INTRODUZIONE

Supabase

Per questo progetto, useremo un servizio di database online gratuito chiamato Supabase. Questo ci permetterà di avere una base data sincronizzata visto che alcuni elementi come personaggi, classi e abilità sono comune a tutte le missioni. Utilizzeremo Python per mandare query SQL al servizio e quindi interrogare il database.

PostgreSQL

Supabase utilizza PostgreSQL come DBMS, al contrario di XAMPP che utilizza MariaDB. Ci sono quindi alcune differenze, la più importante il fatto che Postgres è case sensitive, a differenza di MariaDB. Se una query non funziona, una di queste differenze potrebbe essere la ragione.

Librerie

Per interrogare il database, **servirà le libreria postgres** di python. per installarla il comando è " pip install psycopg2 ". **Come sempre, se avete problemi con pip/librerie, utilizzate un virtual environment per il progetto.**

API

Ci si può interfacciare direttamente con psycopg2 e le sue strutture dati, ma questo risulterebbe non solo leggermente più complicato ma anche ridondante (se ogni gruppo facesse la sua connessione al DB, avremmo un server con multiple connessioni). Per questo vi verrà fornita una libreria custom "queryLib" tramite cui queste operazioni sono semplificate. La libreria rende comunque disponibile la connessione e il cursore nel caso si necessiti di operazioni più avanzate.

UTILIZZO

Dati di ritorno

Ogni query SQL sia questa chiamata con queryLib.execute() o con cursor.execute() ha come ritorno una lista di tuple, ognuna di queste rappresentante una riga della tabella. Ad esempio, la query 'SELECT "Nome", "Cognome" FROM persone LIMIT 2' darà come ritorno la lista di tuple: [("Jonathan", "Joestar"),("Jean Pierre", "Polnareff")]

Tramite libreria custom

La libreria queryLib mette a disposizione le seguenti funzionalità:

- .connetti() : connessione al database. verrà chiamato una sola volta da main.py
- .disconnetti() : disconnessione dal database
- .execute() : esegue una query SQL e ne ritorna il risultato
- .getHeaders() : ritorna una lista rappresentate i nomi delle colonne di una tabella
- .connection : variabile connection della libreria postgres (per usi avanzati)
- .cursor : variabile cursor della libreria postgres (per usi avanzati)

Un esempio di chiamata tramite la libreria:

```
from moduli import queryLib
queryLib.connetti()
print(
   queryLib.execute('SELECT "Vigore" from classi')
)
queryLib.disconnetti()
```

Tramite connessione diretta

Per connettersi al database, si può anche utilizzare direttamente la libreria psycopg2.

Bisognerà importarla e definire alcune variabili:

```
import psycopg2

user = "[USERNAME]"
port = 6543
dbname = "postgres"
host = "aws-0-eu-central-1.pooler.supabase.com"
```

Dopodichè si può stabilire una connessione e definire un cursore:

```
try:
   connection = psycopg2.connect(
       user=user,
       password=r"LA MIA PASSWORD",
       host=host,
       port=port,
       dbname=dbname
   )
   print("Connection successful!")
   cursor:psycopg2.extensions.cursor = connection.cursor()
except Exception as e:
   raise ConnectionError(f"Connessione Fallita: {e}")
```

Tramite il cursore è possibile interrogare il database ed estrarre i dati dell'ultima query:

```
cursor.execute('SELECT "Nome", "Cognome", FROM persone;')
result = cursor.fetchall()
print(result)
```

È importante chiudere la connessione a programma terminato:

```
cursor.close()
connection.close()
print("Connection closed.")
```