Filmilox: Technical Report

Arina Antskaitis, Manuel Xuan Bodner, Etem Karao, Kevin Kraus, Benedikt Niedernhuber, Arsen Oganisyan, Eni Veshi
Ostbayerische Technische Hochschule Amberg-Weiden
Amberg

***Abstract*—Filmilox ist eine Web Plattform zur Bewertung von Filmen. Filmilox verwendet ein React Frontend, das Material UI Bibliotheken einbindet. Das Filmilox Backend setzt auf node.js und Express. Für Persistenz wird die MongoDB Datenbank verwendet. Wir beschreiben in diesem Dokument die wichtigsten Projektziele und geben einen Überblick über die Architektur der Anwendung und Schnittstellen zwischen den Schichten unserer Architektur.**

***Index Terms*—film, review, movie, web, MERN**

I. Einleitung

Ziel der Filmilox Web Plattform ist es, eine Online Plattform zum Bewerten von Filmen zu sein. Dafür sind folgende Kernanforderungen notwendig. Eine Datenbank von Filmen und ihren Bewertungen, die durch Administratoren pflegbar und erweiterbar ist. Es ist essentiell, dass möglichst alle Filme ein Coverbild haben sollen. Filmilox benötigt eine Startseite, die immer eine Übersicht der relevantesten Filme bieten soll. Benutzer mit einem Benutzeraccount sollen in der Lage sein, für Filme Bewertungen abzugeben, die Bewertungen haben einen Wert zwischen 0 und 10 Sternen. Bewertungen können auch Text-Rezensionen enthalten und andere Benutzer können Bewertungen mittels Up- und Downvotes bewerten. Die Plattform soll alle gängigen Geräte und Display-Dimensionen mittels Responsive Webdesign unterstützen. Browserkompatibilität soll mit den aktuellen Versionen von Chrome, Safari und Firefox sichergestellt werden. Als Web Anwendung sollen alle gängigen Betriebssysteme und Endgeräte unterstützt werden. Exotische Bildschirmformate sollen nicht unterstützt werden. Unsere wichtigsten Qualitätsziele sind Sicherheit, Performance-Effizienz und funktionale Software. Da wir Benutzerdaten verwalten, ist Sicherheit eines unserer wichtigsten Qualitätsziele und essentiell für unsere Stakeholder. Aufgrund unseres beschränkten Hosting-Budgets ist ein hoher Grad an Performance Effizienz notwendig. Um am Markt wettbewerbsfähig zu sein, ist es notwendig, dass unsere Software korrekt funktioniert und unsere Kernfunktionen im Bereich Filmbewertungen verlässlich funktionieren. Es soll keine unnötige Funktionalität integriert werden.

II. Randbedingungen

Filmilox als Projekt ist durch die kurze Entwicklungszeit von einer Woche und dem kleinen Entwicklungsteam von 7 Entwicklern eingeschränkt. Unser Budget ist auch substantiell eingeschränkt und liegt bei 0€. Es dürften keine zusätzlichen Kosten anfallen. Entsprechend ist das Feature Set minimal. Aufgrund dieser schweren Einschränkungen haben wir uns auch auf die manuelle Aufnahme einzelner Filme in die Datenbank entschieden, anstatt uns von einer existierenden bezahlten API abhängig zu machen.

III. Kontextabgrenzung

Filmilox ist in sich geschlossen und hat keine externen Schnittstellen, außer der Integration von eingebetteten YouTube Videos, die unsere Filmtrailer Funktionalität bereitstellen.

IV. Architekturübersicht

A. MERN

Express und Node bilden die mittlere Anwendungsebene. Express.js ist ein serverseitiges Webframework und Node.js die beliebte und leistungsstarke JavaScript-Serverplattform. MERN ist der ideale Ansatz, um durchgängig mit JavaScript und JSON zu arbeiten.

B. React.js-Frontend

Die oberste Ebene des MERN-Stacks ist React.js, das deklarative JavaScript-Framework zum Erstellen dynamischer clientseitiger Anwendungen in HTML. Mit React können Sie komplexe Schnittstellen durch einfache Komponenten aufbauen, sie mit Daten auf Ihrem Backend-Server verbinden und sie als HTML rendern. Die Stärke von React besteht darin, zustandsbehaftete, datengesteuerte Schnittstellen mit minimalem Code und minimalem Aufwand zu handhaben und bietet als moderner Web-Framework viele hilfreiche Werkzeuge wie: großartige Unterstützung für Formulare, Fehlerbehandlung, Ereignisse, Listen und mehr.

C. Express.js- und Node.js-Serverebene

Die nächste Ebene nach unten ist das serverseitige Express.js-Framework, das innerhalb eines Node.js-Servers ausgeführt wird. Express.js bezeichnet sich selbst als ”schnelles, unparteiisches, minimalistisches Web-Framework für Node.js”, und das ist es auch. Express.js verfügt über leistungsstarke Modelle für das URL-Routing (Abgleich einer eingehenden URL mit einer Serverfunktion) und die Verarbeitung von HTTP-Anforderungen und -Antworten.

Indem Sie XML-HTTP-Anforderungen (XHRs) oder GETs oder POSTs von Ihrem React.js-Front-End ausführen, können Sie eine Verbindung zu Express.js-Funktionen herstellen, die Ihre Anwendung unterstützen. Diese Funktionen wiederum verwenden die Node.js-Treiber von MongoDB, entweder über Rückrufe für die Verwendung von Promises, um auf Daten in

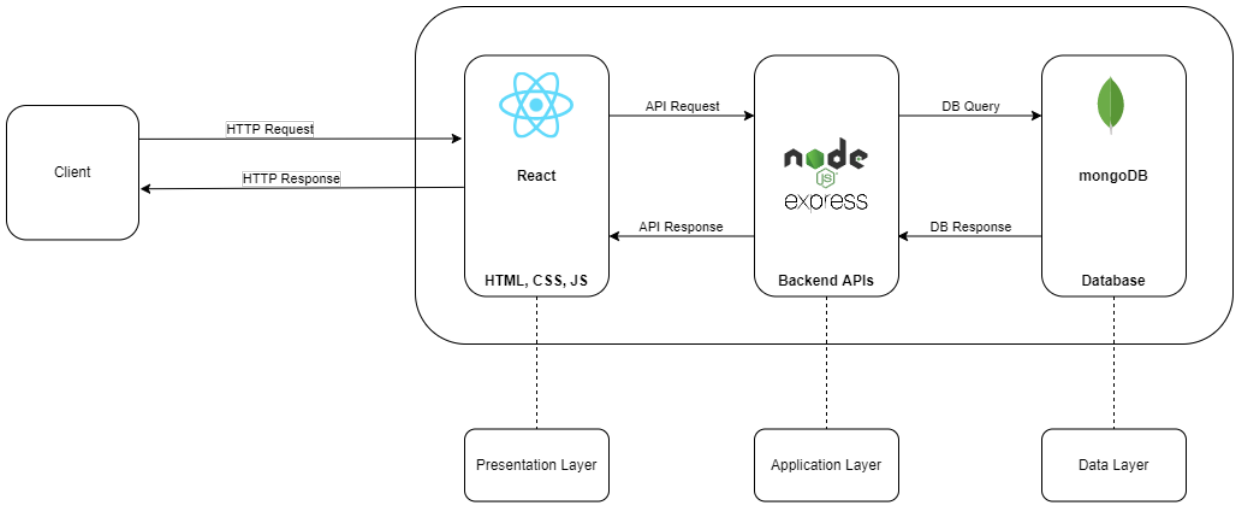

Fig. 1. Übersicht über die technische Architektur der Filmilox Plattform

Ihrer MongoDB-Datenbank zuzugreifen und diese zu aktualisieren.

D. MongoDB-Datenbankebene

Wenn Ihre Anwendung Daten speichert (Benutzerprofile, Inhalte, Kommentare, Uploads, Ereignisse usw.), möchten Sie eine Datenbank, mit der Sie genauso einfach arbeiten können wie mit React, Express und Node.

Hier kommt MongoDB ins Spiel: JSON-Dokumente, die in Ihrem React.js-Frontend erstellt wurden, können an den Express.js-Server gesendet werden, wo sie verarbeitet und (vorausgesetzt, sie sind gültig) für einen späteren Abruf direkt in MongoDB gespeichert werden.

V. Frontend Architektur Übersicht

Zur Authentifizierung gehören die Klassen “Login” und “Register”. Um sich zu registrieren, wird der Benutzer dazu aufgerufen, entweder einen Benutzernamen oder eine E-mail und ein Passwort anzugeben. Zum Einloggen werden ein Identifier und das Passwort benötigt. Der Identifier kann entweder der Username oder die Email-Adresse sein.

In der AppBar gibt es die Möglichkeit, über ein Suchfeld einen bestimmten Film zu finden, wenn dieser in der Datenbank existiert. Falls der gesuchte Film existiert, kann man auf dessen Details-Seite zugreifen. Außerdem kann man über die AppBar auf die Admin Seite (falls der eingeloggte User ein Admin ist) und die UserSettings Seite zugreifen. Weiterhin kann man darüber die Registrierung und Login Seite aufrufen, bzw. sich ausloggen.

In der Admin-Komponente kann ein Benutzer, der als Admin deklariert wurde, neue Filme in die Datenbank hinzufügen. Dabei muss der Filmtitel, eine Filmbeschreibung, ein Trailerlink und das Veröffentlichungsdatum des Filmes angegeben werden. Ein Filmposter kann optional auch hinzugefügt werden. Nach dem Hinzufügen des Films wird dieser auf der Overview Seite angezeigt und eine Details Seite automatisch hinzugefügt. Auf der UserSettings Seite kann der Benutzer ein Profilbild hochladen, um seinen Avatar zu ändern. Standardmäßig werden die ersten 2 Zeichen seines Benutzernamens angezeigt.

Die Hauptseite besteht aus der Overview-Komponente, die sich aus der MovieCard-Komponente zusammensetzt. Die MovieCard hat die wichtigsten Informationen zum Film wie den Titel, das Filmposter, die aktuelle Durchschnittsbewertung und einen Link zum Trailer. Klickt man auf die MovieCard, so landet man auf der Detailseite. Diese Seite ist repräsentiert durch die Klasse MovieDetails. Außer dem Titel, dem Bild, der durchschnittlichen Bewertung und einem Link zum Trailer verfügt MovieDetails über zusätzliche Informationen wie eine Filmbeschreibung, das Erscheinungsdatum und die Benutzerbewertungen. Ein Benutzer, der sich registriert und angemeldet hat, kann zu einem Film eine Bewertung hinterlassen (AddReview). Eine Benutzerbewertung (Review) besteht aus einem Kommentar und einer Sternebewertung. Andere Nutzer können diese Benutzerbewertung gut oder oder schlecht bewerten (like, dislike). Der Autor der Bewertung kann seine Bewertung löschen.

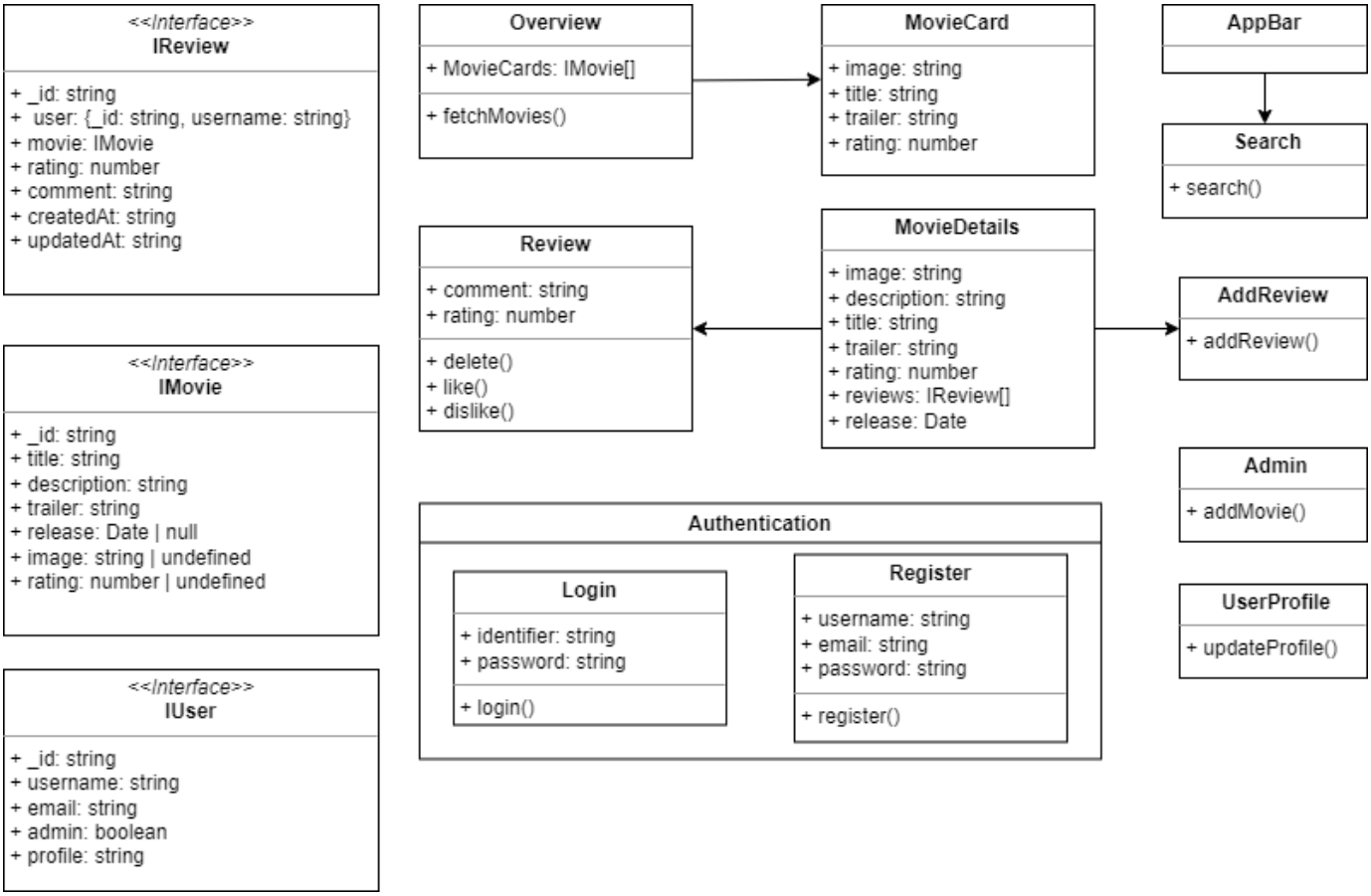

Fig. 2. Klassendiagramm des Filmilox React Frontends

VI. Datenbank Schema

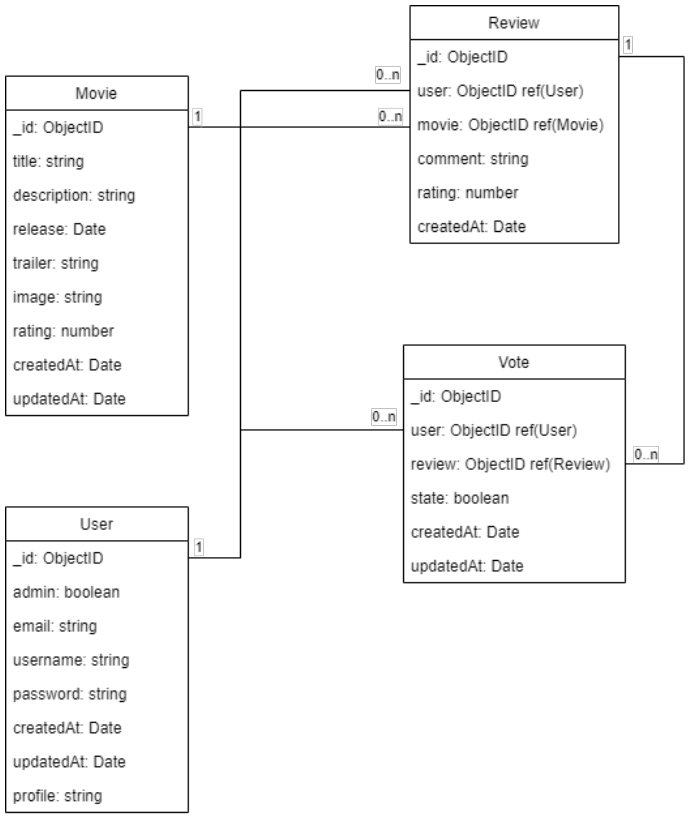

Fig. 3. Datenbank-Schema der Filmilox Datenbank

Zur Datenbank gehören folgende Entitäten: Movie, Review, User und Vote. Ein Film wird durch den Filmtitel, die Filmbeschreibung, das Erscheinungsdatum, einem Trailerlink, dem Filmposter (optional), und der Durchschnittsbewertung definiert. Ein Review hat zwei Referenzen. Eine Referenz zeigt auf den Benutzer, der die Bewertung geschrieben hat. Die andere Referenz zeigt auf den Film, zu dem die Bewertung gehört. Außerdem besteht das Review auch aus einem Kommentar und einer Sternebewertung (von eins bis zehn). Ein Vote besitzt eine Referenz zu der Bewertung (Review), zu der es gehört und eine Referenz zum Benutzer (User), der die Abstimmung hinterlassen hat. State repräsentiert einen Upvote (true) oder Downvote (false). Ein Benutzer ist definiert durch die E-Mail Adresse, den Benutzernamen, das Passwort, das Profilbild (optional) und einen Admin-Flag (true: Admin, false: normaler Benutzer).

Glossar

Admin	Bezeichnung für die Benutzer mit Verwaltungsberechtigung für die Anwendung.
AppBar	Obere Navigationsleiste der Anwendung, typisch für Material UI Anwendungen.
Backend	Für Endbenutzer unsichtbare Ebene unserer Architektur im Schichten Modell.
Downvote	Negative Stimme eines Benutzers für eine Film-Rezension.
Frontend	Die Ebene unserer Architektur mit der, der Endbenutzer interagiert.
GET	Ein HTTP Request um eine Ressource von einem Server anzufordern.
HTTP	Das Hypertext Transfer Protocol, ein Protokol zur Datenübertragung
JSON	JavaScript Object Notation ein Format für strukturierte Dokumente
POST	Ein HTTP Request um Daten an einen Server zu schicken und zu erhalten.
Upvote	Positive Stimme eines Benutzers für eine Film-Rezension