

Python Datenbank und Rest API

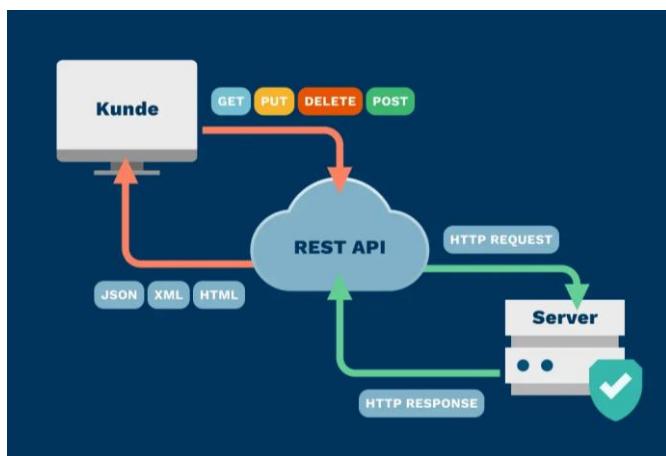
Inhalt

1	Plan.....	2
2	Die Relationale Datenbank MySQL/ MariaDB.....	4
2.1	MySQL Installation	4
2.2	PHPMyAdmin Client	6
2.2.1	XAMPP Konfiguration für den Daten Import.....	6
2.3	Datenbank anlegen.....	8
2.4	Importieren von Tabellen.....	9
2.5	Installation DBeaver	9
2.6	Werkzeuge und Editoren	10
2.7	DBEaver.....	10
2.7.1	Dokumentation	10
2.7.2	Backup Restore	10
2.7.3	Neues DBEaver Projekt	11
3	SQL	13
4	Python und Datenbanken.....	13
4.1	Modul installieren.....	13
4.2	Datenbankverbindung aufbauen.....	14
4.3	Datenklassen mit @dataclass	15
4.4	SQLAlchemy	15
4.4.1	Beispiel.....	16
4.5	Pydantic	16
4.6	Passwortverschlüsselung.....	16
5	Rest /RestFULL	17
5.1.1	Wichtige http Status-Codes.....	18
5.2	RestFull mit FastAPI.....	19
5.3	Schema mit Pydantic	20
5.3.1	Eingehender Request (Client → Server)	20
5.3.2	Ausgehende Response (Server → Client)	20
6	Anhang	21

6.1	VS Code Module	22
6.2	Python Module	22

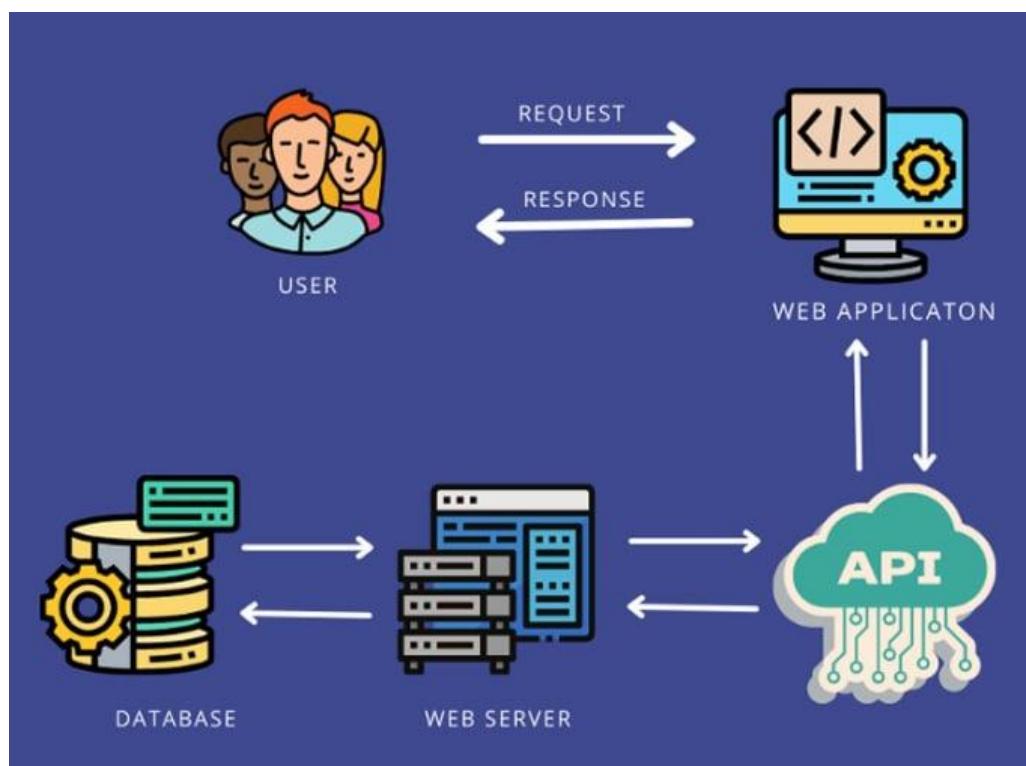
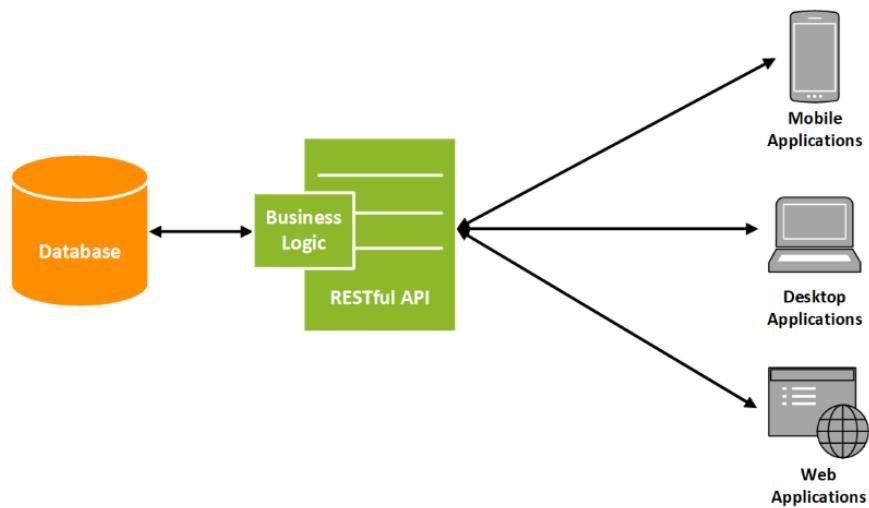
1 Plan

- Python
- Datenbank
- Datenbankframeworks
- Rest-Apis



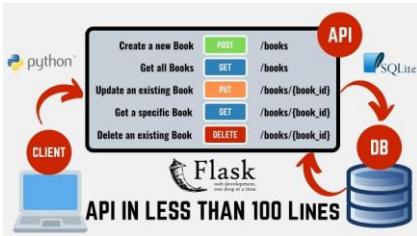
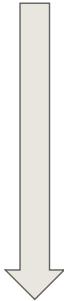
Um komplexe und flexible Anwendungen umsetzen zu können, werden heute Rest-APIs eingesetzt, die mit Datenbanken kommunizieren.

Dieses ermöglicht, dass Daten für beliebige Plattformen eingesetzt werden können.



REIHENFOLGE DER THEMEN

- Datenbank mit MySql
- Datenbank-Entwurf und Abfragen
- Python und Datenbanken (Zugriff und Abfragen)
- Einsatz von Frameworks
- Python und Rest-API
- Einfacher Client als Webseite
- Sicherheit und Optimierung



Ergebnis: eine komplette Server-Anwendung mit allen wesentlichen Komponenten Schnittstellen

8

2 Die Relationale Datenbank MySQL/ MariaDB

MySQL und MariaDB basieren auf der gleichen Implementierung. Das heißt beide Datenbanksysteme sind weitgehendst kompatibel. MySQL gehört zu Oracle und unterliegt verschiedenen Lizenzen. MariaDB ist komplett Open Source und wird von einer Community weiterentwickelt. MariaDB hat in einigen Fällen bessere Performance und auch einige geschlossene Sicherheitslücken.

Obwohl wir von MySQL sprechen, verwenden wir eigentlich genau genommen MariaDB (auch im XAMPP enthalten). Von der Benutzung sind beide Systeme zum großen Teil identisch.

2.1 MySQL Installation

Komplettpaket XAMPP

Herunterladen
Klicken Sie hier für weitere Versionen

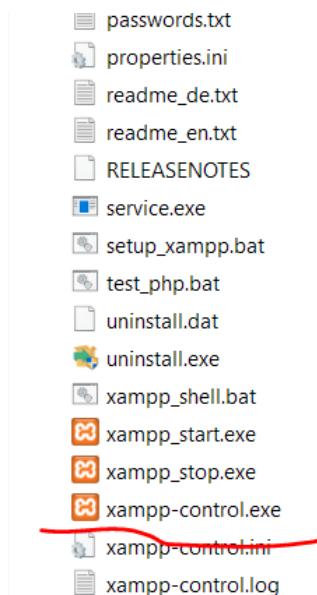
XAMPP für Windows
8.1.6 (PHP 8.1.6)

XAMPP für Linux
8.1.6 (PHP 8.1.6)

XAMPP für OS X
8.1.6 (PHP 8.1.6)

<https://www.apachefriends.org/de/index.html>

Wichtig: unter Windows direkt unter c:/xampp installieren!



Im XAMPP Control Panel muss der Webserver (Apache) und MySQL gestartet werden.

XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]

Modul	Dienst	Modul	PID(s)	Port(s)	Aktionen
		Apache	18332 21476	80, 443	Stoppen Admin Konfig Logs
		MySQL	12976	3306	Stoppen Admin Konfig Logs
		FileZilla			Starten Admin Konfig Logs
		Mercury			Starten Admin Konfig Logs
		Tomcat			Starten Admin Konfig Logs

Konfig

Netstat
Shell
Explorer
Dienste
Hilfe
Beenden

```

13:40:03 [main] Voraussetzungen werden geprüft
13:40:04 [main] Alle Voraussetzungen sind erfüllt
13:40:04 [main] Initialisiere Module
13:40:04 [Apache] XAMPP Apache ist bereits gestartet auf Port 80
13:40:04 [Apache] XAMPP Apache ist bereits gestartet auf Port 443
13:40:04 [mysql] XAMPP MySQL ist bereits gestartet auf Port 3306
13:40:04 [main] Das FileZilla Modul ist deaktiviert
13:40:04 [main] Das Mercury Modul ist deaktiviert
13:40:04 [main] Das Tomcat Modul ist deaktiviert
13:40:04 [main] Starte Check-Timer
13:40:04 [main] Control Panel bereit

```

Mit dem Admin Button von MySQL kommen wir zum MySQL Client PHPMyAdmin:

The screenshot shows the XAMPP Control Panel interface. On the left is a table of services:

Modul	Dienst	PID(s)	Port(s)	Aktionen
Apache		18332 21476	80, 443	Stoppen Admin Konfig Logs
MySQL		12976	3306	Stoppen Admin Konfig Logs
FileZilla				Starten Admin Konfig Logs
Mercury				Starten Admin Konfig Logs
Tomcat				Starten Admin Konfig Logs

On the right are various links: Konfig, Netstat, Shell, Explorer, Dienste, Hilfe, and Beenden. A red circle highlights the 'Admin' button for the MySQL service. Below the table is a log window:

```

13:40:03 [main] Voraussetzungen werden geprüft
13:40:04 [main] Alle Voraussetzungen sind erfüllt
13:40:04 [main] Initialisiere Module
13:40:04 [Apache] XAMPP Apache ist bereits gestartet auf Port 80
13:40:04 [Apache] XAMPP Apache ist bereits gestartet auf Port 443
13:40:04 [mysql] XAMPP MySQL ist bereits gestartet auf Port 3306
13:40:04 [main] Das FileZilla Modul ist deaktiviert
13:40:04 [main] Das Mercury Modul ist deaktiviert
13:40:04 [main] Das Tomcat Modul ist deaktiviert
13:40:04 [main] Starte Check-Timer
13:40:04 [main] Control Panel bereit

```

2.2 PHPMyAdmin Client

2.2.1 XAMPP Konfiguration für den Daten Import

Das Importieren von Daten erfolgt in der Regel über *.sql Skripte. Da diese sehr groß sein können, müssen wir XAMPP so konfigurieren, dass er auch mit großen Dateien keine Probleme hat.

Unter Konfig PHP (php.ini) auswählen.

The screenshot shows the XAMPP Control Panel interface. On the left is a table of services:

Modul	Dienst	PID(s)	Port(s)	Aktionen
Apache		18332 21476	80, 443	Stoppen Admin Konfig Logs
MySQL		12976	3306	Stoppen Admin Konfig Logs
FileZilla				Starten Admin Konfig Logs
Mercury				Starten Admin Konfig Logs
Tomcat				Starten Admin Konfig Logs

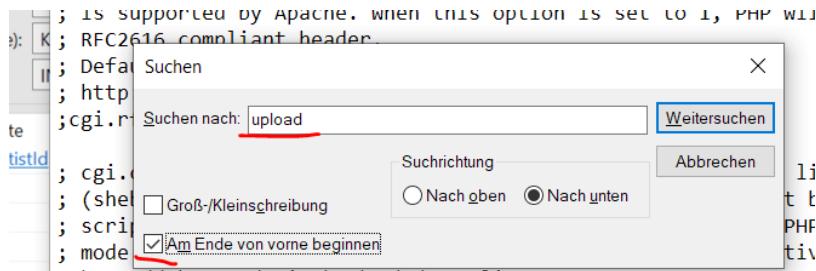
On the right are various links: Netstat, Shell, Explorer, Dienste, Hilfe, and Beenden. A red circle highlights the 'Konfig' button for the MySQL service. Below the table is a log window:

```

3 [main] Voraussetzungen werden geprüft

```

In der Konfigurationsdatei suchen wir nach *upload*:



```
; is supported by Apache. When this option is set to 1, PHP will
; RFC2616 compliant header.
;
; Default search
;
; http
;cgi.r Suchen nach: upload Weitersuchen
; cgi.r Suchrichtung
; (shell Groß-/Kleinschreibung
; script Nach oben Nach unten
; mode Am Ende von vorne beginnen
; http://php.net/cgi.check-shebang-line
;cgi.check_shebang_line=1

;;;;;;;;;;
; File Uploads ;
;;;;;;;;;

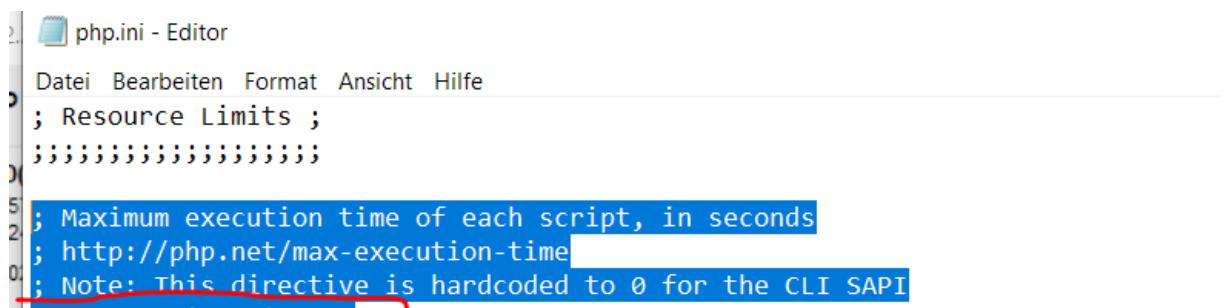
; Whether to allow HTTP file uploads.
; http://php.net/file-uploads
file_uploads=On

; Temporary directory for HTTP uploaded files (will use system
; specified).
; http://php.net/upload-tmp-dir
upload_tmp_dir="C:\xampp\tmp"

; Maximum allowed size for uploaded files.
; http://php.net/upload-max-filesize
upload_max_filesize=2M
```

Kann man z.B. auf 50M (50 MB) erhöhen und Datei speichern.

Des Weiteren empfiehlt es sich die **max_execution_time** zu erhöhen (z.B. 500):



```
php.ini - Editor
Datei Bearbeiten Format Ansicht Hilfe
; Resource Limits ;
;;;;;;;;;

; Maximum execution time of each script, in seconds
; http://php.net/max-execution-time
; Note: This directive is hardcoded to 0 for the CLI SAPI
max_execution_time=30

; Maximum amount of time each script may spend parsing request data. It's a go
; idea to limit this time on production servers in order to eliminate unexpe
; long running scripts.
; Note: This directive is hardcoded to -1 for the CLI SAPI
; Default Value: -1 (Unlimited)
; Development Value: 60 (60 seconds)
```

Achtung, danach muss Apache nochmal gestartet werden, damit die Konfiguration neu eingelesen wird.

2.3 Datenbank anlegen

Unter dem Punkt Datenbanken können wir diese anlegen. Wichtige ist UTF8-Zeichensatz einzustellen:

Achtung! MySQL hat mehrere UTF-8 Zeichensatz Varianten. Wenn für uns deutsche Umlaute relevant sind, sollten wir utf8_general_ci einstellen. Das kann auch für jede Spalte der Tabelle einzeln später

The screenshot shows the 'Datenbanken' (Databases) section of the phpMyAdmin interface. In the top navigation bar, 'Datenbanken' is selected. Below it, there's a form for creating a new database with fields for 'Datenbankname' (set to 'utf8_general_ci') and 'Kollation' (set to 'utf8_general_ci'). A red circle highlights the 'Datenbankname' dropdown. At the bottom of the page, a table lists existing databases: 'artist_ms' (collation utf8_bin), 'course-selection' (collation latin1_swedish_ci), and 'test'. A red circle highlights the 'Anlegen' (Create) button.

Vor dem Anlegen einer Datenbank sollte als Zeichensatz sollte **utf8_general_ci** gewählt werden.

Das ist wichtig für die Darstellung von Sonderzeichen und Umlauten bzw. auch für die Sortierung.

Die Datenbank *bookstore* anlegen.

This screenshot shows the same 'Datenbanken' (Databases) section as the previous one, but with a new database being created. The 'Datenbankname' field now contains 'bookstore'. The 'Kollation' dropdown still shows 'utf8_general_ci'. A red circle highlights the 'Anlegen' (Create) button. The rest of the interface, including the list of existing databases at the bottom, remains the same.

2.4 Importieren von Tabellen

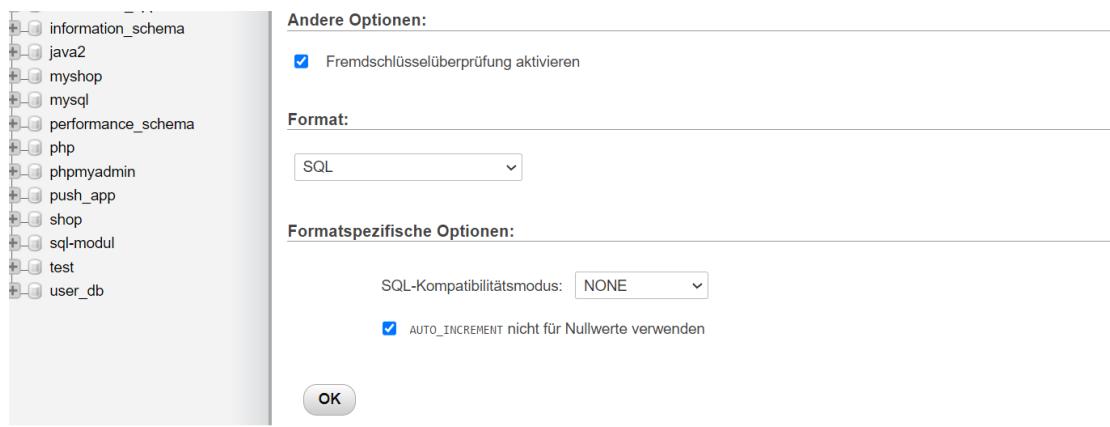
Vor dem Import muss die Datenbank ausgewählt sein.

Danach gehen wir auf Datei auswählen und wählen die Datei bookstore.sql.bz2.

→ Es können komprimierte und unkomprimierte Formate benutzt werden.



Links unten mit Ok bestätigen:



Der Import kann mehrere Minuten dauern! Falls der Import abbricht, sollte vorher die Konfiguration (siehe oben) vorgenommen werden.

2.5 Installation DBeaver

DBeaver ist ein ähnlich wie PHPMyAdmin ein weiterer Client für MySQL und andere relationale Datenbanken. Vor allem das ausführen und verwalten funktioniert mit diesem Programm sehr gut.

Letztendlich ist DBeaver für uns aber nur eine weitere praktische Alternative zu PHPMyAdmin.

Download: <https://dbeaver.io/>

Bitte die Community Edition wählen.

2.6 Werkzeuge und Editoren

2.7 DBEaver

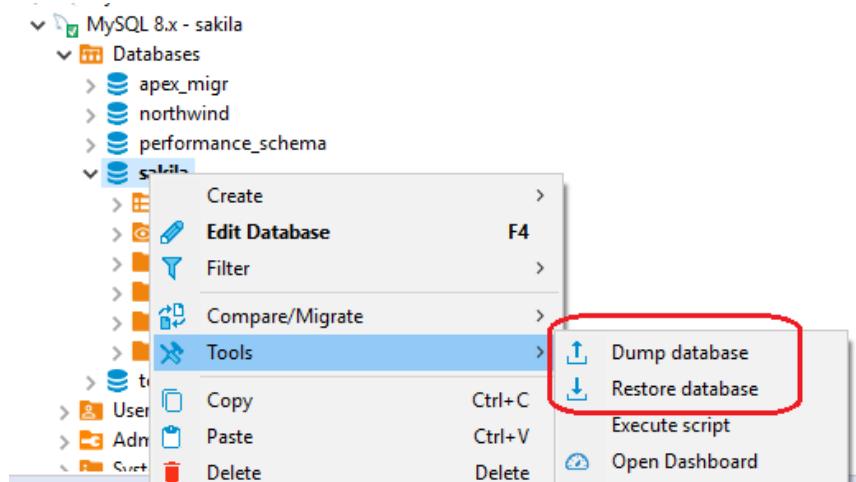
2.7.1 Dokumentation

<https://dbeaver.com/docs/wiki>

2.7.2 Backup Restore

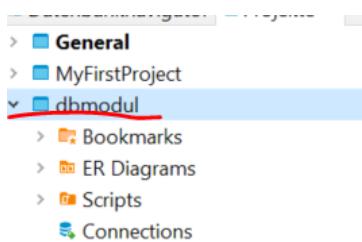
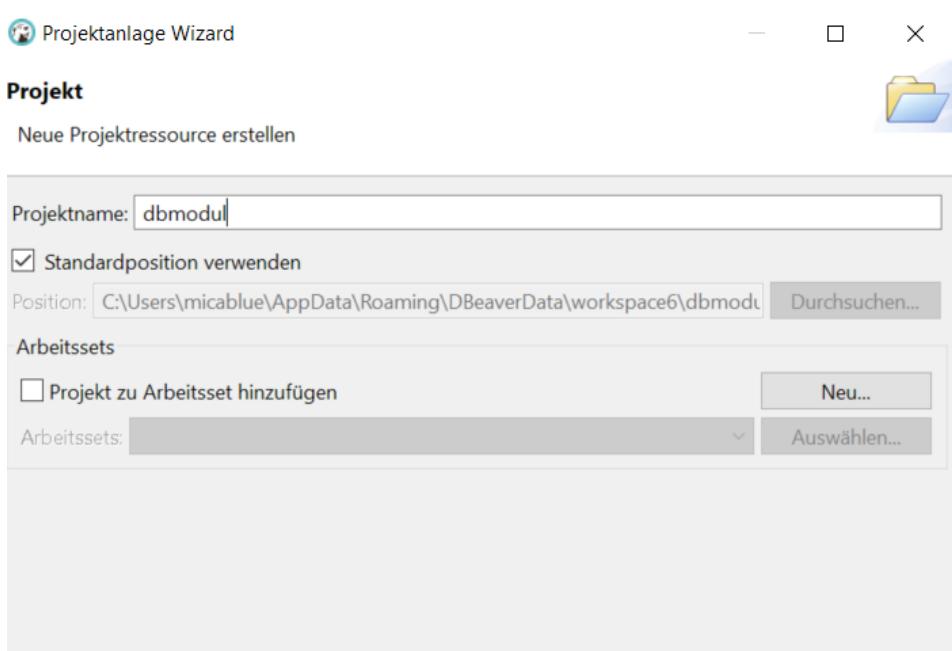
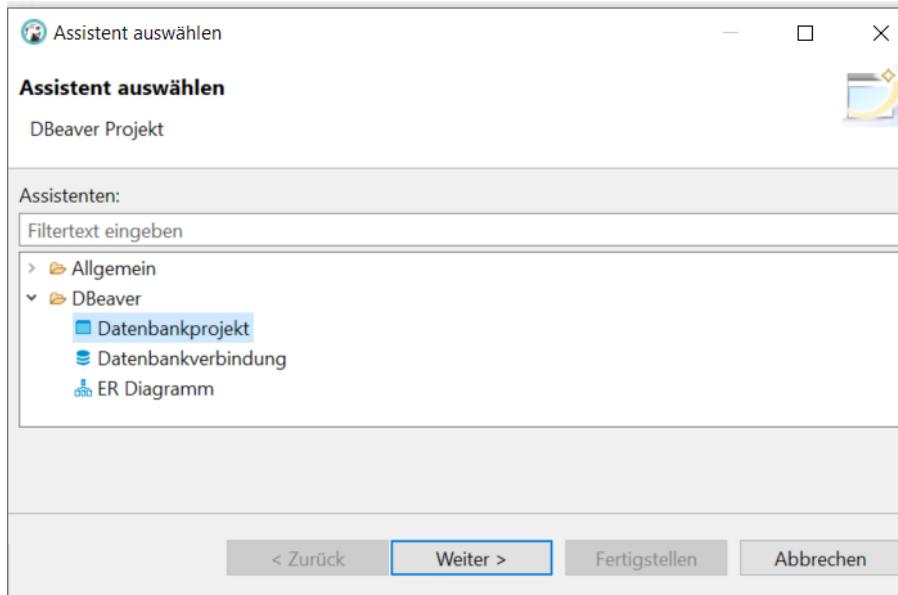
Für Backups werden in der Regel als sql-Datei gespeichert. Speichern im Binärformat ist allerdings auch möglich.

<https://dbeaver.com/docs/wiki/Backup-Restore/>

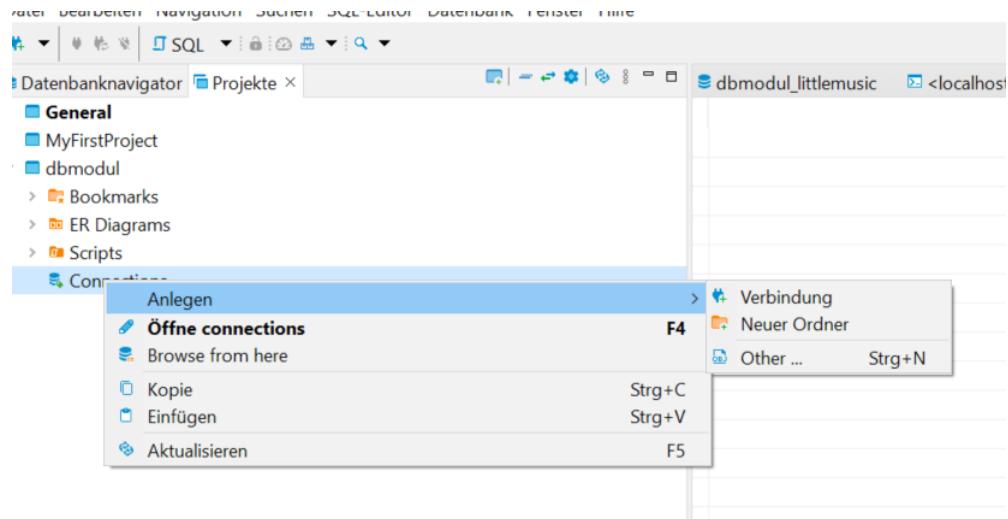


2.7.3 Neues DBEaver Projekt

Menü/ Datei/Neu

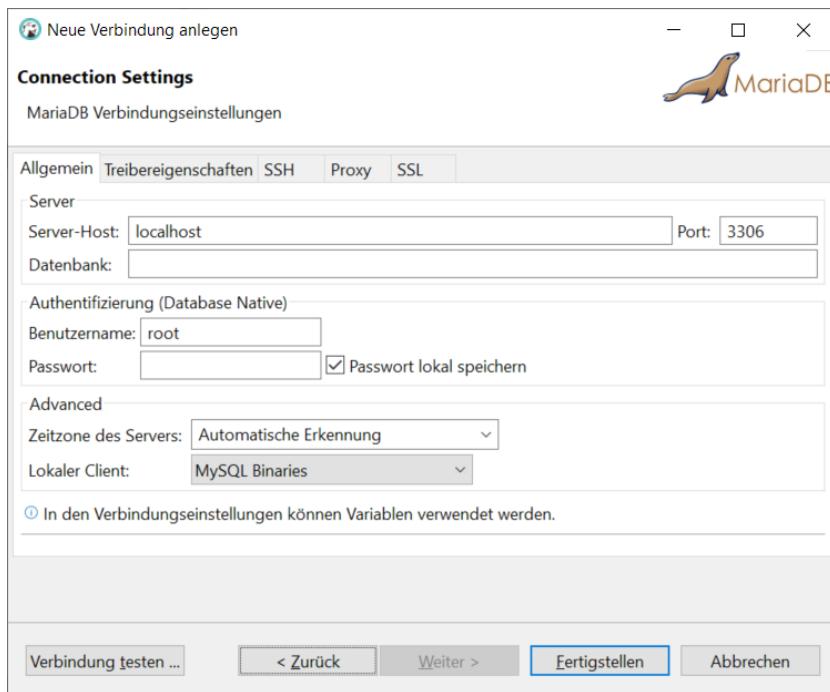


Jetzt kann eine Connection hergestellt werden. Dazu muss die Datenbank vorher gestartet sein.



MariaDB oder MySQL aussuchen

The screenshot shows the 'Neue Verbindung anlegen' (Create new connection) dialog. The title bar says 'Neue Verbindung anlegen'. The main section is titled 'Verbindungstyp auswählen' (Select connection type). It contains the instruction 'Neue Verbindung anlegen. Sollte die Datenbank nicht in der Liste sehen, kann über den Treiber suchen' (Create new connection. If the database is not in the list, it can be searched via the driver). Below this is a search bar labeled 'Geben Sie einen Teil des Namens der zu filternden Datenbank/Treiber' (Enter part of the name of the database/treiber to filter). On the left, a sidebar lists categories: 'All', 'Popular', and 'SQL' (which is selected). Under 'SQL', there are sub-categories: 'NoSQL', 'Analytical', 'Timeseries', 'Embedded', 'Hadoop / BigData', 'Full-text search', and 'Graph databases'. To the right, a grid of connection icons is shown, each with a name below it. The 'MariaDB' icon is highlighted with a blue border. Other icons include SQLite, DB2 LUW, DuckDB, MySQL, Azure Databricks, Azure SQL Server, Babelfish via TDS (beta), Cache, ClickHouse, DB2 z/OS, Denodo 8, Derby Embedded, Derby Server, and Dremio.



3 SQL

→ Siehe Datenbankskript

4 Python und Datenbanken

4.1 Modul installieren

Damit deine Projekte sauber getrennt bleiben, eine virtuelle Umgebung einrichten:

- Am besten in unserem übergeordneten Verzeichnis z.B. **py_workspace**
- Falls notwendig nach: `cd py_workspace`

```
cd PY_WORKSPACE
```

```
python -m venv .venv
```

Windows:

```
.venv\Scripts\Activate
```

macOS/Linux:

```
source .venv/bin/activate
```

Datenbank Treiber Installation

```
pip install mysql-connector-python
```

zusätzlich noch

```
pip install pymysql
```

4.2 Datenbankverbindung aufbauen

```
import mysql.connector

con = mysql.connector.connect(
    user="root",
    password="",
    host="localhost",
    database="testdb"
)
```

```
# Tabelle anlegen
cursor = conn.cursor()
q = ''' CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    email VARCHAR(100)
)'''
```

```
# Datensatz einfügen
insert_query = "INSERT INTO users (name, email) VALUES (%s, %s)"
cursor.execute(insert_query, ("Max", "m@x.com"))
conn.commit()
```

```
# Daten abfragen
cursor.execute("SELECT * FROM users")
for row in cursor.fetchall():
    print(row)
```

4.3 Datenklassen mit @dataclass

@dataclass (python3.7) vereinfachen das schreiben von Model-Klassen:

```
from dataclasses import dataclass
from typing import Optional

@dataclass
class Book:
    id: Optional[int] = None    # beim neuen Buch None (wird von DB vergeben)
    title: str = ""
    author: str = ""
    genre: str = ""
    published_year: int = 0
```

Optional: alle Felder sind Pflicht, sofern sie keinen Defaultwert haben. Durch Optional erreichen wir, dass die *id* entweder *int* oder *None* ist. Das kann für das speichern von Datensätzen sinnvoll sein (auto increment).

4.4 SQLAlchemy

SQLAlchemy ist das populärste Python-Datenbank-Frameworks, um mit relationalen Datenbanken wie SQLite, PostgreSQL, MySQL etc. zu arbeiten.

1. SQLAlchemy ORM (Object Relational Mapper)

- Datenbank-Tabellen als Python-Klassen definieren.
- schreiben, lesen und ändern als Python-Objekte.
- Beziehungen wie ForeignKeys werden automatisch gemanaged.

- Erlaubt das einfache, objektorientierte Arbeiten mit Datenbankinhalten.
- Es ist flexibel: man kann mit wenig Aufwand zwischen verschiedenen Datenbanksystemen wechseln.

4.4.1 Beispiel

Tabelle definieren:

```
from sqlalchemy.orm import declarative_base
from sqlalchemy import Column, Integer, String
```

```
Base = declarative_base()
```

```
class User(Base):
    __tablename__ = "user"
    id = Column(Integer, primary_key=True)
    name = Column(String)
```

Objekt speichern:

```
session.add(User(name="Alice"))
session.commit()
```

Objekt auslesen:

```
alice = session.query(User).filter_by(name="Alice").first()
```

4.5 Pydantic

```
pip install pydantic
```

4.6 Passwortverschlüsselung

```
pip install bcrypt
```

5 Rest /RestFULL

REST steht für **Representational State Transfer** und ist ein Architekturstil für Web-Services. RESTful Web-Services folgen bestimmten Prinzipien, um klar strukturierte und leicht verständliche APIs (Programmierschnittstellen) zu bauen.

Grundidee von REST

- **Ressourcen** (z. B. Benutzer, Artikel, Produkte) werden über **URLs** eindeutig adressiert.
- Jeder URL entspricht eine Ressource oder Sammlung von Ressourcen.
- Über HTTP-Methoden (GET, POST, PUT, DELETE, etc.) wird auf diese Ressourcen zugegriffen oder sie werden verändert.

Prinzip	Erklärung	Beispiel
Ressourcen-Orientierung	APIs repräsentieren Ressourcen (meist als Substantive)	/users, /products/123
HTTP-Methoden als Aktionen	Aktionen sind durch HTTP-Methoden definiert	GET /users → Liste aller Nutzer POST /users → Neuen Nutzer anlegen PUT /users/1 → Nutzer 1 aktualisieren DELETE /users/1 → Nutzer 1 löschen
Zustandslosigkeit	Server speichert keine Session-Infos; jeder Request ist für sich allein verständlich	Alle Authentifizierungsdaten müssen im Request enthalten sein

Prinzip	Erklärung	Beispiel
Repräsentation	Ressourcen werden typischerweise als JSON oder XML übertragen	Client und Server tauschen JSON-Daten aus
Selbstbeschreibende Nachrichten	Requests und Responses enthalten alle Infos (z.B. HTTP-Statuscodes, Header)	404 = nicht gefunden, 200 = OK, 201 = erstellt

5.1.1 Wichtige http Status-Codes

Statuscode	Bedeutung	Beschreibung
1xx – Informativ		
100	Continue	Anfrage wird fortgesetzt
2xx – Erfolg		
200	OK	Anfrage erfolgreich, Antwort enthält die Daten
201	Created	Neue Ressource wurde erfolgreich erstellt
204	No Content	Anfrage erfolgreich, aber keine Daten zurück
3xx – Umleitung		
301	Moved Permanently	Ressource dauerhaft verschoben
302	Found (Temporary Redirect)	Ressource vorübergehend verschoben
4xx – Client-Fehler		
400	Bad Request	Ungültige Anfrage (z.B. fehlende Parameter)
401	Unauthorized	Authentifizierung erforderlich oder fehlgeschlagen
403	Forbidden	Zugriff verweigert
404	Not Found	Ressource nicht gefunden
409	Conflict	Konflikt, z.B. doppelte Daten oder Versionskonflikt
5xx – Server-Fehler		
500	Internal Server Error	Fehler auf Serverseite
503	Service Unavailable	Dienst vorübergehend nicht verfügbar

5.2 RestFull mit FastAPI

Die folgenden Module müssen installiert sein:

- ➔ fastapi
- ➔ uvicorn
- ➔ sqlalchemy
- ➔ pydantic

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Willkommen bei FastAPI!"}

@app.get("/items/{item_id}")
def read_item(item_id: int, q: str = None):
    return {"item_id": item_id, "query": q}

@app.post("/items/")
def create_item(item: dict):
    return {"message": "Item erhalten", "item": item}
```

Starten mit:

```
uvicorn main:app --reload
```

Test im Browser:

- GET <http://127.0.0.1:8000/>
→ { "message": "Willkommen bei FastAPI!" }
 - GET <http://127.0.0.1:8000/items/42?q=test>
→ { "item_id": 42, "query": "test" }
- ➔ Post kann nicht direkt im Browser getestet werden. Unser Server bietet aber ein Swagger-Interface zum Testen an:
<http://127.0.0.1:8000/docs#/>

- /items/ auswählen
- Auf „Try it out“ klicken
- JSON-Body eingeben:

```
json
KopierenBearbeiten
{
    "name": "Apfel",
    "preis": 1.99
}
```

5.3 Schema mit Pydantic

```
from pydantic import BaseModel, EmailStr

class UserCreate(BaseModel):
    name: str
    email: EmailStr

class UserRead(BaseModel):
    id: int
    name: str
    email: EmailStr

    class Config:
        orm_mode = True
```

5.3.1 Eingehender Request (Client → Server)

- Wenn du den Parameter user: UserCreate (ein Pydantic-Modell) in deiner FastAPI-Funktion hast, übernimmt FastAPI intern Folgendes:
 - Es liest den JSON-Body aus der HTTP-Anfrage.
 - Wandelt den JSON-Text automatisch in ein UserCreate-Objekt um (Deserialisierung).
 - Dabei validiert Pydantic die Daten (Typen, Pflichtfelder usw.).

Das passiert **automatisch**, bevor deine Funktion ausgeführt wird.

5.3.2 Ausgehende Response (Server → Client)

```
return {"id": new_user.id, "name": new_user.name, "email": new_user.email}
```

FastAPI wandelt dieses Python-Dict automatisch in JSON um (Serialisierung), bevor es es an den Client zurückschickt.

Das passiert hinter den Kulissen durch FastAPI (bzw. durch Starlette, das FastAPI zugrunde liegt).

6 Anhang

6.1 Unittests

pip install pytest

Wir haben zum Beispiel diese Struktur:

UnitTests/

```
|—— functions.py  
└── tests/  
    └── test_functions.py
```

pytest.ini im Projektverzeichnis:

hier unter UnitTests

[pytest]

testpaths = tests

pythonpath = .

addopts = -v

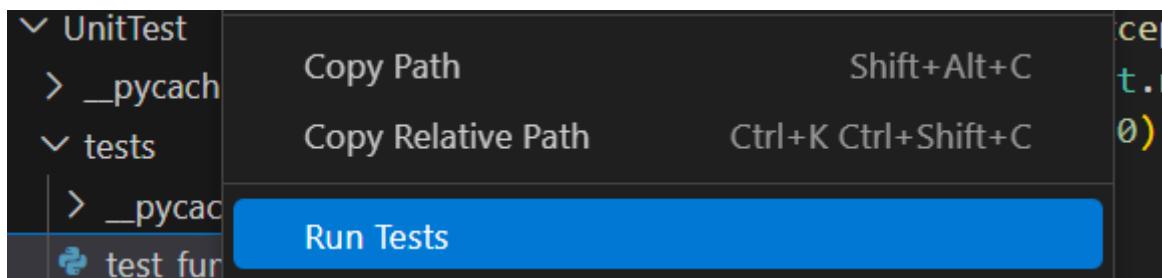
zu testende Funktion: functions.py

```
def sum(a, b):  
    return a + b
```

Test:test_functions.py

```
def test_sum():  
    assert sum(1,2) == 3
```

6.1.1 Test ausführen



6.2 VS Code Module

6.3 Python Module

pip install