▲ Hochschule Harz

Automatisierung und Informatik

Medieninformatik

Hausarbeit

in Programmierung 3

Dozent:
Prof. Jürgen K. Singer

Autor:
Jeremy Grunert
Kirchstraße 34
38855 Wernigerode
jgrnrt@gmail.com
Matrikelnummer: 29265

Wintersemester 2021/22
Wernigerode, 28.02.2022

Inhaltsverzeichnis

1 Installation

Für dieses Projekt ist ein MySQL-Datenbankmanagementsystem notwendig. XAMPP stellt als eine Möglichkeit solch ein System zur Verfügung.

Im XAMPP Control Panel muss das Apache- und das MySQL-Modul gestartet werden. Eine leere Datenbank mit dem Namen „cinelab“ mit Standardvoreinstellungen und ein Benutzer mit dem Namen „cinelab“ müssen via phpMyAdmin angelegt werden. Der Benutzer hat zur einfachen Handhabung globale Rechte und das Passwort „123456“. Die Datenbankinformationen können in der „Select.properties“-Datei im Server-Modul angepasst werden.

Entpacken Sie den Projektordner an einem beliebigen Ort. Die Start-Konfigurationen sind bereits im „.idea“-Ordner beigelegt und können mit der integrierten Entwicklungsumgebung IntelliJ IDEA genutzt werden.

2 Benutzung

Zuerst muss das MySQL-Modul im XAMPP Control Panel gestartet werden. Daraufhin kann der Projektordner mit IntelliJ IDEA geöffnet werden. Im vollständig geladenen IntelliJ IDEA-Fenster werden die Start-Konfigurationen angezeigt.

Sobald die „Server [run]“-Konfiguration gestartet wird, versucht der Server erstmalig eine Verbindung mit der Datenbank herzustellen. Bei einer erfolgreichen Verbindung wird eine neue Tabelle mit Beispielfilmen erstellt. Der Server läuft dann bis zur manuellen Beendigung. Ohne laufendem Server ist ein Client nicht nutzbar.

Anschließend können beliebig viele Instanzen der „Client [run]“-Konfiguration aufgerufen werden. Ein Client sendet beim erstmaligen Starten eine HTTP-GET-Anforderung und präsentiert daraufhin alle vorhandenen Filme in einer TableView.

In der Standard-Ansicht kann man die einzelnen Tabellenspalten sortieren. Des Weiteren bietet das „Search“-Textfeld die Option der Filterung nach allmöglichen Filminformationen. Letztlich kann man beliebig oft die TableView aktualisieren.

Um die Funktionalität des Clients zu erweitern, muss der „User“-SplitMenuButton auf „Admin“ gestellt werden. Damit erscheinen zwei neue Spalten in der TableView und ein neuer „+“-Button. Bei Auswahl des Buttons öffnet sich ein neues Fenster. In diesem Fenster kann ein neuer Film hinzugefügt werden. Die zwei Spalten beinhalten für die besetzten Reihen Bearbeiten- oder Lösch-Buttons. Ein „Edit“-Button öffnet ein ähnliches Fenster wie der „Add“-Button. In diesem Fenster sind die Filminformationen bereits in den entsprechenden Textfeldern hinzugefügt und können einfach bearbeitet werden. Der „-“-Button löscht den ausgewählten Film aus der Datenbank.

3 Vorgehen & Entwürfe

Das Projekt ist in drei Modulen unterteilt. Das Anwenderprogramm wird vom Client-Modul erstellt, welches mit einem Server kommuniziert. Informationen werden als Zeichenketten in Nachrichten-Entitäten übertragen. Der Server hat eine Verbindung zu einer SQL-Datenbank auf welcher alle Filme hinterlegt sind. Das Common-Modul stellt dem Server- und dem Client-Modul die Movie-Klasse zur Verfügung.

3.1 Common-Modul

Mit der Movie-Klasse lassen sich POJOs erstellen.

Die Klasse beinhaltet die notwendigen Felder, zwei Konstruktoren und Getter-Methoden (vgl. Abbildung 1). Der erste Konstruktor in der Movie-Klasse beinhaltet zusätzlich noch den Integer-Parameter für den Filmidentifikator (movieID). Dieser wird gebraucht, sobald die Filme aus der Datenbank an den Client übertragen werden oder ein bestimmter Film vom Client bearbeitet wird. Die Property-Felder bestimmen die Spalteninhalte in der TableView.

Die CustomDeserializer-Klasse überschreibt die deserializer-Methode in der Jackson Bibliothek. Damit kann der JSON Parser einen JSON String in ein neues Movie-Objekt umwandeln.

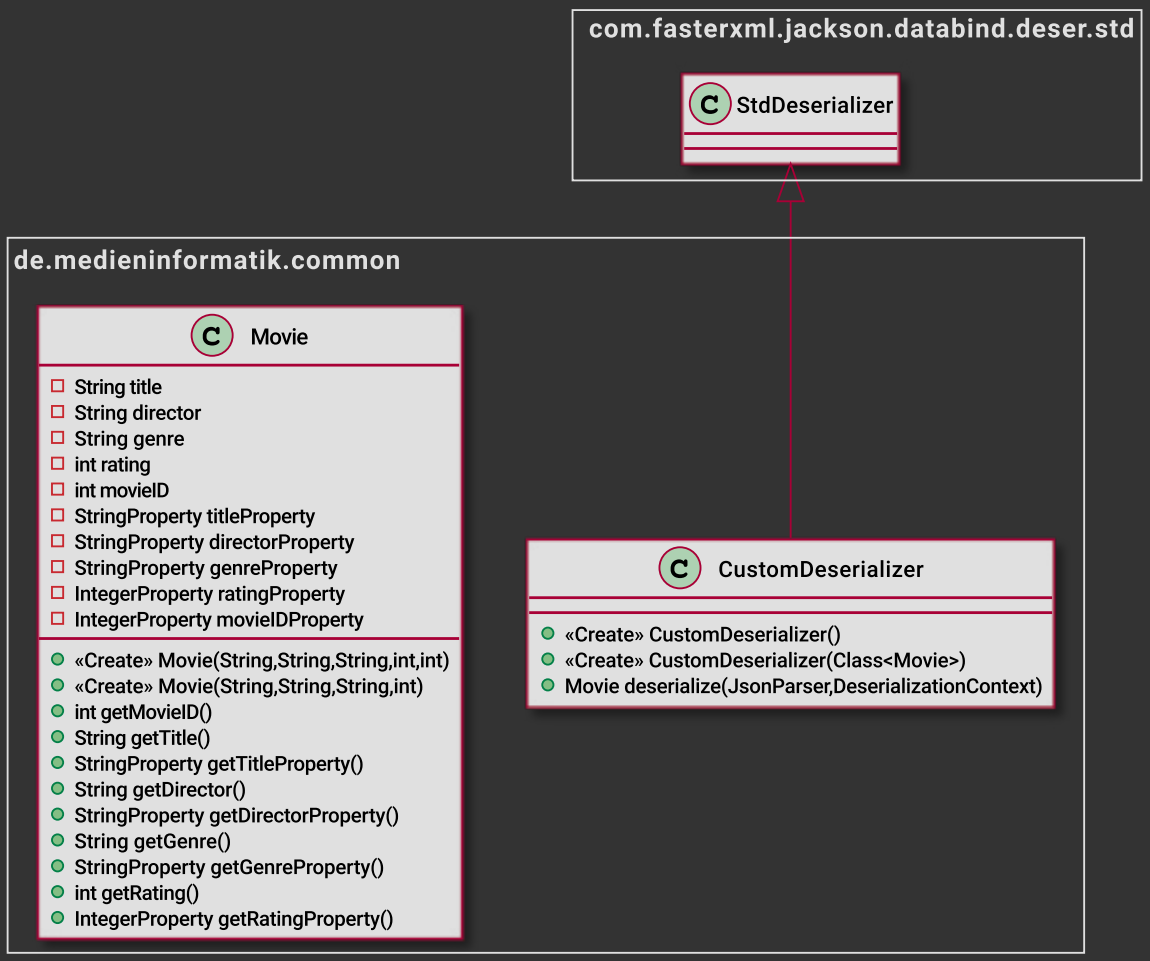

Abbildung 1: Common-Modul dargestellt mittels UML.
Quelle: Eigene Darstellung erstellt mit PlantUML

3.2 Server-Modul

In der Server-Klasse wird mit dem Grizzly Framework ein Server instanziiert. Bevor dieser gestartet wird, stellt die Database-Klasse fest, ob eine Verbindung zur Datenbank hergestellt werden kann. Falls das nicht der Fall ist, dann stürzt der Server ab. Bei einer erfolgreichen Verbindung überprüft die createDatabase-Methode, ob die essentielle Tabelle zur Speicherung der Filme vorhanden ist. Bei einer fehlenden Tabelle wird eine neue mit Beispielfilmen erstellt. Falls eine bereits vorhanden war, wird der Server gestartet, ohne eine neue Tabelle zu erstellen.

Die CineLabApplication-Klasse kombiniert alle Rest- bzw. Ressourcen-Klassen und wird von der Application-Klasse abgeleitet (vgl. Abbildung 2). Für dieses Projekt gibt es lediglich die CineLabRest-Klasse. Diese versucht eine Verbindung zur Datenbank herzustellen und verfügt über vier Antwort-Methoden. Die Kommunikation zwischen dem Client und dem Server funktioniert über die Jakarta RESTful Web Services API.

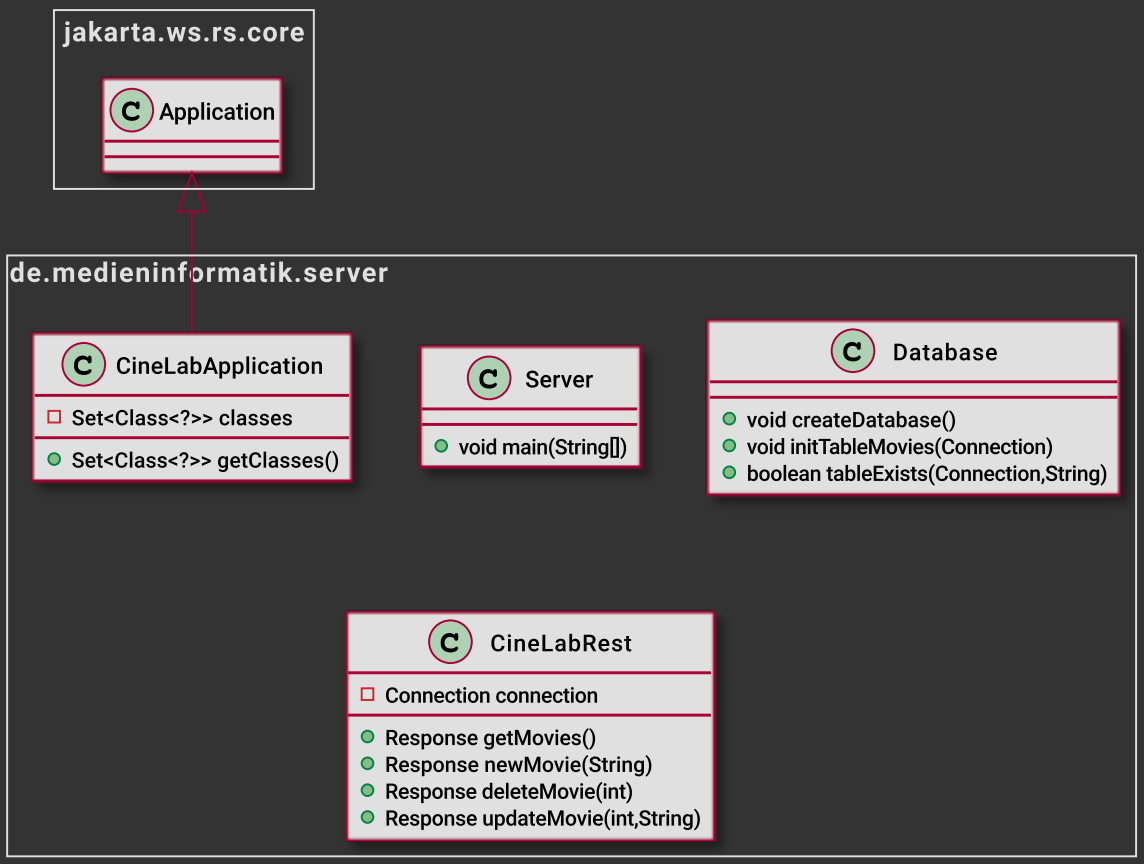

Abbildung 2: Server-Modul dargestellt mittels UML.
Quelle: Eigene Darstellung erstellt mit PlantUML

3.2.1 Get-Antwort

Diese speichert die Filmtabelle aus der Datenbank in einem ResultSet und konvertiert die Werte aus dem ResultSet in Filmobjekte. Die Filmobjekte werden dann in einer Liste gespeichert. Wenn die Liste nicht leer ist, wird sie als JSON String umgewandelt und als Antwort-Entität zum Client versandt. Bei einer leeren Liste wird eine Antwort zum Client mit dem Statuscode „404 Not Found“ übermittelt.

3.2.2 Post-Antwort

Der übergebene JSON String wird in ein Filmobjekt umgewandelt. In einer Statement-Vorlage werden die Filminformationen eingetragen und der Datenbank eingespeist.

Sobald ein Film der Datenbank hinzugefügt wird, erstellt diese automatisch einen Filmidentifikator.

Nach dem erfolgreichen Ausführen des Statements wird dem Client der Statuscode „204 No Content“ übermittelt. Falls das Ausführen nicht möglich war, wird der Statuscode „304 Not Modified“ übertragen.

3.2.3 Delete-Antwort

Mit dem Filmidentifkator-Parameter wird in der Datenbank der Filmeintrag lokalisiert und gelöscht. Wenn die Statement-Vorlage erfolgreich ausgeführt werden konnte, gibt die executeUpdate-Methode die Zeilenanzahl zurück, in dem der Filmeintrag entfernt wurde. Falls eine Null als Zeilenzahl übertragen wird, dann konnte der Film nicht gefunden werden. Dem Client wird dann der „304 Not Modified“-Statuscode übermittelt. Bei einer erfolgreichen Ausführung wird dem Client der Statuscode „204 No Content“ übermittelt.

3.2.4 Put-Antwort

Wie bei der Post-Antwort-Methode wird der übergebene „movieJSON“ String in ein Filmobjekt umgewandelt. Mit der Filmidentifikation wird in der Datenbank der Filmeintrag verändert. In einer Statement-Vorlage werden die neuen Filminformationen eingetragen. Nach dem erfolgreichen Ausführen des Statements wird dem Client der Statuscode „204 No Content“ übermittelt. Falls das Ausführen nicht möglich war, wird der Statuscode „304 Not Modified“ übertragen.

Alle Antworten mit dem Statuscode „204 No Content“ brauchen keine Nachrichten-Entität übermitteln, da diese überflüssig für den Client sind. Nach dem Aufrufen der Methoden Post, Delete oder Put folgt immer die Get-Antwort.

3.3 Client-Modul

Das Model View Controller Entwurfsmuster wird hier angewandt. Die Dialog-Controller-Klassen und die ClientController-Klasse sind die jeweiligen Controller. Die dazugehörigen FXML-Dateien sind die View Elemente. Sobald diese von den Controllern geladen werden beinhalten sie lediglich Sammlungen von Java Objekten, welche von den Controllern verändert werden können.

Die View beinhaltet eine TableView aus dem JavaFX Framework um eine Filmliste von Movie-Objekten zu präsentieren. Diese Liste wird vom Model „CineLab-Client“ bereitgestellt.

Das Model verfügt außerdem über die Funktionen zum Hinzufügen, Bearbeiten und Löschen von Einträgen in der Datenbank. Sobald im Client ein Film erstellt oder ein bereits vorhandener Film bearbeitet wird, dann wird dieser in ein neues Filmobjekt umgewandelt. Dieses Filmobjekt wird in der CineLabClient-Klasse

mit der Jackson Bibliothek in einen JSON-String serialisiert. Der JSON-String fungiert als Nachrichten-Entität für die jeweilige Anfrage.

Die Edit- und DeleteButton-Klassen sind innere Klassen die von der TableCell-Klasse abgeleitet sind (vgl. Abbildung 3).

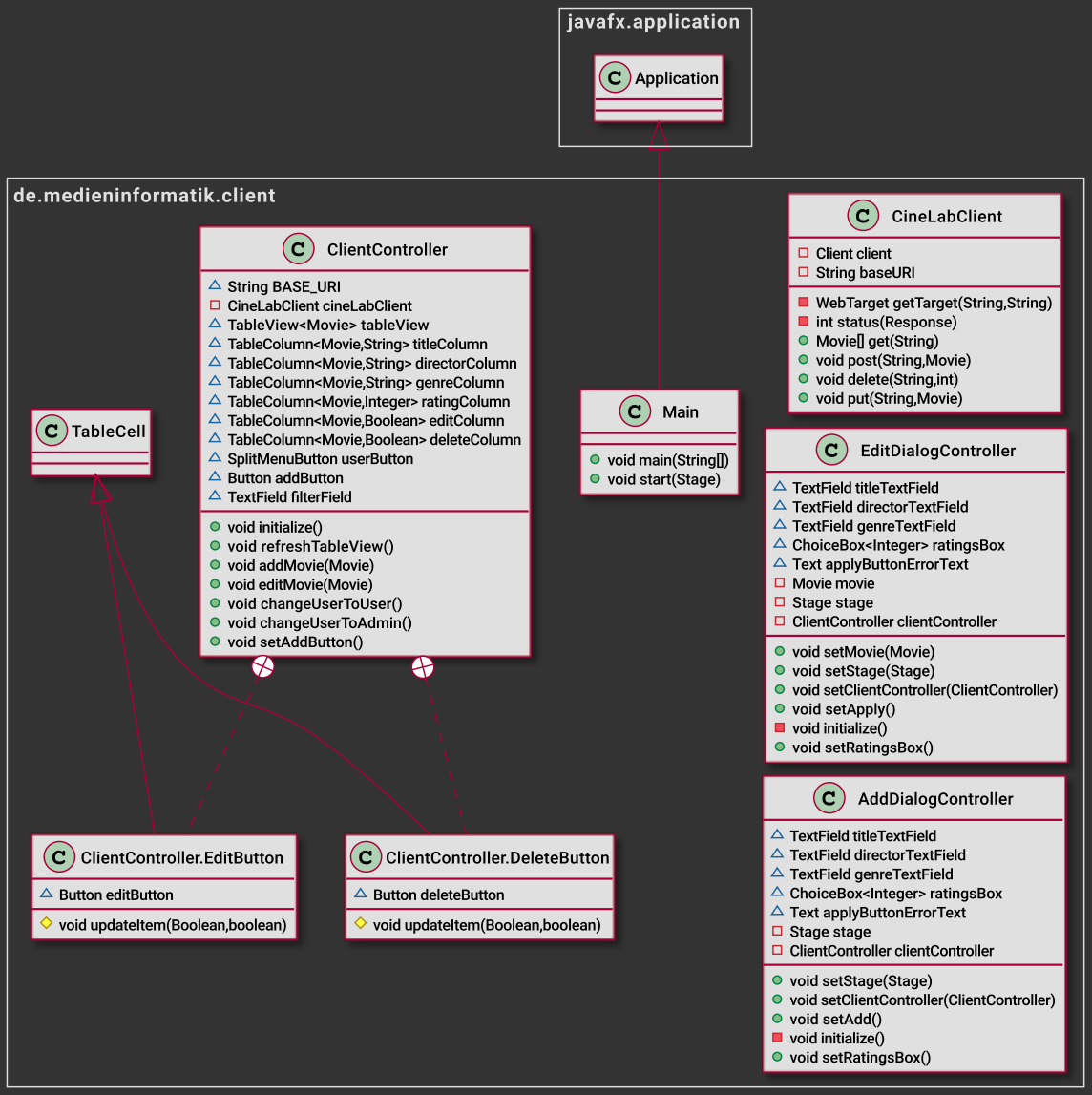

Abbildung 3: Client-Modul dargestellt mittels UML.
Quelle: Eigene Darstellung erstellt mit PlantUML

Die Get-Anfrage übermittelt an den Controller eine Sammlung von Filmobjekten – die Filmliste. Eine Get-Anfrage für einen einzelnen Film ist in diesem Projekt nicht notwendig. Die weiteren Anfragetypen brauchen keine Antworten an den Controller übergeben, da nach allen Anfragen immer die Get-Anfrage aufgerufen wird.

Die Filmliste wird im Controller zu einer gefilterten Liste übertragen, welche auf Veränderungen im Suchtextfeld achtet. Sobald nach einer bestimmten Filminformationen gesucht wird, werden alle übereinstimmenden Filme zu einer sortierten Liste hinzugefügt. Diese wird dann in der TableView angezeigt, sofern es auch Filme in der Filmliste gibt. Ansonsten zeigt die TableView keinen Inhalt.

Abkürzungsverzeichnis

API — Programmierschnittstelle

JSON — JavaScript Object Notation

POJO — Plain Old Java Object

UML — Unified Modeling Language

Literaturverzeichnis

Clientfunktion zum Hinzufügen eines Buttons pro Zeile in der TableView:
Agarwal, A. (2014, 24. März). Delete Rows using row buttons in Javafx TableView. GitHub Gist. Abgerufen am 21. Februar 2022, von https://gist.github.com/abhinayagarwal/9735744

Clientfunktion zum Sortieren und Filtern der TableView:
Jakob, M. (2015, 15. April). JavaFX 8 TableView Sorting and Filtering. Code. Makery. Abgerufen am 21. Februar 2022, von https://code.makery.ch/blog/javafx-8-tableview-sorting-filtering/

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken – dazu gehören auch Internetquellen – als solche kenntlich gemacht habe.

Wernigerode, 25.02.2022