Docker-basierte Webentwicklungsumgebung mit Apache, PHP und MariaDB

Dieses Dokument zeigt eine Mitschrift zur Ausarbeitung eines Arbeitsauftrage/Laborauftrages im Unterrichtsfach **Betriebssysteme & Labor** (Hr. Wofgang Ullner)

Inhalt

1. Zielsetzung

Ziel dieser Dokumentation ist der Aufbau und Vergleich zweier LAMP-Umgebungen (Linux, Apache, MySQL, PHP) unter Windows 11.

Dabei wird die Webserver- und Datenbankumgebung einmal containerisiert mit Docker Compose und einmal lokal mit XAMPP realisiert.
Schwerpunkte sind die Einrichtung der Dienste, die Überprüfung der Datenbankintegrität mittels Fremdschlüsseln sowie die Bewertung beider Ansätze hinsichtlich Funktionalität, Portabilität und Benutzerfreundlichkeit.

2. Geräteübersicht

Device (# & Name)		Rolle	Hypervisor (VM)	Betriebssystem (OS)	IP-Adresse (LAN)
1	Lenovo E16	Hostsystem	direkt, keine VM	Windows 11 Pro (64-bit)	192.168.0.150
2	Docker Desktop (WSL2)	Container-Engine *	WSL2	Linux (Docker Subsystem)	intern über Docker-Bridge, **172.23.64.1**

* für Apache/PHP/MySQL/phpMyAdmin

2.1 Netzwerkeinstellungen (Hardware)

	Device #	# 1	# 2	# 3	# 4	# 5
Devices	IP Adr.	192.168.0.150	intern 172.23.64.1			
	DHCP/static	DHCP	automatisch			
	SUB	255.255.255.0	---			
	Gateway	192.168.0.254	Docker Bridge			
	DNS	automatisch	---			
	Mac	70-08-94-62-5E-FB	automatisch			

2.2 User & Zugangsdaten (Software)

	Geräte #	Anwendung	Benutzername	Passwort	Gerätename	domain (AD)
Client	1	Windows 11 Benutzer	markus	Aa_123456	LENOVO-E16	---
	1	XAMPP / phpMyAdmin	user1	Aa_123456	localhost	—

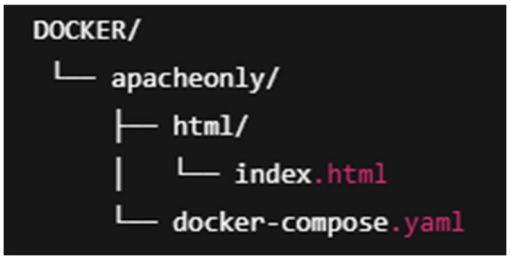
	Geräte #	Anwendung	Benutzername	Passwort	Hostname
Server	2	Apache / PHP (Container)	---	---	apache-php
	2	MySQL Server	*user1*	Aa_123456	mysql-db
	2	MySQL Root	root	rootpass	mysql-db
	2	phpMyAdmin	user1	Aa_123456	phpmyadmin

3. Ausgangssituation

Docker-Dektop sowie Visual-Studio Code war bereits am WIN11 Rechner installiert.

4. Vorbereitung/Schema

- Erstellung eines neuen „Projektordners“ unter **C:\docker.**
- Erstellung von Unterordnern wie z.B: „**apacheonly**“.
- Erstellung eines weiteren Unterordners „**html**“(für index.html bzw. PHP-Dateien)



4.1 Visual Studio Code

Visual Studio Code wird als Editor genutzt um die entsprechenden Dateien in der vorbereiteten Ordnerstruktur zu erstellen bzw. zu bearbeiten.

- Index.html
- Docker-compose.yaml

5. Umsetzung (DOCKER)

In diesem Abschnitt wird die praktische Umsetzung beschrieben. Schrittweise werden verschiedene Varianten einer Webserver-Umgebung aufgebaut, um die Funktionsweise und Zusammenarbeit der einzelnen Komponenten (Apache, PHP, MySQL, phpMyAdmin) zu verstehen.

Ziel ist es, ausgehend von einem einfachen Webserver bis hin zu einer vollständigen LAMP-Stack-Umgebung (Linux, Apache, MySQL, PHP) alle Funktionen nachvollziehbar aufzubauen und zu testen.

5.1 Variante A – Apache only

In dieser Variante wird ein einfacher Apache-Webserver in einem Docker-Container gestartet.

Er liefert statische HTML-Dateien aus (z. B. eine einfache „Hallo Welt“-Seite).

Beschreibung	Ergebnis
Erstelle eine Datei **docker-compose.yml** mit folgendem Grundgerüst im **Ordner apacheonly**	apacheonly > docker-compose.yaml 1 version: "3.8" 2 ▷Run All Services 3 services: 4 ▷ Run Service 5 apache: 6 image: httpd:2.4 7 container_name: apache-only 8 ports: 9 - "8080:80" 10 volumes: 11 - ./html:/usr/local/apache2/htdocs/

Lege im **Ordner html** eine Datei **index.html** an

```
apacheonly > html > <> index.html > ...
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h1>Hallo Docker-Welt! Markus ist mein Name!</h1>
10  </body>
11  </html>
```

Um den Dienst zu starten muss die Docker Desktop App am host-system laufen!

Um den Container zu starten, öffnen wir **CMD,** wechseln in das Verzeichnis wo die **.yaml** liegt und starten den Container

```
cd C:\docker\apacheonly

docker-compose up -d
```

```
C:\docker\apacheonly>docker-compose up -d
time="2025-10-07T18:23:44+02:00" level=warning msg="C:\\docker\\apacheonly\\docker-compose.yaml: the attribute `version`
 is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 2/2
 ✔Network apacheonly_default  Created                                    0.2s
 ✔Container apache-only       Started                                    0.7s

C:\docker\apacheonly>
```

Rufe im Browser auf → http://localhost:8080

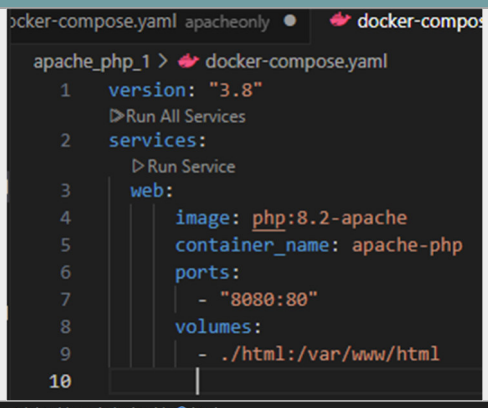
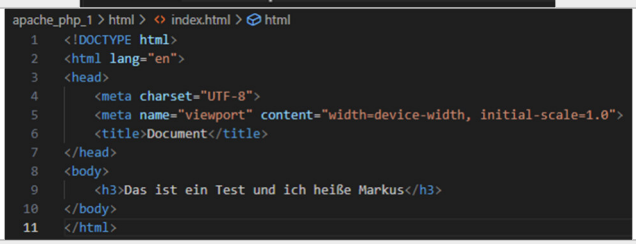
Zum Stoppen des Containers**: docker-compose down**

```
C:\docker\apacheonly>docker-compose down
time="2025-10-07T18:29:07+02:00" level=warning msg="C:\\docker\\apacheonly\\docker-compose.yaml: the attribute `version`
 is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 2/2
 ✔Container apache-only       Removed                                    1.4s
 ✔Network apacheonly_default  Removed                                    0.3s
```

5.2 Variante B.1 – Apache mit PHP

PHP ist „nur“ eine serverseitige Programmiersprache!

- erzeugt HTML, das dann an den Browser geschickt wird
- kann Daten lesen, verarbeiten und speichern
- kann mit Datenbanken kommunizieren (z. B. MySQL)
- kann Dateien lesen/schreiben, E-Mails senden, APIs aufrufen, Benutzer authentifizieren usw.

Beschreibung	Ergebnis
Erstelle eine Datei **docker-compose.yml** mit folgendem Grundgerüst unter C:\docker\apache_php_1	
Lege im Unter-Ordner **html** eine Datei **index.html** an	
Um den Container zu starten, öffnen wir **CMD,** wechseln in das Verzeichnis wo die **.yaml** liegt und starten den Container	cd C:\docker\apache_php_1 docker-compose up -d

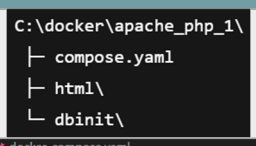
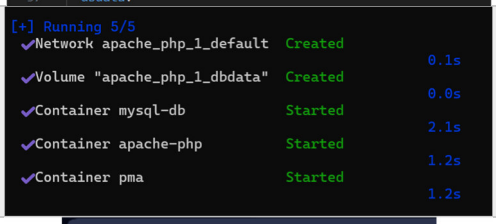
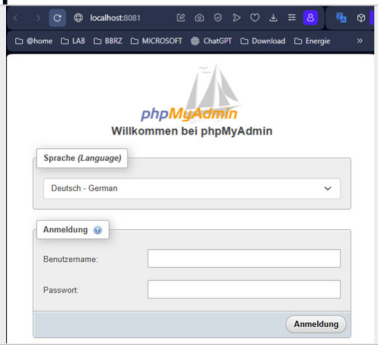
```
C:\docker\apache_php_1>docker-compose up -d
time="2025-10-10T11:32:32+02:00" level=warning msg="C:\\docker\\apache_php_1\\docker-compose.yaml: the attribute `versio
n` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 15/15
 ✔web Pulled                                                                                                    24.4s
   ✔349592d2c6d1 Pull complete                                                                                  20.2s
   ✔4f4fb700ef54 Pull complete                                                                                   0.1s
   ✔08a4a8d8574b Pull complete                                                                                   0.5s
   ✔634ab520a54a Pull complete                                                                                  20.6s
   ✔4a03afdd8816 Pull complete                                                                                   0.8s
   ✔2e39efed0f04 Pull complete                                                                                   1.0s
   ✔e9d7b3818d3e Pull complete                                                                                   0.8s
   ✔ec1283305d46 Pull complete                                                                                  21.1s
   ✔b29d636b5d63 Pull complete                                                                                   0.6s
   ✔cdb5bbef0e17 Pull complete                                                                                   0.8s
   ✔042f7bbd46e8 Pull complete                                                                                  20.5s
   ✔4ad1ce053292 Pull complete                                                                                   1.8s
   ✔3f814cc06e5a Pull complete                                                                                   0.7s
   ✔50e2b6face72 Pull complete                                                                                   1.1s
[+] Running 2/2
 ✔Network apache_php_1_default  Created                                                                          0.1s
 ✔Container apache-php          Started                                                                          1.2s
```

Rufe im Browser auf: http://localhost:8080	

Zum Stoppen des Containers**: docker-compose down**

```
C:\docker\apache_php_1>docker-compose down
time="2025-10-10T13:00:18+02:00" level=warning msg="C:\\docker\\apache_php_1\\docker-com
pose.yaml: the attribute `version` is obsolete, it will be ignored, please remove it to
avoid potential confusion"
[+] Running 2/2
 ✔Container apache-php          Removed
                                1.4s
 ✔Network apache_php_1_default  Removed
                                0.4s
```

5.3 Variante B.2 – Compose um MySQL + phpMyAdmin erweitern

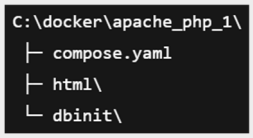
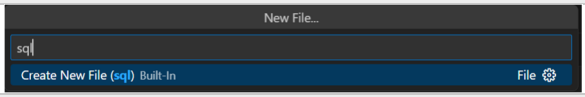
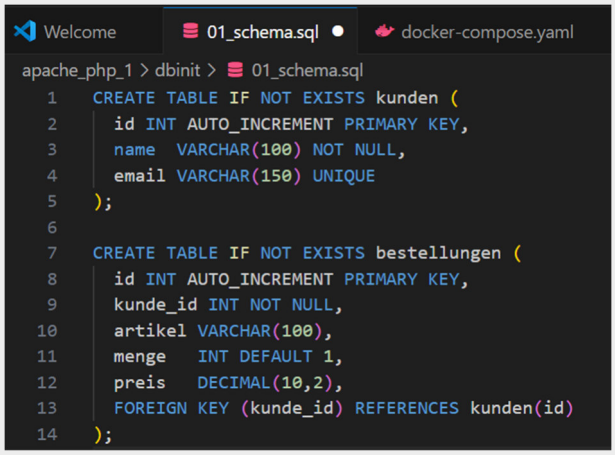
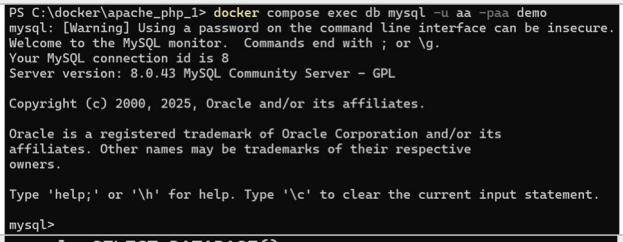
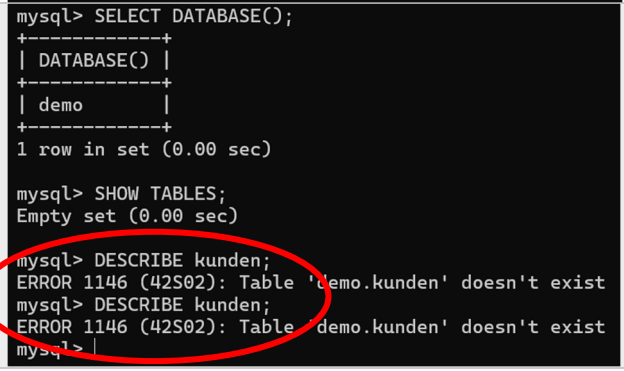
Beschreibung	Ergebnis
neuen Ordner mit dem Namen **dbinit** unter **C:\docker\apache_php_1** erstellen.	
Öffne deine **compose.yaml** und ersetze den Inhalt durch diesen bzw. ergänzen den vorhandenen Eintrag →	
CMD **C:\docker\apache_php_1>** **docker-compose up -d**	
http://localhost:8080/dbtest.php	
http://localhost:8081	

5.3.1 Datenbankeinrichtung / Initialisierung

Um die Datenbank automatisch beim Start des Containers zu erzeugen, wird ein Init-Verzeichnis eingebunden.

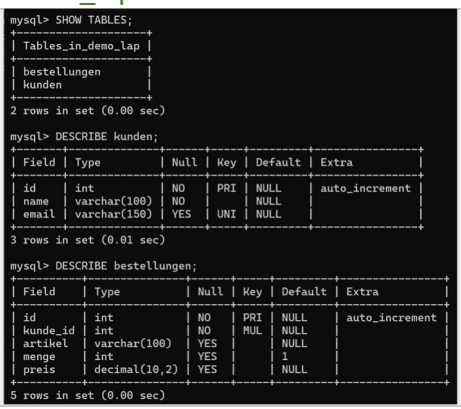
Der MySQL-Container führt alle .sql-Dateien aus, die sich in dbinit/ befinden.

Dadurch wird die Grundstruktur (Tabellen kunden und bestellungen) direkt beim Start erstellt, ohne manuelle Eingaben über phpMyAdmin.

Beschreibung	Ergebnis
in diesen vorher angelegten Ordner „dbinit“ kommt ein neues skript mit dem Namen: „01_schema.sql“	
in Visual Studio einen neuen sql-biult erstellen	
Automatische Initialisierung Tabellen werden beim ersten Start automatisch angelegt **Datenstruktur** Kunden & Bestellungen →(1:n-Beziehung) **Integrität** Fremdschlüssel stellt sicher, dass keine Bestellung ohne gültigen Kunden existiert **Wiederholbar** Dank IF NOT EXISTS kannst du das Skript mehrfach verwenden, ohne Fehler	
docker starten: **docker compose exec db mysql -u aa -paa demo**	
Prüfen durch folgende Eingaben: SELECT DATABASE(); SHOW TABLES; DESCRIBE kunden; DESCRIBE bestellungen;	

MySQL führt die Dateien aus ./dbinit/ nur beim allerersten Start aus, wenn die Datenbank noch leer ist.

Wenn du dbinit später hinzugefügt hast (nachdem der Container schon mal lief), dann ist die Datenbank bereits initialisiert - und MySQL überspringt die Init-Skripte.

Datenbank-Volume löschen (damit Init neu läuft) löscht die Datei →	cd C:\docker\apache_php_1 docker compose down docker volume ls docker volume rm apache_php_1_dbdata
Container erneut starten	docker compose up -d
Prüfen, ob das Init-Script erkannt wurde	docker compose logs db --no-color
In mysql mit **Benutzer**/**Datenbanknamen** einloggen	docker compose exec db mysql -u user1 -p demo_lap
und Tabelle prüfen **SHOW TABLES;** ➔ **OK!** **DESCRIBE kunden;** ➔ **OK!** **DESCRIBE bestellungen;** ➔ **OK**	
Login in myphp: user1 | Aa_123456	http://localhost:8081
Hinweis: SSL-Verbindung In der phpMyAdmin-Statusübersicht wird angezeigt: „Server-Verbindung: SSL wird nicht verwendet“. Diese Meldung weist lediglich darauf hin, dass phpMyAdmin ohne SSL-Verschlüsselung mit dem MySQL-Server kommuniziert. Da beide Container innerhalb des isolierten Docker-Netzwerks laufen, besteht kein Sicherheitsrisiko. In produktiven Umgebungen würde die Verbindung zwischen Webserver und Datenbankserver verschlüsselt erfolgen, z. B. über TLS-Zertifikate und die Parameter --ssl-cert / --ssl-key.	
alle laufenden Prozesse stoppen	docker compose down

Erweiterung/Ergänzung der Datei „01_schema.sql“ für Testzwecke…

Was die Einträge machen:

Abschnitt	Beschreibung
SET NAMES / CHARACTER SET	sorgt für korrekte Zeichencodierung (Umlaute etc.)
CREATE DATABASE demo	legt automatisch die Datenbank an, falls sie fehlt
DROP TABLE IF EXISTS	| löscht alte Tabellen bei erneutem Start
CREATE TABLE …	legt Tabellen `kunden` & `bestellungen` mit Foreign Key an
INSERT INTO …	fügt 3 Kunden und 3 Bestellungen als Beispiel ein

apache_php_1 > dbinit > 01_schema.sql

```sql
SET NAMES utf8mb4;
SET CHARACTER SET utf8mb4;

CREATE DATABASE IF NOT EXISTS demo_lap CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
USE demo_lap;

DROP TABLE IF EXISTS bestellungen;
DROP TABLE IF EXISTS kunden;

CREATE TABLE kunden (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name  VARCHAR(100) NOT NULL,
  email VARCHAR(150) UNIQUE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE bestellungen (
  id INT AUTO_INCREMENT PRIMARY KEY,
  kunde_id INT NOT NULL,
  artikel VARCHAR(100),
  menge   INT DEFAULT 1,
  preis   DECIMAL(10,2),
  FOREIGN KEY (kunde_id) REFERENCES kunden(id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- optional: Demo-Daten
INSERT INTO kunden (name,email) VALUES
('Astrid Mayr','astrid@mayr.at'),
('Astrid Mayer','astrid@mayer.at'),
('Astrid Maier','astrid@maier.at');
```

erneutes beenden, löschen und ausführen des Containers/Datei…

docker compose down
docker volume rm apache_php_1_dbdata
docker compose up -d

Login in die Datenbank per Powershell

docker compose exec db mysql -u user1 -p demo_lap

Prüfen in mysql mit:

SHOW TABLES;

SELECT * FROM kunden;

Kunden in der Datenbank werden angezeigt

```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_demo_lap |
+--------------------+
| bestellungen       |
| kunden             |
+--------------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM kunden;
+----+-----------------+--------------------+
| id | name            | email              |
+----+-----------------+--------------------+
|  1 | Josef Buberl    | josef@buberl.at    |
|  2 | Maria Berger    | maria@berger.at    |
|  3 | Markus Lugstein | markus@lugstein.at |
+----+-----------------+--------------------+
3 rows in set (0.00 sec)
```

5.4 SQL-Befehle ausführen

Es gibt mehrere Wege die Datenbank zu befüllen:

5.4.1 Variante 1: Daten direkt in der MySQL-Konsole

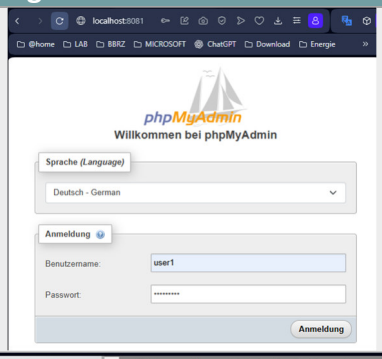
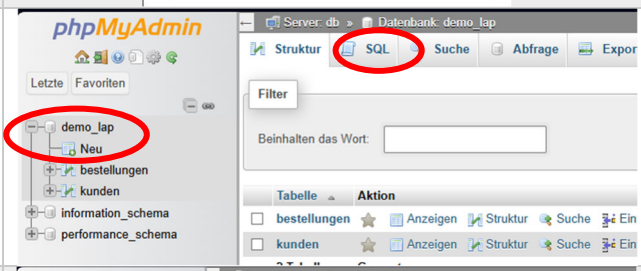
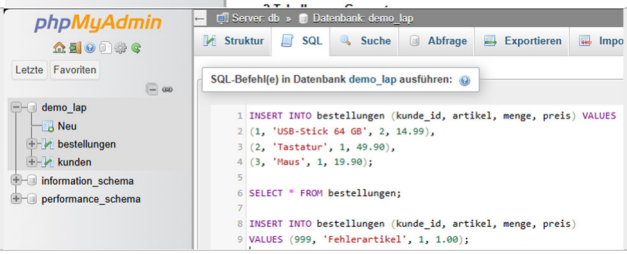
Erklärung	Syntax/Ergebnis
Datenbank starten mit user1 und PW-Abfrage	docker compose exec db mysql -u user1 -p demo_lap
Auswahl der Datenbank „demo_lap“	USE demo_lap;

Die Eingaben können direkt im Terminal der Datenbank so eingefügt werden:
`-- für bestehende Kunden (IDs 1–3)`
`INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES`
`(1, 'USB-Stick 64 GB', 2, 14.99),`
`(2, 'Tastatur', 1, 49.90),`
`(3, 'Maus', 1, 19.90);`
	INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES
	(1, 'USB-Stick 64 GB', 2, 14.99),
	(2, 'Tastatur', 1, 49.90),
	(3, 'Maus', 1, 19.90);

Prüfen ob die Einträge gemacht wurden	SELECT * FROM bestellungen;

Ausgabe im Termin:	mysql> SELECT * FROM bestellungen; +----+-----------+-----------------+-------+-------+ | id | kunde_id | artikel | menge | preis | +----+-----------+-----------------+-------+-------+ | 1 | 1 | USB-Stick 64 GB | 2 | 14.99 | | 2 | 2 | Tastatur | 1 | 49.90 | | 3 | 3 | Maus | 1 | 19.90 | +----+-----------+-----------------+-------+-------+ 3 rows in set (0.00 sec)
Fehlerfall, muss scheitern→ Kunde 999 existiert nicht mysql> INSERT INTO bestellungen (kunde_id, artikel, menge, preis) -> VALUES (999, 'Fehlerartikel', 1, 1.00); ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`demo_lap`.`bestellungen`, CONSTRAINT `bestellungen_ibfk_1` FOREIGN KEY (`kunde_id`) REFERENCES `kunden` (`id`) ON DELETE CASCADE ON UPDATE CASCADE)	INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES (999, 'Fehlerartikel', 1, 1.00);

5.4.2 Variante 2: Über phpMyAdmin

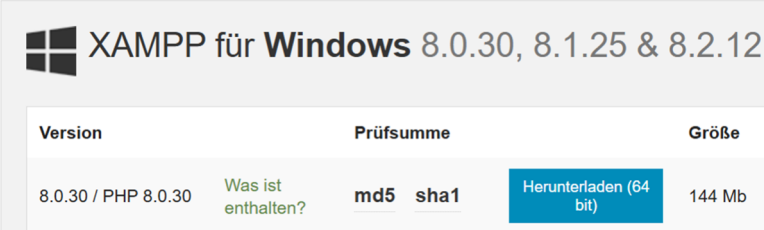
Erklärung	Synytax/Ergebnis
Aufruf im Browser: http://localhost:8081 **Login:** **user1 | Aa_123456**	
Wähle links die Datenbank **demo_lap** Klicke oben auf **SQL**	
Füge dort den gesamten SQL-Block ein INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES (1, 'USB-Stick 64 GB', 2, 14.99), (2, 'Tastatur', 1, 49.90), (3, 'Maus', 1, 19.90); SELECT * FROM bestellungen; INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES (999, 'Fehlerartikel', 1, 1.00); **Klicke auf OK oder Ausführen →**	
Fehlerfall, muss scheitern→ Kunde 999 existiert nicht 	INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES (999, 'Fehlerartikel', 1, 1.00);
Zur Überprüfung der referenziellen Integrität wurden gültige und ungültige Bestellungen eingefügt. Bestellungen mit bestehenden kunde_id-Werten wurden erfolgreich angenommen, während ein Eintrag mit kunde_id=999 vom Fremdschlüsselmechanismus abgewiesen wurde. Damit wurde die Datenkonsistenz bestätigt.	

6. XAMPP (for Windows)

XAMPP ist ein Komplettpaket, das eine lokale Entwicklungsumgebung mitbringt, nämlich

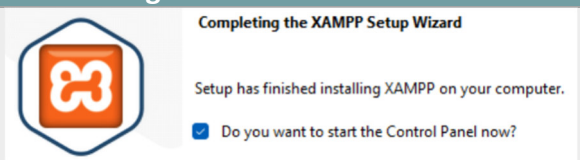
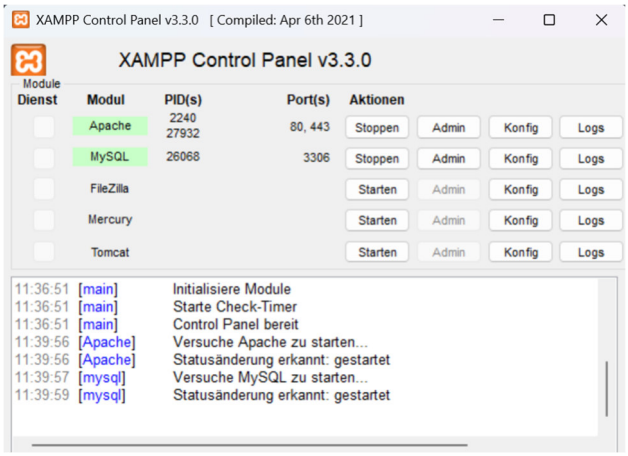
- X = Cross-platform (läuft unter Windows, macOS, Linux)
- A = Apache (Webserver)
- M = MySQL/MariaDB (Datenbank)
- P = PHP
- P = Perl

Quelle: https://www.apachefriends.org/de/download.html

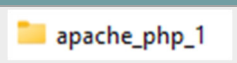
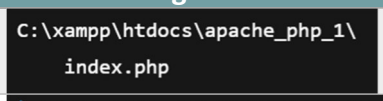
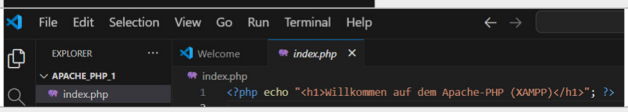


Man startet XAMPP lokal auf dem PC, und bekommt sofort einen funktionierenden Webserver mit PHP und eine MySQL-Datenbank — ohne viel manuell zu installieren.

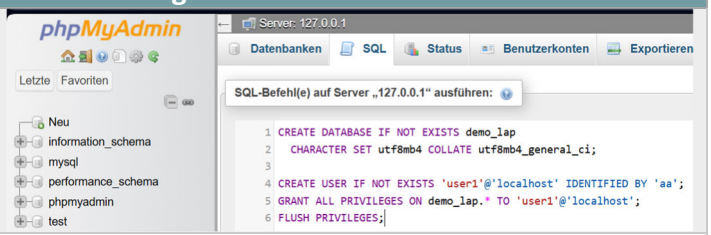
6.1 Installation

Erklärung	Beschreibung
Installation der Software nach dem download **Standardinstallation** Ablageort, Sprache, wählen wenn erwünscht!	
Control Panel von XAMPP ➔ **Apache & MySQL jeweils starten!** **Test:** Apache: http://localhost/ ➔ zeigt XAMPP-Startseite. phpMyAdmin: http://localhost/phpmyadmin ➔ zeigt phpMyAdmin-Startseite.	

6.2 Webroot & Projektordner

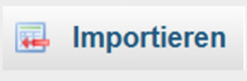
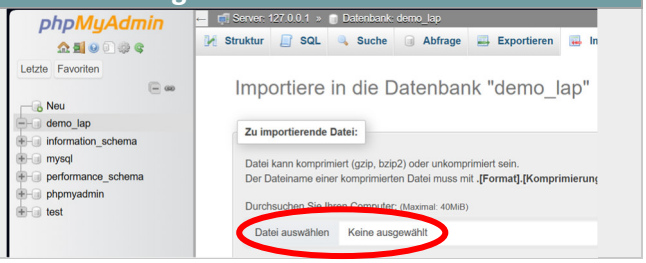
Erklärung	Beschreibung
Projektordner anlegen: 	
in diesen Ordner mit Hilfe VS eine index.php erstellen mit dem Inhalt	
Aufruf per http://localhost/apache_php_1/ oder http://localhost:80/apache_php_1/	

6.3 In phpMyAdmin den Benutzer & Passwort korrekt anlegen

Erklärung	Beschreibung
http://localhost/phpmyadmin öffnen **SQL Tab öffnen und eingeben…. Und OK!** **CREATE DATABASE IF NOT EXISTS demo_lap** **CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;** **CREATE USER IF NOT EXISTS 'user1'@'localhost' IDENTIFIED BY 'Aa_123456';** **GRANT ALL PRIVILEGES ON demo_lap.* TO 'user1'@'localhost';** **FLUSH PRIVILEGES;**	
Unten auf **OK** klicken.	Rechts oben sollte „Abfrage erfolgreich ausgeführt“ erscheinen. Links in der Liste sollte die Datenbank demo_lap sichtbar sein.

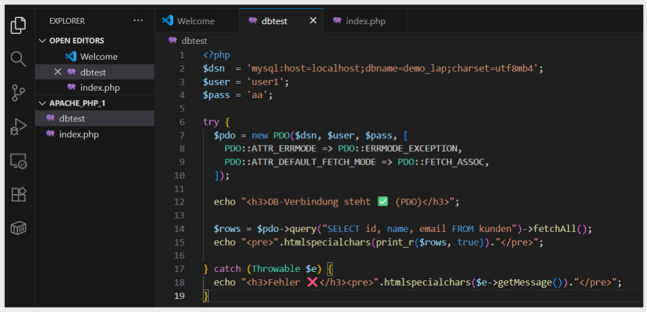
6.4 Schema + Demo-Daten importieren

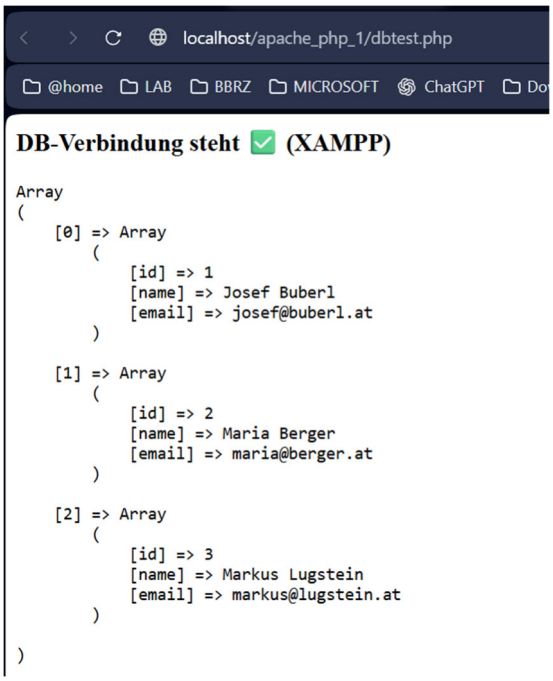
Diese 01_schema.sql Datei ist bereits vorhanden, sie wird aus dem Ordner **C:\docker\apache_php_1\dbinit** importiert.

Erklärung	Beschreibung
In phpMyAdmin links demo_lap wählen → Importieren. 	
Die Datei **01_schema.sql** suchen und hochladen	

6.5 PHP-DB-Verbindung testen

Erklärung	Beschreibung

Erstellung einer neuen php-Datei mit Hilfe von VS unter:
C:\xampp\htdocs\apache_php_1\dbtest.php
`<?php`
`$dsn = 'mysql:host=localhost;dbname=demo_lap;charset=utf8mb4';`
`$user = 'user1';`
`$pass = 'Aa_123456'; // <-- Dein aktuelles Passwort`
 	
`try {`
`$pdo = new PDO($dsn, $user, $pass, [`
`PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,`
`PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,`
`]);`
 	
`echo "<h3>DB-Verbindung steht ☑ (XAMPP)</h3>";`
 	
`$rows = $pdo->query("SELECT id, name, email FROM kunden")->fetchAll();`
`echo "<pre>" . htmlspecialchars(print_r($rows, true)) . "</pre>";`
 	
`} catch (Throwable $e) {`
`echo "<h3>Fehler ❌</h3><pre>" . htmlspecialchars($e->getMessage()) . "</pre>";`
`}`
`?>`
`$rows = $pdo->query("SELECT id, name, email FROM kunden")->fetchAll();`
`echo "<pre>".htmlspecialchars(print_r($rows, true))."</pre>";`
 	
`} catch (Throwable $e) {`
`echo "<h3>Fehler ❌</h3><pre>".htmlspecialchars($e->getMessage())."</pre>";`
`}`
	
	
	Datei **speichern** (Strg+S).

Verbindung testen – Browser öffnen **http://localhost/apache_php_1/dbtest.php**	

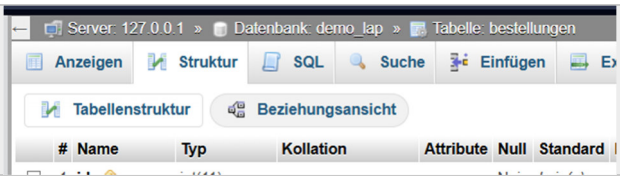
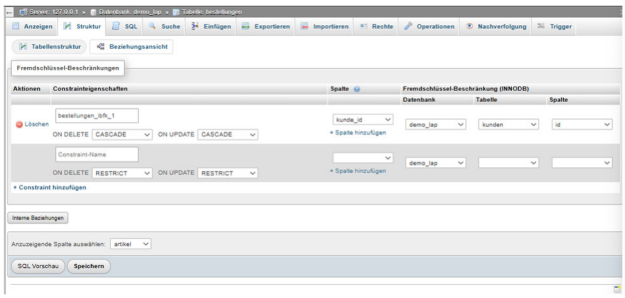
6.6 Fremdschlüssel prüfen (Integrität + CASCADE)

Nach dem erfolgreichen Verbindungstest wird nun die Beziehung zwischen den Tabellen „kunden“ und „bestellungen“ überprüft.

Dabei soll sichergestellt werden, dass:

- eine Bestellung nur für existierende Kunden angelegt werden kann
- und beim Löschen eines Kunden automatisch alle zugehörigen Bestellungen gelöscht werden (ON DELETE CASCADE).

Da der Fremdschlüssel nach dem Import nicht aktiv war, wurde er manuell in phpMyAdmin ergänzt.

Erklärung	Beschreibung
öffnen von MyPhpAdmin→ **http://localhost/phpmyadmin/index.php**	
Tabelle **bestellungen** öffnen → Reiter **Struktur**	
Ansicht wechsel → **Beziehungsansicht**	
Unten auf **„Beziehung anlegen“** klicken Jede Zeile in bestellungen verweist über kunde_id auf kunden.id Wenn du einen Kunden löschst → werden alle Bestellungen dieses Kunden automatisch gelöscht (ON DELETE CASCADE) Wenn du eine Kunden-ID änderst → wird sie auch in den Bestellungen aktualisiert (ON UPDATE CASCADE)	

Zum Test der referenziellen Integrität wurden mehrere Bestellungen eingefügt.

Gültige Kunden-IDs wurden akzeptiert, eine Bestellung mit einer ungültigen Kunden-ID (999)

führte zur Fehlermeldung „Cannot add or update a child row: a foreign key constraint fails“.

Beim Löschen eines Kunden wurden dank ON DELETE CASCADE alle zugehörigen Bestellungen automatisch entfernt.

Damit wurde die Fremdschlüsselbeziehung erfolgreich nachgewiesen.

6.7 Fremdschlüssel & CASCADE-Verhalten

zeigen, dass Bestellungen nur zu gültigen Kunden gehören dürfen,

und dass mit ON DELETE CASCADE abhängige Bestellungen automatisch gelöscht werden.

Erklärung	Beschreibung
phpMyAdmin im Browser öffnen → **http://localhost/phpmyadmin/index.php** **unter SQL eintragen und ausführen.**	-- gültige Bestellungen INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES (1, 'USB-Stick 64 GB', 2, 14.99), (2, 'Tastatur', 1, 49.90), (3, 'Maus', 1, 19.90); -- ungültige Bestellung (soll FEHLSCHLAGEN) INSERT INTO bestellungen (kunde_id, artikel, menge, preis) VALUES (999, 'Fehlerartikel', 1, 1.00); -- prüfen SELECT * FROM bestellungen; ➔ Mit Ok bestätigen

7. Vorgabedokumente

APACHE ONLY

```yaml
version: "3.8"

services:
  apache:
    image: httpd:2.4
    container_name: apache-only
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/local/apache2/htdocs/
```

C:/

!!!
WICHTIG
visual Studio code

localhost:8080

Start mit CMD

Strg N → neue
Strg S → speichern
! in erste Zeile für html

APACHE mit php

```yaml
version: "3.8"

services:
  web:
    image: php:8.2-apache
    container_name: apache-only
    ports:
      - "8080:80"
    volumes:
      - ./html:/var/www/html

```

Index.php im html-Verzeichnis:

Apache/mysql-Connection mit php

apache_startseite_2 > docker-compose.yml > ...

```yaml
version: "3.8"
services:
  web:
    image: php:8.2-apache
    container_name: apache-php
    volumes:
      - ./html:/var/www/html
    ports:
      - "8080:80"
    depends_on:
      - db

  db:
    image: mariadb:10.6
    container_name: mariadb
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: schule
      MYSQL_USER: dbuser
      MYSQL_PASSWORD: Passwort123
    volumes:
      - ./db_data:/var/lib/mysql

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin
    environment:
      PMA_HOST: db
      PMA_USER: dbuser
      PMA_PASSWORD: Passwort123
    ports:
      - "8081:80"
    depends_on:
      - db

volumes:
  db_data:
```

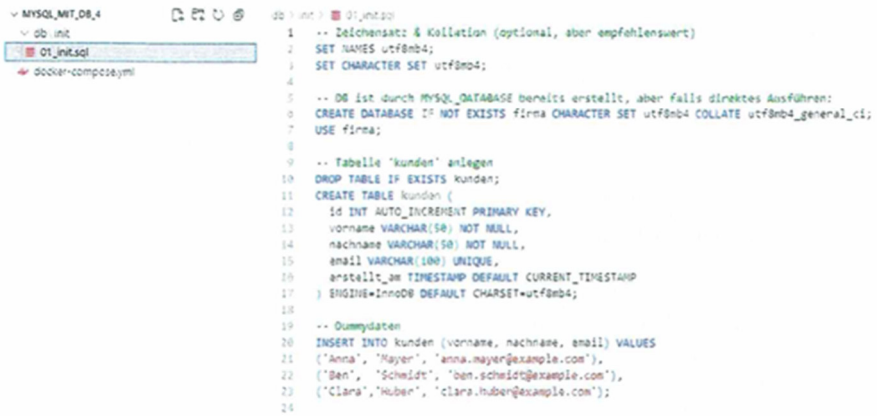

DB_CONNECTION_
html
index.php
docker-compose.yml
html > index.php
<?php
$servername = "db";
$username = "dbuser";
$password = "Passwort123";
$dbname = "schule";
// Verbindung testen
$conn = new mysqli($servername, $username, $password, $dbname);
if ($conn->connect_error) {
die("Verbindung fehlgeschlagen: " . $conn->connect_error);
}
echo "<h1>Willkommen auf dem Apache-PHP-Docker-Container</h1>";
echo " Datenbankverbindung erfolgreich! ";
$conn->close();
?>

```yaml
version: "3.8"

▷Run All Services
services:
  ▷Run Service
  db:
    image: mariadb:10.6
    container_name: mariadb-db
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: firma
      MYSQL_USER: appuser
      MYSQL_PASSWORD: Passwort123
      MARIADB_AUTO_UPGRADE: "1"
    ports:
      - "3306:3306"
    volumes:
      - db_data:/var/lib/mysql
      - ./db/init:/docker-entrypoint-initdb.d:ro
    healthcheck:
      test:
        [
          "CMD",
          "mysqladmin",
          "ping",
          "-h",
          "localhost",
          "-p${MYSQL_ROOT_PASSWORD}",
        ]
      interval: 10s
      timeout: 5s
      retries: 10

  ▷Run Service
  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: phpmyadmin-db
    environment:
      PMA_HOST: db
      PMA_USER: appuser
      PMA_PASSWORD: Passwort123
    ports:
      - "8081:80"
    depends_on:
      db:
        condition: service_healthy

volumes:
  db_data:
```


MYSQL_MIT_DB_4
db_init
01_init.sql
docker-compose.yml
db > init > 01_init.sql
-- Zeichensatz & Kollation (optional, aber empfehlenswert)
SET NAMES utf8mb4;
SET CHARACTER SET utf8mb4;
-- DB ist durch MYSQL_DATABASE bereits erstellt, aber falls direktes Ausführen:
CREATE DATABASE IF NOT EXISTS firma CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
USE firma;
-- Tabelle 'kunden' anlegen
DROP TABLE IF EXISTS kunden;
CREATE TABLE kunden (
id INT AUTO_INCREMENT PRIMARY KEY,
vorname VARCHAR(50) NOT NULL,
nachname VARCHAR(50) NOT NULL,
email VARCHAR(100) UNIQUE,
erstellt_am TIMESTAMP DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- Dummydaten
INSERT INTO kunden (vorname, nachname, email) VALUES
('Anna', 'Mayer', 'anna.mayer@example.com'),
('Ben', 'Schmidt', 'ben.schmidt@example.com'),
('Clara','Huber', 'clara.huber@example.com');