REST-Web-Service

PROGRAMMIERUNG 3
Prüfungshausarbeit

27606 | © Lucas Kahl | WiSe 2020/2021
Prüfer: Prof. Singer
Abgabedatum: 28.02.2021

Inhalt

Abbildungsverzeichnis

» so habe ich gearbeitet…

1. Überlegungen zu Entwurfsmustern

Für die Realisierung der Datenbank Klasse habe ich mich für das **Singleton Entwurfsmuster** entschieden. Dabei ist der Konstruktor der Datenbank-Klasse private , um sicherzustellen, dass der Client keine Möglichkeit hat die Datenbank zu instanziieren. Allein innerhalb der Datenbank ist es möglich eine Instanz zu erzeugen. Innerhalb des Konstruktors wird eine Connection zur Datenbank angelegt. Es darf nur eine Verbindung geben!

Die Singleton Klasse „Datenbank" stellt eine globale Methode zur Verfügung, welche immer die einzigartige Singletoninstanz zurückgibt.
Über die definierte Methode:
public static Datenbank getInstance(){ return DATENBANK; }
wird die einzigartige Instanz an den Client ausgeliefert.
Des weiteren sind Klassen, welche Daten behandeln mit dem **Iterator-Muster** verbunden. Sie müssen iterierbar sein, um gezielt auf die Daten zuzugreifen. Beispielsweise die Datenbank-Methode getData() gibt eine HashMap zurück, welche alle Daten iterierbar beinhaltet. Getter-Methoden ermöglichen gezielten Zugriff auf Attribute.

2. Vorgehen in der Entwicklung

Der Arbeitsprozess begann mit dem Aufsetzen der Serververbindung und der serverseitigen Module. Dabei orientierte ich mich an den in den Fächern Programmierung 3 und Softwaretechnik kennengelernten Methoden.

Zum Aufsetzen der Serververbindung war die Integration einiger Bibliotheken in den entsprechenden Modulen notwendig. Dabei war darauf zu achten, dass alle Maven Bibliotheken die gleiche Versionsnummer tragen. In meinem Fall mussten alle mit 2.X.X beginnen.

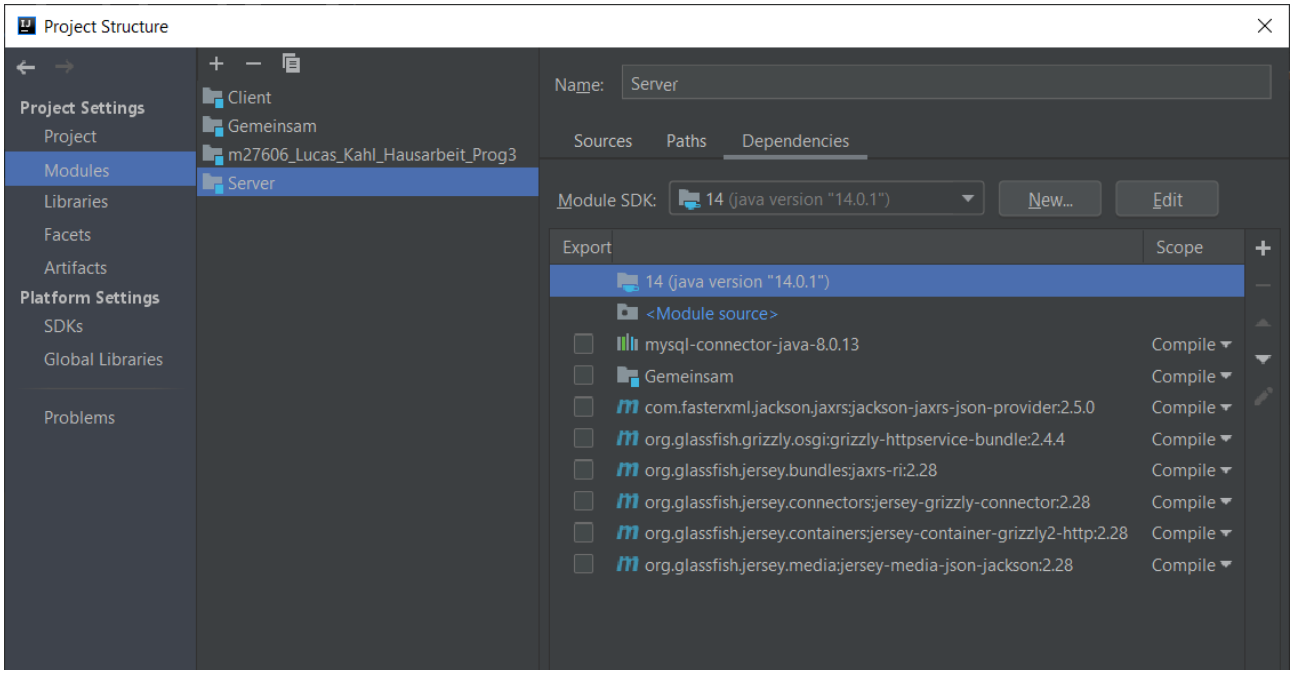

Abbildung 01 Maven-Bibliotheken des Server-Modules

Nachdem eine Serververbindung möglich war, kümmerte ich mich um die Integration der Datenbank. Die Datenbank Klasse stellt die Anfragen an die Datenbank über SQL-Statements und speichert die Ergebnistabellen als ResultSets. Daraus habe ich nun die Daten entnommen und in entsprechende Album- und Song-Objekte überführt.

Um die Verbindung mit der Datenbank herzustellen, musste ich zuerst import java.sql.*; durchführen und eine Connection nach dem Singleton-Entwurfsmuster anlegen.

Nachdem die Datenbankverbindung angelegt war und die Daten organisiert waren, führte ich erste Tests über die Konsole aus. Dabei wollte ich sicherstellen, dass ich entsprechende Speicheradressen der Daten lokal abrufen kann und ich über getter-Methoden gezielte Informationen ausgeben kann.

Als dies auf der Konsole möglich war, begann ich mit dem Entwurf einer GUI. Bei der Recherche stieß ich auf die Software SceneBuilder, welche es einfach ermöglicht, gutaussehende, strukturierte Grafikoberflächen anzulegen. Der intuitive Layoutprozess gestaltete sich gut über den SceneBuilder. Der SceneBuilder liefert eine fxml-Datei zurück, welche in Java geladen werden kann. Die GUI wird auf Grundlage der fxml erzeugt.

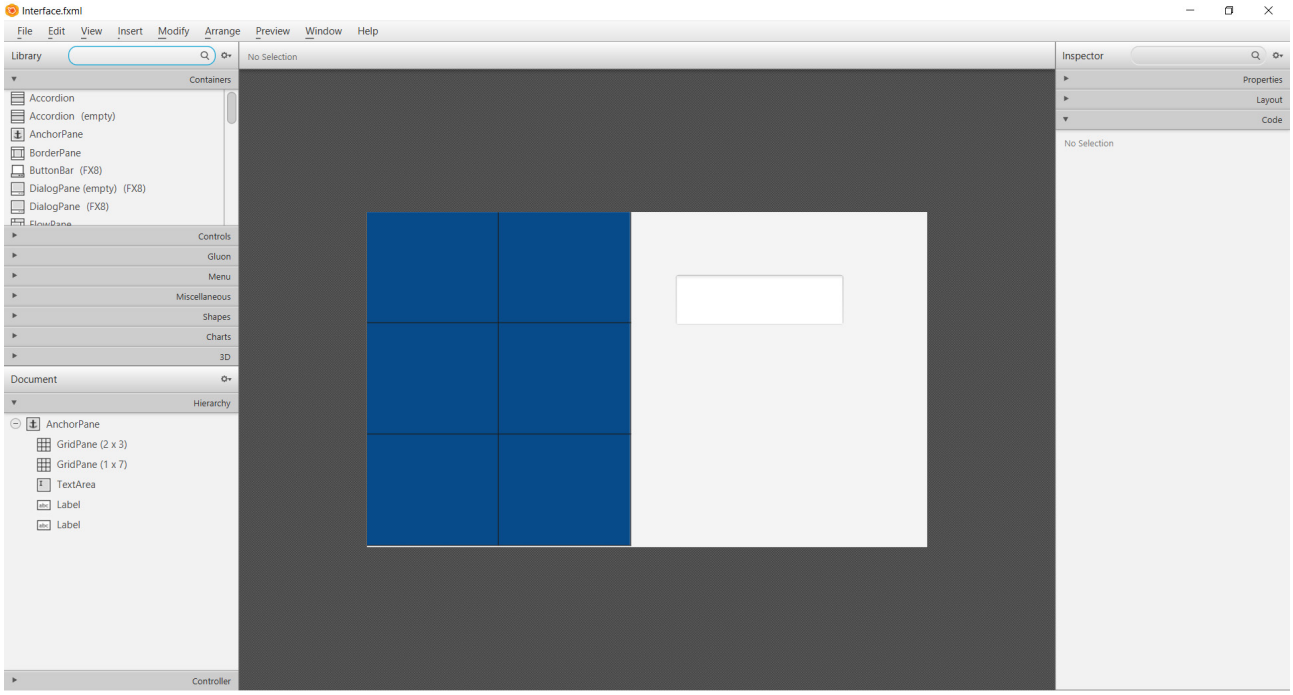

Abbildung 02 GUI-Entwicklung in SceneBuilder

Über die Vergabe von fx:id's war es anschließend möglich, die erstellen Elemente und Bereiche direkt anzusprechen und mit Inhalt zu füllen.

Zum Empfangen und darstellen der Daten mussten dem Client auch einige Bibliotheken bekannt gemacht werden. Dabei galt es ebenso wie beim Server darauf zu achten, dass konsistente Versionsnummern der Bibliotheken verwendet werden.

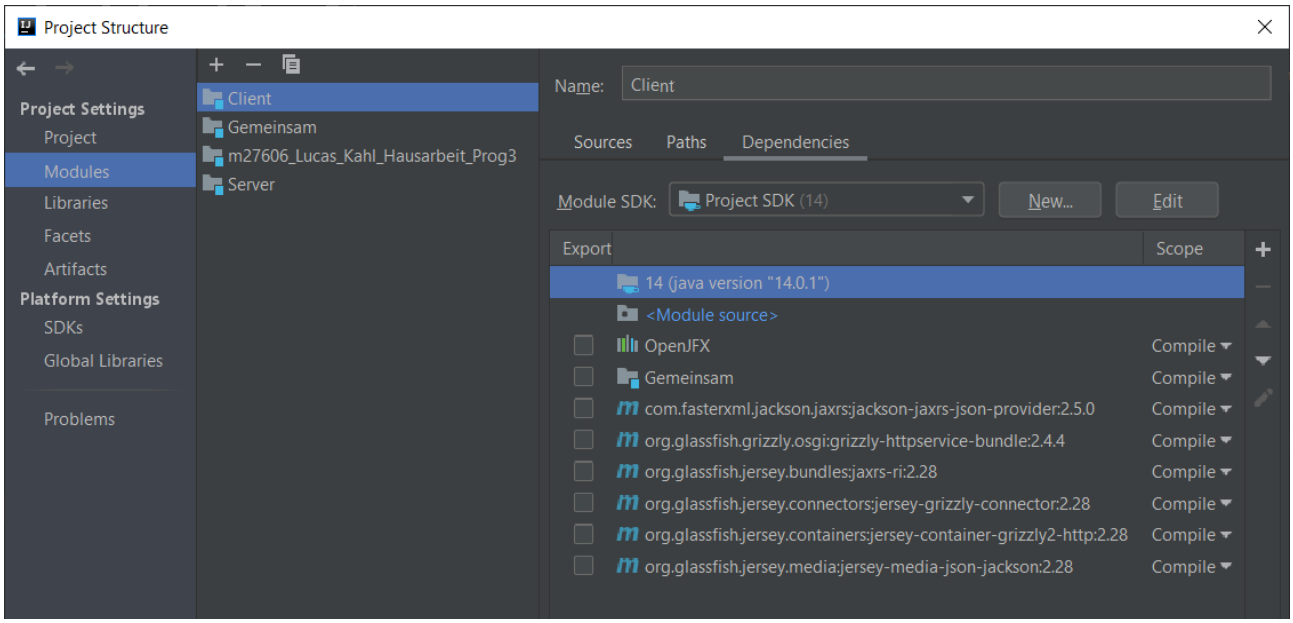

Abbildung 03 Maven-Bibliotheken des Client-Modules

3. Installation und Ausführung

Um das Programm ausführen zu können, sollte der Dienst XAMPP oder ein Equivalent installiert und aktiviert sein. (Dabei die Mysql-Funktionalität)

Die zu installierende Datenbank ist im Projektordner enthalten und kann beispielsweise lokal oder netzwerkweit verfügbar gemacht werden.

Achtung! Wenn die Datenbank netzwerkweit bereitgestellt wird oder online verfügbar gemacht wird, dann muss die URL zur Datenbank im Programmcode innerhalb der Datenbank-Klasse angepasst werden.

Installation der Datenbank lokal:

- über die cmd Konsole (Windows) oder ihr betriebssystemabhängiges Equivalent möglich
- über phpmyadmin -> Importieren -> .sql Datei auswählen und laden

Ich werde in dieser Dokumentation auf die Installation der Datenbank über phpmyadmin eingehen.

Um die Datenbank lokal verfügbar zu machen, legt man im Ordner „data“ unter C:\xampp\mysql\data einen neuen Ordner an. Beispielsweise „Musikdatenbank“.

-> C:\xampp\mysql\data\Musikdatenbank

In diesen Ordner wird jetzt die .sql-Datei kopiert, welche der Abgabe beigelegt ist.

-> musikdatenbank.sql

Zu Testzwecken habe ich diese Abläufe vor der Abgabe selbst an einem anderen PC durchgeführt.

Um die Datenbank zu laden, öffnete ich nun phpmyadmin im Browser und wählte den Reiter Importieren (1). Dort habe ich die sql-Datei über „Durchsuchen“ (2) im Ordner C:\xampp\mysql\data\Musikdatenbank ausgewählt und mit OK bestätigt. In diesem Schritt wurde bei mir die Datenbank strukturell erzeugt, vorerst ohne Inhalte. Anschießend markierte ich die erstellte Datenbank im linken Bereich (3) , dort wo alle Datenbänke aufgeführt sind, und wählte erneut „Importieren“ (4) . Dabei habe ich die selbe Datei erneut geladen(5), wobei es dieses Mal die Inhalte einfügte. Die Datenbank ist nun lokal installiert und mit Werten gefüllt.

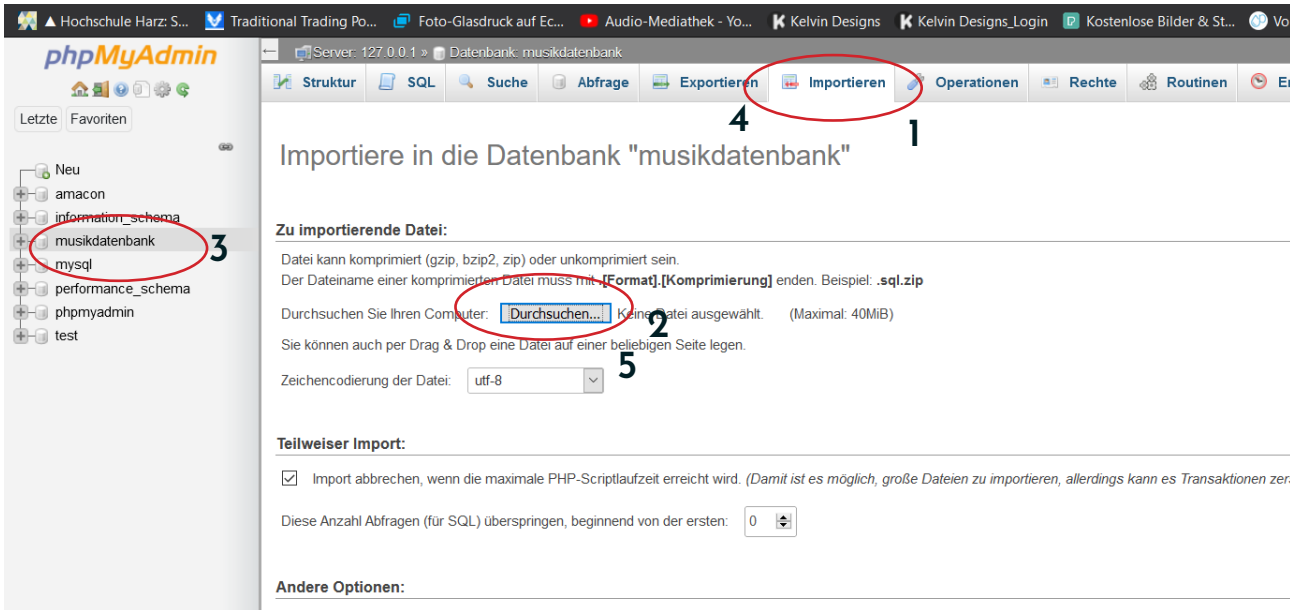

Abbildung 04 phpmyadmin - Import der Datenbank

Nach erfolgreichem zweiten Laden war die Datenbank mit allen Strukturen und Inhalten eingefügt.

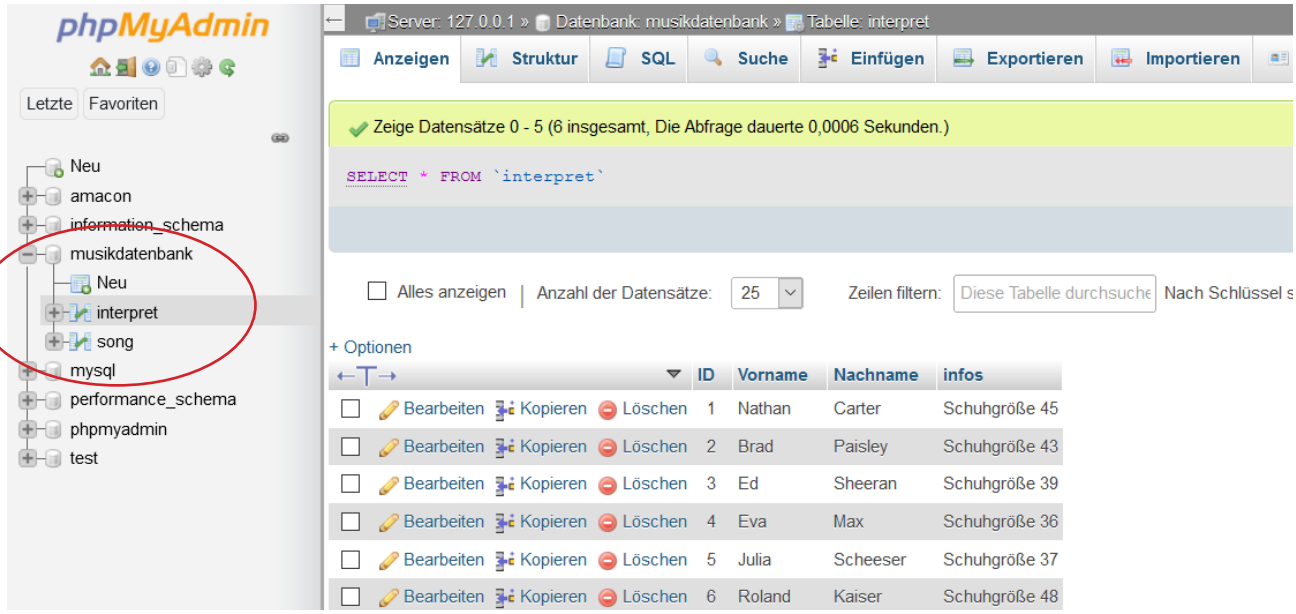

Abbildung 05 phpmyadmin - erfolgreicher Import der Datenbank anschaulich

Zur Probe nutzte ich das Windows cmd um ebenfalls dort auf die Datenbank zuzugreifen. Ebenfalls testete ich einige SQL-Statements.

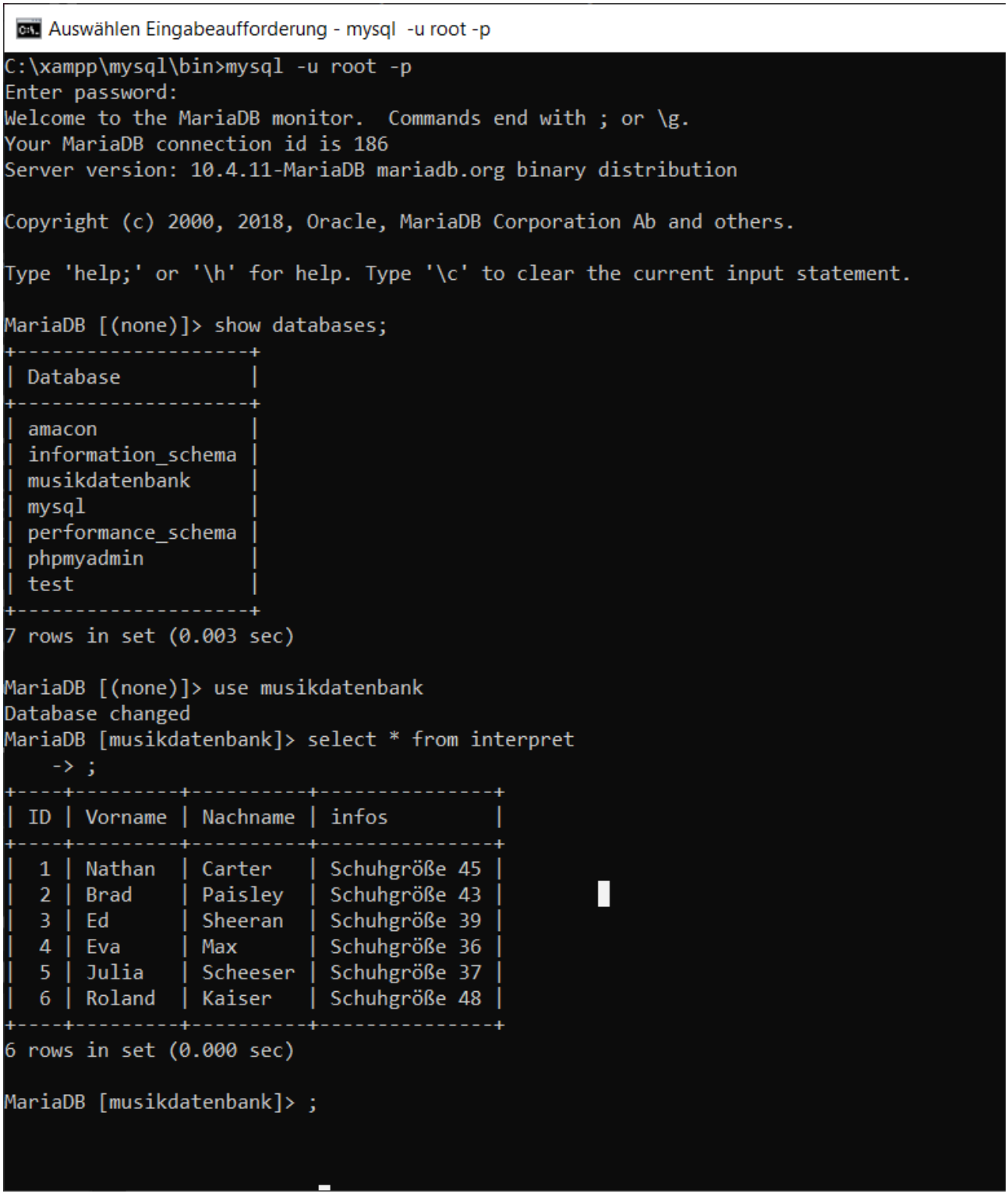

Abbildung 06 cmd - Datenbankzugriff und Funktionskontrolle

Ist die Datenbank lokal installiert, so kann die URL in der Datenbank Klasse gegebenenfalls angepasst werden. xampp nutzt im Regelfall den Port 3306 für mysql. Der, dem Port folgende, Pfadname hängt von der Benennung der Datenbank ab. Man muss auf Groß/ Kleinschriebung achten (6)

```
* Konstruktor ist private, sodass sich die Singleton Klasse nur selbst instanziieren kann
*
* es darf nur eine connection zur Datenbank geben!
*/
rivate Datenbank() {
    try {
        Connection connection = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/musikdatenbank",  user: "ro
        statement01 = connection.createStatement();
    }catch (SQLException e){
        System.err.println("Es gibt wohl einen Fehler bei der Datenbankverbindung.");
    }
```

6

Abbildung 07 IntelliJ - Hinweis auf URL-Überprüfung

Sobald die Datenbank installiert ist, kann das Programm ausgeführt werden.
Zuerst sollte in IntelliJ der Server gestartet werden. Anschließend können beliebig viele, voneinander unabhängige Clienten gestartet werden.

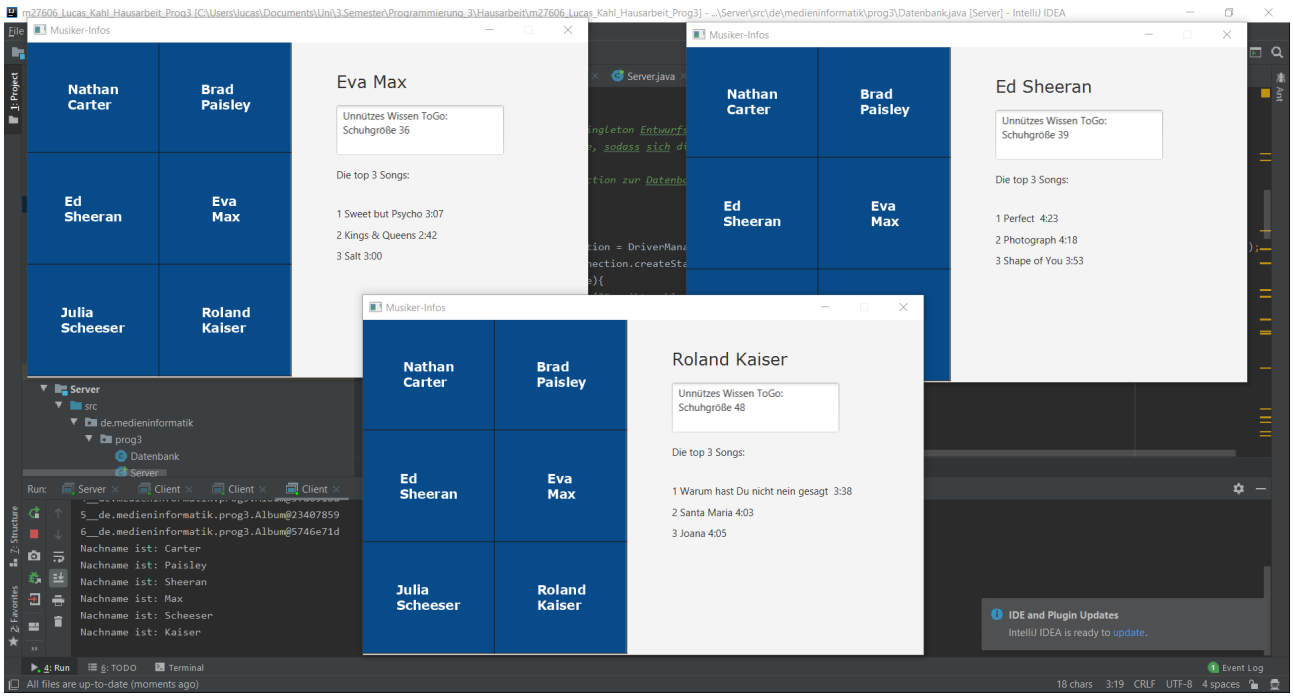

Abbildung 08 GUI - Ansicht zur Nutzung durch mehrere Clienten

4. Quellen:

Vorlesungsunterlagen und Praxisbeispiele der Fächer Programmierung 3 bei Prof. Singer und Softwaretechnik bei Herrn Scheithauer

SceneBuilder:
https://gluonhq.com/products/scene-builder/