
        updateSQL = "UPDATE restituzioni " +
                    "SET Voto = '" + getStrings()["scoreDropdown"].get() + "'"
        whereSQL = "WHERE Nome = '" + getUser() + "' " +
                   "AND Password = '" + hashedPassword + "';"

        # if the user left a comment, add it in the query
        if widgets["commentTextBox"].get(1.0, "end-1c") != "":
            updateSQL += ", Commento = '" +
                         widgets["commentTextBox"].get(1.0, "end-1c") + "' "
        else:
            updateSQL += " "

        # send the query
        sendMySQL(updateSQL + whereSQL)


    # update the user's owned book in the "utenti" table
    sendMySQL("UPDATE utenti " +
              "SET IDLibro = NULL " +
              "WHERE Nome = '" + getUser() + "' " +
              "AND Password = '" + hashedPassword + "';")

    # update the UI to show the owned book
    getStrings()["booksOwned"].set("Libro: ")
    setBooksOwned("")
```

6.2 - Codice per prendere in prestito un libro:

widgets.py (355-400):

```
# if there is no book selected
if len(widgets["books"].selection()) == 0:
      # notify the user
      messagebox.showerror("Errore di selezione",
                           "Non è stato selezionato nessun libro")
      return()

# hashed password to check if it matches in the database
hashedPassword = sha256(getPassword().encode('utf-8')).hexdigest()
# string containing the selected book ID
selectedBook = str(widgets["books"].item(widgets["books"].selection())
                                                          ["values"][0])

# query to get the book owned by the current user
ownedBook = sendMySQL("SELECT IDLibro " +
                      "FROM utenti " +
                      "WHERE Nome = '" + getUser() + "' "
                      "AND Password = '" + hashedPassword + "';")

# if the user already owns a book
if not(ownedBook[0][0] is None):
      # notify the user
      messagebox.showerror("Errore di prestito",
                           "Possiedi già un libro in prestito")
      return()


# query to check if the selected book is already owened by another user
availability = sendMySQL("SELECT IDLibro " +
                         "FROM utenti " +
                         "WHERE IDLibro = '" + selectedBook + "';")

# if the book is already borrowed
if len(availability) != 0:
      # notify the user
      messagebox.showerror("Errore di selezione",
                           "Questo libro è già stato preso in prestito")
      return()
```

```
# query to update  the user's owned book ID with the selected one
sendMySQL("UPDATE utenti " +
          "SET IDLibro = '" + selectedBook + "' " +
          "WHERE Nome = '" + getUser() + "' " +
          "AND Password = '" + hashedPassword + "';")

# update the UI to show the owned book
setBooksOwned(widgets["books"].item(widgets["books"].selection())["values"]
                                                                    [2])
getStrings()["booksOwned"].set("Libro: " +
     widgets["books"].item(widgets["books"].selection())["values"][2])
```

6.3 - Validazione utenti:

Come si può notare nel codice del client, la password degli utenti viene criptata tramite algoritmo di hashing SHA-256. Questo metodo consente l'occultazione della visibilità della password nel database, così che chiunque riesca ad avere accesso alla base dati (più in specifico alla tavola "utenti") non sarà in grado di decifrare le password salvate.

Prima di consentire una qualsiasi operazione che riguarda gli utenti, il client verifica l'identità dell'utente tramite nome utente e password (convertita tramite hash function) confrontandole con i valori della tavola "utenti".

7 - Repository Github:

Codice sorgente per database, client e script di installazione, file originale BIBLIOTECA.csv e relazione sono disponibili su:

https://github.com/Joshua-Micheletti/progettoBasiDati