# Prototyping Projektdokumentation

Name: Thinley Sheda

E-Mail: shedathi@students.zhaw.ch

URL der deployten Anwendung: <https://shedadb.netlify.app/>

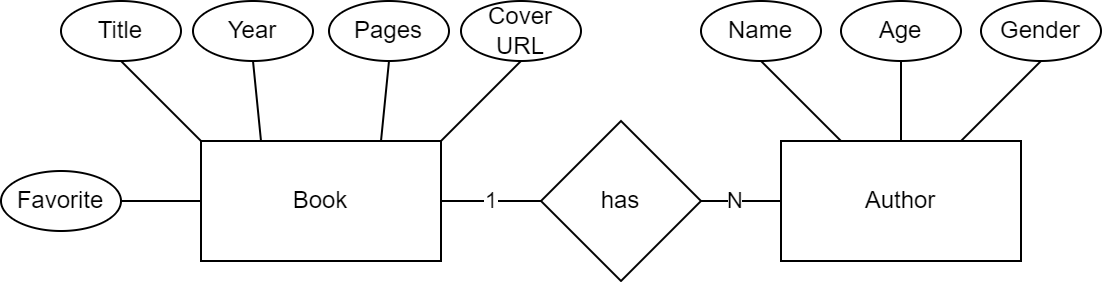
## Einleitung

ShedaDB ist eine übersichtliche und einfach zu bedienende Buchdatenbank, mit der Literaturbegeisterte ihre persönliche Sammlung verwalten können. Die Grundidee besteht darin, Leserinnen und Lesern eine Plattform zu bieten, auf der sie ihre Bücher nicht nur speichern, sondern auch besser organisieren und neue Favoriten entdecken können.

Die wichtigsten Funktionen der Anwendung umfassen das Anlegen und Verwalten von Büchern mit Titel, Autor, Genre und weiteren Details. Zudem können Lieblingslisten erstellt werden, um besonders geschätzte Bücher schnell im Blick zu behalten. Ein Statistikbereich unterstützt die Nutzerinnen und Nutzer dabei, Einblicke in das eigene Leseverhalten zu erhalten – beispielsweise in Bezug auf Anzahl oder Kategorien der Bücher. Dank der leistungsfähigen Such- und Filterfunktion lassen sich einzelne Titel oder bestimmte Genres in Sekundenschnelle finden. Für die kontinuierliche Erweiterung der persönlichen Bibliothek ermöglicht ShedaDB das komfortable Hinzufügen neuer Bücher.

ShedaDB ist benutzerfreundlich, ansprechend gestaltet und lässt sich individuell anpassen. Zukünftig sind zusätzliche Funktionen wie Empfehlungen, Buchrezensionen und Fortschrittsverfolgung geplant, damit die Buchverwaltung noch umfassender und inspirierender wird.

## Datenmodell



Ein Buch wird durch die Entität **Book** dargestellt und besitzt die Attribute:

* **Title:** Der Titel des Buches
* **Year:** Das Erscheinungsjahr
* **Pages:** Die Anzahl der Seiten
* **Cover URL:** Ein Link zum Coverbild
* **Favorite:** Eine Kennzeichnung, ob das Buch ein Favorit ist

Ein Autor wird durch die Entität **Author** dargestellt und verfügt über die Attribute:

* **Name:** Der Name des Autors
* **Age:** Das Alter des Autors
* **Gender:** Das Geschlecht des Autors

Die Beziehung zwischen Buch und Autor wird durch die Verbindung has definiert, welche eine 1:N-Beziehung darstellt. Das bedeutet, dass jedes Buch mindestens einen Autor haben muss, während ein Autor mehrere Bücher geschrieben haben kann.

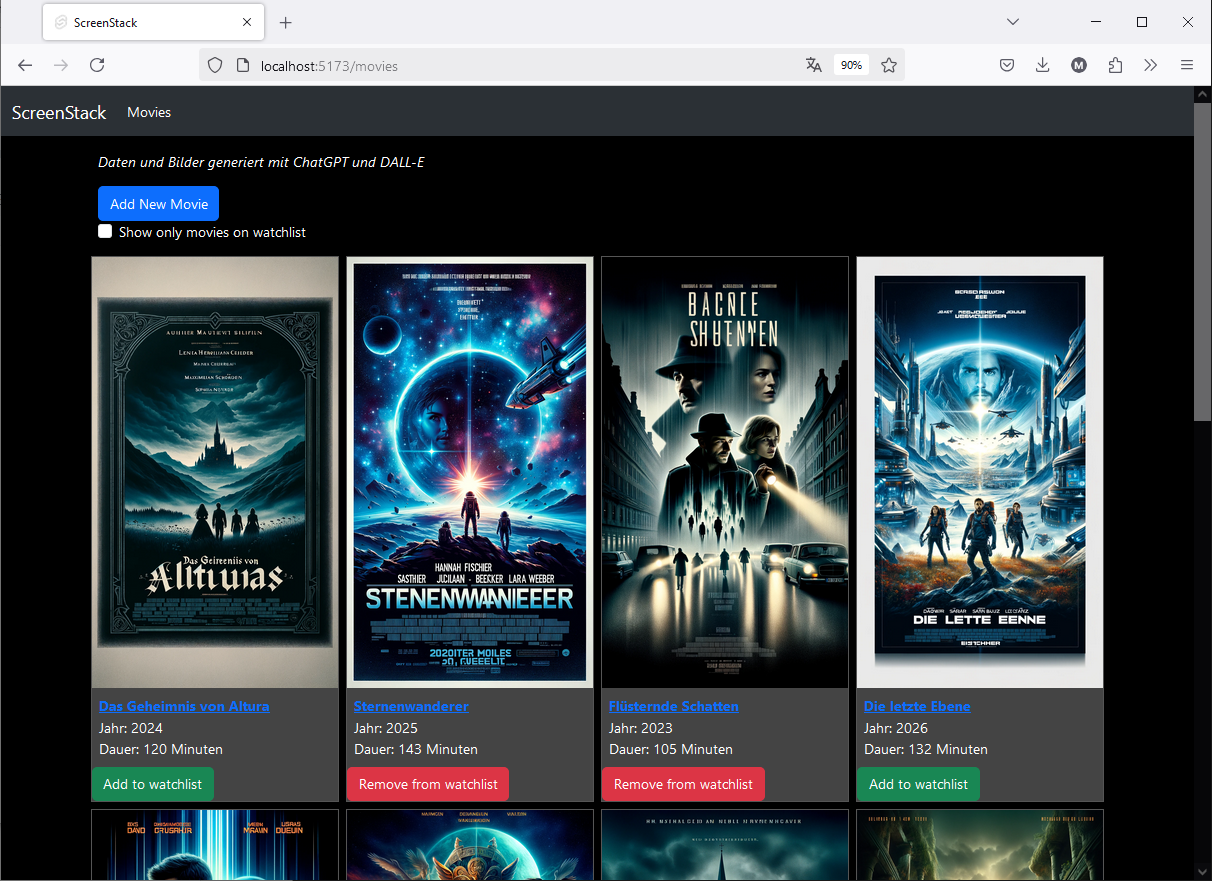
## Beschreibung der Anwendung

*[Beschreibung der einzelnen Pages und Funktionen inklusive Screenshots. Die Funktionalität und die Workflows müssen anhand der Screenshots und Textbeschreibungen nachvollziehbar sein. Der Code muss nicht beschrieben werden.]*

*Beispiel:*

### Filmübersicht

Route: /books

**

Auf dieser Seite werden alle Filme aufgelistet. Für die Darstellung der einzelnen Filme wird die Komponente MovieCard.svelte verwendet. Jeder Film kann zur Watchlist hinzugefügt oder von dieser entfernt werden. Per Checkbox «Show only movies on watchlist» können die Filme ausgeblendet werden, die nicht auf der Watchlist stehen. Über den Button «Add New Movie» gelangt man zur Seite /movies/create.

Dateien:

* lib/components/MovieCard.svelte
* routes/movies/+page.svelte
* routes/movies/+page.server.js

## Erweiterungen

*[Liste der Erweiterungen, die über die Grundanforderungen hinausgehen, siehe Punkt «Erweiterungen» in der Aufgabenstellung. Achten Sie auf Vollständigkeit. Es werden nur die hier beschriebenen Erweiterungen bewertet. Der Code muss grundsätzlich nicht beschrieben werden, aber es sollten Hinweise gegeben werden, wo die Erweiterung implementiert wurde (Dateiname, Funktionsname, etc.). Screenshots sind nur nötig, falls die Erweiterung nicht unter «Beschreibung der Anwendung» beschrieben wurde.]*

*Beispiel:*

### Watchlist-Filter

Auf der Seite /movies können via Checkbox Filme ausgeblendet werden, die nicht auf der Watchlist stehen. Dies wurde mit einem Formular und einer MongoDB-Query in der Funktion getMoviesOnWatchlist() in der Datei lib/db.js realisiert.

Datei(en):

* lib/db.js
* routes/movies/+page.svelte
* routes/movies/+page.server.js

### Dark Mode Filter