TECH STARTER

1.6.1. Gruppenaufgabe Aufgaben-Tracker

Use Case

Organisiere deine Aufgaben und Projekte effektiv mit einem persönlichen Aufgaben-Tracker. Behalte den Überblick über anstehende Arbeiten, priorisiere Aufgaben und verwalte Fristen, um deine Produktivität zu steigern.

Aufgabenbeschreibung

- Datenbank erstellen:
  - Tabelle `tasks` mit den Feldern:
    - `id` (Primärschlüssel)
    - `name` (VARCHAR)
    - `description` (TEXT)
    - `created` (DATETIME)
    - `due` (DATETIME)
    - `prio` (INTEGER)
    - `state` (VARCHAR)

- Funktionen:
  - Aufgabe hinzufügen
  - Aufgabe aktualisieren
  - Aufgabe löschen
  - Aufgaben anzeigen, sortiert nach Priorität oder Fälligkeitsdatum

- **Statusverwaltung**:
  - Definiere Statuswerte wie "Offen", "In Bearbeitung", "Abgeschlossen".

Tipps

- Datum und Uhrzeit:
  - Verwende das `datetime` Modul in Python für Datumsoperationen.
  - Achte auf das richtige Format beim Speichern in der Datenbank.

- Sortierung und Filterung:
  - Nutze SQL-Befehle, um die Aufgaben nach Priorität oder Fälligkeitsdatum zu sortieren.
  - Ermögliche das Filtern von Aufgaben nach Status.

Code Snippets

1. Aufgabe hinzufügen:

```
from datetime import datetime

def add_task(name, desc, duedate, priority):
    connection = create_connection()
    cursor = connection.cursor()
    created_at = datetime.now()
    duedate = datetime.strptime(due, "%Y-%m-%d")
    status = "Offen"
    sql = "INSERT INTO task (aname, description, createdat, due_Date,
prio, state) VALUES (%s, %s, %s, %s, %s, %s)"
    values = (taskname, description, createdate, duedate, prio1, statw)
    cursor.execute(sql, values)
    connection.commit()
    cursor.close()
    connection.close()
```

Kleine Falle: Achte darauf, dass das Datum im richtigen Format eingegeben wird. Ein Formatfehler kann zu einem Absturz des Programms führen.

2. Aufgaben anzeigen nach Priorität:

```
def view_tasks_by_priority():
    connection = create_connection()
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM todo ORDER BY priority ASC")
    result = cursor.fetchall()
    for row in result:
        print(f"ID: {row[7]}, Aufgabe: {row[6]}, Priorität: {row[5]},
Fällig bis: {row[4]}, Status: {row[2]}")
    cursor.close()
    connection.close()
```

Kleine Falle: Überprüfe, ob die Indizes in `row` korrekt sind, besonders wenn du die Reihenfolge der Felder in der Tabelle geändert hast.

Allgemeine Hinweise für alle Projekte

- Netzwerkverbindung zwischen VMs:
- Stelle sicher, dass die Python-VM die Datenbank-VM über das Netzwerk erreichen kann.

- Datenbank-Berechtigungen:
  - Erstelle einen Datenbankbenutzer mit den notwendigen Berechtigungen (SELECT, INSERT, UPDATE, DELETE).
  - **Kleine Falle:** Wenn du die Standard-Benutzeranmeldeinformationen verwendest, könnte das Sicherheitsrisiken mit sich bringen.

- Dokumentation:
  - Kommentiere deinen Code ausführlich.
  - Erstelle eine README-Datei mit Installationsanweisungen und einer Beschreibung, wie das Programm verwendet wird.

- Versionskontrolle:
  - Verwende Git, um Änderungen an deinem Code zu verfolgen.
  - Kleine Falle: Vergiss nicht, sensible Informationen wie Passwörter oder geheime Schlüssel aus dem Repository auszuschließen.

Abschließender Hinweis:

Achte darauf, jeden Schritt gründlich zu lesen und zu verstehen, bevor du ihn ausführst oder kopierst. Manchmal können kleine Details einen großen Unterschied machen. Viel Spaß beim Programmieren!