Progetto Tepsit Query

Questa relazione è stata creata da Deiana Andrea Mario, 5A info, per spiegare in cosa consista questo progetto.

Indice :

Cos’è questo progetto?

In questo progetto si vedranno come due o più materie di studio si possano intersecare in un compito e questa relazione ha lo scopo di spiegare cosa sia questo progetto, come è stato sviluppato e cosa si può capire da questo esercizio.

Nella seconda settimana di settembre si è avuta la necessità di creare un programma che da
python si collega ad un database in un server presente nella scuola in aule diverse; per accedere a questo database sono state fornite delle credenziali per accedere al sito che gestiva il database (phpmyadmin).

Le specifiche del programma sono state scritte in un foglio che poi è stato consegnato a noi studenti; in questo foglio si è specificato il problema, delle richieste aggiuntive, le credenziali per accedere al sito del database e un esempio di codice per creare un database (visto che in quel periodo di tempo non avevamo ancora introdotto i database in altre materie).

Il programma usato

Per poter risolvere questo esercizio si è usato PYcharm che fornisce un'interfaccia utente fruibile e dinamica.

PYcharm è un IDE scaricabile gratuitamente con la community edition che offre un'interfaccia collaborativa e efficiente, rivolta a chi non è un esperto nell'ambito della programmazione. Infatti quando si incomincia a scrivere, il programma mostra delle parole chiave che possono essere quelle che si vogliono inserire con anche una spiegazione di cosa vogliono dire (Ctrl + Alt sulla parola chiave). Sia in PYcharm si possono importare le librerie che servono al programmatore, per questo progetto ci servirà la libreria socket, mysql.connector, convertitore e threading; dove è presente "as" lo si importa con una scrittura diversa (nella maggiorparte dei casi è un abbreviazione).

Per poter utilizzare la libreria mysql.connector si è dovuto usare il comando [pip install <nome_libreria>]

che permette di installare librerie o elementi esterni che non esistono su python.

La libreria socket serve per la comunicazione tra client (utente che si collega per cercare un servizio) e un server (ente che offre quel servizio che si cerca) che comunicano attraverso un protocollo chiamato TCP (Transmission Control Protocol, è un protocollo del livello 4 del modello ISO-OSI); quando si vuole programmare con la libreria socket è frequente un errore in cui bisogna cambiare porta di comunicazione perché il programma non riesce a chiudere la porta prima del termine della comunicazione.

Mysql.connector serve per comunicare con il database da python e in questo progetto vengono richieste le CRUD (offerte da mysql.connector) che sarebbero le operazioni base che un programmatore deve saper fare. Le CRUD sono:

C - Create, aggiunge o crea un elemento nel database;

R - Read, legge gli elementi di un database;

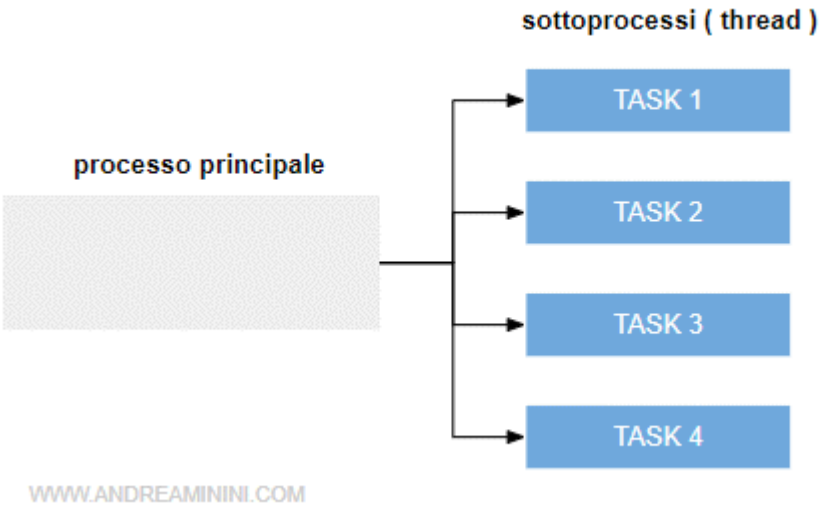
U - Update, aggiorna gli elementi in un database;

D - Delete, elimina uno o più elementi in un database.

La libreria convertitore è una libreria creata da un terzo ente che permette di prendere i risultati delle

CRUD e di poterli mandare al client attraverso delle funzioni integrate.

La libreria Threading serve per abilitare la comunicazione tra un server e più utenti (nella immagine di sotto proposta il processo principale è il server e i sottoprocessi sono i client);



in questo progetto è richiesto un programma che riesca a far comunicare n-utenti con il loro database con un solo server.

Le CRUD

Le CRUD su python vengono implementate principalmente con questo codice:

```python
def db_leggi():
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    cur = conn.cursor()
    query = "SELECT * FROM dipendenti_deiana_andrea_mario"
    cur.execute(query)
    dati = cur.fetchall()
    print(dati)
    return dati
#-----------------------------------------------------------------------------------------------------------
def db_elimina(i):
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    cur = conn.cursor()
    query = f"DELETE FROM `dipendenti_deiana_andrea_mario` WHERE id={i} "
    cur.execute(query)
    conn.commit()
#-----------------------------------------------------------------------------------------------------------
```

```python
def db_inserisci(parametri):
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    clausole = "("
    for key in parametri:
        clausole += f"{key},"
    clausole = clausole[:-1]
    clausole += ") VALUES ("
    for val in parametri.values():
        clausole += f"'{val}',"
    clausole = clausole[:-1]
    clausole += ")"
    cur = conn.cursor()
    query = f"INSERT INTO dipendenti_deiana_andrea_mario {clausole}"
    print(query)
    cur.execute(query)
    conn.commit()
#-----------------------------------------------------------------------------------------------------------
```

```python
def db_modifica(parametri):
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    #UPDATE `dipendenti_deiana_andrea_mario` SET
    #`id`=[value-1],`nome`=[value-2],`cognome`=[value-3],`posizione_lavorativa`=[value-4],`settore_di_lavoro`=[value-5],`turno`=[value-6],`data_di_assunzione`=[value-7] WHERE 1
    clausole = ""
    for key, value in parametri.items():
        clausole += f"{key} = '{value}' "
    cur = conn.cursor()
    query = f"UPDATE `dipendenti_deiana_andrea_mario` SET {clausole} WHERE 1"
    print(query)
    cur.execute(query)
    conn.commit()
#-----------------------------------------------------------------------------------------------------------
```

In ogni funzione sopra mostrata è presente una parte dove accede in automatico alla pagina web del database con le credenziali, password, la porta e l'indirizzo ip fornito.

Con solo questo codice non si può iniziare una comunicazione con il database, infatti server anche:

```
def comm_tabella (conn,data):
    while True:
        conn.send("Benvenuto utente, cosa vuole fare con la sua tabella ? : \n C-Create; \n R-Read; \n U-Update; \n D-Delete.".encode())
        data = conn.recv(1024)
# ------------------------------------------------------------------------------------------
        if data.decode() == "C" :
           p = conn.recv(1024)
           parametri = C.bytes_to_dict(p)
           conn.send("elemento aggiunto .".encode())
# ------------------------------------------------------------------------------------------
        if data.decode() == "R":
            lista = list(db_leggi())
            conn.send(C.list_to_bytes(lista))
# ------------------------------------------------------------------------------------------
        if data.decode() == "U":
            p = conn.recv(1024)
            parametri = C.bytes_to_dict(p)
            conn.send("elemento modificato .".encode())
# ------------------------------------------------------------------------------------------
        if data.decode() == "D":
            conn.send("inserisci l'operaio da eliminare [inserisci l'id]".encode())
            i = conn.recv(1024)
            db_elimina(i.decode())
            conn.send("è stato eliminato l'operaio".encode())
#-------------------------------------------------------------------------------------------
```

Con il codice che viene mostrato di sopra si attuano le CRUD; di seguito verranno spiegate le linee di codice sopra mostrate.

```
def db_leggi():
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    cur = conn.cursor()
    query = "SELECT * FROM dipendenti_deiana_andrea_mario"
    cur.execute(query)
    dati = cur.fetchall()
    print(dati)
    return dati
#-------------------------------------------------------------------------------------------
def db_elimina(i):
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    cur = conn.cursor()
    query = f"DELETE FROM `dipendenti_deiana_andrea_mario` WHERE id={i} "
    cur.execute(query)
    conn.commit()
#-------------------------------------------------------------------------------------------
```

La funzione db_leggi serve a ottenere i dati del database e a stamparli su python e prende in input i che sarebbe un valore inserito dall'utente, eseguendo sul database il comando query [SELECT * FROM <nome_tabella>] in cui :
- SELECT * indica di selezionare tutte le entri della tabella e di salvarle in una tupla;
- FROM <nome_tabella> indica ad un generico comando da quale tabella deve prendere i dati.

La funzione db_elimina elimina un record dal database (non verra più mostrato quel record) con il comando query [DELETE FROM <nome_tabella> WHERE id={valore(i)}] in cui:
-DELETE FROM <nome_tabella> indica da quale tabella si vuole eliminare il record;
-WHERE id={valore} indica con quale criterio bisogna eliminare il/i record, cioè in questo caso bisognerà inserire un valore usando la formattazione f-string (f"delete from <nome_tabella> where id={valore}") che ci permette di inserire un valore con una variabile esterna in una stringa che in questo caso è il comando query.

```
def db_inserisci(parametri):
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    clausole = "("
    for key in parametri:
        clausole += f"{key},"
    clausole = clausole[:-1]
    clausole += ") VALUES ("
    for val in parametri.values():
        clausole += f"'{val}',"
    clausole = clausole[:-1]
    clausole += ")"
    cur = conn.cursor()
    query = f"INSERT INTO dipendenti_deiana_andrea_mario {clausole}"
    print(query)
    cur.execute(query)
    conn.commit()
#-----------------------------------------------------------------------------------------------------------------------
```

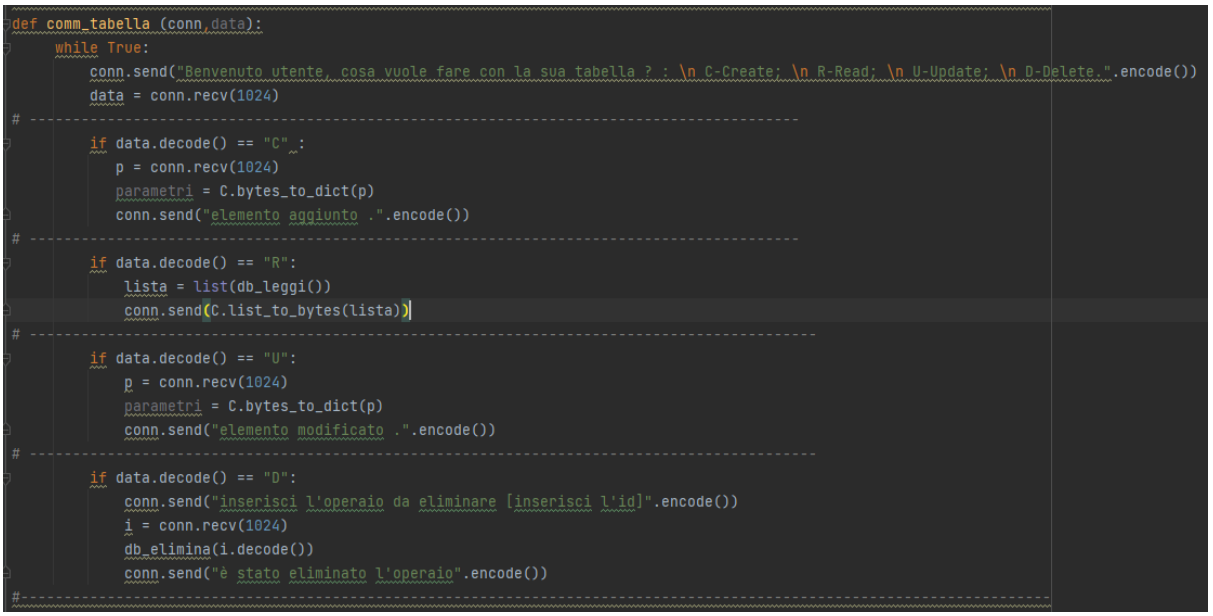
Con la funzione db_inserisci si inserisce un elemento nel database tramite dei valori che l'utente ha inserito e quei valori vengono poi presi dalla key che è un dizionario che contiene il nome della colonna del database con il valore della entry.

Il comando che viene usato è [f"INSERT INTO <nome_tabella> {clausola}"] in cui:
-INSERT INTO <nome_tabella> specifica in quale tabella bisogna inserire i dati;
-{clausola} che è un dizionario composto dai valori inseriti dall'utente con il nome della colonna in cui deve essere scritto, dentro a clausole, oltre ai dati è stato scritto anche parte del comando originale, cioè ["(nome_colonna) VALUES ([value-1],[value-2],[value-n])] in cui VALUES specifica che ci andranno i valori e [value-n] sono i valori che l'utente inserirà; nella funzione viene passato [parametri] che contiene i valori che ha inserito l'utente.

```
def db_modifica(parametri):
    conn = mysql.connector.connect(
        host="10.10.0.10",
        user="andreamario_deiana",
        password="deiana1234",
        database="5ATepsit",
        port=3306,
        )
    #UPDATE `dipendenti_deiana_andrea_mario` SET
    # `id`=[value-1],`nome`=[value-2],`cognome`=[value-3],`posizione_lavorativa`=[value-4],`settore_di_lavoro`=[value-5],`turno`=[value-6],`data_di_assunzione`=[value-7] WHERE 1
    clausole = ""
    for key, value in parametri.items():
        clausole += f"{key} = '{value}' "
    cur = conn.cursor()
    query = f"UPDATE `dipendenti_deiana_andrea_mario` SET {clausole} WHERE 1"
    print(query)
    cur.execute(query)
    conn.commit()
#-----------------------------------------------------------------------------------------------------------------------
```

Infine, con la funzione db_modifica che prende in input parametri, inserisce nel database dei valori che l'utente ha inserito, seguendo la logica della clausola

(spiegata in precedenza), eseguendo questo comando query [f”UPDATE <nome_tabella> SET {clausole} WHERE 1”] in cui:
-UPDATE <nome_tabella> specifica in quale tabella bisogna aggiornare i dati;
-SET {clausole} che serve per inserire nel comando i valori che l’utente ha specificato di aggiornare, sfruttando una key che è composta dal nome della colonna con il valore da modificare;
-WHERE 1 indica che il comando sia corretto.



```
def comm_tabella (conn,data):
    while True:
        conn.send("Benvenuto utente, cosa vuole fare con la sua tabella ? : \n C-Create; \n R-Read; \n U-Update; \n D-Delete.".encode())
        data = conn.recv(1024)
# ------------------------------------------------------------------------------------------
        if data.decode() == "C" :
           p = conn.recv(1024)
           parametri = C.bytes_to_dict(p)
           conn.send("elemento aggiunto .".encode())
# ------------------------------------------------------------------------------------------
        if data.decode() == "R":
            lista = list(db_leggi())
            conn.send(C.list_to_bytes(lista))
# ------------------------------------------------------------------------------------------
        if data.decode() == "U":
            p = conn.recv(1024)
            parametri = C.bytes_to_dict(p)
            conn.send("elemento modificato .".encode())
# ------------------------------------------------------------------------------------------
        if data.decode() == "D":
            conn.send("inserisci l'operaio da eliminare [inserisci l'id]".encode())
            i = conn.recv(1024)
            db_elimina(i.decode())
            conn.send("è stato eliminato l'operaio".encode())
#-------------------------------------------------------------------------------------------
```

Queste quattro funzioni vengono gestite da un ulteriore funzione che ha il compito di chiamare una determinata funzione quando l’utente metta una lettera dell’acronimo CRUD; questa funzione si chiama comm_tabella che è composta da un [while true] e 4 if che servono a chiedere cosa vuole fare l’utente. Dove è presente la lettera p vengono presi i valori che l’utente ha inserito e li mette in parametri che poi verra usato nelle funzioni.

```python
if __name__ == "__main__" :
    Lock = t.Lock()
    HOST = 'localhost'                  # Nome simbolico che rappresenta il nodo locale
    PORT = 50016                        # Porta non privilegiata arbitraria
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((HOST, PORT))
    s.listen(1)
    conn, addr = s.accept()
    print('Connected by', addr)
    thread = []
    lista_connessioni = []
    i = 0
    TRY = 0
    while TRY < 2 :
        # ------------------------------------------------------------------------------------------------
        lista_connessioni.append(s.accept())
        data = conn.recv(1024)
        # ------------------------------------------------------------------------------------------------
        if data.decode() == "topogigio" :
            if i == 1:
                thread.append(t.Thread(target=comm_tabella (conn,data), args=(lista_connessioni, 0)))
                thread.append(t.Thread(target=comm_tabella (conn,data), args=(lista_connessioni, 1)))
                thread[0].start()
                thread[1].start()
                thread[0].join()
                thread[1].join()
            conn.send("password accettata, premere invio. \n".encode())
            th = t.Thread(target=comm_tabella(conn,data),args=(conn,data))
        else :
            TRY = TRY + 1
            conn.send("inserisci di nuovo la password :".encode())
    conn.close()
```

Questo è il corpo principale del programma sul server multithreading con i lock.

Il server crea il socket che appartiene alla famiglia AF_INET che serve per le comunicazioni di rete e questo socket verrà usato dagli utenti che si connetteranno per accedere al database; per gestire i vari utenti si è deciso che si userà la libreria (con le relative funzioni) multithreading che gestisce le comunicazioni con più utenti.

Nel server è presente anche un sistema di sicurezza attraverso l'uso di una password che se sbagliata 3 volte si disconnette (il server) in automatico, ma se la password è corretta allora inserisce in una lista di connessioni la sessione dell'utente che poi lavorerà con il database (per il programma che ho creato la password sarà "topogigio").

Oltre al codice del server deve essere presente anche il codice dell'utente che è principalmente questo :

```python
import socket
import convertitore as C

# client

if __name__ == "__main__":
    HOST = 'localhost'
    PORT = 50016
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((HOST, PORT))
    print("benvenutto,inserire la password: ")
    messaggio = input()
    s.send(messaggio.encode())
    while True:
        if messaggio == "STOP":
            s.close()
        data = s.recv(1024)
        print(data.decode())
        messaggio = input()
        s.send(messaggio.encode())
        if messaggio=="C":
            #id,nome,cognome,posizione_lavorativa,settore_di_lavoro,turno,data_di_assunzione
            nome = input("inserisci nome da inserire nei dati: ")
            cognome = input("inserisci cognome da inserire nei dati: ")
            posizione_lavorativa = input("inserisci posizione lavorativa da inserire nei dati: ")
            settore_di_lavoro = input("inserisci settore di lavoro da inserire nei dati: ")
            turno = input("inserisci turno da inserire nei dati: ")
            data_di_assunzione = input("inserisci data di assunzione da inserire nei dati [ format : AA-MM-GG ]: ")
            par = {"nome": nome, "cognome": cognome, "posizione_lavorativa":posizione_lavorativa,
                   "settore_di_lavoro": settore_di_lavoro, "turno":turno, "data_di_assunzione":data_di_assunzione}
            s.send(C.dict_to_bytes(par))
        if messaggio=='R':
            data = s.recv(1024)
            dati = C.bytes_to_list(data)
            print(dati)
```

```python
if messaggio=="U":
    #id,nome,cognome,posizione_lavorativa,settore_di_lavoro,turno,data_di_assunzione
    nome = input("inserisci il nome da modificare: ")
    cognome = input("inserisci il cognome da modificare: ")
    posizione_lavorativa = input("inserisci la posizione lavorativa da modificare: ")
    settore_di_lavoro = input("inserisci il settore di lavoro da modificare: ")
    turno = input("inserisci il turno da modificare: ")
    data_di_assunzione = input("inserisci la data di assunzione da modificare [ format : AA-MM-GG ]: ")
    par = {"nome": nome, "cognome": cognome, "posizione_lavorativa":posizione_lavorativa,
           "settore_di_lavoro": settore_di_lavoro, "turno":turno, "data_di_assunzione":data_di_assunzio
    s.send(C.dict_to_bytes(par))
```

In questo codice l'utente si connette al server attraverso un socket che per accettare la connessione gli serve una password (che ipoteticamente sarà fornita da un terzo ente); una volta accettata la password l'utente può decidere cosa vuole fare con il propio database.

L'utente può inserire una lettera dell'acronimo CRUD per decidere cosa fare e se non vuole fare niente

deve inserire STOP (per forza in maiuscolo); nel codice dell’utente sono gestiti vari casi per ognuna delle CRUD, per la creazione e e l’update fa inserire i dati e li salva in un dizionario che poi passerà al server (che poi diventeranno clausola),per la lettura non richiede niente visto che non bisogna creare o aggiornare il database, il caso dell’eliminazione non viene gestito perché il dato viene chiesto soltanto l’id del dipendente.

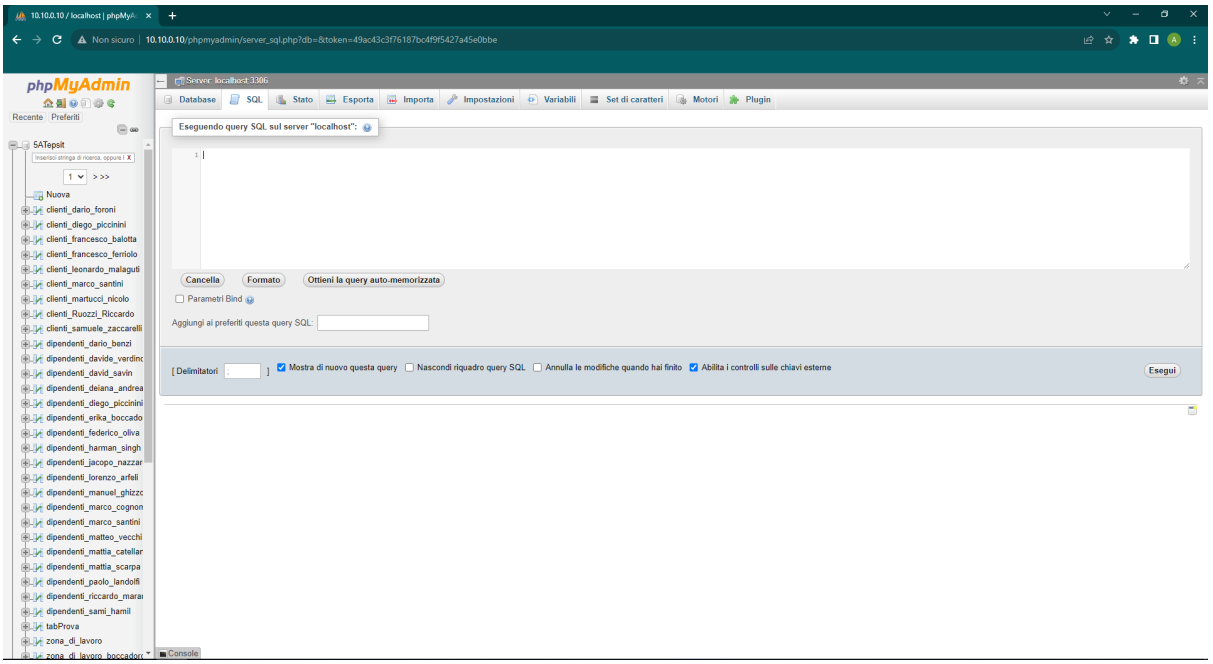
Il database

Il database è gestito tramite un programma online chiamato phpMyAdmin che permette di realizzare le query con molti programmi, nel nostro caso con python.

Questo programma può essere avviato in vari modi tra cui :

- avviando XAMPP (nel nostro lavoro non servirà XAMPP);
- inserendo l'indirizzo IP della barra di ricerca del browser usato.

Ecco un pratico esempio di come PhpMyadmin funzioni :



-Questo programma è principalmente GUI con piccoli elementi di programmazione a riga comandi, esplicitati dal sito;

-A sinistra si può vedere come ci sia il logo del programma e sotto al logo ci siano i database che principalmente è 5Atepsit e i database che sono presenti;

-Al centro della pagina è presente un menu di scelta che permette di scegliere come operare sul proprio database, scegliendo tra :

1)Database, mostra tutte le tabelle esistenti nel server;

2)SQL, permette di inserire i comandi per le tabelle come CREATE, READ o DELETE (dopo aver scritto un comando bisogna premere ESEGUI [situato a destra] per eseguirlo);

3)Stato, mostra lo stato del database con il peso nel server del database, da quanto tempo è attivo il database, il traffico (le varie operazioni fatte da altre persone) delle azioni e il tipo di connessioni;

4)Esporta, permette di salvare precisamente il database scelto su una chiavetta o un altro dispositivo per poi averlo su un altro dispositivo;

5)Importa, permette di salvare il database esportato sul proprio dispositivo scelto;

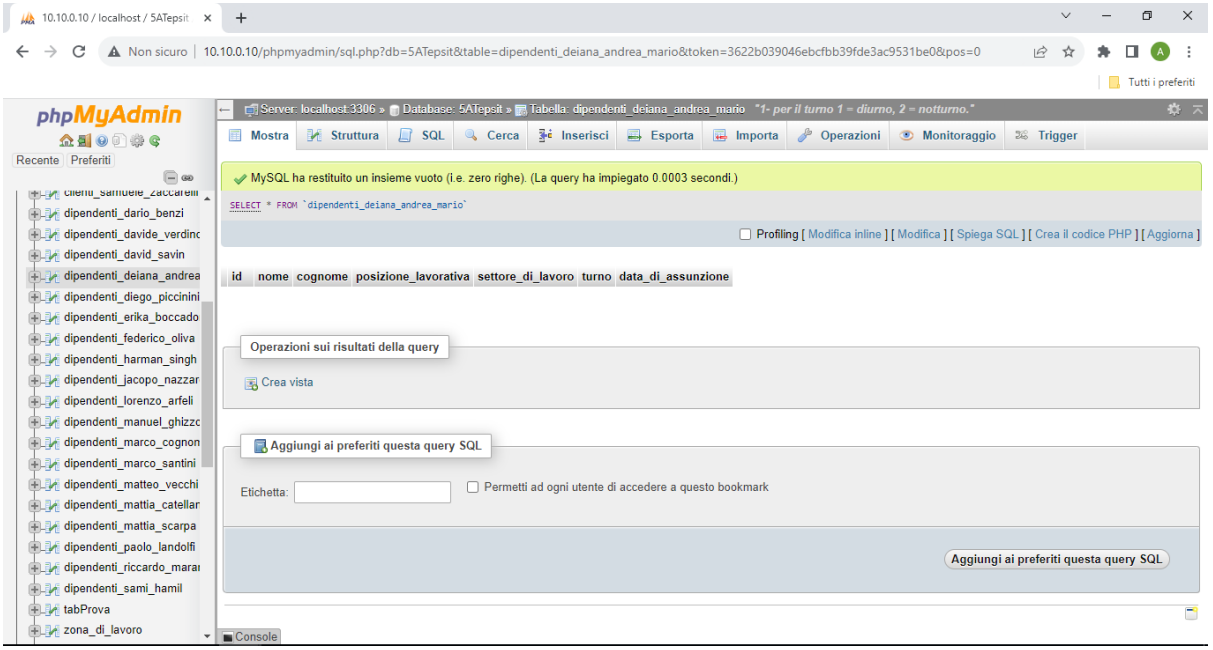
6)Impostazioni, mostra le informazioni generali del database e del programma che si sta usando per il database;

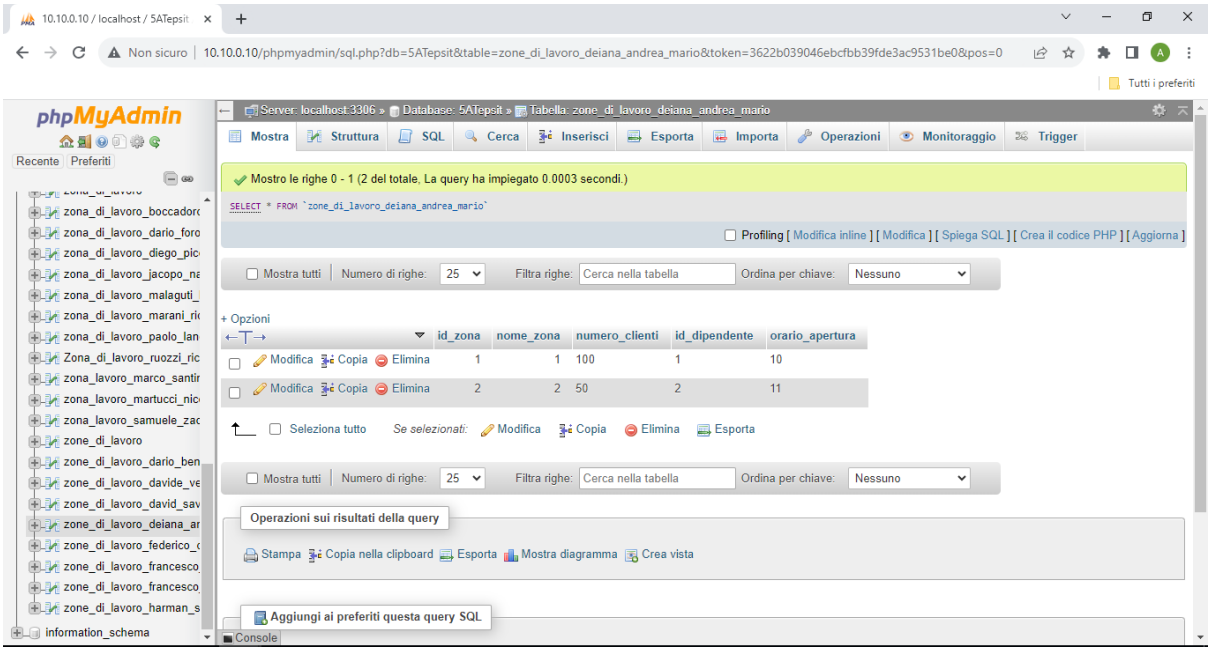
7)Variabili, mostra tutte le variabili nei vari database creati;

9)Motori, mostra con quale programma il database è gestito e la sua descrizione;

10)Plugin, permette di installare tramite phpmyadmin dei programmi secondari che aiutano alla gestione del database.

Una volta scelto uno dei nostri database verrà mostrata questa schermata :





Si può notare come sia cambiata soltanto la schermata al centro con nuove scelte, mantenendo le altre cose descritte in precedenza; le nuove scelte comparse sono :

1)Mostra, permette di vedere come è strutturato il database, mostrando gli elementi e i record presenti;

2)Struttura, si concentra sui valori delle colonne e le loro caratteristiche come il tipo di dati da inserire, se non può avere valori nulli e quanto lungo può essere il valore (lunghezza dei caratteri);

3)SQL è la zona dove si possono effettuare i comandi e dove si effettuano le query (SELECT,DELETE,ecc…);

4)Cerca, permette di cercare un elemento/i per uno o più valori inseriti (come una SELECT);

5)Inserisci al posto di scrivere il comando CREATE, PhpMyadmin permette di inserire gli elementi tramite un'interfaccia grafica;

6)Esporta e Importa, [Importa] permette di salvare il proprio database in un dispositivo di archiviazione qualsiasi (USB,cloud,il proprio disco, ecc…), [Esporta] permette di scaricare il proprio database nel dispositivo corrente;

7)Operazioni è una parte del programma che permette di gestire il database, modificando per esempio i diritti di una tabella come il fatto di poterla leggerla senza inserire elementi o entrambe ;

8)Monitoraggio mostra le varie operazioni che si son fatte sul database.

id_zona	nome_zona	numero_clienti	id_dipendente	orario_apertura
1	produzione	100	01	12.00
2	produzione	100	02	12.00

id	nome	cognome	posizione_lavorativa	settore_di_lavoro	turno	data_di_assunzione
1	andrea	deiana	produzione	primario	diurno	2023-07-03
2	andrea	mario	produzione	primario	notturno	2023-07-04
3	mario	deiana	produzione	secondario	diurno	2023-07-05
4	luca	deiana	qualità	secondario	notturno	2023-07-03

Le tabelle mostrate sono un esempio delle tabelle in cui l'utente (dipendente delle risorse umane) dovrà lavorare.

Richieste aggiuntive

Ulteriori funzionalità desiderate:

Si possono implementare a piacere le seguenti funzionalità per ottenere voti via via più alti:

- controllare i dati che inserisce il client per fare **l'escape** delle stringhe e rendere il software sicuro rispetto ad attacchi di tipo SQL injection (https://it.wikipedia.org/wiki/SQL_injection)
- nel caso (e solo in questo caso) l'utente risorse umane cancelli (con una **delete**) l'anagrafica di un dipendente; tramite una chiamata al modulo **stmpllib_mail.py** caricato da Malvezzi sullo stream classroom del corso il 19 marzo, fa si che arrivi una mail di avviso (a un indirizzo mail che indicate voi, consiglio uno vostro che usate poco così da poter controllare se arriva e non creare spam in uno che usate) con scritto che dipendente è stato cancellato a che ora e che giorno.
- usando un software di database design disegna le relazioni tra le due tabelle del progetto, consiglio **https://dbdiagram.io/home**
- Produci una documentazione tecnica, con **word** o **google doc** in cui descrivi il funzionamento e la logica del tuo software, includi se lo hai fatto il disegno di relazioni tra le tabelle.
- (**Se sei carico**) Metti un interfaccia grafica con cui l'utente interagisca al posto della riga di comando (consiglio **tkinter,** libreria python semplice e basilare. Malvezzi l'ha usata a inizio anno, potete chiedergli il modulo .py come esempio se volete).
- (**se sei carichissimo**) metti il modulo server su una macchina in cloud (suggerisco https://it.wikipedia.org/wiki/Amazon_EC2), cambiane l'indirizzo IP da locale a quello del server in cloud, così che la comunicazione client server avvenga davvero tra due macchine differenti.

Dopo aver completato il programma si è richiesto di aggiungere delle informazioni per avere un programma più completo, aumentando così la valutazione generale del programma; ecco le richieste aggiuntive:

- Fare l'escape delle stringhe per rendere il programma sicuro;
- Far sì che una volta eliminato un elemento arrivi una mail di avviso tramite un programma caricato precedentemente da uno studente che ha frequentato questo indirizzo;
- Usando DB Diagram creare un disegno delle relazioni delle due tabelle; DB Diagram serve a

disegnare le relazioni di due tabelle in cui i dati li deve fornire l'utente che sta usando il programma;
- Produrre una documentazione/relazione tecnica del programma;
- Usare la libreria Tkinter per scrivere del codice che permetta di usare l'interfaccia grafica di python;
- Collegare il database a un server cloud.

Considerazioni finali

Per completare questo progetto ci sono voluti 3 mesi di lavoro compiuto non solo a scuola ma anche a casa; questo progetto è molto utile non solo a capire come si possa interagire con un database da python, ma è utile anche per ripassare argomenti precedenti a quelli di quinta e a capire cosa potrebbe essere richiesto nel mondo del lavoro.

In questa relazione si è cercato di essere il più preciso possibile con il linguaggio tecnico, fornendo molti esempi riportati da delle immagini; le immagini riportate e il programma consegnato non includono i due casi delle tabelle perchè i comandi e il codice da creare è lo stesso.

Relazione scritta da Deiana Andrea Mario.