

Prototyping Projektdokumentation

Name: Thinley Sheda

E-Mail: shedathi@students.zhaw.ch

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

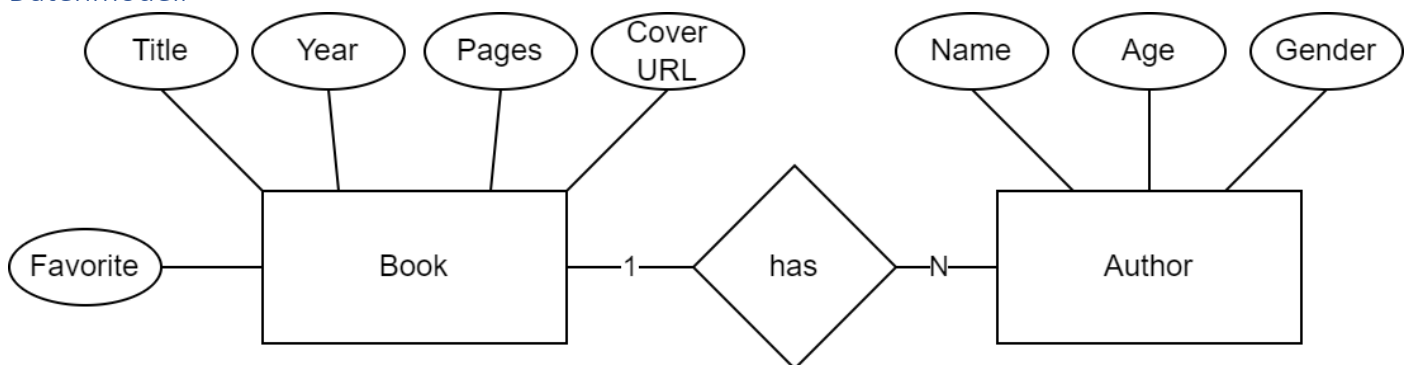
Einleitung

ShedaDB ist eine moderne, benutzerfreundliche Plattform, die es Leserinnen und Lesern ermöglicht, ihre Buchsammlung sowie Informationen zu Autorinnen und Autoren effizient zu verwalten und zu entdecken. Der Grundgedanke von ShedaDB besteht darin, Literaturbegeisterten eine umfassende Lösung zu offerieren, die nicht nur die Organisation von Büchern und Autorinnen und Autoren, sondern auch die Generierung neuer Favoriten sowie die Bereicherung des Leseerlebnisses ermöglicht.

Zu den Kernfunktionen gehört das Buchmanagement, bei dem Nutzende Bücher mit Details wie Titel, Autor, Erscheinungsjahr und Seitenzahl erstellen und verwalten können. Darüber hinaus ermöglicht ShedaDB die Verwaltung von Autorenprofilen, die mit Büchern verlinkt werden können, sowie das Erstellen von Favoritenlisten, um geschätzte Bücher und Autoren schnell wiederzufinden. Eine leistungsstarke Such- und Filterfunktion erleichtert das Auffinden von Büchern oder Autorinnen und Autoren nach Titel, Jahr oder Namen und sorgt dafür, dass wichtige Informationen immer griffbereit sind. Zusätzlich bietet die Plattform eine Dark-Mode-Funktion, die zwischen Light- und Dark-Mode wechseln lässt, um eine komfortable Nutzung zu jeder Tageszeit zu gewährleisten.

Die Plattform eignet sich sowohl für Gelegenheitsleserinnen und -leser als auch für passionierte Bücherliebhaberinnen und -liebhaber und bietet eine nahtlose Möglichkeit, die eigene Sammlung stetig zu erweitern. ShedaDB ermöglicht es den Nutzenden, eine strukturierte Übersicht ihrer Bücher und Autoren zu schaffen.

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

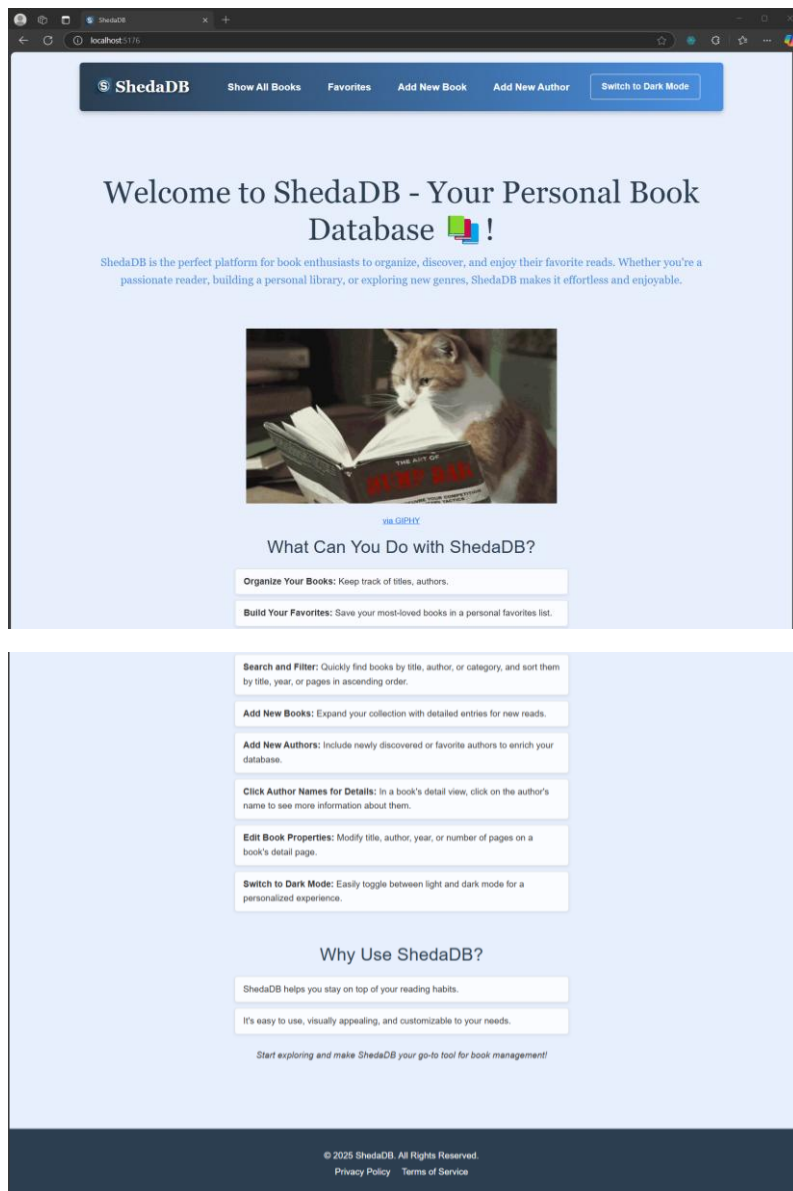
- **Age:** Das Alter des Autors
- **Gender:** Das Geschlecht des Autors

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

Hauptseite

Route: /



Die Hauptseite von ShedaDB begrüßt die Nutzenden mit einem einladenden Header und einem zentral platzierten animierten Buch-GIF, welches der Plattform eine lebendige Wirkung verleiht. Eine klar strukturierte Übersicht der Funktionen demonstriert, wie ShedaDB Buchliebhaberinnen und -liebhaber dabei unterstützt, ihre Sammlungen effizient zu organisieren. Die Nutzende können Bücher und Autorinnen und Autoren verwalten, Favoriten speichern sowie mit leistungsstarken Such- und Filterfunktionen effizient Informationen selektieren. Zusätzlich bietet die Plattform eine intuitive Möglichkeit, neue Bücher und Autoren hinzuzufügen oder bestehende Einträge zu bearbeiten. Durch die Option, zwischen Light- und Dark-Mode zu wechseln, passt sich ShedaDB perfekt den individuellen Vorlieben an und optimiert das Nutzererlebnis.

Dateien

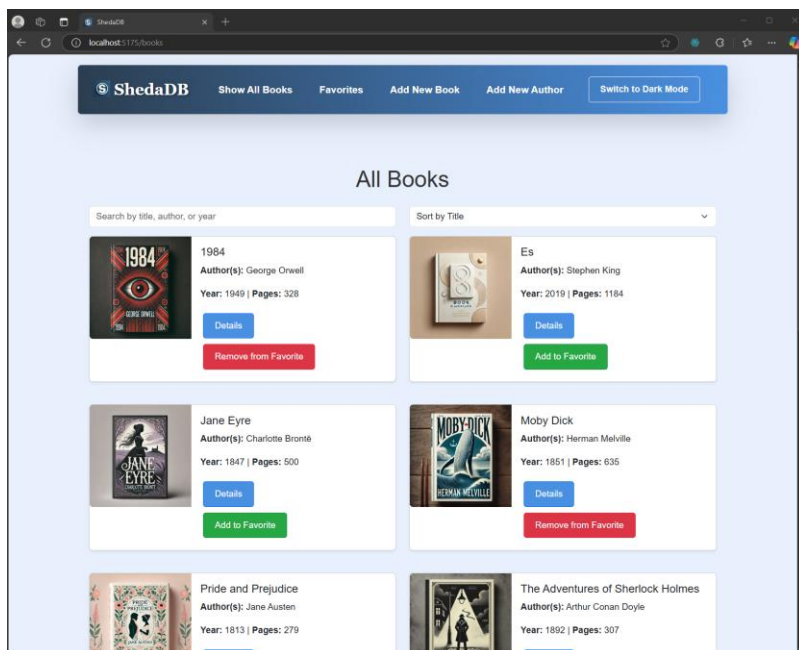
- **routes/+page.svelte:** Beinhaltet die Struktur und den Inhalt der Hauptseite.
- **svelte/transition:** Bietet die fly-Animation, die verwendet wird, um Elemente sanft mit einer Bewegung und Transparenzänderung einzublenden.
- **svelte/easing:** Liefert die backOut-Easing-Funktion, die der Animation einen schwungvollen Effekt am Ende verleiht.

Workflow

- Der Nutzer öffnet die Anwendung und landet auf der Hauptseite.
- Durch die animierte Darstellung erhält er einen Überblick über die Möglichkeiten der Plattform.
- Von hier aus kann der Nutzer über die Navigationsleiste zu weiteren Seiten wechseln, wie z. B. /books oder /books/create.

Buchübersicht

Route: /books, Navigationsleiste: Show All Books



Auf dieser Seite erfolgt eine übersichtliche Darstellung der Bücher, die durch eine Such-, Filter- und Sortierfunktion ergänzt wird. Für die Darstellung jedes einzelnen Buches wird die Komponente BookCard.svelte verwendet, die Informationen wie Titel, Autor(en), Erscheinungsjahr und Seitenanzahl enthält. Die Favoritenliste ermöglicht es, Bücher zu Ihren Favoriten hinzuzufügen oder von dieser Liste zu entfernen. Zusätzlich ermöglicht die Suchleiste das gezielte Auffinden von Büchern anhand von Titel, Autor(en) oder Erscheinungsjahr. Darüber hinaus besteht für die Nutzende die Möglichkeit, die Bücherliste nach Titel, Jahr oder Seitenanzahl aufsteigend zu sortieren. Durch Anklicken des Buttons "Details" wird der Nutzer zur Detailansicht des jeweiligen Buches weitergeleitet.

Dateien:

- **lib/components/BookCard.svelte:** Zeigt die Informationen und Aktionen für ein einzelnes Buch an.
- **routes/books/+page.svelte:** Lädt und rendert alle Bücher als Liste, integriert Such-, Filter- und Sortierfunktionen und nutzt die BookCard-Komponente.
- **routes/books/+page.server.js:** Lädt die Daten aus der Datenbank und definiert Aktionen zum Hinzufügen und Entfernen von Büchern aus den Favoriten.
- **svelte/store:** Stellt writable und derived zur Verfügung, die hier verwendet werden, um den Suchbegriff und die Sortierkriterien zu verwalten sowie die Bücherliste dynamisch zu filtern und zu sortieren.

Funktionen:

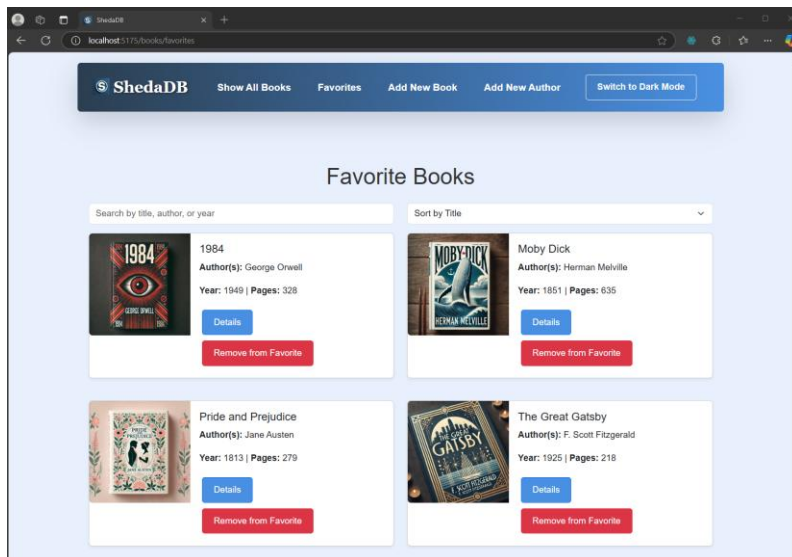
1. **Bücher anzeigen:** Alle Bücher werden aus der Datenbank geladen und in einer Kartenansicht dargestellt.
2. **Such- und Filterfunktion:** Nutzende können gezielt nach Büchern suchen, z. B. nach Titel, Autorinnen oder Erscheinungsjahr. Die Ergebnisse werden in Echtzeit gefiltert.
3. **Sortierfunktion:** Bücher können nach Titel, Erscheinungsjahr oder Seitenanzahl sortiert werden, um die Navigation zu erleichtern.
4. **Favoriten verwalten:** Über „Add to Favorite“ und „Remove from Favorite“ können Bücher direkt zur Favoritenliste hinzugefügt oder entfernt werden.
5. **Details anzeigen:** Der „Details“-Button führt zur Detailansicht eines Buches, in der weitere Informationen verfügbar sind.
6. **Leerer Zustand:** Wenn keine Bücher vorhanden sind, wird der Hinweis „No books found.“ angezeigt.

Workflows:

- **Bücher anzeigen:** Beim Laden der Seite werden alle Bücher aus der Datenbank abgerufen und in einer übersichtlichen Kartenansicht dargestellt.
- **Suche und Filterung:** Die Nutzende können mithilfe der Suchleiste gezielt nach Büchern suchen. Die Ergebnisse werden in Echtzeit aktualisiert, ohne dass die Seite neu geladen wird.
- **Sortierung:** Über ein Dropdown-Menü können Bücher nach Titel, Jahr oder Seitenanzahl sortiert werden. Die Sortierung wird dynamisch angewendet.
- **Favoriten verwalten:** Mithilfe der Buttons können Bücher als Favoriten markiert oder daraus entfernt werden. Änderungen werden sofort gespeichert.
- **Details anzeigen:** Ein Klick auf den „Details“-Button führt die Nutzende zur Detailseite des ausgewählten Buches, um weitere Informationen einzusehen.

Favoriten

Route: /books/favorites, Navigationsleiste: Show All Favorites



Auf dieser Seite erfolgt die Darstellung sämtlicher Bücher, die von Nutzenden als Favoriten markiert wurden. Die einzelnen Bücher werden mittels der BookCard.svelte-Komponente dargestellt, die Informationen wie Titel, Autoren, Erscheinungsjahr und Seitenanzahl sowie einen Button zum Entfernen von Favoriten enthält. Bei Abwesenheit von Favoriten wird ein entsprechender Hinweistext angezeigt.

Funktionen:

1. **Favoriten anzeigen:** Nur Bücher, die mit favorite: true markiert sind, werden auf dieser Seite dargestellt. Die Favoritenbücher werden übersichtlich in Kartenform angezeigt.

2. **Von Favoriten entfernen:** Nutzende können Bücher direkt von der Favoritenliste entfernen, indem sie den Button „Remove from favorite“ in der jeweiligen Buchkarte klicken. Die Änderung wird serverseitig verarbeitet, und die Ansicht wird automatisch aktualisiert.
3. **Leerer Zustand:** Wenn keine Favoriten vorhanden sind, wird der Hinweis „You have no favorite books yet. Start adding some!“ angezeigt.

Dateien:

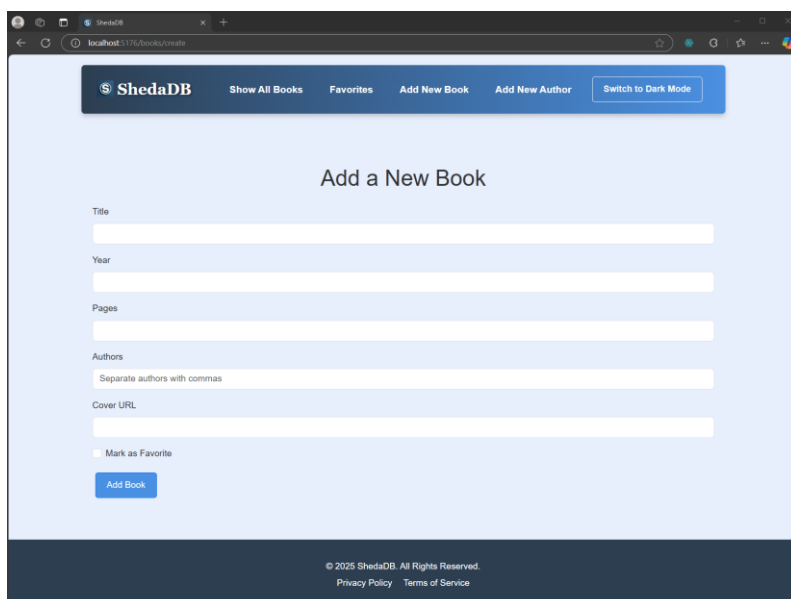
- **lib/components/BookCard.svelte:** Stellt die Details eines Buches dar und bietet den Button „Remove from favorite“ für die Aktion.
- **routes/books/favorites/+page.svelte:** Rendert die Favoritenliste und zeigt entweder die Bücher oder den Hinweistext an.
- **routes/books/favorites/+page.server.js:** Lädt die Favoritenbücher aus der Datenbank und definiert die Action, um Bücher aus der Favoritenliste zu entfernen.
- **svelte/store:** Stellt writable und derived zur Verfügung, die hier verwendet werden, um den Suchbegriff und die Sortierkriterien zu verwalten sowie die Bücherliste dynamisch zu filtern und zu sortieren.

Workflows:

- **Favoriten laden:** Beim Aufruf der Seite werden alle Favoritenbücher aus der Datenbank abgerufen und in der Kartenansicht angezeigt.
- **Bücher durchsuchen und sortieren:** Nutzerinnen können die Favoritenbücher nach Titel, Autorinnen oder Jahr durchsuchen und sortieren. Die Ergebnisse werden dynamisch gefiltert und aktualisiert.
- **Favoriten verwalten:** Mit dem Button „Remove from favorite“ können Bücher aus der Favoritenliste entfernt werden. Die Änderungen werden serverseitig gespeichert, und die Liste aktualisiert sich automatisch.
- **Hinweistext anzeigen:** Wenn keine Favoriten vorhanden sind, wird ein Hinweistext angezeigt, um die Nutzer*innen zu motivieren, Favoriten hinzuzufügen.

Buch erstellen

Route: /books/create, Navigationsleiste: Add New Book



The screenshot shows a web browser window displaying the 'Add a New Book' form. The browser's address bar shows 'localhost:5173/books/create'. The application's navigation bar includes the 'ShedaDB' logo and links for 'Show All Books', 'Favorites', 'Add New Book', 'Add New Author', and a 'Switch to Dark Mode' button. The form itself has a title 'Add a New Book' and several input fields: 'Title', 'Year', 'Pages', 'Authors' (with a note 'Separate authors with commas'), and 'Cover URL'. There is also a checkbox labeled 'Mark as Favorite' and an 'Add Book' button at the bottom. The footer of the application shows the copyright notice '© 2025 ShedaDB. All Rights Reserved.' and links to 'Privacy Policy' and 'Terms of Service'.

Auf dieser Seite besteht für Nutzern die Möglichkeit, ein neues Buch hinzuzufügen. Zu diesem Zweck ist das Ausfüllen eines benutzerfreundlichen Formulars erforderlich. Die CreateBook.svelte-Komponente offeriert eine übersichtliche Möglichkeit, Details wie Titel, Erscheinungsjahr, Seitenanzahl, Autorinnen und ein optionales Cover-Bild einzugeben. Darüber hinaus besteht die Möglichkeit, ein Buch direkt als Favorit zu markieren.

Funktionen:

1. **Buch hinzufügen:** Nutzer können über das Formular ein neues Buch mit den erforderlichen Details wie Titel, Jahr, Seitenanzahl und Autoren erstellen.
2. **Direkte Favoriten-Markierung:** Über die Checkbox „Mark as Favorite“ kann ein Buch direkt beim Erstellen als Favorit hinzugefügt werden.
3. **Validierung:** Alle Pflichtfelder (Titel, Jahr, Seitenanzahl, Autoren) sind mit der required-Eigenschaft versehen, um unvollständige Eingaben zu verhindern.
4. **Erfolgs-/Fehlermeldungen:** Nach dem Absenden des Formulars zeigt die Seite eine grüne Erfolgsmeldung oder eine rote Fehlermeldung, falls das Erstellen nicht erfolgreich war.

Dateien:

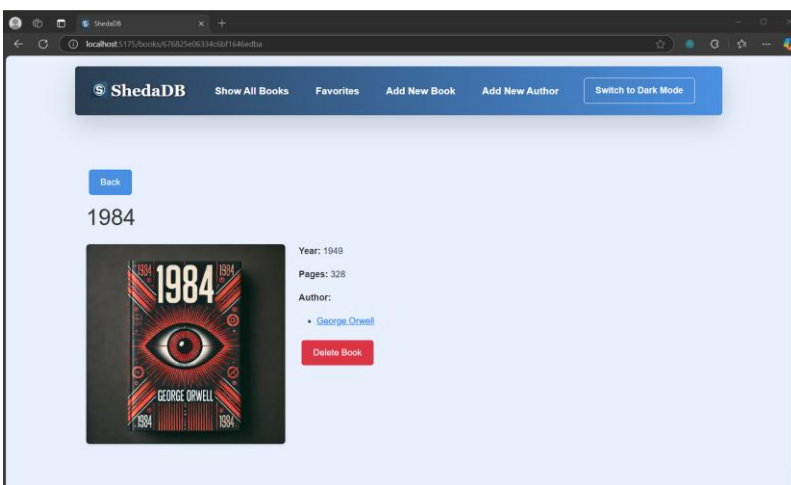
- **lib/components/CreateBook.svelte:** Die Komponente enthält das Formular und kümmert sich um die Eingabevalidierung und Anzeige von Meldungen.
- **routes/books/create/+page.svelte:** Bindet die CreateBook-Komponente ein und übergibt Erfolgs- oder Fehlermeldungen an die Komponente.
- **routes/books/create/+page.server.js:** Enthält die serverseitige Action, um die Formulardaten entgegenzunehmen und ein neues Buch in der Datenbank zu erstellen.

Workflow:

- **Navigation zur Seite:** Nutzende navigieren zur Seite /books/create, um ein neues Buch hinzuzufügen.
- **Formulärausfüllung:** Das Formular wird angezeigt, und die Nutzerinnen können die erforderlichen Felder wie Titel, Jahr, Seitenanzahl und Autorinnen ausfüllen. Optional kann ein Cover-Bild hinzugefügt und das Buch als Favorit markiert werden.
- **Absenden der Daten:** Nach dem Absenden werden die Daten per POST an die Action ?/create gesendet.
- **Verarbeitung der Daten:** Die serverseitige Action speichert die Daten in der Datenbank und gibt eine entsprechende Erfolgsmeldung oder einen Fehler zurück.
- **Anzeige von Meldungen:** Die Seite zeigt eine grüne Erfolgsmeldung oder eine rote Fehlermeldung und bleibt für weitere Eingaben offen.

Buchdetails

Route: /books/[book_id]



Update Book

Title
1984

Year
1949

Pages
328

Authors
George Orwell

Cover URL
/images/1984.png

Save Changes

© 2025 SheduDB. All Rights Reserved.
Privacy Policy Terms of Service

Auf dieser Seite werden die Details eines ausgewählten Buches angezeigt, einschliesslich Titel, Erscheinungsjahr, Seitenanzahl, Autorinnen und Autoren und Cover-Bild. Neben der Darstellung der Informationen besteht für die Nutzenden die Option, das Buch zu bearbeiten oder zu löschen. Die Autorinnen und Autoren sind als anklickbare Links dargestellt, die zu den individuellen Detailseiten der Autorinnen und Autoren führen.

Funktionen:

- Buchdetails anzeigen:** Die Seite lädt die Daten des ausgewählten Buches und stellt sie übersichtlich dar, einschliesslich Titel, Jahr, Seitenanzahl, Autoren und Cover.
- Autoren verlinken:** Die Autoren des Buches werden als anklickbare Links angezeigt. Jeder Link führt zur Detailseite des jeweiligen Autors (/authors/[name]).
- Buch bearbeiten:** Ein Formular erlaubt die Aktualisierung von Titel, Jahr, Seitenanzahl, Autoren und Cover-Bild des Buches. Nach dem Speichern wird eine Erfolgsmeldung angezeigt.
- Buch löschen:** Über einen Button kann das Buch direkt gelöscht werden. Die Aktion wird serverseitig durchgeführt, und der Nutzer wird nach dem Löschen zur Übersicht aller Bücher (/books) weitergeleitet.

Dateien:

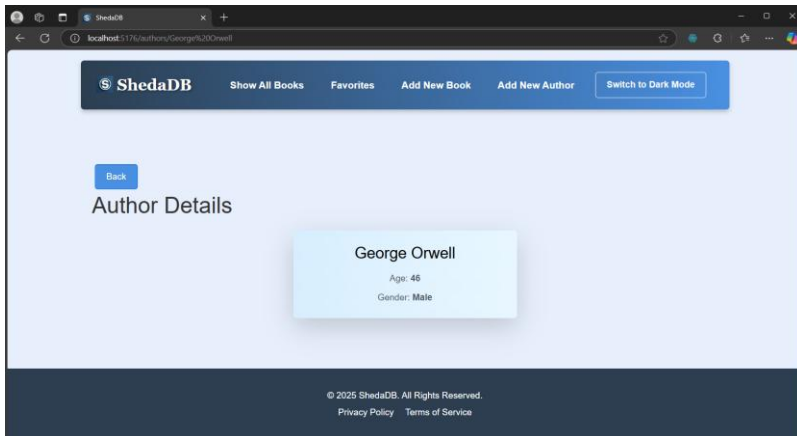
- routes/books/[book_id]/+page.svelte:** Rendert die Buchdetails, enthält die Logik für die Anzeige der Autorenlinks sowie die Formulare für das Bearbeiten und Löschen eines Buches.
- routes/books/[book_id]/+page.server.js:** Lädt die Buchdetails aus der Datenbank und enthält die serverseitigen Actions für das Aktualisieren und Löschen eines Buches.

Workflow:

- Navigation zur Detailseite:** Der Nutzer wählt ein Buch in der Übersicht unter /books aus und wird zur Detailseite (/books/[book_id]) weitergeleitet.
- Datenanzeige:** Die Seite lädt die Buchdetails aus der Datenbank und zeigt diese in einer übersichtlichen Ansicht an. Die Autorinnen und Autoren des Buches werden als Links angezeigt, die zu deren Detailseiten führen.
- Bearbeitung des Buches:** Über ein Formular können die Details des Buches aktualisiert und gespeichert werden. Eine Erfolgsmeldung wird nach erfolgreicher Aktualisierung angezeigt.
- Löschen eines Buches:** Mit dem Button „Delete Book“ wird das Buch gelöscht. Nach erfolgreichem Löschen wird der Nutzer zurück zur Buchübersicht (/books) weitergeleitet.

Autorendetails

Route: /authors/[name]



Auf dieser Seite werden die Details eines spezifischen Autors präsentiert, einschliesslich Namen, Alter und Geschlecht. Die Visualisierung der Daten erfolgt durch die Verwendung der wiederverwendbaren `AuthorCard.svelte`-Komponente. Durch die Integration einer zusätzlichen Entität (Autoren) und einer detaillierten Anzeige, die über eine eigens dafür entwickelte Route zugänglich ist, werden die grundlegenden Anforderungen erweitert.

Funktionen:

- Autorendetails anzeigen:** Die Seite lädt die Daten eines spezifischen Autors aus der Datenbank und präsentiert diese in der `AuthorCard`-Komponente. Angezeigt werden Name, Alter und Geschlecht des Autors.
- Navigation zurück:** Über den Button „Back“ können Nutzer bequem zur vorherigen Seite zurückkehren, z. B. zur Buchdetailansicht oder zur Übersicht.
- Fehlerbehandlung:** Wenn der angeforderte Autor nicht gefunden wird, wird der Nutzer entweder zur Buchübersicht (`/books`) oder auf eine "Not Found"-Seite weitergeleitet.

Dateien:

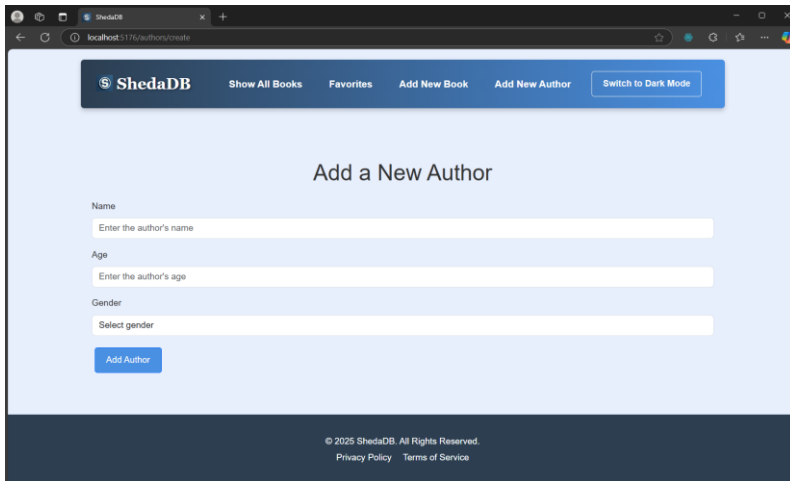
- `lib/components/AuthorCard.svelte`:** Wiederverwendbare Komponente zur Darstellung von Autoreninformationen (Name, Alter, Geschlecht).
- `routes/authors/[name]/+page.svelte`:** Rendert die Detailseite eines Autors und bindet die `AuthorCard`-Komponente ein.
- `routes/authors/[name]/+page.server.js`:** Lädt die Autorendaten aus der Datenbank und behandelt Fehler wie fehlende Datensätze.

Workflow:

- Navigation zur Detailseite:** Nutzende klicken in der Buchdetailansicht (`/books/[book_id]`) auf den Namen eines Autors. Dies führt sie zur Autorendetailseite unter `/authors/[name]`.
- Datenabruf:** Die Seite lädt die Daten des Autors mithilfe der Serverfunktion aus der Datenbank.
- Anzeige der Details:** Die `AuthorCard`-Komponente zeigt die Informationen des Autors übersichtlich an.
- Fehlerbehandlung:** Falls der Autor nicht existiert, wird eine Umleitung zur Buchübersicht oder auf eine „Not Found“-Seite vorgenommen.

Autor erstellen

Route: `/authors/create`, Navigationsleiste: Add New Author



Auf dieser Seite besteht für Nutzerinnen und Nutzer die Möglichkeit, neue Autorinnen hinzuzufügen. Zu diesem Zweck steht ihnen ein benutzerfreundliches Formular zur Verfügung, das sie durch Ausfüllen nutzen können. Die `CreateAuthor.svelte`-Komponente bietet ein intuitives Interface, um Details wie Name, Alter und Geschlecht einzugeben. Nach dem Absenden des Formulars verarbeitet die Seite die Eingaben und gibt eine entsprechende Erfolgs- oder Fehlermeldung zurück.

Funktionen:

1. **Autor hinzufügen:** Nutzende können einen neuen Autor erstellen, indem sie ein Formular ausfüllen und absenden. Erforderliche Angaben sind Name, Alter und Geschlecht.
2. **Validierung:** Alle Felder sind Pflichtfelder. Die Eingaben werden validiert, um sicherzustellen, dass vollständige und gültige Daten übermittelt werden.
3. **Erfolgs-/Fehlermeldungen:** Eine grüne Erfolgsmeldung zeigt an, wenn der Autor erfolgreich erstellt wurde. Bei fehlerhaften Eingaben wird eine rote Fehlermeldung angezeigt, um die Nutzenden auf das Problem hinzuweisen.

Dateien:

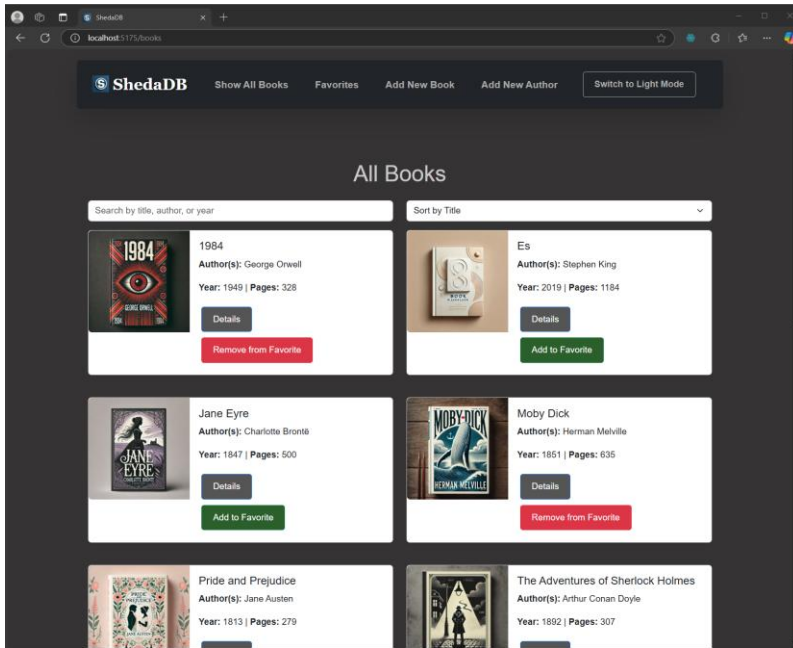
- **lib/components/CreateAuthor.svelte:** Diese Komponente enthält das Formular, kümmert sich um die Validierung und zeigt Erfolgs-/Fehlermeldungen an.
- **routes/authors/create/+page.svelte:** Rendert die `CreateAuthor`-Komponente und übergibt Erfolgs- oder Fehlermeldungen an sie.
- **routes/authors/create/+page.server.js:** Enthält die serverseitige Action, um die Formulardaten entgegenzunehmen und den neuen Autor in der Datenbank zu speichern.

Workflow:

- **Navigation zur Seite:** Nutzende navigieren zur Seite `/authors/create`, um einen neuen Autor hinzuzufügen.
- **Formularausfüllung:** Das Formular wird angezeigt und kann mit den erforderlichen Informationen (Name, Alter, Geschlecht) ausgefüllt werden.
- **Absenden der Daten:** Nach dem Absenden werden die Formulardaten per POST an die Action `?/create-author` gesendet.
- **Verarbeitung der Daten:** Die serverseitige Action speichert die Daten in der Datenbank und gibt eine Erfolgsmeldung oder einen Fehler zurück.
- **Anzeigen der Meldung:** Die Seite zeigt die entsprechende Meldung an (grün für Erfolg, rot für Fehler) und bleibt für weitere Eingaben offen.

Erweiterungen

Dark Mode



Der sogenannte Dark Mode ermöglicht es den Nutzerinnen und Nutzern, die Farbgestaltung der Anwendung in einen dunklen Modus zu ändern. Dies verbessert die Lesbarkeit in dunklen Umgebungen und schonen die Augen. Die Einstellung wird im Browser gespeichert, sodass sie auch nach einem Neuladen der Seite erhalten bleibt. Die Implementierung verbindet Svelte-Stores, localStorage und CSS-Overrides, um eine reibungslose und benutzerfreundliche Erfahrung zu gewährleisten.

Funktionen:

1. Umschalten des Dark Modes:

- Nutzer können über einen Button in der Navigationsleiste zwischen Light Mode und Dark Mode wechseln.
- Der Button zeigt dynamisch an, welcher Modus aktiv ist, und ermöglicht ein nahtloses Umschalten.

2. Persistenz der Einstellung:

- Der Dark Mode wird im localStorage des Browsers gespeichert, sodass die Einstellung auch nach einem Seitenneuladen oder erneutem Öffnen der Anwendung erhalten bleibt.

Implementierung:

- State Management mit Svelte Store:**
 - Datei:** lib/stores/theme.js
 - Enthält den darkMode-Store, der den aktuellen Zustand des Dark Modes verwaltet.
- Integration in die Navbar:**
 - Datei:** lib/components/Navbar.svelte
 - Der Dark Mode wird mit Hilfe von onMount und dem darkMode-Store verwaltet.
 - Der Button „Switch to Light/Dark Mode“ erlaubt das Umschalten des Modus, aktualisiert den Zustand im darkMode-Store und speichert die Einstellung im localStorage.
- CSS-Overrides für den Dark Mode:**
 - Datei:** styles.css
 - Die Klasse .dark-mode wird auf das <body>-Element angewendet und überschreibt die Standardstile.
 - Anpassungen umfassen Änderungen für Hintergrundfarben, Textfarben, Buttons, Navbar und andere UI-Elemente.

Workflow:

- Seitenladen:**

- Beim Öffnen der Anwendung wird der gespeicherte Modus aus dem localStorage ausgelesen.
- Die CSS-Klasse .dark-mode wird basierend auf der gespeicherten Einstellung auf das <body>-Element angewendet.
- **Umschalten des Modus:**
 - Beim Klick auf den Button „Switch to Light/Dark Mode“ wird der darkMode-Store aktualisiert.
 - Die CSS-Klasse .dark-mode wird dynamisch hinzugefügt oder entfernt.
- **Speichern der Einstellung:**
 - Der aktuelle Zustand des Dark Modes wird im localStorage gespeichert, um die Einstellung dauerhaft zu sichern.

Dateien:

- **lib/stores/theme.js:**
 - Enthält den Svelte Store darkMode zur Verwaltung des Modus.
- **lib/components/Navbar.svelte:**
 - Implementiert die Logik und den Button zum Umschalten zwischen Light und Dark Mode.
- **styles.css:**
 - Enthält die CSS-Overrides für die Klasse .dark-mode, einschliesslich Anpassungen für Hintergrund, Textfarben und UI-Elemente.

Such- und Filterfunktion

Sort by Title ▾

Die Seite /books und /books/favorites wurde mit einer leistungsstarken Such- und Filterfunktion erweitert, die es Nutzern ermöglicht, gezielt nach Büchern zu suchen und die Ergebnisse dynamisch zu sortieren. Mit dieser Erweiterung wird die Navigation durch grosse Buchsammlungen vereinfacht und die Benutzerfreundlichkeit erhöht.

Funktionen:

1. **Suchfunktion:**
 - a. Nutzer können nach Büchern suchen, indem sie Titel, Autoren oder Erscheinungsjahr in ein Suchfeld eingeben.
 - b. Die Ergebnisse werden in Echtzeit gefiltert, sodass nur die relevanten Bücher angezeigt werden.
 - c. Durch die Implementierung eines Debouncing-Mechanismus wird die Leistung optimiert und unnötige Abfragen vermieden.
2. **Filterfunktion:**
 - a. Die Ergebnisse können dynamisch nach verschiedenen Kriterien sortiert werden:
 - i. Titel (alphabetisch)
 - ii. Erscheinungsjahr (aufsteigend)
 - iii. Seitenanzahl (aufsteigend)
 - b. Änderungen an den Sortioptionen werden sofort angewendet.
3. **Benutzerfreundliche Rückmeldungen:**
 - a. Falls keine Bücher den Suchkriterien entsprechen, wird eine Meldung mit „No books found.“ angezeigt, um den Nutzer zu informieren.

Dateien:

- **routes/books/+page.svelte:**
 - Hier wurde die Such- und Filterlogik implementiert. Die Suchabfragen und Sortierkriterien werden als reaktive Stores verwaltet. Die Ergebnisse werden dynamisch aktualisiert und in der BookCard-Komponente angezeigt.
 - Die Debouncing-Funktion sorgt dafür, dass die Suchanfragen performant bleiben.
- **lib/components/BookCard.svelte:**

- Zeigt die Informationen zu den einzelnen Büchern an und bleibt für die Anzeige der gefilterten Ergebnisse unverändert.

Workflow:

- **Seite laden:** Beim Laden der Seite werden alle Bücher aus der Datenbank angezeigt.
- **Suchabfrage:** Der Nutzer gibt ein Schlüsselwort in das Suchfeld ein (z. B. Titel, Autor oder Jahr). Die Suchergebnisse werden dynamisch gefiltert.
- **Filterung und Sortierung:** Der Nutzer kann die Sortierkriterien über ein Dropdown-Menü ändern. Die Ergebnisse werden in Echtzeit nach den gewählten Kriterien sortiert.
- **Anzeige der Ergebnisse:** Die gefilterten und sortierten Bücher werden in der BookCard-Komponente angezeigt. Wenn keine Ergebnisse vorliegen, wird eine entsprechende Meldung eingeblendet.

Hinweise zur Implementierung:

- **Such- und Filterlogik:**
 - Implementiert in `routes/books/+page.svelte` mithilfe der `writable`- und `derived`-Stores aus Svelte.
 - Die Funktion `debounceSearch()` optimiert die Eingabeverarbeitung durch eine Verzögerung von 300ms.
- **Reaktive Anzeige:**
 - Die Filter- und Sortierlogik wird über einen `derived`-Store realisiert, der die Bücher dynamisch filtert und sortiert.

Autorendetails anzeigen

Auf der Seite `/authors/[name]` werden die Details eines spezifischen Autors (Name, Alter, Geschlecht) dargestellt. Falls der Autor nicht existiert, wird der Nutzer auf eine alternative Seite weitergeleitet.

- **Implementierung:**
 - **Dateien:**
 - **`lib/components/AuthorCard.svelte`:** Wiederverwendbare Komponente zur Anzeige von Autoreninformationen.
 - **`routes/authors/[name]/+page.svelte`:** Rendert die Detailseite des Autors und verwendet die `AuthorCard`-Komponente.
 - **`routes/authors/[name]/+page.server.js`:** Lädt die Autorendaten aus der Datenbank und behandelt Fehler.
- **Workflow:**
 - Der Nutzer klickt auf einen Autorennamen in der Buchdetailansicht (`/books/[book_id]`) und wird zur Detailseite des Autors weitergeleitet.
 - Die Informationen des Autors werden aus der Datenbank geladen und angezeigt.

Autor erstellen

Auf der Seite `/authors/create` können Nutzer über ein Formular neue Autoren hinzufügen.

- **Implementierung:**
 - **Dateien:**
 - **`lib/components/CreateAuthor.svelte`:** Enthält das Formular, kümmert sich um die Validierung und zeigt Erfolgs-/Fehlermeldungen an.
 - **`routes/authors/create/+page.svelte`:** Rendert die `CreateAuthor`-Komponente und übergibt Erfolgs-/Fehlermeldungen.
 - **`routes/authors/create/+page.server.js`:** Verarbeitet die Formulardaten serverseitig und speichert den neuen Autor in der Datenbank.
 - **Workflow:**

- Nutzer navigiert zur Seite `/authors/create`.
- Ein Formular wird angezeigt, um Name, Alter und Geschlecht des Autors einzugeben.
- Nach dem Absenden werden die Daten serverseitig validiert und in der Datenbank gespeichert.

Buch bearbeiten

Die Detailseite eines Buches (`/books/[book_id]`) enthält ein eingebettetes Formular, um die Buchdetails zu aktualisieren.

- **Implementierung:**
 - **Dateien:**
 - **`routes/books/[book_id]/+page.svelte`:** Zeigt die Buchdetails an und enthält das Formular zur Bearbeitung.
 - **`routes/books/[book_id]/+page.server.js`:** Verarbeitet die Aktualisierung serverseitig.
 - **Workflow:**
 - Nutzer bearbeitet Felder wie Titel, Jahr, Seitenanzahl, Autoren oder Cover-URL.
 - Änderungen werden über ein Formular gesendet und serverseitig in der Datenbank aktualisiert.
 - Erfolgsnachricht wird angezeigt, sobald die Änderungen gespeichert wurden.

Komponentenstruktur

Die Anwendung besteht aus 6 wiederverwendbaren Komponenten, was über die Mindestanforderung von 5 hinausgeht.

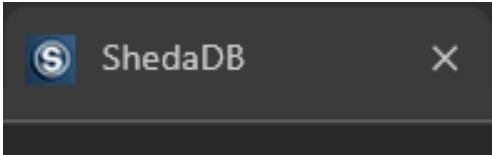
- **Komponenten:**
 - **`AuthorCard.svelte`:** Zeigt Details eines spezifischen Autors.
 - **`BookCard.svelte`:** Zeigt Details eines Buches in der Buchliste.
 - **`CreateAuthor.svelte`:** Formular zum Erstellen eines neuen Autors.
 - **`CreateBook.svelte`:** Formular zum Erstellen eines neuen Buches.
 - **`Footer.svelte`:** Fusszeile der Anwendung.
 - **`Navbar.svelte`:** Navigationsleiste mit Dark-Mode-Umschalter und Links zu verschiedenen Seiten.

Datenbankoperationen

Insgesamt wurden 8 verschiedene Datenbankfunktionen implementiert, um Bücher und Autoren zu verwalten.

- **Funktionen:**
 - **`getBooks`:** Alle Bücher abrufen.
 - **`getFavoriteBooks`:** Nur favorisierte Bücher abrufen.
 - **`getBook`:** Details eines spezifischen Buches abrufen.
 - **`createBook`:** Ein neues Buch erstellen.
 - **`updateBook`:** Buchdetails aktualisieren.
 - **`deleteBook`:** Buch löschen.
 - **`getAuthorByName`:** Details eines spezifischen Autors abrufen.
 - **`createAuthor`:** Einen neuen Autor erstellen.

Anpassung des Tab Logos



Im Rahmen der Implementierung einer Erweiterung wurde eine Anpassung des Tab-Logos (Favicon) der Anwendung vorgenommen, um ihr ein individuelles Erscheinungsbild zu verleihen. Zu diesem Zweck wurde in der Datei `app.html` im Bereich des Head-Tags ein Link-Tag implementiert, der auf die Datei `%sveltekit.assets%/images/logo.png` verweist. Das neu implementierte Logo befindet sich im Ordner `static/images` der Anwendung und wird nun als Favicon in der Browserleiste angezeigt. Diese Anpassung verbessert die visuelle Identität der Anwendung und erleichtert die Wiedererkennung, insbesondere bei geöffneten mehreren Tabs.

Animationen und Interaktivität

Der Fokus der Erweiterungen liegt auf der Implementierung von Animationen und interaktiven Effekten, die über die grundlegenden Anforderungen hinausgehen. Die implementierten Animationen dienen der Optimierung der Benutzererfahrung, wobei insbesondere Übergänge, visuelle Rückmeldungen und dynamische Inhalte integriert werden.

Dateiverweise:

1. Hauptseite (+page.svelte):

- a. Animation beim Laden von Überschriften und Texten:
- b. Transition fly mit Svelte für Überschriften und Absätze.
- c. Animationen sorgen für ein dynamisches Erscheinungsbild der Willkommensnachricht und der Feature-Beschreibung.
- d. Datei: `routes/+page.svelte`.

2. Autorenkarte (AuthorCard.svelte):

- a. Hover-Effekte:
 - i. Schatteneffekte (box-shadow) beim Überfahren der Karte und dynamische Skalierung der Karte beim Hover.
- b. Animationen für Inhalte:
 - i. fade für den Autorennamen und fly für das Geschlecht.
 - ii. scale für die Altersanzeige des Autors.
- c. Datei: `lib/components/AuthorCard.svelte`.

3. Buchkarte (BookCard.svelte):

- a. Hover-Effekte für das Cover-Bild:
- b. Rotation und Skalierung beim Hover.
- c. Animationen:
 - i. fly für Titel und Details, fade und scale für Buttons.
- d. Visuelle Rückmeldungen:
 - i. Hintergrundveränderung der Karte bei Hover und ein Übergang zu einer helleren Farbschattierung.
- e. Datei: `lib/components/BookCard.svelte`.

4. Navbar (Navbar.svelte):

- a. Dark-Mode-Integration:
 - i. Dynamische Umschaltung zwischen Light Mode und Dark Mode mit optischen Rückmeldungen.
- b. Hover-Effekte für Links:
- c. Farbanpassungen und Hintergrundhervorhebungen.
- d. Interaktivität:
 - i. Sanfte Übergänge für Buttons und Links.

e. Datei: lib/components/Navbar.svelte.

Zusätzliche Hinweise:

- **Implementierung der Animationen:**
 - Animationen wurden mithilfe der in Svelte integrierten Transition-Funktionen (fly, fade, scale) erstellt.
 - Effekte wie hover und box-shadow werden durch CSS-Transitions ergänzt.
- **Warum diese Erweiterungen sinnvoll sind:**
 - Die Animationen und interaktiven Elemente machen die Anwendung nicht nur optisch ansprechender, sondern verbessern auch die Benutzerfreundlichkeit und die Interaktion mit der Plattform.
- **Screenshots:**
 - Screenshots sind nicht erforderlich, da die Animationen direkt durch die oben aufgeführten Dateien und Funktionen beschrieben wurden.