*Abstract*

## Zielsetzung

Ziel des Projekts Medianizer war die Konzeption und Implementierung einer Desktop-Anwendung zur Verwaltung physischer Medienbestände. Die Anwendung ermöglicht das Erfassen, Suchen, Bearbeiten und Löschen von Medienobjekten in einer SQLite-Datenbank und bietet eine intuitive Benutzeroberfläche auf Basis von JavaFX.

Im Mittelpunkt stand dabei die Umsetzung einer sauberen, wartbaren Architektur, die konsequent dem MVC-Prinzip (Model-View-Controller) folgt, sowie die Integration einer persistenten Datenspeicherung. Die Entwicklung erfolgte vollständig in Java unter Verwendung von Maven als Build-Management-System.

## Projektverlauf und Vorgehen

Die Implementierung des Projekts erfolgte innerhalb von 10 Werktagen. Zuvor wurde eine detaillierte Zeitplanung erstellt, die die Abschnitte Entwicklung, Test und Dokumentation beinhaltete. Diese Struktur half dabei, den Überblick über den Fortschritt zu behalten und die gesetzten Meilensteine termingerecht zu erreichen. Die Erstellung der Ausführbaren Anwendung, sowie die Abgabe erfolgten an einem zusätzlichen Werktag.

Die Umsetzung begann mit der Modellierung der zentralen Klassenstruktur im Model, die die Entitäten Film und CD sowie die Datenbankverwaltung (DatabaseManager) umfasst. Darauf aufbauend wurden schrittweise die View-Komponenten (JavaFX-Oberflächen) und Controller-Logik implementiert.

Die Erweiterbarkeit und Testbarkeit des Systems wurden durch die Kommunikation zwischen diesen Schichten über klar definierte Schnittstellen, sowie eine vorausschauende Struktur realisiert.

## Herausforderung und Problemlösung

Eine der größten technischen Herausforderungen trat im Zusammenhang mit der Versionsverwaltung über GitHub auf. Ein fehlerhafter Commit führte dazu, dass lokale Änderungen als bereits übermittelt markiert wurden. Dies verursachte anhaltende Konflikte mit Maven, das danach nicht mehr korrekt über die Kommandozeile ausgeführt werden konnte. Dieses Problem erforderte eine manuelle Bereinigung des Repositorys und eine Neukonfiguration der lokalen Entwicklungsumgebung. Trotz dieses Rückschlags konnte die Projektarbeit ohne signifikante Zeitverzögerung fortgesetzt werden. Darüber hinaus war es aufgrund der Größe der ausführbaren Anwendung nicht möglich diese auf GitHub hochzuladen, daher ist diese lediglich über die Abgabefunktion von Pebblepad erfolgt (s. Anmerkung 1 – Anhang Abstract).

Ein weiterer Aspekt betraf die Integration von JavaFX in die Tests. Da grafische Elemente nicht direkt in automatisierten Testumgebungen ausgeführt werden können, wurde ein AlertHelper eingeführt, der Warn- und Informationsmeldungen im Testmodus unterdrückt. Diese Lösung ermöglichte eine reibungslose Testautomatisierung, ohne die Produktionslogik verändern zu müssen. Die für die Tests vorgenommenen Änderungen sind in den Commits auf GitHub nachzuvollziehen, wurden aber für das Endprodukt wieder entfernt, da diese auch in realen Projekten nicht mit ausgeliefert werden.

Auch der Umgang mit den Prüfungsphasen stellte eine organisatorische Herausforderung dar. Aufgrund des engen Zeitplans war es nicht möglich, nach jeder Phase auf Feedback zu warten. Stattdessen wurde in iterativen Zyklen gearbeitet, wodurch die Entwicklung zügig und dennoch strukturiert voranschreiten konnte. (s. Anmerkung 2 – Anhang Abstract)

Am Tag des Releases fiel ein kleiner unentdeckter Bug auf, der zwar schnell behoben werden konnte, aber dennoch die Annahme der Risikoanalyse bestätigt, dass das Risiko mittel-hoch war.

## Ergebnisse und Bewertung

Die Anwendung konnte vollständig umgesetzt und getestet werden. Sämtliche Unit- und Integrationstests verliefen erfolgreich. Besonders die Trennung der Logik in klar abgegrenzte MVC-Komponenten erwies sich als entscheidend für die Stabilität und Erweiterbarkeit des Systems. Die SQLite-Datenbank erwies sich als zuverlässige, leichtgewichtige Lösung für die Persistenzschicht, und JavaFX bot eine moderne, flexible Plattform zur Realisierung der Benutzeroberfläche.

Die Zielsetzung, eine funktionsfähige, intuitive und technisch saubere Anwendung zu entwickeln, wurde erreicht. Die implementierten Funktionen – insbesondere das Einfügen, Suchen, Bearbeiten und Löschen von Medien – arbeiteten in allen Tests fehlerfrei zusammen. Darüber hinaus war die Teststrategie mit einer klaren Trennung zwischen Unit-, Integrations- und Systemtests ein wichtiger Beitrag zur Qualitätssicherung.

Für zukünftige Projekte wäre eine klarere Trennung zwischen Entwicklungs- und Testdatenbank, eine frühere Automatisierung der Testprozesse und eine Erweiterung der Systemtests sinnvoll. Zudem könnten zusätzliche Funktionen wie Export-Optionen oder eine erweiterte Suchlogik die Anwendung weiter verbessern.

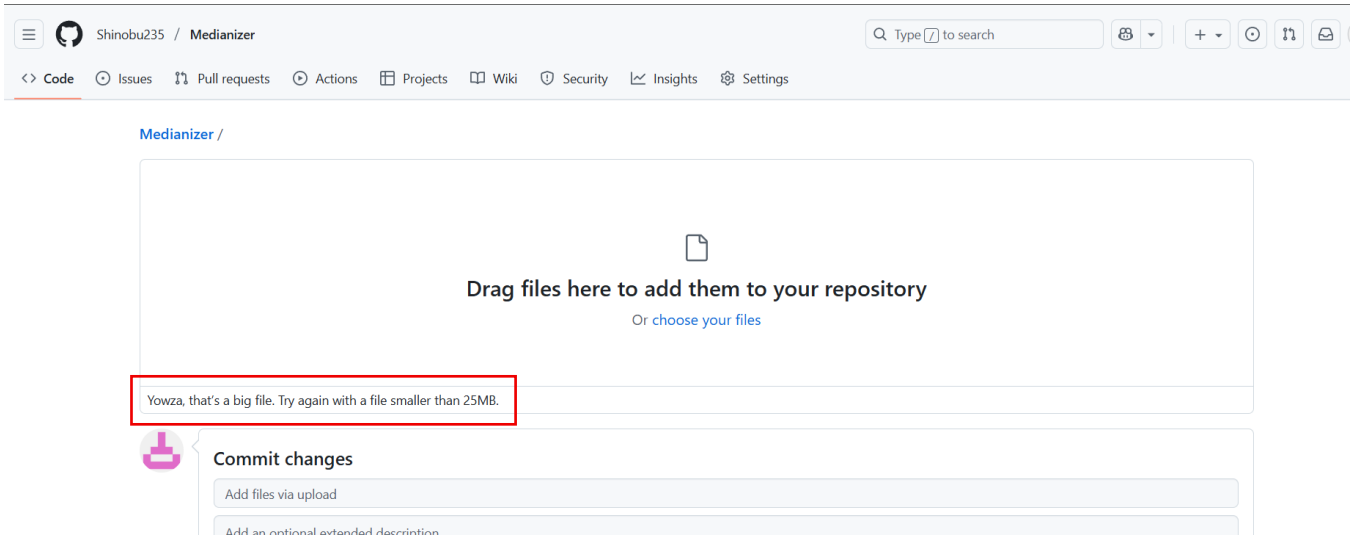
## Fazit

Das Projekt Medianizer war sowohl technisch als auch organisatorisch erfolgreich. Die Anwendung erfüllt alle funktionalen Anforderungen, ist stabil und leicht wartbar. Trotz einzelner technischer Hürden konnte der Zeitplan für die Implementierung eingehalten werden und die strukturierte Vorgehensweise führte zu einer hohen Codequalität. Die Arbeit bot wertvolle praktische Einblicke in die Softwareentwicklung unter realistischen Bedingungen und stärkte insbesondere die Fähigkeiten im Bereich Architekturplanung, Testing und Fehlerbehebung.

Anhang Abstract

## Anmerkung 1

Die ausführbare Anwendung konnte aufgrund einer Fehlermeldung von GitHub nicht dort bereitgestellt werden. Sie befindet sich stattdessen auf Pebblepad. Auf dem folgenden Screenshot ist die Fehlermeldung zu erkennen:




## Anmerkung 2

Nach der Projektplanung lagen sowohl die Dokumentation als auch die ausführbare Anwendung vor. Da in den vorherigen Phasen jedoch nicht auf das Feedback gewartet werden konnte, wurde die Notwendigkeit einer Überarbeitung deutlich. Aus diesem Grund wurde der Abgabetermin nach Absprache verschoben. Es wurden jedoch nur Anpassungen an den Dokumenten, nicht jedoch an der ausführbaren Anwendung vorgenommen, sodass der Projektplan dennoch eingehalten wurde.