

„LibraNova“: Full-Stack-Bibliotheksmanagementapp für die Ausleihe von Büchern

”LibraNova“ – A Full-Stack Library Management System for Lending Books

Bearbeiter: Mohammed Salih Mezraoui

Bachelor-Abschlussarbeit

Betreuer: Prof. Dr.-Ing. Georg Schneider

Trier, den 15.08.2025

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Konzeption und Umsetzung von *LibraNova*, einer modernen, webbasierten Full-Stack-Anwendung zur digitalen Verwaltung einer Bibliothek sowie zur effizienten Organisation von Buchausleihen. Ziel des Projekts ist es, sowohl Nutzer:innen als auch Administrator:innen eine intuitive, leistungsfähige und sichere Plattform bereitzustellen, die den gesamten Ausleihprozess digital unterstützt.

Die Anwendung wurde mit **Spring Boot** im Backend und **React** mit Type-Script im Frontend realisiert. Nutzer:innen können Bücher recherchieren, deren Verfügbarkeit in Echtzeit prüfen sowie Rezensionen lesen und verfassen. Administrator:innen erhalten Werkzeuge zur Verwaltung des Buchbestands und zur Kommunikation mit den Nutzer:innen.

Zur Absicherung sensibler Funktionen werden moderne Authentifizierungs- und Autorisierungsverfahren eingesetzt, darunter **JWT** und **OAuth2** via Okta. Für kostenpflichtige Bibliotheksdienste ist die **Stripe API** eingebunden, um sichere Zahlungen zu ermöglichen. Die persistente Datenhaltung erfolgt über die **MySQL-Datenbank**.

Die Anwendung ist in **deutscher und englischer Sprache** verfügbar und legt großen Wert auf **Barrierefreiheit**, um einen breiten Nutzerzugang zu gewährleisten. Im Rahmen der Entwicklung wurde besonderer Wert auf eine responsive Benutzeroberfläche, ein konsistentes UX-Design sowie die Qualitätssicherung durch automatisierte Tests mit **JUnit** und **Mockito** gelegt. *LibraNova* demonstriert, wie durch den Einsatz moderner Webtechnologien eine skalierbare und benutzerfreundliche Lösung für das Bibliotheksmanagement realisiert werden kann.

Abstract

This thesis presents *LibraNova*, a modern, web-based full-stack application designed to facilitate the digital management of a library and the efficient organization of book lending. The project aims to provide both users and administrators with an intuitive, performant, and secure platform that fully supports the lending process digitally.

The application is built using **Spring Boot** for the backend and **React** with TypeScript for the frontend. Users can search for books, check their real-time availability, and read or write reviews. Administrators are equipped with tools to manage the book inventory and communicate with users.

To secure sensitive functions, modern authentication and authorization techniques such as **JWT** and **OAuth2** via Okta are employed. For paid library services, the **Stripe API** ensures secure payment processing. Persistent data storage is handled via the **MySQL database**.

The application supports both **English and German languages** and emphasizes **accessibility** to ensure broad user access. Special attention was given to a responsive user interface, consistent UX design, and quality assurance through automated testing using **JUnit** and **Mockito**. *LibraNova* demonstrates how modern web technologies can be leveraged to create a scalable and user-friendly solution for library management.

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Ziele der Arbeit	1
2	Verwandte Arbeiten	3
2.1	Enföhrung	3
2.2	Existierende Systeme	3
2.2.1	Thalia Website	3
2.2.2	Stadtbücherei Trier	4
2.3	Beitrag von LibraNova	4
2.4	Fazit	5
3	Grundlagen	6
3.1	Back-End Technologien	6
3.1.1	Java	6
3.1.2	Spring Boot	6
3.1.3	Hibernate	7
3.1.4	REST-API	8
3.1.5	Spring Security	8

Inhaltsverzeichnis	v
3.1.6 HTTPS und SSL/TLS	8
3.1.7 Stripe API	9
3.2 Front-End Technologien	9
3.2.1 HTML/CSS	9
3.2.2 Bootstrap	10
3.2.3 TypeScript	10
3.2.4 React	10
3.2.5 i18next	11
3.3 Datenbanktechnologien	11
3.3.1 MySQL	11
3.3.2 MySQL-Workbench und SQL	11
3.4 Authentifizierungs- und Autorisierungsprotokolle	12
3.4.1 Okta	12
3.4.2 JWT	12
3.4.3 OAuth2	12
3.4.4 OpenID Connect	13
3.5 Version Control mit Git und GitHub	13
3.6 Testen und Qualitätssicherung	13
3.6.1 Unit-Tests mit JUnit 5	13
3.6.2 Mocking mit Mockito	14
3.6.3 API-Tests mit Postman	14
3.7 Modellierung mit UML	14
4 Konzept	15
4.1 Blockbild der Architektur	15

4.2 UML-Diagramme zur Konzeption	16
4.2.1 Klassendiagramm	16
4.2.2 Sequenzdiagramme	18
4.3 Anforderungen an das System	20
4.3.1 Funktionale Anforderungen	20
4.3.2 Nicht-funktionale Anforderungen	20
4.3.3 Technische Anforderungen	21
4.4 Wichtige Algorithmen (Pseudocode)	21
4.4.1 Buchausleihe	21
4.4.2 Verlängerung der Buchausleihe	22
5 Realisierung	23
5.1 Backend-Architektur	23
5.1.1 Projektinitialisierung und Startklasse	23
5.1.2 Modulare Paketstruktur des Backends	25
5.1.3 Teststrategie und Testintegration	27
5.2 Frontend-Struktur	28
5.2.1 Einstiegspunkt der React-Anwendung	28
5.2.2 React-Projektstruktur	30
5.2.3 Internationalisierung mit i18next	32
5.3 Datenbankmodell und Persistenz	35
5.3.1 Datenbankschema	35
5.3.2 Datenpersistenz mit Spring Data und Hibernate	37
6 Implementierung	39
6.1 Ausleihprozess eines Buches	39

6.1.1 Backend-Prozess	39
6.1.2 Frontend-Prozess	43
6.2 Verfahren zur Rückgabe eines Buches	45
6.2.1 Backend-Prozess	45
6.2.2 Frontend-Prozess	46
7 Beispiel	49
7.1 Startseite	49
7.1.1 Header und Navigation	49
7.1.2 Footer	50
7.1.3 Hauptteil	50
7.2 Seitenübersicht	52
7.2.1 Buchsuche	52
7.2.2 Buchseite	52
7.3 Bibliotheksaktivität	54
7.3.1 Ausleihen	54
7.3.2 Ausleihhistorie	55
7.4 Bibliotheksdienste	55
7.5 Bezahlungsseite	57
7.6 Admin-Bereich	58
7.6.1 Neues Buch hinzufügen	58
7.6.2 Bücher verwalten	58
7.6.3 Nachrichten	59
8 Anwendungsszenarien	60
8.1 Bildungs- und E-Learning-Plattformen	60

Inhaltsverzeichnis	VIII
8.2 Mobile Nutzung	61
8.3 Personalisierte Empfehlungen	61
8.4 Fernleihe und Reservierungsbenachrichtigungen	61
8.5 Reservierungsbenachrichtigungen	61
9 Resümee und Ausblick	62
Literaturverzeichnis	63
Systemanforderungen	67
Abkürzungsverzeichnis	71
Selbstständigkeitserklärung	72

Abbildungsverzeichnis

4.1	Blockbild der Systemarchitektur	15
4.2	Klassendiagramm der Anwendung <i>Libranova</i>	17
4.3	Sequenzdiagramm des Login-Vorgangs, angelehnt an [OT25a]	18
4.4	Zahlungsablauf mit Stripe, angelehnt an [MT25a]	19
5.1	Modularer Aufbau des Backends	26
5.2	Modularer Aufbau des Frontends	30
5.3	ER-Modell der Datenbank	36
7.1	Kopfzeile der App <i>Libranova</i>	49
7.2	Kopfzeile der App <i>Libranova</i>	50
7.3	Karussell und Buch-Browser-Schaltfläche	51
7.4	Karussell und Buch-Browser-Schaltfläche	51
7.5	Benutzeroberfläche der Suchseite	52
7.6	Benutzeroberfläche der Buchseite	53
7.7	Benutzeroberfläche der Ausleihseite	54
7.8	Benutzeroberfläche der Ausleihverlaufsseite	55
7.9	Benutzeroberfläche zum Versenden von Anfragen	56
7.10	Nachrichtenverlauf zwischen Nutzer und Bibliothek	56

7.11 Benutzeroberfläche bei ausstehenden Zahlungen	57
7.12 Benutzeroberfläche bei keinen offenen Zahlungen	57
7.13 Benutzeroberfläche bei keinen offenen Zahlungen	58
7.14 Benutzeroberfläche bei keinen offenen Zahlungen	59
7.15 Benutzeroberfläche bei keinen offenen Zahlungen	59

1

Einleitung

In einer zunehmend digitalen Welt wird die effiziente Verwaltung von Informationen und Ressourcen immer wichtiger. Bibliotheken – als zentrale Einrichtungen der Wissensvermittlung – stehen vor der Herausforderung, ihre Prozesse zu modernisieren und den veränderten Anforderungen der Nutzer:innen gerecht zu werden. Webbasierte Anwendungen bieten großes Potenzial, um den Zugang zu Medien, die Organisation von Ausleihen und die Kommunikation zwischen Nutzer:innen und Verwaltung zeitgemäß und benutzerfreundlich zu gestalten.

Vor diesem Hintergrund wurde die Anwendung *LibraNova* entwickelt. Ziel ist es, eine moderne Full-Stack-Webanwendung bereitzustellen, die Bibliotheksprozesse effizient, sicher und barrierefrei digitalisiert – sowohl für Nutzer:innen als auch Administrator:innen.

Im Folgenden werden die Motivation und die Ziele dieser Arbeit vorgestellt.

1.1 Motivation

LibraNova adressiert genau diese Schwachstellen: Die Anwendung soll zeigen, wie durch moderne Webtechnologien – wie **Spring Boot**, **React** und **TypeScript** – eine skalierbare, intuitive und sichere Lösung für das Bibliotheksmanagement geschaffen werden kann. Besondere Schwerpunkte liegen auf der Mehrsprachigkeit (Deutsch und Englisch), Barrierefreiheit und responsiven Benutzerführung, um eine möglichst breite Zielgruppe anzusprechen und digitale Teilhabe aktiv zu fördern.

1.2 Ziele der Arbeit

Das Ziel dieser Arbeit besteht in der Konzeption, Umsetzung und Dokumentation einer webbasierten Bibliotheksanwendung, die folgende Kernanforderungen erfüllt:

- Bereitstellung einer intuitiven Benutzeroberfläche für die Recherche, Ausleihe und Bewertung von Büchern.
- Ermöglichung einer effizienten Verwaltung des Buchbestands durch Administrator:innen.
- Integration moderner Authentifizierungs- und Autorisierungsmechanismen (**JWT**, **OAuth2** via Okta).
- Einbindung eines sicheren Zahlungssystems über die **Stripe API** für kostenpflichtige Dienste.
- Mehrsprachige Bereitstellung der Anwendung (Deutsch und Englisch).
- Berücksichtigung von Aspekten der Barrierefreiheit zur Förderung inklusiver Nutzung.
- Umsetzung eines **responsiven Designs** zur optimalen Darstellung auf verschiedenen Endgeräten (Desktop, Tablet, Smartphone).
- Sicherstellung von Softwarequalität durch automatisierte Tests mit **JUnit** und **Mockito**.

Darüber hinaus soll die Anwendung modular und erweiterbar gestaltet sein, um zukünftige Funktionalitäten problemlos integrieren zu können. Durch diese Arbeit soll ein Beitrag zur praxisnahen Entwicklung moderner Webanwendungen im Bildungs- und Informationsbereich geleistet werden.

2

Verwandte Arbeiten

In diesem Kapitel werden ausgewählte bestehende Systeme untersucht, die ähnliche Funktionen wie LibraNova bereitstellen. Ziel ist es, deren Stärken und Schwächen zu analysieren, um die Positionierung und den Mehrwert der eigenen Anwendung im Vergleich zum aktuellen Stand der Technik (*State-of-the-Art*) zu verdeutlichen.

2.1 Einführung

Die Analyse bestehender Systeme ist ein zentraler Bestandteil wissenschaftlicher Arbeiten, da sie den aktuellen Stand der Technik (*State-of-the-Art*) verdeutlicht. Durch den Vergleich mit etablierten Lösungen lassen sich Stärken, Schwächen und bestehende Lücken identifizieren. Auf dieser Grundlage können die eigenen Beiträge klar positioniert und die Motivation für die Entwicklung eines neuen Systems nachvollziehbar begründet werden.

2.2 Existierende Systeme

Dieser Abschnitt untersucht ausgewählte bestehende Systeme, die ähnliche Funktionen wie LibraNova bieten. Ziel ist es, deren Merkmale, Stärken und Schwächen darzustellen und aufzuzeigen, wie sich die eigene Anwendung im Vergleich zum aktuellen Stand der Technik (*State-of-the-Art*) positioniert.

2.2.1 Thalia Website

Beschreibung: Thalia ist ein führender Buchhändler im deutschsprachigen Raum mit einem stetig wachsenden Netz aus derzeit über 500 Filialen in Deutschland, Österreich und der Schweiz. Neben dem stationären Handel betreibt Thalia einen

umfangreichen Online-Shop und baut seine Präsenz im E-Commerce kontinuierlich aus [Gmb25a].

Funktionen: Neben dem Online-Shop bietet Thalia mit der *Lesen & Hören* App eine mobile Lösung zum Lesen und Hören von eBooks und Hörbüchern. Nutzer können ihre Titel in der tolino-Cloud speichern, offline lesen, Lesefortschritte automatisch synchronisieren sowie Schriftart, -größe und Layout individuell anpassen. Zusätzliche Funktionen wie Sammlungen und ein Nachtmodus erhöhen den Lesekomfort [Gmb25c].

Sprachverfügbarkeit: Obwohl Thalia ein breites Sortiment an Büchern in verschiedenen Sprachen (z. B. Englisch, Französisch, Spanisch) anbietet, ist die Website-Oberfläche ausschließlich auf Deutsch verfügbar. Auch manche Buchbeschreibungen sind nur auf Deutsch verfügbar, was die Navigation und Informationsbeschaffung für nicht-deutschsprachige Nutzer zusätzlich erschwert und die allgemeine Nutzerfreundlichkeit beim Online-Browsing reduziert [Gmb25b].

2.2.2 Stadtbücherei Trier

Überblick: Die Stadtbücherei Trier stellt mehr als 90.000 Medien zur Verfügung, einschließlich Bücher, Hörbücher, Musik-CDs sowie Computer- und Konsolenspiele, die sowohl vor Ort genutzt als auch ausgeliehen werden können. Auf fünf lichtdurchfluteten Etagen stehen Sitz- und Lernbereiche, PC-Arbeitsplätze, Drucker und ausleihbare Laptops zur Verfügung. Informationen zum Medienangebot sind über den Onlinekatalog verfügbar (<https://opac.trier.de/>) [Tri25].

OPAC-System: Das OPAC-System der Stadtbücherei Trier ist keine Single-Page Application (SPA). Eine Analyse mit den Browser-Entwicklertools zeigt, dass bei jeder Navigation innerhalb des OPAC – selbst beim Anwenden einfacher Filter wie der Suche nach Medien eines bestimmten Jahres – die gesamte HTML-Seite vollständig neu geladen wird, anstatt Inhalte dynamisch nachzuladen, wie es bei einer echten SPA der Fall wäre. Auch die Hauptwebsite (<https://www.stadtbumcherei-trier.de/startseite/>) ist nicht vollständig responsiv, was die Nutzerfreundlichkeit insbesondere auf mobilen Geräten einschränkt.

2.3 Beitrag von LibraNova

LibraNova adressiert die identifizierten Schwächen bestehender Systeme auf mehreren Ebenen:

- **Bilinguale Verfügbarkeit:** Die Anwendung ist vollständig auf Deutsch und Englisch verfügbar, was die Nutzerbasis deutlich erweitert und die

Zugänglichkeit für internationale oder nicht-deutschsprachige Nutzer verbessert.

- **Single-Page Application (SPA):** Dank der Verwendung von React als Frontend-Framework verhält sich LibraNova als SPA. Inhalte werden dynamisch nachgeladen, wodurch bei Navigation und Filteranwendung kein vollständiger Seiten-Reload erfolgt [Pur25]. Dies erhöht die Effizienz und verbessert die Nutzererfahrung.
- **Responsives Design:** Die gesamte Anwendung ist vollständig responsiv gestaltet, wodurch alle Funktionen auf Desktop-, Tablet- und Mobilgeräten optimal genutzt werden können.

2.4 Fazit

Die Analyse bestehender Systeme zeigt, dass Thalia und die Stadtbücherei Trier jeweils Stärken besitzen, jedoch auch wesentliche Einschränkungen aufweisen. So ist beispielsweise Thalia ausschließlich auf Deutsch verfügbar, während das OPAC-System der Stadtbücherei Trier keine Single-Page Application ist und die Hauptwebsite nicht vollständig responsiv ist. LibraNova positioniert sich im State-of-the-Art, indem es diese Schwächen adressiert: Die Anwendung ist vollständig bilingual (Deutsch und Englisch), vollständig responsiv und als SPA umgesetzt, sodass Inhalte dynamisch nachgeladen werden. Auf diese Weise werden bekannte Konzepte bestehender Plattformen übernommen, jedoch entscheidend verbessert, was die Notwendigkeit und den Mehrwert der eigenen Entwicklung von LibraNova verdeutlicht.

3

Grundlagen

In diesem Kapitel werden die zentralen Technologien vorgestellt, die für die Konzeption und Entwicklung der Bibliotheksanwendung *LibraNova* von Bedeutung sind.

3.1 Back-End Technologien

In diesem Abschnitt werden die eingesetzten Back-End-Technologien vorgestellt, wobei der Fokus auf Spring Boot liegt – dem zentralen Framework zur Implementierung einer robusten, skalierbaren und sicheren serverseitigen Logik sowie RESTful APIs für die Anwendung *LibraNova*.

3.1.1 Java

Java ist eine weit verbreitete, objektorientierte Programmiersprache, die für ihre Plattformunabhängigkeit, Stabilität und umfangreichen Standardbibliotheken bekannt ist. Aufgrund ihrer Vielseitigkeit und Leistungsfähigkeit wird sie häufig für die Entwicklung von Back-End-Anwendungen eingesetzt. In der Bibliotheksanwendung *LibraNova* wird Java in der Version 17 verwendet, um eine robuste, wartbare und skalierbare serverseitige Logik bereitzustellen [Ora25].

3.1.2 Spring Boot

Spring Boot ist ein Framework, das auf dem Spring Framework aufbaut und speziell entwickelt wurde, um schnelle, effiziente und skalierbare Anwendungen zu erstellen. Es bietet eine Vielzahl von Features, die den Entwicklungsprozess beschleunigen und vereinfachen, insbesondere für Java-basierte Webanwendungen [Spr25a].

Gründe für die Wahl von Spring Boot

Spring Boot wurde für *LibraNova* gewählt, da es die Entwicklung serverseitiger Anwendungen deutlich vereinfacht und beschleunigt. Entscheidende Vorteile sind:

- **Schneller Entwicklungsstart:** Vorkonfigurierte Abhängigkeiten und Best Practices ermöglichen sofortiges Beginnen der Entwicklung.
- **Einfache Bereitstellung:** Eigenständig ausführbare Anwendungen mit eingebautem Tomcat-Server benötigen keinen externen Applikationsserver.
- **Automatische Konfiguration:** Frameworks und Bibliotheken werden automatisch erkannt, was den Konfigurationsaufwand reduziert.
- **Produktionsreife Funktionen:** Integrierte Features wie Konfigurationsdateien, Monitoring und Health Checks vereinfachen den produktiven Betrieb [Spr25b].

Verwendete Abhängigkeiten

- **Spring Data REST:** Ermöglicht die automatische Bereitstellung von RESTful Webservices auf Basis von Spring-Data-Repositories. Es erstellt durchsuchbare Endpunkte, unterstützt HAL, Paginierung, Sortierung und Filterung, ganz ohne manuelle Controller [ST25b].
- **Spring Data JPA:** Vereinfacht die Implementierung von Repository-Schichten auf Basis der Java Persistence API. Entwickler definieren Interfaces, während Spring die Implementierung liefert. Unterstützt abgeleitete Suchmethoden, Paginierung, dynamische Abfragen und Auditierung von Entitäten [ST25a].

3.1.3 Hibernate

Hibernate ist ein typensicheres Java-ORM-Framework, das automatische Abbildung zwischen Objekten und relationalen Datenbanken ermöglicht, JPA unterstützt, komplexe Abfragen, Transaktionen, Caching und Performance-Optimierung bietet. Es fördert idiomatische Java-Persistenz, HQL-Abfragen, Datenbankkompatibilität, Skalierbarkeit und Entwicklerproduktivität, reduziert Boilerplate-Code, gewährleistet ACID-Eigenschaften und vereinfacht die Entwicklung von Enterprise-Anwendungen [HT25].

3.1.4 REST-API

REST (Representational State Transfer) ist ein Architekturstil für verteilte Hypermedia-Systeme, der von Roy Fielding im Jahr 2000 in seiner Dissertation vorgestellt wurde. Seitdem hat sich REST als eine der am weitesten verbreiteten Methoden zur Entwicklung von webbasierten APIs etabliert [Gup25]. Es definiert zentrale Prinzipien, die ein Webservice erfüllen muss, um als RESTful zu gelten:

- **Uniform Interface:** Ressourcen werden über eine einheitliche Schnittstelle eindeutig identifiziert und mittels standardisierter HTTP-Methoden (GET, POST, PUT, DELETE) manipuliert. Nachrichten sind selbstbeschreibend und können Hypermedia-Links enthalten.
- **Stateless:** Jeder Client-Request enthält alle notwendigen Informationen; der Server speichert keinen Sitzungszustand.
- **Client-Server-Architektur:** Trennung von Benutzeroberfläche und Datenhaltung ermöglicht unabhängige Weiterentwicklung und bessere Skalierbarkeit [Gup25]

Im *LibraNova*-Projekt wird das Backend mit Spring Boot umgesetzt, um RESTful-Endpunkte für die Verwaltung verschiedener Ressourcen bereitzustellen. Die API-Pfade, beispielsweise `/api/books`, sind klar strukturiert, um einen konsistenten Zugriff auf die Daten zu gewährleisten.

3.1.5 Spring Security

Spring Security ist ein anpassbares Framework zur Verwaltung von Authentifizierung und Autorisierung in Java-Anwendungen. Es bietet wiederverwendbare Module, Schutz gegen Web-Schwachstellen wie CSRF und flexible Integrationsmöglichkeiten für unterschiedliche Anwendungsfälle. Unsachgemäße Konfiguration kann jedoch Sicherheitsrisiken bergen [IRM⁺20]

Im *LibraNova*-Projekt wird Spring Security eingesetzt, um die REST-API-Endpunkte abzusichern und den Zugriff zu kontrollieren.

3.1.6 HTTPS und SSL/TLS

HTTPS sichert die Kommunikation zwischen Browser und Webserver durch TLS-Verschlüsselung (früher SSL). Öffentlicher Schlüssel verschlüsselt Daten, die nur

der private Schlüssel auf dem Server entschlüsseln kann. So werden sensible Informationen geschützt, und die Verbindung gewährleistet Vertraulichkeit, Integrität und Authentizität. Standardmäßig erfolgt die Kommunikation über Port 443 [Clo25].

In *LibraNova* laufen sowohl das Backend als auch das Frontend über HTTPS in ihren jeweiligen Ports unter Verwendung eines selbstsignierten Zertifikats.

3.1.7 Stripe API

Die Stripe API ermöglicht die nahtlose Integration von Zahlungsfunktionen in Anwendungen. Sie orientiert sich an REST-Prinzipien, verwendet ressourcenorientierte URLs, überträgt Anfragen im Formularformat und liefert Antworten im JSON-Format. Die Authentifizierung erfolgt über API-Schlüssel, und die API unterstützt sowohl Test- als auch Echtzeitmodi. Sie bietet umfassende Funktionen für Zahlungsabwicklung, Kundenverwaltung und Abo-Management sowie Zugriff auf das gesamte Stripe-Portfolio, einschließlich *Payments*, *Billing*, *Connect*, *Terminal*, *Issuing* und *Treasury*, wodurch individuelle Zahlungsprozesse und Automatisierungen effizient umgesetzt werden können [ST25c].

In *LibraNova* wurde Stripe eingesetzt, um eine moderne und sichere Zahlungsabwicklung zu ermöglichen.

3.2 Front-End Technologien

Dieser Abschnitt befasst sich mit den verwendeten Frontend-Technologien, darunter HTML, CSS, Bootstrap, TypeScript, React und i18next. Sie ermöglichen gemeinsam eine moderne, responsive und benutzerfreundliche Oberfläche für die Anwendung sowie eine effiziente Umsetzung der Internationalisierung.

3.2.1 HTML/CSS

HTML und CSS bilden die Grundlage für die Entwicklung und Gestaltung von Webseiten. HTML definiert die Struktur und den Inhalt der Seite, während CSS für das visuelle Erscheinungsbild zuständig ist – einschließlich Layout, Farben und Schriftarten. Gemeinsam ermöglichen sie die Erstellung ansprechender und übersichtlich strukturierter Benutzeroberflächen [HTM25a, HTM25b].

3.2.2 Bootstrap

Bootstrap ist ein weit verbreitetes, quelloffenes Framework zur Entwicklung responsiver und mobiler Webanwendungen. Es stellt eine Vielzahl vordefinierter CSS-Klassen sowie JavaScript-Komponenten bereit, die eine schnelle und konsistente Gestaltung von Benutzeroberflächen ermöglichen [Boo25].

In *LibraNova* wurde Bootstrap eingesetzt, um das Layout flexibel zu gestalten und sicherzustellen, dass sich die Benutzeroberfläche auf verschiedenen Bildschirmgrößen (z. B. Desktop, Tablet, Smartphone) dynamisch anpasst. Hierfür wurden gezielt Klassen wie `container`, `row` und `col-md-*` verwendet.

3.2.3 TypeScript

TypeScript wurde von Microsoft entwickelt und ist eine Programmiersprache, die JavaScript erweitert und optionale statische Typisierung sowie moderne Sprachfunktionen bietet. Sie ermöglicht es Entwickler:innen, viele häufige Fehler bereits während der Entwicklungsphase zu erkennen, wobei das ursprüngliche Laufzeitverhalten von JavaScript erhalten bleibt. Für die Entwicklung der React-basierten Benutzeroberfläche wurde bewusst TypeScript anstelle von reinem JavaScript gewählt. Die Gründe dafür liegen in der besseren Code-Wartbarkeit, der verbesserten Autovervollständigung in modernen IDEs und der höheren Typsicherheit, die gerade in größeren Anwendungen wie einem Bibliotheksverwaltungssystem eine zentrale Rolle spielt. [Cor25a, Bra25].

3.2.4 React

React ist eine JavaScript-Bibliothek zum Erstellen von Benutzeroberflächen, indem kleine, wiederverwendbare Komponenten wie Buttons und Text kombiniert werden. Es verwendet einen deklarativen Ansatz, der nur die Teile der UI effizient aktualisiert, die sich ändern, wodurch der Code leichter verständlich und leichter zu debuggen ist. Komponenten kapseln ihre eigene Logik und ihren Zustand und können zu komplexen Oberflächen zusammengesetzt werden. Da die Komponenten in JavaScript geschrieben sind, ermöglichen sie einen reibungslosen Datenfluss ohne direkte Abhängigkeit vom DOM. Zur besseren Strukturierung können Komponenten in separate Dateien ausgelagert und bei Bedarf importiert werden. React ist besonders beliebt für die Entwicklung von Single-Page-Anwendungen (SPAs), die schnelle und nahtlose Nutzererlebnisse ohne vollständiges Neuladen der Seite bieten [MP25b, MP25a, MP25c].

Die Kombination von React im Frontend mit Spring Boot im Backend ist weit verbreitet, da Spring Boot RESTful APIs bereitstellt, die React über HTTP-Anfragen konsumieren kann. Diese Trennung von Frontend und Backend unterstützt eine

klare Architektur, fördert Skalierbarkeit und erleichtert die Wartung. Über REST APIs können dynamisch Daten wie Bücher, Benutzerinformationen oder Ausleihvorgänge in Echtzeit geladen und aktualisiert werden, was für eine Bibliotheksanwendung essenziell ist [Pur25].

3.2.5 i18next

i18next ist ein umfassendes Internationalisierungs-Framework für JavaScript, das Web-, Mobile- und Desktop-Plattformen unterstützt. Es bietet Funktionen wie automatische Spracherkennung, flexibles Laden von Übersetzungen und Erweiterbarkeit durch Plugins. Kompatibel mit wichtigen Frontend-Frameworks, ist es auf Skalierbarkeit und Benutzerfreundlichkeit ausgelegt. Diese Bibliothek wurde für *LibraNova* aufgrund ihrer Popularität und ihres robusten Ökosystems ausgewählt [i1825].

3.3 Datenbanktechnologien

In diesem Abschnitt werden die wichtigsten Datenbanktechnologien erläutert, die für die Implementierung der *LibraNova*-Anwendung verwendet wurden. Im Fokus stehen dabei MySQL als Datenbanksystem, MySQL Workbench als grafisches Verwaltungswerkzeug sowie SQL als zugrunde liegende Abfragesprache.

3.3.1 MySQL

MySQL ist ein weit verbreitetes Open-Source-Datenbankmanagementsystem, das Daten in relationalen Tabellen speichert und über SQL zugänglich macht. Es bietet hohe Geschwindigkeit, Zuverlässigkeit und Skalierbarkeit [Cor25c].

3.3.2 MySQL-Workbench und SQL

MySQL Workbench ist ein grafisches Werkzeug zur Modellierung, Verwaltung und Migration von MySQL-Datenbanken. Es bietet Funktionen wie visuelles Datenbankdesign, Ausführung von SQL-Abfragen, Serveradministration sowie Datenmigration aus anderen Datenbanksystemen [Cor25b].

SQL (Structured Query Language), entwickelt von Don Chamberlin und Ray Boyce bei IBM, ist eine standardisierte Sprache zur Verwaltung relationaler Datenbanken. Sie ermöglicht das Einfügen, Abfragen, Aktualisieren und Löschen von

Daten mit wenigen Befehlen. Durch JOIN-Operationen können Daten aus mehreren Tabellen effizient verknüpft und Redundanzen vermieden werden. SQL ist ein Kernbestandteil moderner datengetriebener Anwendungen und kommt in nahezu allen relationalen Datenbanksystemen zum Einsatz [IBM25].

3.4 Authentifizierungs- und Autorisierungsprotokolle

Sichere Benutzerverwaltung und Zugriffskontrolle sind für Webanwendungen entscheidend. Technologien wie Okta, JWT, OAuth2 und OpenID Connect sorgen für standardisierte Authentifizierung und Autorisierung. Im Folgenden wird ihre Nutzung in der Anwendung beschrieben.

3.4.1 Okta

Okta ist ein cloudbasierter Identitätsdienst, der sicheren Zugriff auf Anwendungen und Geräte ermöglicht. Es bietet Single Sign-On (SSO), Multi-Faktor-Authentifizierung (MFA) und Integration mit lokalen Verzeichnissen wie Active Directory. Okta erleichtert das Identitäts- und Zugriffsmanagement über verschiedene Systeme hinweg [OT25b]. Die Okta-Integration wird verwendet, um die sichere Benutzerverwaltung auszulagern, einschließlich Passwortverwaltung, Token-Ausgabe und rollenbasierter Zugriffskontrolle. Dadurch wird ein standardisierter Authentifizierungsprozess implementiert, der verschiedene Benutzerrollen unterstützt und Berechtigungen direkt in Okta verwaltet.

3.4.2 JWT

JWT ist ein offener Standard zur sicheren Übertragung von Informationen als signiertes JSON-Objekt. Es wird hauptsächlich für die Autorisierung genutzt, indem es nach der Anmeldung den Zugriff auf geschützte Ressourcen ermöglicht. Außerdem gewährleistet JWT die Integrität und Authentizität der übertragenen Daten [JT25b]. In der Anwendung sichern Okta-JWTs die API-Endpunkte. Spring Security ist als OAuth2-Resource-Server konfiguriert und validiert die Tokens, die Benutzer-Claims wie „userType“ enthalten, um rollenbasierte Zugriffssteuerung zu ermöglichen. So wird eine sichere, tokenbasierte Authentifizierung und feingranulare Autorisierung im Backend gewährleistet.

3.4.3 OAuth2

OAuth 2.0 ist ein Sicherheitsstandard, der Drittanbieteranwendungen ermöglicht, im Namen von Nutzern auf Ressourcen zuzugreifen, ohne deren Anmeldedaten

preiszugeben. Es basiert auf dem Austausch von Zugriffstokens, was die Sicherheit erhöht [RHHH20]. In der Anwendung wird OAuth 2.0 zusammen mit Okta genutzt, um JWTs auszustellen und zu validieren. Diese Tokens autorisieren den Zugriff auf geschützte Backend-Ressourcen, wodurch eine sichere und rollenbasierte Zugriffskontrolle gewährleistet wird.

3.4.4 OpenID Connect

OpenID Connect ist ein Authentifizierungsprotokoll, das auf OAuth 2.0 basiert und die sichere Überprüfung der Benutzeridentität ermöglicht. Es liefert standardisierte Nutzerinformationen und unterstützt eine einfache Integration in verschiedene Anwendungen [OT25c]. Die Anwendung nutzt OpenID Connect mit Okta zur sicheren Anmeldung und zur Verwaltung von Benutzerrollen und Zugriffsrechten.

3.5 Version Control mit Git und GitHub

Git ist ein kostenloses, verteiltes Versionskontrollsystem, das durch hohe Geschwindigkeit, effizientes Branch-Management und flexible Arbeitsabläufe besticht. Es ermöglicht die einfache Verwaltung von Projekten jeder Größe und unterstützt parallele Entwicklungsprozesse durch lokale Branches [GT25a].

GitHub ergänzt Git als cloudbasierte Plattform zum Speichern, Teilen und gemeinsamen Entwickeln von Code. Es erleichtert das Nachverfolgen von Änderungen, die Code-Review durch andere Entwickler sowie die koordinierte Zusammenarbeit, ohne unbeabsichtigte Auswirkungen auf den Hauptzweig zu riskieren [GT25b]. Beide Werkzeuge haben maßgeblich zur effizienten Verwaltung und Versionskontrolle des Codes im Verlauf dieses Projekts beigetragen.

3.6 Testen und Qualitätssicherung

Zur Sicherstellung der Softwarequalität wurden im Projekt Unit-Tests, Mocking und API-Tests eingesetzt, um Funktionen, API-Endpunkte und Abhängigkeiten zuverlässig zu überprüfen.

3.6.1 Unit-Tests mit JUnit 5

JUnit 5 ist ein modernes Testframework für Java, das aus mehreren Modulen besteht: der JUnit Platform, JUnit Jupiter und JUnit Vintage. Die JUnit Platform

bildet die Basis zur Ausführung von Tests auf der JVM und ermöglicht die Integration verschiedener Testengines. JUnit Jupiter bietet die Programmierschnittstelle und Erweiterungsmöglichkeiten für das Schreiben und Ausführen neuer Tests, während JUnit Vintage die Kompatibilität zu älteren JUnit-Versionen sicherstellt. JUnit 5 setzt mindestens Java 8 voraus und wird von gängigen Entwicklungsumgebungen und Build-Tools wie IntelliJ IDEA, Maven und Gradle unterstützt. Die modulare Architektur erleichtert sowohl das Schreiben als auch das Ausführen von Tests in modernen Java-Projekten [JT25a].

3.6.2 Mocking mit Mockito

Mockito ist ein weit verbreitetes Mocking-Framework für Java, das speziell für das Schreiben von klaren und leicht lesbaren Unit-Tests entwickelt wurde. Es ermöglicht das einfache Erstellen von Scheinobjekten (Mocks), mit denen sich das Verhalten von Abhängigkeiten gezielt simulieren und testen lässt. Dank seiner übersichtlichen API trägt Mockito zu einer besseren Verständlichkeit der Tests und zu nachvollziehbaren Fehlermeldungen bei. Es wird von der Entwicklergemeinschaft intensiv genutzt und zählt zu den beliebtesten Bibliotheken im Java-Ökosystem [MT25b].

3.6.3 API-Tests mit Postman

Postman ist eine umfassende Plattform für die Arbeit mit APIs und unterstützt den gesamten Lebenszyklus – von der Planung über das Testen bis hin zur Bereitstellung und Überwachung. Mit einer intuitiven Oberfläche und zahlreichen Funktionen ermöglicht Postman das Speichern, Dokumentieren und Testen von API-Endpunkten in einem zentralen Repository. Es bietet Werkzeuge zur Spezifikation, Mock-Erstellung und Automatisierung von Tests, wodurch die Entwicklung beschleunigt und die Zusammenarbeit im Team vereinfacht wird [PT25].

3.7 Modellierung mit UML

Die Unified Modeling Language (UML) ist ein wesentliches Werkzeug im Bereich Entwurf. UML ist eine standardisierte Modellierungssprache, die eine Vielzahl von Diagrammen bietet, um unterschiedliche Aspekte eines Systems darzustellen. Diese Diagramme helfen dabei, komplexe Systeme zu visualisieren, zu dokumentieren und zu kommunizieren [Bel25]. Eine Auswahl relevanter Diagramme zur Konzeption der Webanwendung wird im folgenden Kapitel vorgestellt, um einen Überblick über den strukturellen Aufbau und das Zusammenspiel der Systemkomponenten zu geben.

4

Konzept

Dieses Kapitel beschreibt das grundlegende Konzept der Anwendung *Libranova*. Es erläutert zunächst die Systemarchitektur und die wichtigsten technischen Komponenten, bevor zentrale Abläufe anhand von UML-Diagrammen veranschaulicht werden. Anschließend werden die funktionalen, nicht-funktionalen und technischen Anforderungen dargestellt, gefolgt von der Beschreibung wesentlicher Algorithmen in Pseudocode. Ziel ist es, sowohl die logische Struktur als auch die Interaktion der einzelnen Systemteile transparent und nachvollziehbar darzustellen.

4.1 Blockbild der Architektur

Das Gesamtsystem basiert auf einer klassischen Client-Server-Architektur. Die Webanwendung besteht aus zwei Hauptkomponenten: dem Frontend, implementiert mit React, und dem Backend, realisiert mit Spring Boot. Die Kommunikation erfolgt über eine REST-API. Persistente Daten werden in einer MySQL-Datenbank gespeichert.

Abbildung 4.1 zeigt ein Blockdiagramm der Systemarchitektur von *Libranova*. Es veranschaulicht die Aufteilung der Anwendung in verschiedene logische Schichten und deren Zusammenspiel:

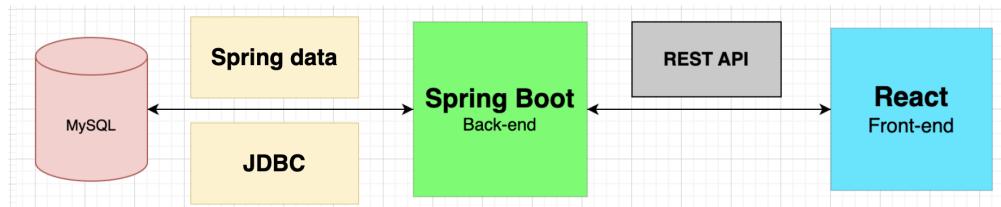


Abbildung 4.1: Blockbild der Systemarchitektur

- **React Frontend:** Die Benutzeroberfläche wurde mit React umgesetzt. Sie bietet eine moderne, komponentenbasierte Nutzererfahrung. Benutzeraktionen wie

Anmeldung, Buchsuche oder Ausleihe werden hier initiiert und über HTTP-Anfragen (im JSON-Format) an die REST-API weitergeleitet.

- **REST API:** Die Schnittstelle zwischen Frontend und Backend folgt dem REST-Architekturstil. Die Kommunikation erfolgt über klar definierte Endpunkte unter Verwendung der HTTP-Methoden **GET**, **POST**, **PUT** und **DELETE**.
- **Spring Boot Backend:** Diese Schicht verarbeitet eingehende API-Anfragen, übernimmt die Geschäftslogik, führt Validierungen durch und steuert Datenbankzugriffe. Die Benutzerverwaltung und Zugriffskontrolle erfolgt hier über die Integration mit Okta.
- **Spring Data JPA, Spring Data REST und JDBC:** Für den Datenbankzugriff wird Spring Data JPA verwendet, das eine deklarative und effiziente Abfrageerstellung über Repository-Interfaces ermöglicht. Über Spring Data REST werden ausgewählte Repository-Methoden automatisch als REST-Endpunkte bereitgestellt. Intern erfolgt die Kommunikation mit der MySQL-Datenbank über JDBC als Treiberschicht.
- **MySQL Database:** Relationale Datenbank zur Speicherung persistenter Daten, strukturiert durch Entitäten wie **Buch**, **Rezension** und **Historie**.

4.2 UML-Diagramme zur Konzeption

Zur Visualisierung des Systemkonzepts werden im Folgenden zentrale UML-Diagramme vorgestellt.

4.2.1 Klassendiagramm

Die Abbildung 4.2 zeigt das Klassendiagramm der Anwendung *LibraNova*. Das Diagramm gliedert sich in vier Hauptbereiche: Bibliotheksbereich, Zahlungssystem, Kommunikation und Benutzerverwaltung.

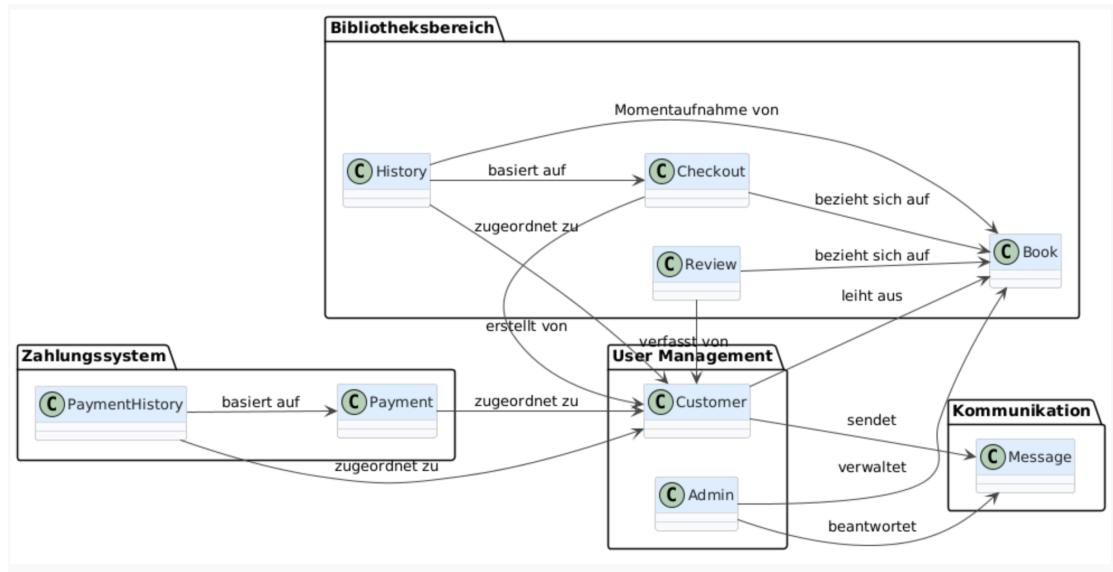


Abbildung 4.2: Klassendiagramm der Anwendung *Libranova*

Im Bibliotheksbereich repräsentiert die Klasse **Book** zentrale Buchinformationen. **Checkout** modelliert aktive Ausleihen, während **History** abgeschlossene Ausleihen als Momentaufnahme speichert und auf den **Checkout**-Vorgängen basiert. Nutzerbewertungen werden über die Klasse **Review** abgebildet, die einem Buch zugeordnet und von einem **Customer** erstellt wird.

Das Zahlungssystem besteht aus den Klassen **Payment** und **PaymentHistory**. **Payment** speichert einzelne Zahlungsinformationen, während **PaymentHistory** auf diesen Zahlungen aufbaut und dem Nutzer einen Überblick über seine bisherigen Transaktionen bietet.

Die Kommunikation zwischen Nutzern und Administratoren wird durch die Klasse **Message** modelliert. Kunden können Nachrichten an Administratoren senden, die diese beantworten.

Im Bereich Benutzerverwaltung werden die Rollen **Customer** und **Admin** dargestellt. **Admin** verwaltet Bücher und beantwortet Nachrichten, während **Customer** Bücher ausleiht, bewertet, Nachrichten versendet und Zahlungen tätigt. Die Klassen sind eng über Beziehungen verknüpft: **Review** und **Checkout** beziehen sich auf **Book** und **Customer**, **History** basiert auf **Checkout**-Daten, und **PaymentHistory** baut auf **Payment**-Vorgängen auf.

Das Diagramm verdeutlicht die logische Struktur der Anwendung, die Verantwortlichkeiten der Klassen und die Interaktionen zwischen den Systemkomponenten.

4.2.2 Sequenzdiagramme

Abbildung 4.3 zeigt das Sequenzdiagramm des Benutzer-Login-Vorgangs innerhalb der Anwendung. Ziel dieses Prozesses ist die sichere Authentifizierung des Benutzers sowie die Bereitstellung eines Access Tokens, das den Zugriff auf geschützte Ressourcen ermöglicht.

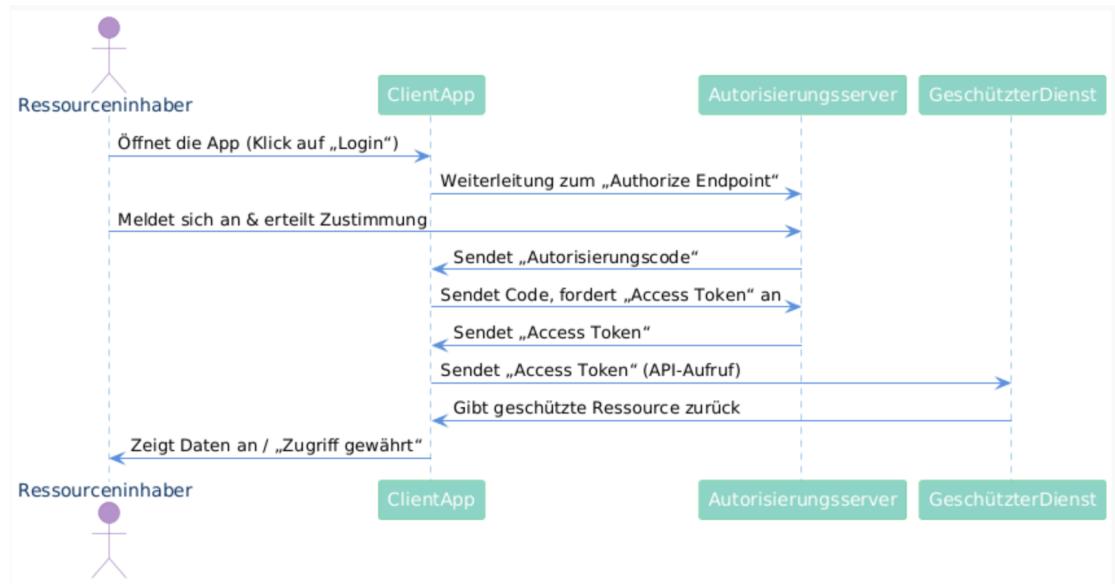


Abbildung 4.3: Sequenzdiagramm des Login-Vorgangs, angelehnt an [OT25a]

Im Login-Prozess sind mehrere Komponenten beteiligt. Der Ressourceninhaber (Benutzer) startet den Anmeldevorgang und erteilt die Zustimmung zur Autorisierung. Die Client-Anwendung (React) fungiert als Schnittstelle zwischen Benutzer und System. Der Autorisierungsserver (Okta) authentifiziert den Benutzer, stellt einen Autorisierungscode aus und tauscht diesen gegen ein Access Token ein. Schließlich überprüft der Ressourcenserver (Spring Boot) die Gültigkeit des Tokens und stellt bei erfolgreicher Authentifizierung die geschützten Ressourcen bereit.

Der Ablauf gliedert sich in folgende Schritte: Zunächst öffnet der Benutzer die React-Anwendung und startet den Login-Prozess. Die Client-Anwendung leitet den Benutzer an den Autorisierungsserver weiter, wo sich der Benutzer anmeldet und der Autorisierung zustimmt. Anschließend sendet der Autorisierungsserver einen Autorisierungscode an die Client-Anwendung, welcher gegen ein Access Token eingetauscht wird. Dieses Token wird lokal gespeichert. Bei einem geschützten API-Aufruf wird das Token an den Ressourcenserver übermittelt. Der Server prüft die Gültigkeit des Tokens und gibt bei erfolgreicher Prüfung die geschützten Daten an die Client-Anwendung zurück, sodass sie dem Benutzer angezeigt werden.

Nach der Beschreibung der Benutzeranmeldung wird nun der Bezahlvorgang über Stripe betrachtet.

Abbildung 4.4 zeigt den Ablauf eines Bezahlvorgangs mit Stripe. Das Sequenzdiagramm verdeutlicht, wie die React-Anwendung, das Spring Boot-Backend und der externe Zahlungsdienstleister Stripe zusammenwirken, um einen sicheren und effizienten Zahlungsprozess zu gewährleisten.

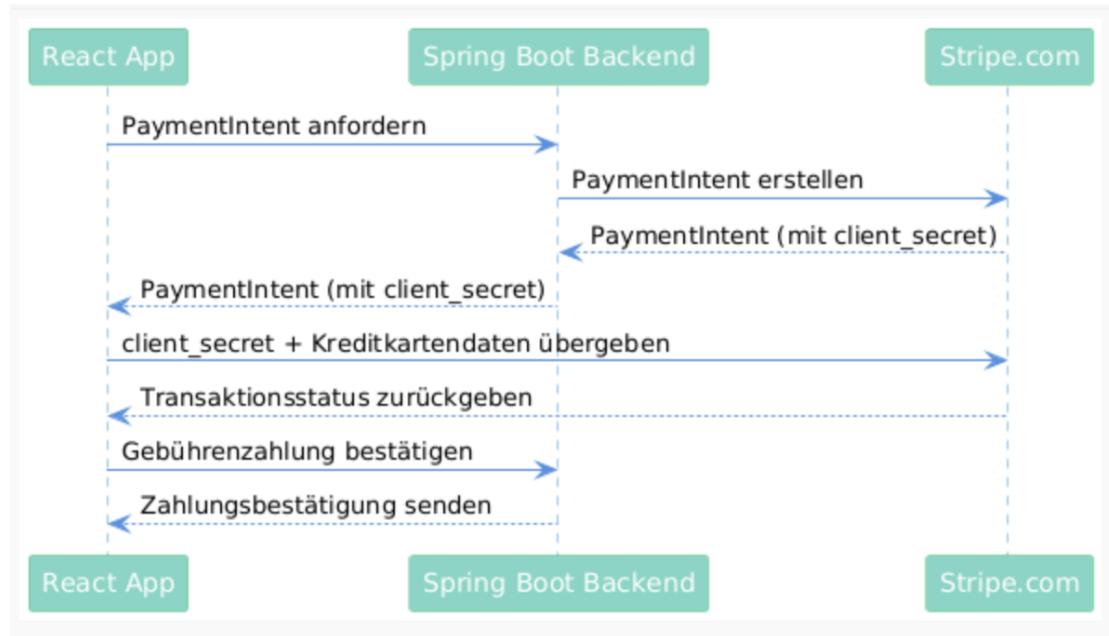


Abbildung 4.4: Zahlungsablauf mit Stripe, angelehnt an [MT25a]

Der Zahlungsprozess beginnt, wenn die React-Anwendung eine Anfrage an den Backend-Endpunkt `/api/payments/secure/intent` sendet, um einen neuen *PaymentIntent* zu erstellen. Das Backend leitet diese Anfrage an Stripe weiter und übermittelt dabei die erforderlichen Parameter wie Betrag, Währung und Zahlungsmethode. Stripe generiert daraufhin einen neuen *PaymentIntent* und gibt diesen, inklusive des *client_secret*, an das Backend zurück. Das Backend sendet den *client_secret* an die React-Anwendung, die ihn zusammen mit den Kreditkartendaten direkt an Stripe übermittelt. Stripe verarbeitet die Zahlungsinformationen und gibt den Transaktionsstatus zurück an das Frontend. Abschließend informiert das Frontend das Backend über die erfolgreiche Zahlung, sodass Datenbanken aktualisiert oder Zahlungshistorien gepflegt werden können, und das Backend bestätigt die erfolgreiche Zahlungsabwicklung.

Ein *PaymentIntent* ist ein von Stripe bereitgestelltes Objekt, das die Absicht einer Zahlung repräsentiert und den gesamten Zahlungsablauf verwaltet – von der

Erstellung über die Authentifizierung bis hin zur endgültigen Bestätigung der Zahlung.

Der *client-secret* ist ein einzigartiger, von Stripe generierter Schlüssel für jeden *PaymentIntent*. Er dient als sicherer Zugriffstoken, mit dem das Frontend bestimmte Informationen über die jeweilige Zahlung abrufen kann, beispielsweise den aktuellen Status der Transaktion. Der *client-secret* ist ausschließlich auf diese eine Zahlung beschränkt und kann nicht für andere Aktionen verwendet werden.

Der Endpunkt `POST /api/payments/secure/intent` nimmt ein JSON-Objekt mit Zahlungsinformationen wie Betrag und Währung entgegen. Intern ruft der Controller die Methode `generatePaymentIntent()` auf, die über das Stripe-SDK einen neuen *PaymentIntent* erstellt. Dabei werden der Betrag (`amount`), die Währung (`currency`) und der Zahlungstyp (auf Kreditkarte beschränkt) festgelegt. Das vom Stripe-SDK zurückgegebene *PaymentIntent*-Objekt, inklusive *client-secret*, wird anschließend vom Backend an das Frontend übermittelt, um den Bezahlvorgang fortzusetzen.

4.3 Anforderungen an das System

Die folgenden Unterabschnitte beschreiben die zentralen Anforderungen an das System. Dazu zählen funktionale, nicht-funktionale sowie technische Anforderungen, die für eine strukturierte Umsetzung der Anwendung erforderlich sind.

4.3.1 Funktionale Anforderungen

Funktionale Anforderungen definieren, welche Dienste das System leisten soll und wie es sich bei bestimmten Eingaben oder in bestimmten Situationen verhalten soll.

Die Tabelle A.1 zeigt alle funktionalen Anforderungen der Anwendung.

4.3.2 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen beschreiben die Qualitätsmerkmale und Randbedingungen des Systems, wie etwa Leistung, Sicherheit, Benutzbarkeit und Zuverlässigkeit

Die Tabelle A.2 fasst die wichtigsten nicht-funktionalen Anforderungen für LibraNova zusammen.

4.3.3 Technische Anforderungen

Technische Anforderungen beschreiben, welche Technologien, Werkzeuge und Methoden für die Entwicklung und den Betrieb von LibraNova verwendet werden. Dazu gehören zum Beispiel Programmiersprachen, Frameworks, Schnittstellen und Protokolle.

Die Tabelle A.3 zeigt die wichtigsten technischen Anforderungen von LibraNova.

4.4 Wichtige Algorithmen (Pseudocode)

In diesem Abschnitt werden die wichtigsten Algorithmen dargestellt, die zentrale Abläufe des Systems beschreiben. Die Abläufe werden aus der Perspektive der Benutzerinteraktionen sowie der dahinterliegenden Systemlogik erläutert. Dabei werden wesentliche Prüfungen und Bedingungen berücksichtigt, um einen klaren und nachvollziehbaren Überblick über die Funktionsweise zu geben.

4.4.1 Buchausleihe

Der folgende Pseudocode 4.1 beschreibt den vollständigen Ablauf der Buchausleihe aus der Perspektive des Benutzers sowie der Systemlogik. Dabei werden wichtige Prüfungen berücksichtigt, wie die Anmeldung des Benutzers, die Verfügbarkeit von Exemplaren, die maximale Anzahl ausgeliehener Bücher sowie etwaige überfällige Rückgaben oder offene Gebühren.

```

1      1. Benutzer öffnet die Detailseite eines Buches.
2      2. System prüft:
3          a. Ist der Benutzer angemeldet?
4              - Nein → Zeige Button >> Anmelden << (Weiterleitung zur
               Login-Seite)
5              - Ja → Weiter zu Schritt 3
6      3. Hat der Benutzer dieses Buch bereits ausgeliehen?
7          - Ja → Zeige Hinweis >> Bereits ausgeliehen <<
8          - Nein → Weiter zu Schritt 4
9      4. Hat der Benutzer weniger als 5 Bücher ausgeliehen?
10         - Nein → Zeige Hinweis >> Maximale Anzahl an Büchern
               erreicht <<
11         - Ja → Weiter zu Schritt 5
12      5. Sind Exemplare des Buches verfügbar?
13         - Nein → Zeige deaktivierten Button >> Ausleihen <<
14         - Ja → Zeige aktiven Button >> Ausleihen <<
15      6. Klickt der Benutzer auf >> Ausleihen <<:
16          a. Hat der Benutzer überfälligen Bücher oder unbezahlten Gebü
               hren?

```

```

17      - Nein -> Lege Ausleiheintrag in der Datenbank an (7 Tage
           Leihfrist)
18      - Ja -> Zeige Meldung: >> Bitte geben Sie überfällige Bü
           cher zurück und begleichen Sie offene Gebühren, bevor
           Sie neue Bücher ausleihen können. <<

```

Listing 4.1: Pseudocode für den Ausleihvorgang eines Buches

4.4.2 Verlängerung der Buchausleihe

Der folgende Pseudocode 4.2 beschreibt den vollständigen Ablauf, wie Nutzer die Leihfrist eines ausgeliehenen Buches verlängern können. Dabei werden die Bedingungen geprüft, unter denen eine Verlängerung möglich ist, sowie die Benutzeroberfläche entsprechend angepasst.

```

1. Benutzer öffnet seine Ausleihen und klickt bei einem Buch auf
   >> Ausleihe verwalten <<.
2. System zeigt zwei Optionen:
   a. >> Zurückgeben <<
   b. >> Verlängern <<
3. System prüft für den Button >> Verlängern <<:
   a. Ist das Rückgabedatum überschritten?
   b. Wurde das Buch aus dem System gelöscht?
4. Wenn eine der Bedingungen erfüllt ist:
   a. Zeige deaktivierten Button mit dem Text >> Verlängerung
      nicht möglich <<.
5. Wenn keine der Bedingungen zutrifft:
   a. Button >> Verlängern << ist aktiv.
   b. Benutzer klickt auf >> Verlängern <<.
   c. System verlängert die Leihfrist um 7 Tage.
   d. Verbleibende Tage werden aktualisiert.

```

Listing 4.2: Pseudocode für die Verlängerung einer Buchausleihe

5

Realisierung

In diesem Kapitel werden die zentralen Komponenten der Systemarchitektur vorgestellt. Es beginnt mit dem modular aufgebauten Backend auf Basis von Spring Boot, gefolgt von der Struktur und Internationalisierung des Frontends mit React. Abschließend wird das Datenbankmodell erläutert, einschließlich der Persistenzlogik mit Spring Data und Hibernate.

5.1 Backend-Architektur

Das Backend basiert auf Spring Boot und folgt einer klar strukturierten, modularen Architektur. Beschrieben werden der Einstiegspunkt, die Paketstruktur sowie die eingesetzte Teststrategie zur Sicherstellung der Codequalität.

5.1.1 Projektinitialisierung und Startklasse

Spring Boot vereinfacht die Entwicklung von Java-basierten Webanwendungen durch konventionsbasierte und automatisierte Projektkonfiguration. Dieses Framework bietet eine standardisierte Struktur, automatisches Abhängigkeitsmanagement und integrierte Komponenten, die den Entwicklungsprozess erheblich beschleunigen.

In diesem Abschnitt wird die Struktur des Projekts erläutert, wobei der Schwerpunkt auf der Projektinitialisierung, den zentralen Einstiegspunkt der Anwendung sowie wesentliche Konfigurationsmechanismen liegt.

- **Projektinitialisierung (Spring Initializr):** Die Anwendung *LibraNova* wurde mithilfe des Online-Tools *Spring Initializr* (<https://start.spring.io/>) erzeugt. Dieses Tool ermöglicht die einfache Auswahl von Projektparametern wie Abhängigkeiten, Java-Version und Build-Tool (z. B. Maven oder Gradle) und

generiert eine startbereite Projektstruktur inklusive grundlegender Dateien und Verzeichnisse.

- **Einstiegspunkt der Anwendung – Main-Klasse:** Die zentrale Einstiegsklasse der Anwendung befindet sich im `src/main/java`-Verzeichnis und enthält die `main`-Methode. Sie ist mit der Annotation `@SpringBootApplication` versehen, welche drei wichtige Spring-Annotationen kombiniert:

- `@Configuration` – Kennzeichnet die Klasse als Quelle für Bean-Definitionen.
- `@EnableAutoConfiguration` – Aktiviert die automatische Konfiguration basierend auf den eingebundenen Abhängigkeiten.
- `@ComponentScan` – Ermöglicht das automatische Auffinden von Komponenten, Services und Repositories im angegebenen Paket und dessen Unterpaketen.

Nachfolgend 5.1 ist der Quellcode der Main-Klasse dargestellt:

```

1      @SpringBootApplication
2      public class SpringBootLibraryApplication {
3
4          public static void main(String[] args) {
5              SpringApplication.run(SpringBootLibraryApplication.class,
6                  args);
7
8      }

```

Listing 5.1: Einstiegspunkt der Spring Boot Anwendung

Erläuterung der Main-Klasse:

- `@SpringBootApplication`: Diese Annotation bündelt die Konfiguration, Auto-Konfiguration und Komponentensuche in einer einzigen Annotation und ist der zentrale Startpunkt für die Spring-Boot-Anwendung.
- `public class SpringBootLibraryApplication`: Definition der Hauptklasse, die die Anwendung repräsentiert.
- `public static void main(String[] args)`: Die Main-Methode dient als Einstiegspunkt der Java-Anwendung. Sie wird beim Programmstart aufgerufen.
- `SpringApplication.run(...)` : Diese Methode startet den eingebetteten Server, initialisiert den Spring Application Context und lädt alle Komponenten, Beans und Konfigurationen.
- **Konfiguration über `application.properties`:** Die Datei `application.properties` im Verzeichnis `src/main/resources` enthält

zentrale Konfigurationen der Anwendung, wie z. B. Datenbankverbindung, REST-Basis-Pfad und andere anwendungsspezifische Einstellungen.

```

1      # Name der Anwendung
2      spring.application.name=libranova

4      # Datenbankverbindung (MySQL)
5      spring.datasource.url=jdbc:mysql://localhost:3306/libranova_db
6      spring.datasource.username=username
7      spring.datasource.password=password

9      # Dialekt von Hibernate
10     spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.
11         MySQLDialect

12     # Basis-REST-Pfad
13     spring.data.rest.base-path=/api

```

Listing 5.2: Beispielhafte application.properties Datei

Erläuterung der wichtigsten Einstellungen:

- `spring.application.name`: Definiert den Namen der Anwendung, z. B. für Logs oder Monitoring.
- `spring.datasource.url`: Verbindungs-URL zur MySQL-Datenbank inklusive Host, Port und Datenbankname.
- `spring.datasource.username`: Benutzername für die Datenbankverbindung.
- `spring.datasource.password`: Passwort für den Datenbankzugang.
- `spring.jpa.properties.hibernate.dialect`: Gibt den SQL-Dialekt für JPA/Hibernate an (hier MySQL).
- `spring.data.rest.base-path`: Legt den Basis-URL-Pfad für automatisch generierte REST-Endpunkte fest (z. B. `/api`).
- **Eingebetteter Webserver (Tomcat):** Spring Boot integriert standardmäßig einen eingebetteten Webserver (standardmäßig Tomcat). Dadurch kann die Anwendung direkt als eigenständiger Prozess gestartet werden, ohne dass ein separater Webserver installiert oder konfiguriert werden muss [Spr25b].

5.1.2 Modulare Paketstruktur des Backends

Im Folgenden wird die interne Struktur des Backends detailliert betrachtet. Dazu gehört die Aufteilung der Anwendung in verschiedene Pakete, die jeweils eigene Verantwortlichkeiten übernehmen und so zur Übersichtlichkeit und Wartbarkeit beitragen.

Eine schematische Übersicht der Paketstruktur ist in **Abbildung 5.1** unten dargestellt.

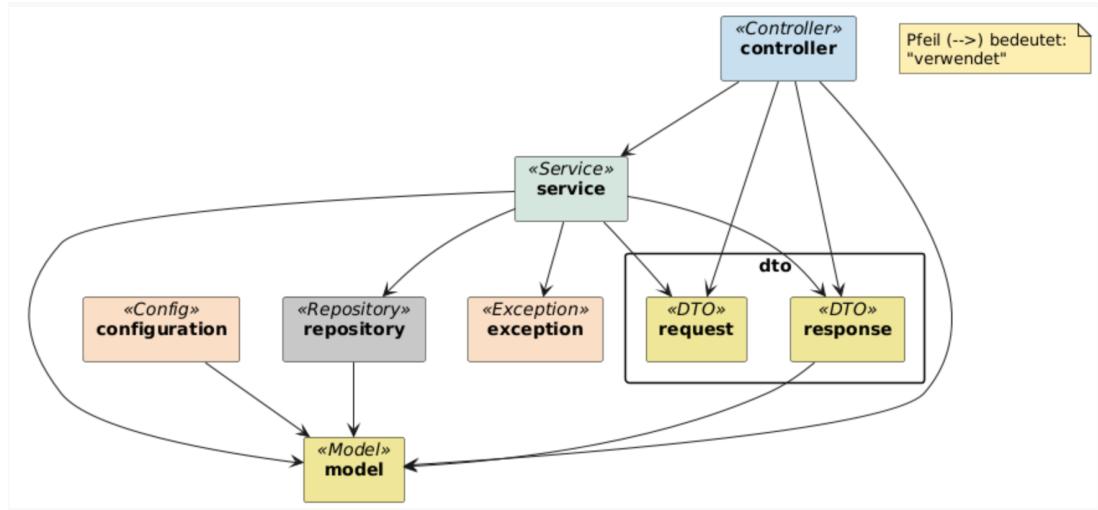


Abbildung 5.1: Modularer Aufbau des Backends

Im Folgenden werden die Pakete und ihre jeweiligen Verantwortlichkeiten beschrieben.

- **model:** Enthält die JPA-Entitäten, die die Tabellen der MySQL-Datenbank abbilden. Jede Klasse in diesem Paket entspricht einer Datenbanktabelle und definiert deren Attribute und Beziehungen. Diese Entitäten bilden die Grundlage für den Datenzugriff über das Repository.
- **dto:** Dient dem strukturierten Datenaustausch zwischen Client und Server, ohne interne Entitäten direkt preiszugeben. Es gibt zwei Unterpakete:
 - **request:** definiert Datenstrukturen für eingehende Anfragen wie etwa Formularinhalte oder Suchparameter,
 - **response:** definiert Rückgabeformate, die speziell für die Client-seitige Anzeige oder Weiterverarbeitung optimiert sind.

Die Verwendung von DTOs erhöht die Sicherheit und Flexibilität des Datenmodells.

- **exception:** Beinhaltet zentrale Komponenten zur Fehlerbehandlung. In der aktuellen Implementierung ist eine benutzerdefinierte Ausnahme enthalten, die bei nicht verfügbaren Büchern geworfen wird. Diese Ausnahme verbessert die Verständlichkeit von Fehlermeldungen auf der Client-Seite.

- **repository:** Beinhaltet Interfaces zur Datenzugriffsabstraktion mittels **Spring Data JPA**. Sie ermöglichen CRUD-Operationen auf den JPA-Entitäten, ohne dass eigene SQL-Statements geschrieben werden müssen. Damit wird der Datenzugriff stark vereinfacht und typsicher umgesetzt.
- **service:** Kapselt die Geschäftslogik der Anwendung. Hier werden Anfragen aus den Controllern verarbeitet, Daten validiert und Repository-Zugriffe koordiniert. Die Services dienen als zentrale Steuerungseinheit zwischen Controller-Logik und Datenbankzugriff.
- **controller:** Beinhaltet die REST-Controller zur Entgegennahme und Verarbeitung von HTTP-Anfragen. Sie dienen als Schnittstelle zwischen Client und Server und leiten die Anfragen zur weiteren Verarbeitung an die Service-Schicht weiter. Zudem bereiten sie die Daten so auf, dass sie für den Client verständlich und verwertbar sind.
- **configuration:** Enthält zentrale Sicherheitskonfigurationen der Anwendung. Hier werden der Zugriff auf HTTP-Endpunkte sowie Authentifizierungsmechanismen mittels JWT und OAuth2 (Okta) definiert und gesteuert.

5.1.3 Teststrategie und Testintegration

Um die Qualität und Zuverlässigkeit des Backends sicherzustellen, wurde ein automatisiertes Testkonzept implementiert. Dabei kommen vor allem Unit-Tests mit **JUnit** sowie Mocking mit **Mockito** zum Einsatz.

Die Tests befinden sich im Verzeichnis `src/test/java` und folgen der Paketstruktur des Produktivcodes, um eine klare Zuordnung zu ermöglichen.

- **JUnit** wird verwendet, um einzelne Komponenten isoliert zu testen und deren Verhalten zu validieren.
- **Mockito** ermöglicht das Erzeugen von Mock-Objekten, um Abhängigkeiten während der Tests zu simulieren und somit isolierte Testumgebungen zu schaffen.

Integration der Tests in den Build-Prozess erfolgt über das verwendete Build-Tool **Maven**, wodurch die Tests automatisiert ausgeführt werden können und eine kontinuierliche Qualitätssicherung gewährleistet ist.

5.2 Frontend-Struktur

In diesem Abschnitt wird die Architektur des Frontends erläutert. Beginnend mit dem Einstiegspunkt der React-Anwendung, werden anschließend die Projektstruktur sowie die Integration der Internationalisierung mittels i18next vorgestellt.

5.2.1 Einstiegspunkt der React-Anwendung

Das Frontend der Anwendung wurde mit **React** und **TypeScript** auf Basis von **Create React App (CRA)** entwickelt. CRA bietet eine sofort einsatzbereite Entwicklungsumgebung inklusive Webpack-Konfiguration, Hot-Reloading, Testing-Setup und TypeScript-Support [Fac25].

In diesem Abschnitt wird die Struktur der React-Anwendung beschrieben, wobei der Schwerpunkt auf der Projektinitialisierung, dem Einstiegspunkt sowie zentralen Konfigurationsdateien liegt.

- **Projektinitialisierung mit CRA:** Die Anwendung wurde über folgendes Kommando initialisiert (siehe Listing 5.3):

```
1 npx create-react-app react-library-app --template typescript
```

Listing 5.3: Projektinitialisierung mit Create React App

Dieses Kommando setzt sich wie folgt zusammen:

- **npx:** Führt ein npm-Paket temporär aus, ohne es global zu installieren.
- **create-react-app:** Das offizielle Tool zur Erzeugung von React-Projekten, welches ein komplettes Setup mit Build-Tooling, Linter und Tests erstellt.
- **react-library-app:** Der Name des Projektverzeichnisses, das automatisch erstellt wird.
- **--template typescript:** Gibt an, dass die Anwendung mit TypeScript anstelle von JavaScript erstellt werden soll.

Dadurch wurde eine vollständige Projektstruktur erzeugt, einschließlich Konfigurationsdateien, TypeScript-Unterstützung und einer initialen Komponentenstruktur.

- **Einstiegspunkt – index.tsx:** Die Datei `src/index.tsx` (siehe Listing 5.4) bildet den Einstiegspunkt der React-Anwendung. Dort wird die Hauptkomponente `<App />` in das DOM eingebunden:

```

1      import React from 'react';
2      import ReactDOM from 'react-dom/client';
3      import './index.css';
4      import { App } from './App';
5      import { BrowserRouter } from 'react-router-dom';
6      import 'bootstrap-icons/font/bootstrap-icons.css';
7      import { loadStripe } from '@stripe/stripe-js';
8      import { Elements } from '@stripe/react-stripe-js';
9      import './i18n';

11     const stripePromise = loadStripe('my_stripe_public_key');

13     const root = ReactDOM.createRoot(
14       document.getElementById('root') as HTMLElement
15     );
16     root.render(
17       <BrowserRouter>
18         <Elements stripe={stripePromise}>
19           <App />
20         </Elements>
21       </BrowserRouter>
22     );

```

Listing 5.4: Einstiegspunkt der React-Anwendung

Die Datei `index.tsx` dient als zentrales Einstiegsskript für die React-Anwendung. Sie importiert zunächst alle notwendigen Module und Abhängigkeiten wie React selbst, den ReactDOM-Client, die globale CSS-Datei, die Hauptkomponente `App` sowie zusätzliche Bibliotheken für Routing (`react-router-dom`), UI-Icons (Bootstrap Icons), Zahlungsintegration (Stripe) und Internationalisierung (`i18n`).

Im nächsten Schritt wird Stripe über `loadStripe` mit dem öffentlichen Schlüssel initialisiert und in einer Promise-Variable gespeichert. Anschließend wird mit Hilfe von `ReactDOM.createRoot(...)` eine sogenannte „Root“-Instanz erzeugt, welche das `<div>` mit der ID `root` im HTML-Dokument referenziert. Das `<div>` mit der ID `root` befindet sich in der Datei `public/index.html`.

Im letzten Schritt erfolgt das eigentliche Rendern der Anwendung. Hierbei wird die Komponente `<App />` in den Kontext von `<BrowserRouter>` und `<Elements>` eingebettet, um Routing- und Zahlungsfunktionen global bereitzustellen.

- **Backend-Kommunikation über Umgebungsvariablen:** Hier wird die Port-URL des Backends definiert, welche von der React-Anwendung zur Kommunikation mit dem Backend genutzt wird. Diese Konfiguration erfolgt über die Datei `.env` (siehe Listing 5.5).

Die Umgebungsvariable `REACT_APP_API_URL` gibt die Adresse an, unter der die REST-API des Backends erreichbar ist. Damit diese Variable im Code verwendet werden kann, muss sie zwingend mit dem Präfix `REACT_APP_` beginnen. Die-

se Variable wird beispielsweise im Code zur Konfiguration der API-Endpunkte verwendet, z. B. `fetch(process.env.REACT_APP_API_URL + '/books')`.

```
1 REACT_APP_API_URL='https://localhost:8443/api'
```

Listing 5.5: Frontend-Umgebungsvariable für Backend-Zugriff in .env-Datei

5.2.2 React-Projektstruktur

Im Folgenden wird die interne Struktur der React-Frontend-Anwendung detailliert betrachtet. Die Anwendung ist in verschiedene Verzeichnisse unterteilt, die jeweils spezifische Verantwortlichkeiten übernehmen und dadurch zur Übersichtlichkeit, Modularität und Wartbarkeit beitragen.

Eine schematische Übersicht der Verzeichnisse und zentralen Dateien ist in **Abbildung 5.2** unten dargestellt.

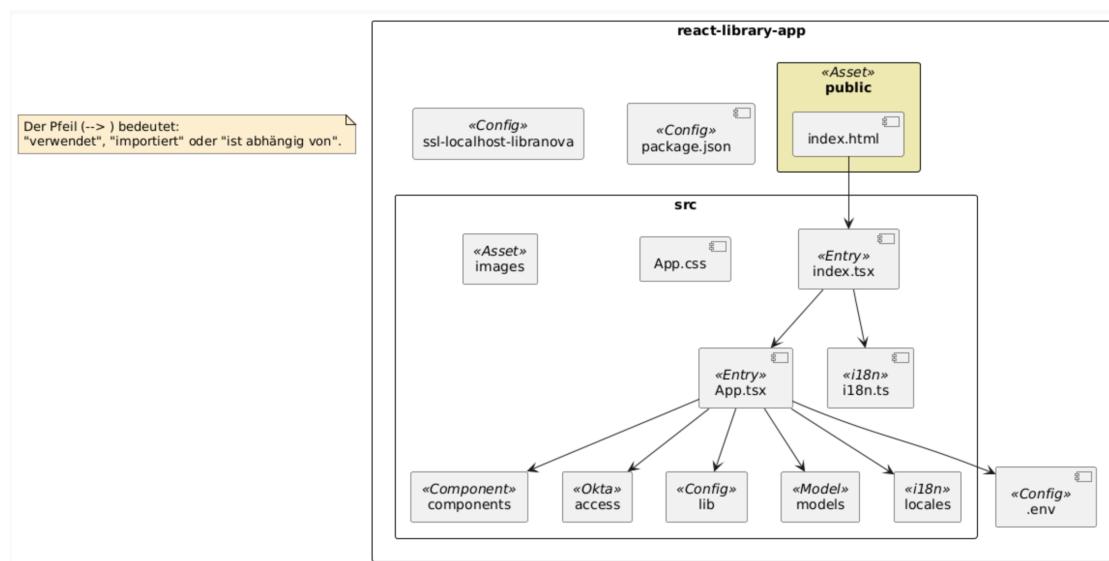


Abbildung 5.2: Modularer Aufbau des Frontends

Anschließend werden die wichtigsten Ordner und Dateien sowie ihre jeweiligen Aufgabenbereiche beschrieben.

- **index.html:** Statisches HTML-Grundgerüst der Anwendung. Enthält das `<div>` mit der ID `root`, in das React die App rendernt. Zusätzlich werden hier externe Ressourcen wie Bootstrap und Stripe eingebunden.

- **package.json:** Zentrale Konfigurationsdatei für das React-Projekt. Definiert alle Projektabhängigkeiten, Skripte zur Ausführung (z. B. Starten, Bauen, Testen), sowie weitere Metadaten der Anwendung.
- **ssl-local-libranova:** Enthält SSL-Zertifikat (`localhost.crt`) und privaten Schlüssel (`localhost.key`) für die lokale HTTPS-Entwicklung.
- **index.tsx:** (siehe Listing 5.4) — bereits zuvor beschrieben im Abschnitt zur Einstiegspunktstruktur.
- **App.css:** Enthält benutzerdefinierte globale CSS-Regeln zur Gestaltung zentraler UI-Komponenten, darunter Header, Buttons, Bilder, Effekte sowie Media Queries zur responsiven Darstellung.
- **images:** Enthält statische Bilddateien, die in der Benutzeroberfläche verwendet werden.
- **i18n.ts:** Initialisiert die Mehrsprachigkeit mittels `i18next` mit englischen und deutschen Übersetzungen sowie automatischer Sprachenerkennung.
- **App.tsx:** Zentrale Komponente der Anwendung, die das Routing und die Navigation steuert. Sie bindet verschiedene Seiten und Komponenten ein und integriert die Authentifizierung mit Okta. Header und Footer sorgen für das Layout, während geschützte Routen durch `Security` verwaltet werden.
- **components:** Enthält die wiederverwendbaren UI-Komponenten der Anwendung, wie Header, Footer, Hauptseite, Suchseite und weitere funktionale Elemente. Diese Komponenten bilden die Benutzeroberfläche und sind modular aufgebaut, um Wartbarkeit und Erweiterbarkeit zu gewährleisten.
- **access:** Beinhaltet die Integration und Steuerung des Okta-Authentifizierungswidgets. Die Komponenten `OktaLoginWidget` und `OktaSignInWidget` ermöglichen das Anmelden, Handhaben von Login-Events und Weiterleitung nach erfolgreicher Authentifizierung.
- **lib:** Enthält die Datei `oktaConfig.ts`, die die Einstellungen für die Okta-Authentifizierung definiert, wie Client-ID, Autorisierungsserver (Issuer), Redirect-URI und Berechtigungen (Scopes). Diese Konfiguration ist zentral für die Anbindung der Okta-Authentifizierung im Frontend.
- **models:** Enthält Klassen, die zentrale Datenstrukturen der Anwendung modellieren, wie z. B. `Book`. Sie dienen als Grundlage für den strukturierten Datenaustausch innerhalb des Frontends sowie zwischen Frontend und Backend.
- **locales:** Beinhaltet sprachspezifische JSON-Dateien (`en`, `de`) für die Internationalisierung der UI mittels `react-i18next`.

- **.env**: Diese Datei definiert Umgebungsvariablen für das Projekt. Dazu gehören Pfade zu SSL-Zertifikat und -Schlüssel für die HTTPS-Kommunikation im lokalen Entwicklungsumfeld sowie die Backend-URL (`REACT_APP_API_URL`) für die React-Anwendung.

5.2.3 Internationalisierung mit i18next

Zur Umsetzung der Mehrsprachigkeit im Frontend wurde das `i18next`-Framework in Kombination mit `react-i18next` und `i18next-browser-languagedetector` verwendet. Nach der Installation dieser Abhängigkeiten wird in der Datei `i18n.ts` (siehe Listing 5.6) die Initialisierung des Übersetzungssystems vorgenommen.

```

1 import i18n from "i18next";
2 import { initReactI18next } from "react-i18next";
3 import LanguageDetector from "i18next-browser-languagedetector";

5 import translationEN from "./locales/en/translation.json";
6 import translationDE from "./locales/de/translation.json";

8 const resources = {
9     en: { translation: translationEN },
10    de: { translation: translationDE }
11};

13 i18n
14 .use(LanguageDetector) // Detektiert die Sprache des Benutzers
15 .use(initReactI18next) // Bindet i18next an React
16 .init({
17     resources,
18     fallbackLng: "en", // verwenden Sie Englisch als Fallback-Sprache
19     interpolation: {
20         escapeValue: false // React bereits vor XSS-Angriffen schützt
21     }
22 });

24 export default i18n;

```

Listing 5.6: Initialisierung von i18next in `i18n.ts`

Hier ist eine Zeile-für-Zeile-Erklärung der Datei `i18n.ts`, die die Internationalisierung der React-Anwendung initialisiert:

- `import i18n from "i18next";`
Importiert die Hauptbibliothek `i18next`, die die Internationalisierung ermöglicht.
- `import { initReactI18next } from "react-i18next";`
Importiert die React-spezifische Integration, um `i18next` mit React zu verbinden.

- `import LanguageDetector from "i18next-browser-languagedetector";`
Importiert ein Modul zur automatischen Erkennung der Sprache des Benutzers im Browser.
- `import translationEN from "./locales/en/translation.json";`
Importiert die englischen Übersetzungen aus der JSON-Datei.
- `import translationDE from "./locales/de/translation.json";`
Importiert die deutschen Übersetzungen aus der JSON-Datei.
- `const resources = {
 en: { translation: translationEN },
 de: { translation: translationDE }
};`
Definiert die verfügbaren Sprachressourcen mit den jeweiligen Übersetzungen.
- `i18n
 .use(LanguageDetector)
 .use(initReactI18next)
 .init({
 resources,
 fallbackLng: "en",
 interpolation: {
 escapeValue: false
 }
 });`
Initialisiert `i18next` mit: automatischer Spracherkennung, React-Integration, Sprachressourcen, Standard-Fallbacksprache Englisch, und deaktiviert Escape-Mechanismen, da React bereits sicher ist.
- `export default i18n;`
Exportiert die konfigurierte Instanz, damit sie im Projekt verwendet werden kann.

Die sprachspezifischen Übersetzungen sind in den Ordner `en` und `de` als strukturierte `.json`-Dateien organisiert.

Ein typisches Beispiel (siehe Listing 5.7) für die Verwendung in einer React-Komponente ist:

```

1  import { useTranslation } from 'react-i18next';
3  const { t } = useTranslation();
5  <button>{t("checkout.thankYouReview")}</button>

```

Listing 5.7: Beispielhafte Nutzung von `useTranslation`

In den entsprechenden JSON-Dateien sind die Übersetzungen wie folgt definiert:

en.json:

```

1   "checkout": {
2     "thankYouReview": "Thank you for your review"
3   }

```

Listing 5.8: Englische Übersetzung in `en.json`

de.json:

```

1   "checkout": {
2     "thankYouReview": "Vielen Dank für deine Bewertung"
3   }

```

Listing 5.9: Deutsche Übersetzung in `de.json`

Wie in Listing 5.8 und Listing 5.9 zu sehen, werden die Schlüssel `checkout.thankYouReview` mit den jeweiligen Texten in Englisch und Deutsch belegt.

Zur Laufzeit kann die Sprache über ein Dropdown-Menü gewechselt werden. Im `Header.tsx` (siehe Listing 5.10) der Anwendung befindet sich ein Sprachumschalter, der die aktuelle Sprache visuell mit einem Icon anzeigt und dem Benutzer den Wechsel zwischen Deutsch und Englisch ermöglicht:

```

1   <div className="dropdown">
2     <button>
3       {i18n.language === 'de' ? '[DE]' : '[EN]'}
4     </button>
5     <ul className="dropdown-menu">
6       <li><button onClick={() => i18n.changeLanguage('en')}>[EN] English
7           </button></li>
8       <li><button onClick={() => i18n.changeLanguage('de')}>[DE] Deutsch
9           </button></li>
10      </ul>
11    </div>

```

Listing 5.10: Sprachumschalter im Header

Hier ist die Zeilen-für-Zeilen-Erklärung des Sprachumschalters:

- `<div className="dropdown">`: Definiert einen Dropdown-Container für die Sprachwahl.
- `<button>`: Der Button zeigt die aktuell ausgewählte Sprache an. Die Darstellung erfolgt dynamisch: Falls die Sprache Deutsch ist (`i18n.language === 'de'`), wird `[DE]` angezeigt, sonst `[EN]`.

- `<ul className="dropdown-menu">`: Die Dropdown-Liste mit Sprachoptionen.
- `<button onClick={() => i18n.changeLanguage('en')}>`: Ein Button zum Wechseln auf Englisch. Beim Klick wird die Sprache zu Englisch gewechselt.
- `<button onClick={() => i18n.changeLanguage('de')}>`: Ein Button zum Wechseln auf Deutsch. Beim Klick wird die Sprache zu Deutsch gewechselt.

Damit wird gewährleistet, dass die Benutzeroberfläche abhängig von der gewählten Sprache dynamisch angepasst wird.

5.3 Datenbankmodell und Persistenz

In diesem Abschnitt werden das Datenbankschema sowie die Umsetzung der Datenpersistenz mit Spring Data und Hibernate erläutert.

5.3.1 Datenbankschema

Die Anwendung verwendet ein relationales MySQL-Datenbanksystem zur Speicherung persistenter Informationen wie Büchern, Ausleihen, Zahlungen, Bewertungen und Nachrichten.

Die folgenden Tabellen wurden modelliert (siehe Abbildung 5.3):

- `book`: Enthält Informationen zu Büchern, einschließlich Titel, Autor, Beschreibung, Kategorie, Bild sowie verfügbarer Exemplare.
- `checkout`: Repräsentiert aktive Ausleihen. Jede Ausleihe ist mit einem Benutzer (über die E-Mail-Adresse) und einem Buch (über `book_id`) verknüpft. Enthält außerdem Ausleih- und Rückgabedatum.
- `review`: Enthält Nutzerbewertungen zu Büchern. Jede Bewertung ist einem Benutzer und einem Buch zugeordnet. Felder umfassen Bewertung (Zahl), Datum und Bewertungstext.
- `messages`: Dient der Kommunikation zwischen Benutzern und Administratoren. Enthält Fragen, Antworten sowie den Status (`closed`).
- `history`: Speichert vergangene Ausleihen. Hält Buchinformationen (z. B. Titel, Autor, Beschreibung, Bild) sowie Ausleih- und Rückgabedatum fest.

- **payment**: Speichert Zahlungsinformationen, einschließlich Betrag und E-Mail-Adresse des Benutzers.

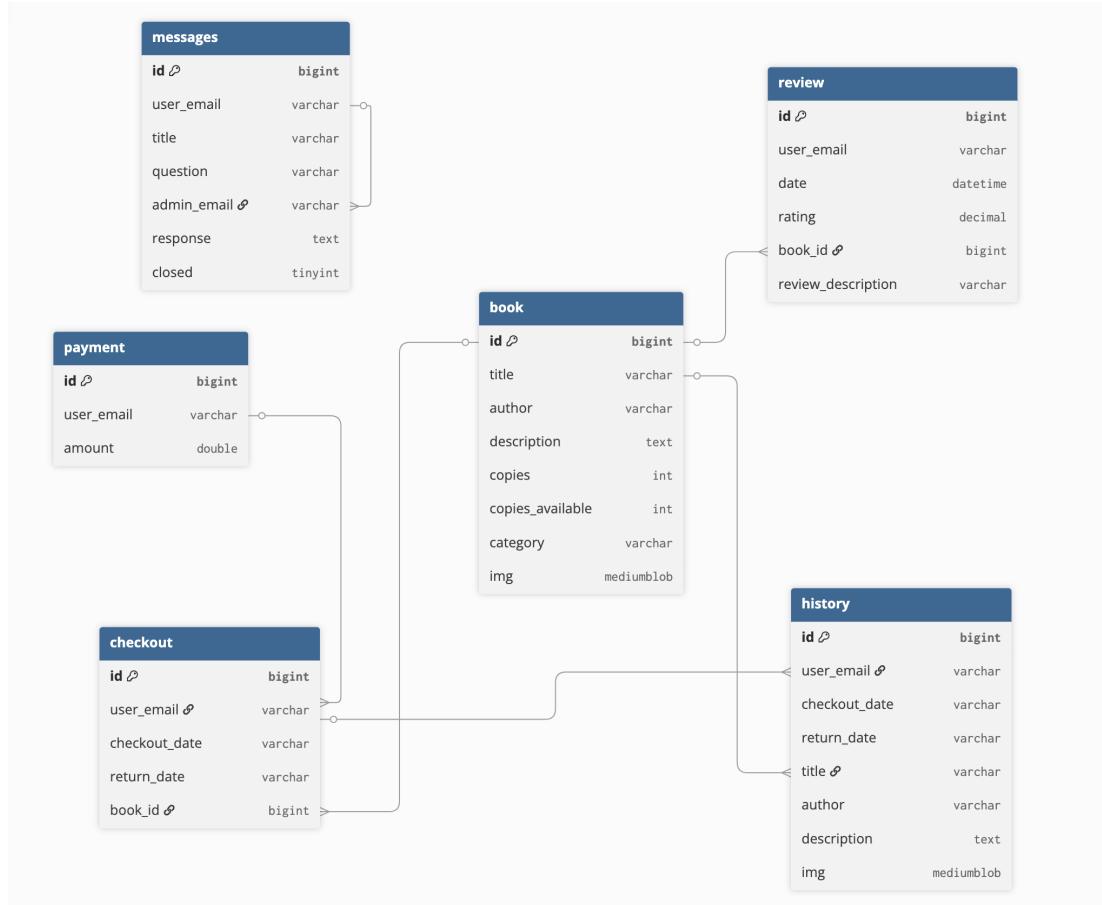


Abbildung 5.3: ER-Modell der Datenbank

Die logischen Beziehungen zwischen den Tabellen lassen sich wie folgt beschreiben:

- Jede **checkout**-Instanz referenziert genau ein Buch über **book_id**.
- Jede **review** ist einem Buch (**book_id**) und einem Benutzer (**user_email**) zugeordnet.
- Eine **message** verknüpft einen Benutzer mit einem Administrator über die jeweiligen E-Mail-Adressen.
- Ein **history**-Eintrag entspricht einer früheren Ausleihe eines Benutzers und speichert redundante Buchdaten.

- **payment**-Einträge entstehen nur bei verspäteter Rückgabe und sind über die Benutzer-E-Mail mit der Ausleihe verknüpft.

5.3.2 Datenpersistenz mit Spring Data und Hibernate

Die Datenpersistenz wird mit Spring Boot in Verbindung mit Hibernate und Spring Data umgesetzt. Die Entitäten werden über Annotationen wie `@Entity`, `@Table`, `@Column`, `@Id`, und Beziehungen wie `@OneToOne` definiert.

Spring Data JPA und Spring Data REST übernehmen automatisch die Abbildung der Entitäten auf die Datenbanktabellen sowie die Bereitstellung von REST-APIs.

Beispiel: Entity-Klasse für Payments

```

1  @Entity
2  @Table(name = "payment")
3  @Data
4  public class Payment {

6      @Id
7      @GeneratedValue(strategy = GenerationType.IDENTITY)
8      private Long id;

10     @Column(name = "user_email")
11     private String userEmail;

13     @Column(name = "amount")
14     private double amount;
15 }
```

Listing 5.11: JPA-Entity Payment

Diese Klasse stellt ein einfaches Beispiel für eine Entity dar, wie sie in Spring Data JPA und Hibernate verwendet wird. Die Annotationen übernehmen die automatische Zuordnung der Klasse zu einer Datenbanktabelle und ermöglichen eine saubere Trennung zwischen Domänenlogik und Datenpersistenz. Hier eine kurze Erklärung der wichtigsten Komponenten:

- `@Entity`: Markiert die Klasse als JPA-Entity. Hibernate erkennt sie dadurch als persistente Klasse, die einer Tabelle in der Datenbank entspricht.
- `@Table(name = "payment")`: Gibt explizit an, dass die Klasse mit der Datenbanktabelle `payment` verknüpft ist. Falls die Annotation fehlt, würde standardmäßig der Klassennamen als Tabellenname verwendet werden.
- `@Id`: Definiert das Feld `id` als Primärschlüssel der Tabelle.
- `@GeneratedValue(strategy = GenerationType.IDENTITY)`: Legt fest, dass der Primärschlüssel von der Datenbank automatisch (z.B. mittels `AUTO_INCREMENT`) generiert wird. Dies ist typisch für MySQL.

- `@Column(name = "...")`: Weist jedes Attribut explizit einer Spalte in der Tabelle zu. Das ist besonders nützlich, wenn sich Feld- und Spaltennamen unterscheiden (z. B. `userEmail` → `user_email`).
- `@Data` (aus Lombok): Generiert automatisch Getter, Setter, `equals()`, `hashCode()` und `toString()`-Methoden, um Boilerplate-Code zu vermeiden. Dies erhöht die Lesbarkeit und spart Entwicklungszeit.

Beispiel: Automatische Repository-Methoden mit Spring Data JPA and rest

```

1  public interface PaymentRepository extends JpaRepository<Payment, Long> {
2      Payment findByUserEmail(String userEmail);
3  }
```

Listing 5.12: Payments-Repository-Schnittstelle

Dieses Interface definiert die Datenzugriffsschicht für die `Payment`-Entität. Durch die Erweiterung von `JpaRepository<Payment, Long>` stellt Spring automatisch alle grundlegenden CRUD-Operationen bereit. `Payment` bezeichnet die Entitätsklasse, `Long` ist der Datentyp des Primärschlüssels.

Die Methode `findByUserEmail(String userEmail)` nutzt die Konventionen von Spring Data JPA: Anhand des Methodennamens erkennt Spring automatisch, dass nach dem Attribut `userEmail` gesucht werden soll, und generiert im Hintergrund die entsprechende SQL-Abfrage.

Da Spring Data REST als Abhängigkeit eingebunden ist, werden automatisch REST-Endpunkte für das Repository verfügbar gemacht – ohne dass man eigene Controller oder Services definieren muss. Die Endpunkte folgen dabei einer standardisierten Struktur:

- GET `/payments` → Alle Zahlungen abrufen
- POST `/payments` → Neue Zahlung erstellen
- GET `/payments/1` → Zahlung mit ID 1 abrufen
- DELETE `/payments/1` → Zahlung mit ID 1 löschen

So ermöglichen Spring Data JPA und Spring Data REST gemeinsam eine saubere Trennung von Datenmodell und Zugriffsschicht – bei minimalem Implementierungsaufwand.

6

Implementierung

Im Rahmen dieses Kapitels werden exemplarisch zwei wesentliche Funktionalitäten der Webanwendung behandelt – der Ausleihprozess sowie das Verfahren zur Rückgabe eines Buches.

6.1 Ausleihprozess eines Buches

Im Folgenden wird der Ausleihprozess eines Buches aus Sicht des Backends sowie des Frontends detailliert beschrieben.

6.1.1 Backend-Prozess

In diesem Abschnitt wird der gesamte Backend-Prozess der Buchausleihe detailliert beschrieben. Die folgende Darstellung umfasst die beteiligten Repositories, die Service-Schicht inklusive aller Hilfsmethoden sowie den REST-Endpunkt im Controller.

Zugriff auf Daten: Repositories

1. PaymentRepository

Da das `PaymentRepository` bereits im vorherigen Kapitel (siehe Listing 5.12 im Zusammenhang mit Spring Data REST und JPA vorgestellt wurde, wird an dieser Stelle lediglich darauf verwiesen. Es wird für die Überprüfung offener Zahlungen verwendet.

2. CheckoutRepository

Für den Ausleihprozess wurde in diesem Repository (siehe Listing 6.1) diese Methoden genutzt:

```

1  public interface CheckoutRepository extends JpaRepository<Checkout,
2      Long> {
3      List<Checkout> findByUserEmail(String userEmail);
4      Checkout findByUserEmailAndBookId(String userEmail, Long bookId);
5  }

```

Listing 6.1: CheckoutRepository.java

Die erste Methode ruft alle ausgeliehenen Bücher eines bestimmten Nutzers anhand seiner E-Mail-Adresse ab. Die zweite Methode ruft einen einzelnen Checkout-Eintrag aus der Datenbank ab, der zur angegebenen E-Mail-Adresse des Benutzers und der ID des Buches gehört.

Geschäftslogik: Die Methode checkoutBook im BookService

Die Methode `checkoutBook` (siehe Listing 6.2) enthält den gesamten Ablauf der Buchausleihe. Im Folgenden wird die Methode vollständig dargestellt und im Anschluss schrittweise erklärt:

```

1  public Book checkoutBook(String userEmail, Long bookId) throws
2      Exception {
3      Book book = bookRepository.findById(bookId)
4          .orElseThrow(() -> new BookNotFoundException("Book with ID
5              " + bookId + " is not available."));
6
7      checkAvailability(book, userEmail);
8
9      List<Checkout> userCheckouts = checkoutRepository.
10         findByUserEmail(userEmail);
11      boolean hasOverdueBooks = hasOverdueBooks(userCheckouts);
12
13      Payment payment = paymentRepository.findByUserEmail(userEmail)
14          ;
15
16      if ((payment != null && payment.getAmount() > 0) || (payment
17          != null && hasOverdueBooks)) {
18          throw new Exception("The loan has been blocked due to
19              outstanding payments or overdue books.");
20      }
21
22      if (payment == null) {
23          createZeroPayment(userEmail);
24      }
25
26      decrementBookStock(book);
27      createCheckoutRecord(userEmail, book);
28
29      return book;
30  }

```

Listing 6.2: checkoutBook() Methode im BookService.java

Erklärung:

- **Zeile 2–3:** Das Buch wird anhand der ID geladen. Wenn es nicht existiert, wird eine Ausnahme geworfen.
- **Zeile 5:** Es wird geprüft, ob das Buch verfügbar ist und noch nicht vom Benutzer ausgeliehen wurde.
- **Zeile 7:** Alle bisherigen Ausleihen des Benutzers werden geladen.
- **Zeile 8:** Es wird überprüft, ob überfällige Bücher dabei sind.
- **Zeile 10:** Die Zahlungsinformationen des Benutzers werden geladen.
- **Zeile 12–14:** Falls offene Zahlungen oder überfällige Bücher vorhanden sind, wird eine Sperre ausgelöst.
- **Zeile 16–18:** Wenn kein Zahlungseintrag vorhanden ist, wird einer mit 0 Euro erstellt.
- **Zeile 20:** Der Buchbestand wird um eins reduziert.
- **Zeile 21:** Ein neuer Ausleihdatensatz wird erstellt.
- **Zeile 23:** Das Buchobjekt wird zurückgegeben.

Hilfsmethoden:

```

1     private void checkAvailability(Book book, String userEmail) {
2         if (checkoutRepository.findByUserEmailAndBookId(userEmail,
3             book.getId()) != null) {
4             throw new BookNotAvailableException("The book has already
5                 been borrowed.");
6         }
7         if (book.getCopiesInStock() <= 0) {
8             throw new BookNotAvailableException("No copies available
9                 for loan.");
10        }
11    }

```

Listing 6.3: checkAvailability()

Diese Methode 6.3 Prüft, ob das Buch bereits vom Benutzer ausgeliehen wurde und Verhindert Ausleihe, wenn keine Kopien mehr verfügbar sind.

```

1     private boolean hasOverdueBooks(List<Checkout> checkouts) {
2         LocalDate today = LocalDate.now();
3
4         for (Checkout checkout : checkouts) {
5             LocalDate returnDate = LocalDate.parse(checkout.getReturnDate());
6             if (returnDate.isBefore(today)) {
7                 return true;
8             }
9         }
10        return false;
11    }

```

Listing 6.4: hasOverdueBooks()

Diese Methode 6.4 überprüft, ob in der übergebenen Liste von Checkout-Einträgen mindestens ein Buch enthalten ist, dessen Rückgabedatum vor dem heutigen Datum liegt – also überfällig ist.

```

1  private void createZeroPayment(String userEmail) {
2      Payment newPayment = new Payment();
3      newPayment.setUserEmail(userEmail);
4      newPayment.setAmount(0.0);
5      paymentRepository.save(newPayment);
6  }

```

Listing 6.5: createZeroPayment()

Diese Methode 6.5 erstellt eine neue Zahlung mit dem Betrag 0.0 für die angegebene Benutzer-E-Mail und speichert sie in der Datenbank über das PaymentRepository.

```

1  private void decrementBookStock(Book book) {
2      book.setCopiesInStock(book.getCopiesInStock() - 1);
3      bookRepository.save(book);
4  }

```

Listing 6.6: decrementBookStock()

Diese Methode 6.6 reduziert den Lagerbestand des übergebenen Buchs um eins und speichert die Änderung in der Datenbank.

```

1  private void createCheckoutRecord(String userEmail, Book book) {
2      Checkout checkout = new Checkout(userEmail, LocalDate.now().
3          toString(),
4          LocalDate.now().plusDays(CHECKOUT_PERIOD_DAYS).toString(), book.
5          getId());
6      checkoutRepository.save(checkout);
7  }

```

Listing 6.7: createCheckoutRecord()

Diese Methode 6.7 erstellt einen neuen Ausleihdatensatz für das angegebene Buch und den Benutzer mit dem aktuellen Datum und speichert ihn in der Datenbank.

Schnittstelle zum Frontend: BookController

Der REST-Endpunkt (siehe 6.8) empfängt Anfragen zur Ausleihe und leitet sie an den Service weiter:

```

1  @PutMapping("/secure/checkout")
2  public Book checkoutBook(Authentication authentication,
3      @RequestParam Long bookId) throws Exception {
4      String userEmail = authentication.getName();
5      return bookService.checkoutBook(userEmail, bookId);

```

```
6     }
```

Listing 6.8: checkoutBook() im BookController.java

- `@PutMapping`: Definiert den Pfad zum Ausleih-Endpunkt.
- `Authentication`: Ermöglicht Zugriff auf die Benutzerinformationen über Spring Security.
- Die Methode ruft den Ausleihprozess im Service auf und gibt das ausgeliehene Buch zurück.

6.1.2 Frontend-Prozess

Im Frontend wird der Ausleihprozess in der Komponente `CheckoutBook` umgesetzt. Hierzu werden zwei zentrale Methoden verwendet: `checkoutBook()` für den API-Aufruf und `renderButton()` zur Darstellung der passenden Benutzeroberfläche.

API-Aufruf zur Ausleihe eines Buches

Die folgende Methode (siehe 6.9) übernimmt den API-Aufruf an das Backend, um ein Buch auszuleihen:

```
1  async function checkoutBook() {
2      const apiUrl = `${process.env.REACT_APP_API_URL}/books/secure/
3          checkout?bookId=${bookId}`;
4      const response = {
5          method: 'PUT',
6          headers: {
7              Authorization: `Bearer ${authState?.accessToken?.
8                  accessToken}`,
9              'Content-Type': 'application/json'
10         }
11     };
12     const res = await fetch(apiUrl, response);
13     if (!res.ok) {
14         setShowError(true);
15     }
16     setShowError(false);
17     setIsBookCheckedOut(true);
18 }
```

Listing 6.9: checkoutBook() Methode in CheckoutBook.tsx

Erklärung:

- `apiUrl`: Baut die URL für den API-Endpunkt mit dem Buch-ID als Parameter.
- `response`: Enthält die Methode (PUT) und den Authentifizierungs-Token.
- `fetch()`: Sendet die Anfrage an das Backend.
- `!res.ok`: Falls der Server einen Fehler zurückgibt, wird ein Fehler angezeigt.

- `setShowError(false)`: Versteckt die Fehlermeldung, falls alles korrekt lief.
- `setIsBookCheckedOut(true)`: Setzt den Status, dass das Buch nun ausgeliehen ist.

Benutzeroberfläche: Auswahl der richtigen Aktion

Die Methode `renderButton()` (siehe 6.10) entscheidet, welcher Button oder Hinweis dem Benutzer angezeigt wird. Es werden vier Möglichkeiten unterschieden:

```

1  function renderButton() {
2      if (props.isAuthenticated) {
3          if (!props.isCheckedOut && props.currentLoans < 5) {
4              return (
5                  <button onClick={() => props.checkoutBook()}>
6                      className="btn btn-success btn-lg">
7                          {t("checkout.checkout")}</button>
8                  )
9          } else if (props.isCheckedOut) {
10              return (<p><b>{t("checkout.alreadyCheckedOut")}</b></p>
11                  >)
12          } else if (!props.isCheckedOut) {
13              return (<p className="text-danger">{t("checkout.
14                  maxReached")}</p>
15          )
16      }
17      return (
18          <Link to="/login" className="btn btn-success btn-lg">
19              {t("checkout.signIn")}</Link>
20      )
21  }

```

Listing 6.10: `renderButton()` in `CheckoutAndReviewBox.tsx`

Erklärung:

- `if (props.isAuthenticated)`: Zeigt die Optionen nur für eingeloggte Benutzer.
- `!props.isCheckedOut && props.currentLoans < 5`: Ausleih-Button erscheint, wenn das Buch noch nicht ausgeliehen ist und weniger als 5 aktive Ausleihen bestehen.
- `props.isCheckedOut`: Zeigt Meldung, dass Buch bereits ausgeliehen wurde.
- `!props.isCheckedOut` bei 5 Ausleihen: Zeigt Fehlermeldung zur max. Grenze.
- `else`: Bei nicht eingeloggten Nutzern erscheint der Login-Link.

Einbindung des Buttons in die Oberfläche

Die Methode `renderButton()` wird innerhalb des return-Blocks der Komponente `CheckoutAndReviewBox.tsx` wie folgt eingebunden: `renderButton()`.

6.2 Verfahren zur Rückgabe eines Buches

Im Folgenden wird das Verfahren zur Rückgabe eines Buches sowohl im Backend als auch im Frontend erläutert.

6.2.1 Backend-Prozess

Der Rückgabeprozess nutzt die bereits beschriebenen Repositories 6.1 und 5.12.

Geschäftslogik: Die Methode `returnBook` im `BookService.java`

Die zentrale Logik zur Rückgabe befindet sich in der folgenden Methode 6.11:

```

1  public void returnBook(String userEmail, Long bookId) throws
2      ParseException {
3      Book book = bookRepository.findById(bookId)
4          .orElseThrow(() -> new BookNotAvailableException("Book with ID
5              " + bookId + " is not available."));
6
7      Checkout checkout = checkoutRepository.
8          findByUserEmailAndBookId(userEmail, bookId);
9      if (checkout == null) {
10          throw new BookNotAvailableException("No active checkout
11              found for book with ID " + bookId + ".");
12      }
13
14      book.setCopiesInStock(book.getCopiesInStock() + 1);
15      bookRepository.save(book);
16
17      LocalDate dueDate = LocalDate.parse(checkout.getReturnDate());
18      LocalDate today = LocalDate.now();
19      long daysOverdue = ChronoUnit.DAYS.between(dueDate, today);
20
21      if (daysOverdue > 0) {
22          Payment payment = paymentRepository.findByUserEmail(
23              userEmail);
24          if (payment == null) {
25              payment = new Payment();
26              payment.setUserEmail(userEmail);
27              payment.setAmount(0.0);
28          }
29          payment.setAmount(payment.getAmount() + daysOverdue * 2);
30          paymentRepository.save(payment);
31      }
32
33      checkoutRepository.deleteById(checkout.getId());
34
35      History history = History.builder()
36          .userEmail(userEmail)
37          .checkoutDate(checkout.getCheckoutDate())
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
765
766
767
767
768
768
769
769
770
771
772
773
774
775
775
776
777
777
778
778
779
779
780
781
782
783
784
785
785
786
787
787
788
788
789
789
790
791
792
793
794
795
795
796
797
797
798
798
799
799
800
801
802
803
803
804
805
805
806
806
807
807
808
808
809
809
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
```

```

33         .returnedDate(LocalDate.now().toString())
34         .title(book.getTitle())
35         .author(book.getAuthor())
36         .overview(book.getOverview())
37         .image(book.getImage())
38         .build();
39
40     historyRepository.save(history);
41 }
```

Listing 6.11: returnBook() Methode in BookService.java

Erklärung:

- `findById()`: Holt das Buch-Objekt aus der Datenbank oder wirft eine Exception.
- `findByUserEmailAndBookId()`: Holt die Ausleihe dieses Nutzers für das Buch.
- Wenn kein Checkout existiert, wird eine Ausnahme geworfen.
- Das Buch wird wieder auf Lager erhöht und gespeichert.
- Das Rückgabedatum wird geprüft, um eventuelle Verspätungsgebühren zu berechnen.
- Falls überfällig, wird der Betrag berechnet (z. B. 2 Euro pro Tag) und gespeichert.
- Der Checkout-Eintrag wird gelöscht.
- Abschließend wird ein History-Eintrag mit allen Buch- und Ausleihinformationen gespeichert.

Rückgabe-Endpunkt im Controller

Die Controller-Methode(siehe 6.12) ist für die Entgegennahme der Anfrage aus dem Frontend zuständig:

```

1  @PutMapping("/secure/return")
2  public void returnBook(Authentication authentication,
3  @RequestParam Long bookId) throws ParseException {
4      String userEmail = authentication.getName();
5      bookService.returnBook(userEmail, bookId);
6  }
```

Listing 6.12: returnBook() Endpoint in BookController.java

Erklärung:

- Der Endpunkt verarbeitet eine PUT-Anfrage mit dem Buch-ID als Parameter.
- Die E-Mail des Benutzers wird über das Authentication-Objekt extrahiert.
- Der Service übernimmt die Logik für die Rückgabe.

6.2.2 Frontend-Prozess

Die Rückgabe eines Buches erfolgt durch einen Button innerhalb der Benutzeroberfläche der ausgeliehenen Bücher. Der Button ist wie folgt definiert (siehe 6.13):

```

1  <button
2    onClick={() => props.returnBook(props.userLoanSummary.book.id)}
3    type="button"
4    data-bs-dismiss="modal"
5    className="btn btn-outline-success rounded-pill py-1 px-2 mb-2
6      small"
7  >
8    {t("loanDetails.returnBook")}
9  </button>

```

Listing 6.13: returnBook-Button in LoanDetailsModal.tsx

Erklärung:

- `onClick={() => props.returnBook(props.userLoanSummary.book.id)}`: Beim Klick wird die Funktion `returnBook()` mit der `bookId` des ausgeliehenen Buches aufgerufen.
- `data-bs-dismiss="modal"`: Schließt das Bootstrap-Modal nach dem Klick automatisch.
- `className="..."`: Definiert das Styling des Buttons (grün, umrandet, abgerundet, klein).
- `t("loanDetails.returnBook")`: Holt den lokalisierten Text für „Buch zurückgeben“ aus der Übersetzungsdatei.

Beim Klicken auf den Button wird die Methode `returnBook()` (siehe 6.14) aufgerufen, welche als `async` Funktion in der Datei `loans.tsx` definiert ist:

```

1  async function returnBook(bookId: number) {
2    const apiUrl = `${process.env.REACT_APP_API_URL}/books/secure/
3      return?bookId=${bookId}`;
4    const requestOptions = {
5      method: 'PUT',
6      headers: {
7        'Authorization': `Bearer ${authState?.accessToken?.
8          accessToken}`,
9        'Content-Type': 'application/json'
10      }
11    };
12    const response = await fetch(apiUrl, requestOptions);
13    if (!response.ok) {
14      throw new Error('Something went wrong while returning the
15        book!');
16    }
17    setCheckout(!checkout);
18  }

```

Listing 6.14: returnBook() in Loans.tsx

Erklärung:

- `const apiUrl = ...`: Erstellt die API-URL mit Query-Parameter `bookId`, um das Rückgabe-Ende im Backend anzusprechen.

- `const requestOptions = {}`: Konfiguriert die HTTP-Anfrage mit Methode und Headern.
- `method: 'PUT'`: Gibt an, dass es sich um eine PUT-Anfrage handelt (für Rückgabe geeignet).
- `Authorization: Bearer ...`: Hängt das JWT-Token im Header an, um die Anfrage zu authentifizieren.
- `'Content-Type': 'application/json'`: Gibt das Format der übertragenen Daten an.
- `const response = await fetch(...)`: Führt die Anfrage asynchron aus und wartet auf die Antwort.
- `if (!response.ok)`: Überprüft, ob ein Fehler vom Server zurückgegeben wurde.
- `throw new Error(...)`: Löst bei Fehlern eine Ausnahme mit passender Meldung aus.
- `setCheckout(!checkout)`: Aktualisiert den `checkout`-State, um die Oberfläche neu zu laden bzw. den Rückgabezustand zu reflektieren.

Beispiel

Dieses Kapitel bietet einen umfassenden Überblick über die Funktionalität und Benutzerfreundlichkeit der Anwendung. Anhand einer Reihe detaillierter Screenshots wird die Bedienung des Systems anschaulich dargestellt und analysiert. Dadurch erhalten die Leserinnen und Leser einen praxisnahen Einblick in den Umgang mit der Anwendung sowie die einzelnen Arbeitsschritte.

7.1 Startseite

Die Startseite der Anwendung dient als zentraler Einstiegspunkt für alle Nutzergruppen. Sie bietet einen Überblick über die wichtigsten Funktionen der App, wie die Buchsuche, aktuelle Empfehlungen und den Zugang zu weiterführenden Diensten. Die Benutzeroberfläche ist übersichtlich gestaltet und ermöglicht eine intuitive Navigation durch die verschiedenen Inhalte. In diesem Teil wird gezeigt, wie die Startseite strukturiert ist und welche interaktiven Elemente den Nutzerinnen und Nutzern zur Verfügung stehen.

7.1.1 Header und Navigation

Die folgende Abbildung 7.1 zeigt den Aufbau des Headers mit dem dynamischen Navigationsmenü, das sich je nach Benutzerrolle unterscheidet:



Abbildung 7.1: Kopfzeile der App *Libranova*

- **Besucher:** Anzeige des *Libranova*-Logos sowie der Menüpunkte **Startseite** und **Bücher Suchen**. Rechts oben befindet sich ein **Login**-Button. Außerdem

steht ein Dropdown-Menü zur Auswahl der Sprache (Deutsch oder Englisch) zur Verfügung.

- **Eingeloggte Nutzer:** Zusätzlich zu den Besucher-Menüpunkten sind **Bibliotheksaktivität** (Einblick in ausgeliehene Bücher und Ausleihhistorie) sowie **Überfällige Gebühren** (Anzeige möglicher Gebühren für verspätete Rückgaben) sichtbar. Rechts oben wird ein **Logout**-Button angezeigt.
- **Administratoren:** Alle Menüpunkte der eingeloggten Nutzer plus der Menüpunkt **Admin**, welcher Verwaltungsfunktionen für den Buchbestand (Hinzufügen, Ändern, Löschen) und die Bearbeitung von Kundenanfragen umfasst.

7.1.2 Footer

Die folgende Abbildung 7.2 zeigt die Fußzeile der Anwendung. Sie enthält das Copyright © Libranova App, Inc sowie die Menüpunkte **Startseite** und **Bücher suchen**.



Abbildung 7.2: Kopfzeile der App *Libranova*

7.1.3 Hauptteil

Karussell und Buch-Browser

Die folgende Abbildung 7.3 zeigt ein Karussell zur Darstellung ausgewählter Bücher sowie einen Button, der zur Suchseite für Bücher weiterleitet.

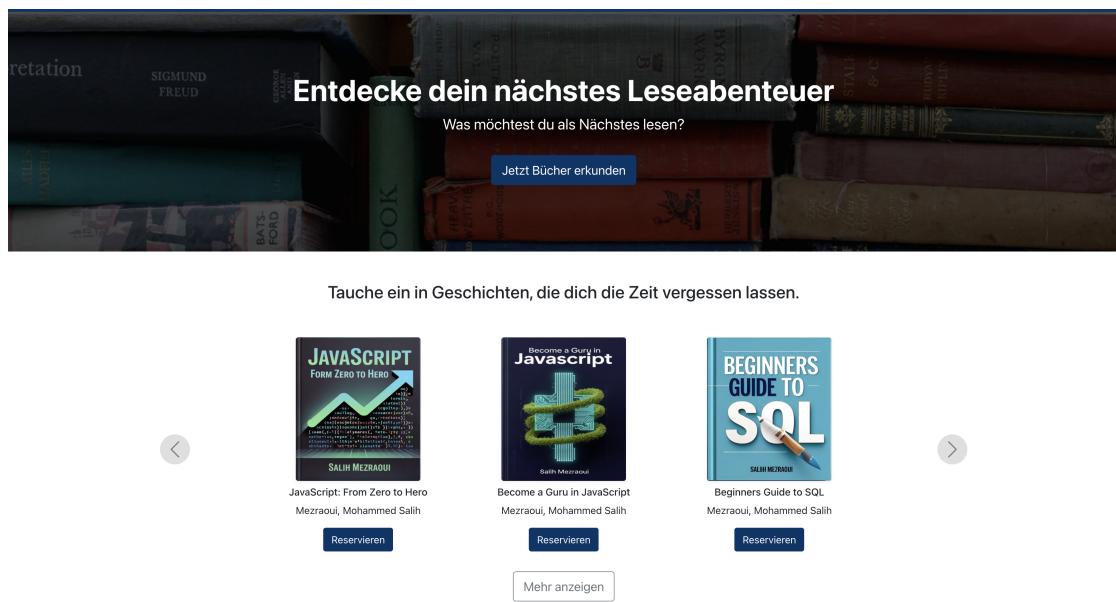


Abbildung 7.3: Karussell und Buch-Browser-Schaltfläche

Die folgende Abbildung 7.4 zeigt einen Link zu den Bibliotheksdiensten, der angezeigt wird, wenn der Benutzer angemeldet ist. Andernfalls erscheint die Option ‚Registrieren‘.

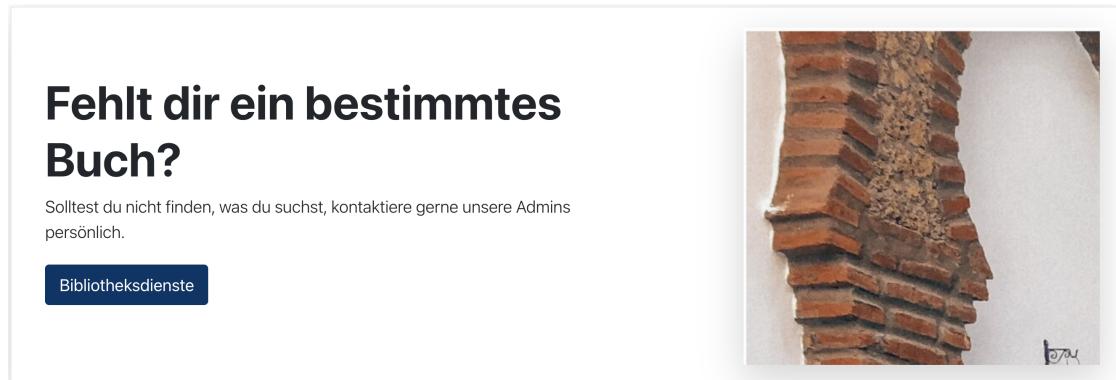


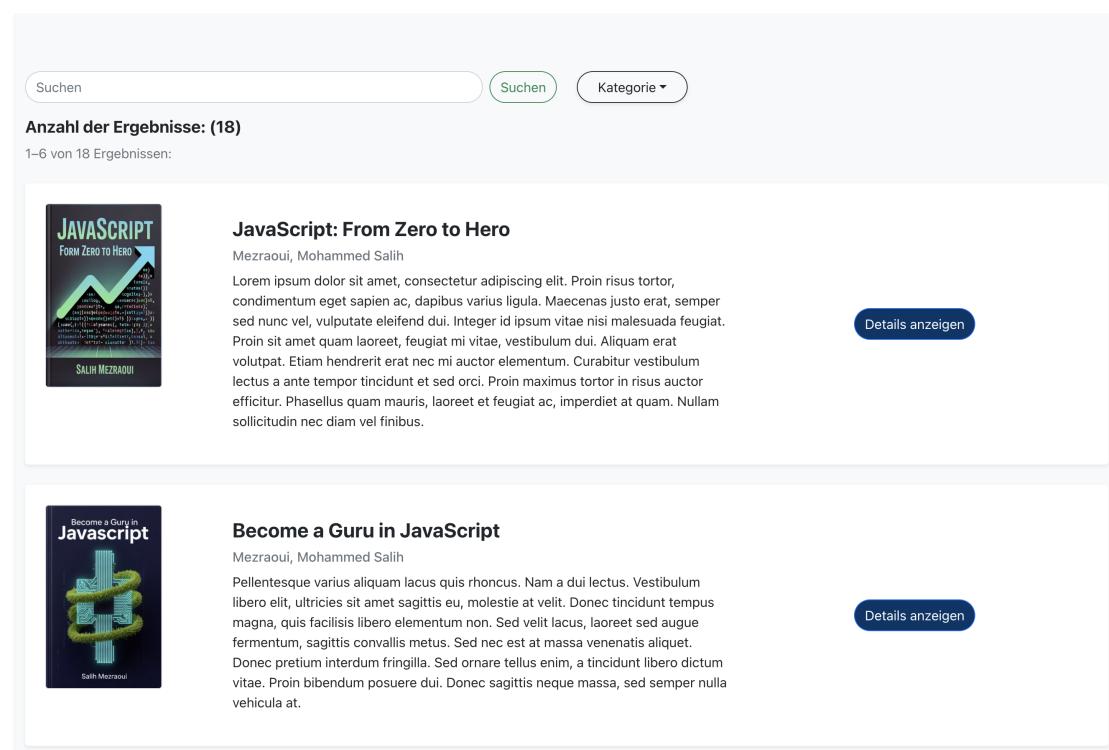
Abbildung 7.4: Karussell und Buch-Browser-Schaltfläche

7.2 Seitenübersicht

Dieser Abschnitt gibt einen Überblick über zentrale Benutzeroberflächen der Anwendung, insbesondere die Buchsuchseite und die Buchdetailseite, und beschreibt deren wichtigste Funktionselemente.

7.2.1 Buchsuche

Die untenstehende Abbildung 7.5 zeigt die Benutzeroberfläche der Suchseite. Sie enthält ein Suchfeld mit Schaltfläche sowie ein Dropdown-Menü zur Auswahl von Kategorien. Nutzer können entweder nach Stichwörtern, Kategorien oder einer Kombination aus beiden suchen. Die Suchergebnisse werden in paginierter Form dargestellt und beinhalten jeweils den Buchtitel, den Autor, eine Kurzbeschreibung sowie eine Schaltfläche zur Detailansicht des jeweiligen Buches.



The screenshot shows a search results page with the following layout:

- Header:** A search bar with the placeholder "Suchen" and a green "Suchen" button, followed by a "Kategorie ▾" button.
- Section Header:** "Anzahl der Ergebnisse: (18)"
- Text:** "1–6 von 18 Ergebnissen:"
- Result 1:**
 - Thumbnail:** Book cover for "JavaScript: From Zero to Hero" by Salih Mezraoui.
 - Title:** "JavaScript: From Zero to Hero"
 - Author:** Mezraoui, Mohammed Salih
 - Text:** "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus."
 - Button:** "Details anzeigen"
- Result 2:**
 - Thumbnail:** Book cover for "Become a Guru in JavaScript" by Salih Mezraoui.
 - Title:** "Become a Guru in JavaScript"
 - Author:** Mezraoui, Mohammed Salih
 - Text:** "Pellentesque varius aliquam lacus quis rhoncus. Nam a dui lectus. Vestibulum libero elit, ultricies sit amet sagittis eu, molestie at velit. Donec tincidunt tempus magna, quis facilisis libero elementum non. Sed velit lacus, laoreet sed augue fermentum, sagittis convallis metus. Sed nec est at massa venenatis aliquet. Donec pretium interdum fringilla. Sed ornare tellus enim, a tincidunt libero dictum vitae. Proin bibendum posuere dui. Donec sagittis neque massa, sed semper nulla vehicula at."
 - Button:** "Details anzeigen"

Abbildung 7.5: Benutzeroberfläche der Suchseite

7.2.2 Buchseite

Die untenstehende Abbildung 7.6 zeigt die Benutzeroberfläche der Buchseite.



Neueste Rezensionen

example2user@email.com	April 17, 2025	★★★★★
Second books is pretty good book overall		
testuser@email.com	June 4, 2025	★★★★★
didn't cover what i wanted		

[Alle Rezensionen anzeigen](#)

Abbildung 7.6: Benutzeroberfläche der Buchseite

Die Buchdetailseite stellt umfassende Informationen zu einem einzelnen Buch bereit und enthält folgende Elemente:

- Darstellung des Buchcovers.
- Anzeige des Buchtitels und des Autors.
- Buchbeschreibung mit der Möglichkeit, zwischen deutscher und englischer Sprache zu wechseln, unabhängig von der Spracheinstellung der Gesamtanwendung.
- Bewertungssystem zur Anzeige der durchschnittlichen Nutzerbewertung.
- Rechte Seitenleiste mit folgenden Informationen:
 - Anzahl der vom aktuellen Benutzer ausgeliehenen Exemplare,
 - Anzahl der derzeit verfügbaren Exemplare,
 - Gesamtanzahl der im Bestand befindlichen Exemplare.

- Übersicht der Nutzerbewertungen zum Buch.
- Link zur vollständigen Liste aller Reviews.

7.3 Bibliotheksaktivität

In diesem Abschnitt werden die Funktionen zur Verwaltung aktueller und vergangener Ausleihen dargestellt.

7.3.1 Ausleihen

Die untenstehende Abbildung 7.7 zeigt das Design der **Ausleihen** Seite.

Aktuelle Ausleihen

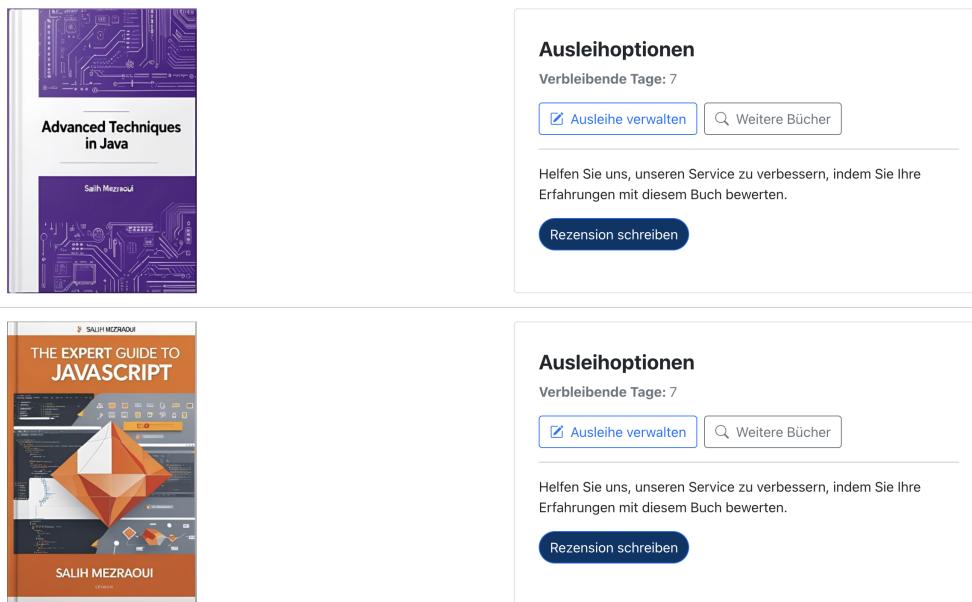


Abbildung 7.7: Benutzeroberfläche der Ausleihseite

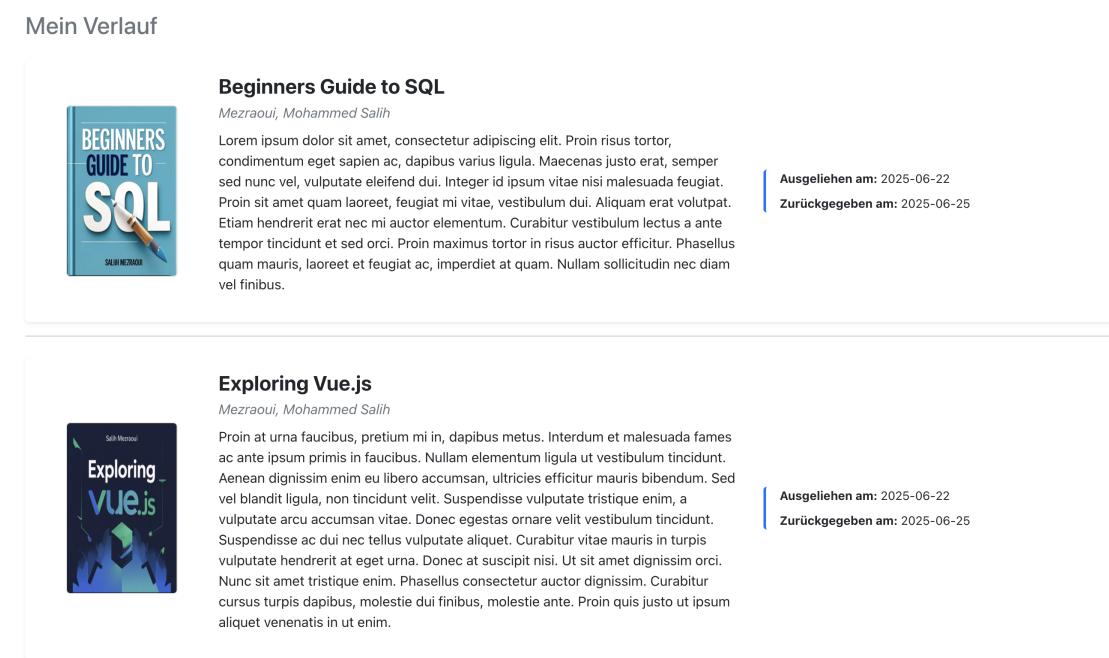
Die Ausleiheseite ermöglicht es Nutzerinnen und Nutzern, ihre aktuell ausgeliehenen Bücher zu verwalten. Sie enthält folgende Elemente:

- Liste aller aktuell ausgeliehenen Bücher.
- Anzeige der verbleibenden Tage bis zur Rückgabe jedes Buches.

- Verwaltungsoptionen pro Buch:
 - Rückgabe des Buches,
 - Verlängerung der Leihfrist.
- Schaltfläche zur Suche nach weiteren Büchern.
- Link zum Verfassen einer Rezension für das jeweilige Buch.

7.3.2 Ausleihhistorie

Die untenstehende Abbildung 7.8 zeigt das Design der Seite **Ausleihverlauf**. Diese Seite enthält die Historie der vom Nutzer ausgeliehenen Bücher, einschließlich des Ausleih- und Rückgabedatums.



The screenshot displays the 'Ausleihverlauf' (Loan History) page with two entries:

Mein Verlauf

Beginners Guide to SQL
Mezraoui, Mohammed Salih

Loan history for 'Beginners Guide to SQL':

- Ausgeliehen am: 2025-06-22
- Zurückgegeben am: 2025-06-25

Exploring Vue.js
Mezraoui, Mohammed Salih

Loan history for 'Exploring Vue.js':

- Ausgeliehen am: 2025-06-22
- Zurückgegeben am: 2025-06-25

Abbildung 7.8: Benutzeroberfläche der Ausleihverlaufsseite

7.4 Bibliotheksdienste

In diesem Abschnitt werden die Bibliotheksdienste vorgestellt, insbesondere die Benutzeroberfläche zur Übermittlung von Anfragen sowie der persönliche Nachrichtenverlauf mit den jeweiligen Antworten.

Die folgende Abbildung 7.9 zeigt die Benutzeroberfläche, über die Benutzer einzelne Anfragen oder Nachrichten an die Administratoren der Bibliothek senden können.

The screenshot shows a web-based message composition interface. At the top, a dark header bar contains the text 'Neue Nachricht senden' in white. Below this, the interface is divided into two main sections: 'Betreff' (Subject) and 'Anfrage' (Query). The 'Betreff' section contains a text input field with the placeholder 'Betreff eingeben'. The 'Anfrage' section contains a larger text input field with the placeholder 'Schreiben Sie Ihre Nachricht...'. At the bottom of the interface is a dark blue button labeled 'Nachricht absenden' (Send message).

Abbildung 7.9: Benutzeroberfläche zum Versenden von Anfragen

Die Abbildung 7.10 veranschaulicht den Verlauf sämtlicher ausgetauschter Nachrichten, einschließlich der gestellten Fragen und der dazugehörigen Antworten.

The screenshot displays a list of exchanged messages. The top message is from 'testuser@email.com' with the subject 'Fall #1: example title' and the content 'sample question'. The response is from 'Administrator: admin@libranova.com' with the content 'Here is the response'. The second message is from 'testuser@email.com' with the subject 'Fall #2: samle title' and the content 'Sample Question'. The response is from 'Administrator: admin@libranova.com' with the content 'Here is ur response'.

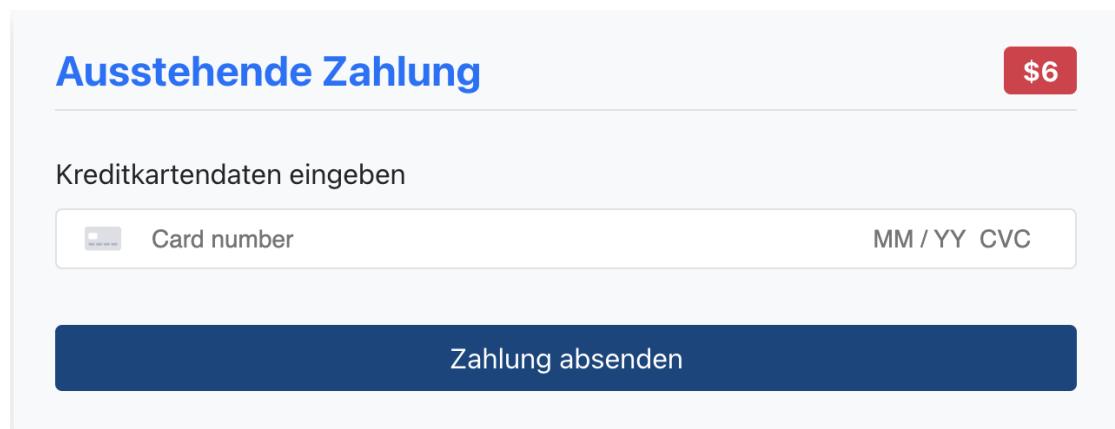
Ihr Nachrichtenverlauf	
Fall #1: example title	
testuser@email.com	
sample question	
Antwort	
Administrator: admin@libranova.com	
Here is the response	
 Fall #2: samle title	
testuser@email.com	
Sample Question	
Antwort	
Administrator: admin@libranova.com	
Here is ur response	

Abbildung 7.10: Nachrichtenverlauf zwischen Nutzer und Bibliothek

7.5 Bezahlungsseite

In diesem Abschnitt wird die Benutzeroberfläche zur Verwaltung von Zahlungsinformationen und offenen Gebühren vorgestellt.

Abbildung 7.11 zeigt das Layout, das angezeigt wird, wenn ein Benutzer ausstehende Zahlungen zu begleichen hat.



The screenshot shows a payment interface with the following elements:

- Ausstehende Zahlung** (Outstanding payment) in blue text at the top left.
- \$6** in a red box at the top right.
- Kreditkartendaten eingeben** (Enter credit card data) in bold black text.
- Card number** input field with a placeholder icon.
- MM / YY CVC** input field.
- Zahlung absenden** (Send payment) button in blue.

Abbildung 7.11: Benutzeroberfläche bei ausstehenden Zahlungen

Wenn keine offenen Gebühren vorliegen, wird dem Nutzer die in Abbildung 7.12 dargestellte Ansicht präsentiert. Sie bestätigt, dass derzeit keine Zahlungen erforderlich sind.



Abbildung 7.12: Benutzeroberfläche bei keinen offenen Zahlungen

7.6 Admin-Bereich

Der Admin-Bereich bietet eine dedizierte Oberfläche zur Verwaltung der Bibliotheksressourcen und zur Interaktion mit Benutzeranfragen. In diesem Abschnitt werden die Verwaltungsfunktionen dargestellt, die einem Administrator zur Verfügung stehen.

7.6.1 Neues Buch hinzufügen

Die erste Funktion (siehe 7.13) ermöglicht es dem Administrator, neue Bücher in das System aufzunehmen. Hierzu gibt er relevante Informationen wie Titel, Autor, Kategorie, eine kurze Beschreibung sowie die Anzahl der verfügbaren Exemplare an. Zusätzlich kann ein Bild des Buchcovers hochgeladen werden. Nach dem Ausfüllen der Felder wird durch Klicken auf die Schaltfläche *"Buch hinzufügen"* ein neuer Eintrag erstellt.

Abbildung 7.13: Benutzeroberfläche bei keinen offenen Zahlungen

7.6.2 Bücher verwalten

Der Administrator kann bestehende Bücher verwalten, indem er die Anzahl der verfügbaren Exemplare anpasst oder Bücher vollständig aus dem System entfernt. Die folgende Abbildung 7.14 zeigt die Verwaltungsoberfläche, über die solche Änderungen vorgenommen werden können.

Gesamtanzahl Bücher: 18

1-6 von 18 Ergebnissen werden angezeigt

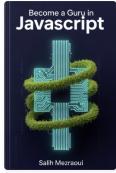
JavaScript: From Zero to Hero



Autor: Mezraoui, Mohammed Salih

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin risus tortor, condimentum eget sapien ac, dapibus varius ligula. Maecenas justo erat, semper sed nunc vel, vulputate eleifend dui. Integer id ipsum vitae nisi malesuada feugiat. Proin sit amet quam laoreet, feugiat mi vitae, vestibulum dui. Aliquam erat volutpat. Etiam hendrerit erat nec mi auctor elementum. Curabitur vestibulum lectus a ante tempor tincidunt et sed orci. Proin maximus tortor in risus auctor efficitur. Phasellus quam mauris, laoreet et feugiat ac, imperdiet at quam. Nullam sollicitudin nec diam vel finibus.

Become a Guru in JavaScript



Autor: Mezraoui, Mohammed Salih

Pellentesque varius aliquam lacus quis rhoncus. Nam a dui lectus. Vestibulum libero elit, ultricies sit amet sagittis eu, molestie at velit. Donec tincidunt tempus magna, quis facilisis libero elementum non. Sed velit lacus, laoreet sed augue fermentum, sagittis convallis metus. Sed nec est at massa venenatis aliquet. Donec pretium interdum fringilla. Sed ornare tellus enim, a tincidunt libero dictum vitae. Proin bibendum posuere dui. Donec sagittis neque massa, sed semper nulla vehicula at.

 **Gesamt:** 1

 **Verfügbar** 1

Anzahl erhöhen Anzahl verringern

 **Löschen**

Abbildung 7.14: Benutzeroberfläche bei keinen offenen Zahlungen

7.6.3 Nachrichten

In diesem Bereich kann der Administrator auf Anfragen von Benutzern antworten. Die Benutzerfrage wird angezeigt und kann direkt über das vorgesehene Textfeld beantwortet werden. Eine Schaltfläche ermöglicht das Senden der Antwort. Die Abbildung 7.15 zeigt die zugehörige Benutzeroberfläche.

 **Ausstehende Fragen**

 **Fall #11: samle title**
testuser@email.com

 : Can I search by combining the category and also the name of the book?

 **Antwort**
Geben Sie Ihre Antwort hier ein...

Antwort senden

Abbildung 7.15: Benutzeroberfläche bei keinen offenen Zahlungen

Anwendungsszenarien

Die im Rahmen dieses Projekts entwickelte Anwendung zur Verwaltung von Büchern ist flexibel einsetzbar und bietet zahlreiche Erweiterungsmöglichkeiten über den ursprünglichen Anwendungsfall hinaus. Dank ihrer modularen Architektur, der Nutzung von REST-APIs und der durchdachten Benutzeroberfläche lässt sich das System leicht an verschiedene Nutzungsszenarien anpassen.

Im Folgenden werden exemplarisch mehrere Anwendungsbereiche vorgestellt, in denen das System sinnvoll eingesetzt oder erweitert werden könnte. Dazu zählen insbesondere der Bildungsbereich und E-Learning-Plattformen, die mobile Nutzung durch Erweiterung zu nativen Apps sowie innovative Funktionen wie personalisierte Buchempfehlungen, Fernleihe sowie Echtzeit-Reservierungsbenachrichtigungen. Diese Szenarien verdeutlichen das Potenzial des Systems, weit über die klassische Bibliotheksverwaltung hinaus einen Mehrwert für Nutzerinnen und Nutzer zu schaffen.

8.1 Bildungs- und E-Learning-Plattformen

E-Learning-Plattformen und Bildungseinrichtungen könnten das entwickelte System als Backend-Lösung zur Bereitstellung von digitalen Lernmaterialien wie Kursunterlagen, wissenschaftlichen Artikeln oder Videoinhalten nutzen. Die integrierte Bewertungs- und Kommentarfunktion ermöglicht es Lernenden, Inhalte zu bewerten und Rezensionen zu verfassen, was wiederum anderen Nutzerinnen und Nutzern bei der Auswahl geeigneter Materialien hilft. Das System könnte zudem erweitert werden, um PDF-Dateien, E-Books und Videos mit Download- oder Streamingrechten zu verwalten und bereitzustellen. Darüber hinaus könnten individuelle Lernpfade unterstützt werden, indem relevante Ressourcen personalisiert vorgeschlagen werden.

8.2 Mobile Nutzung

Durch die RESTful-Architektur der Anwendung ist eine einfache Erweiterung um eine mobile App möglich. Dies ermöglicht Nutzerinnen und Nutzern, bequem von unterwegs auf ihre ausgeliehenen Bücher, Rezensionen und weiteren Inhalten zuzugreifen. Die mobile Nutzung steigert die Flexibilität und Benutzerfreundlichkeit der Anwendung erheblich, indem sie den Zugriff jederzeit und überall ermöglicht – sei es auf iOS- oder Android-Geräten.

8.3 Personalisierte Empfehlungen

Eine zukünftige Erweiterung des Systems könnte die Implementierung KI-basierter Vorschläge sein, die Nutzerinnen und Nutzern auf Basis ihrer Lesehistorie und Präferenzen individuell zugeschnittene Buchempfehlungen anbieten. Dies könnte durch die Integration von Machine-Learning-Algorithmen realisiert werden, die das Nutzerverhalten analysieren. Dadurch würde die Nutzererfahrung verbessert und die Nutzung des Systems attraktiver gestaltet.

8.4 Fernleihe und Reservierungsbenachrichtigungen

Das System könnte durch die Integration von Fernleihfunktionen erweitert werden, sodass Nutzerinnen und Nutzer Bücher aus dem Katalog anderer Bibliotheken ausleihen können. Automatisierte Anfragen und Rückgabeprozesse würden diesen Vorgang effizient gestalten.

8.5 Reservierungsbenachrichtigungen

Das System kann erweitert werden, um Nutzern Echtzeit-Benachrichtigungen zu senden, sobald ein reserviertes Buch wieder verfügbar ist. Diese Funktion verbessert die Nutzerzufriedenheit, da Interessenten sofort informiert werden und somit ihre Ausleihe schneller planen können. Die Integration solcher Benachrichtigungen kann über E-Mail, Push-Nachrichten oder andere Kommunikationskanäle erfolgen.

Resümee und Ausblick

In dieser Arbeit wurde die Konzeption und Realisierung einer modernen, benutzerfreundlichen und sicheren Bibliotheksmanagement-Plattform namens LibraNova vorgestellt. Das Ziel war es, eine skalierbare und wartbare Lösung zu entwickeln, die sowohl den Anforderungen der Nutzerinnen und Nutzer als auch der Bibliothekadministration gerecht wird. Dabei stand die Integration aktueller Webtechnologien wie Spring Boot im Backend, React mit TypeScript im Frontend sowie eine umfassende Sicherheitsarchitektur mit Okta, JWT, OAuth2 und OpenID Connect im Fokus. Zudem wurde die Plattform um eine Zahlungsfunktion erweitert, die die Stripe API nutzt, um beispielsweise etwaige Gebühren reibungslos und sicher abzuwickeln.

Die Anwendung ist mehrsprachig ausgelegt und unterstützt sowohl Deutsch als auch Englisch, um eine breitere Nutzerbasis anzusprechen. Darüber hinaus legt das System großen Wert auf Responsiveness, sodass die Plattform auf verschiedenen Endgeräten — sei es Desktop, Tablet oder Smartphone — optimal genutzt werden kann. Die Umsetzung erfolgte durch modulare Architekturen, die REST-APIs nutzen, um eine flexible Anpassung an verschiedene Nutzungsszenarien zu ermöglichen. Wesentliche Funktionen wie die Buchsuche, das Ausleihmanagement, Nutzerbewertungen und die administrative Verwaltung wurden effizient implementiert und durch systematische Tests abgesichert.

Die Plattform unterstützt zudem zukünftige Erweiterungen, etwa Cloud-Bereitstellung, erweiterte Testabdeckung sowie personalisierte Empfehlungen und Benachrichtigungen. Mit LibraNova wurde eine innovative Lösung geschaffen, die den digitalen Wandel im Bibliothekswesen aktiv mitgestaltet und die Nutzererfahrung durch intuitive Bedienung, hohe Sicherheitsstandards sowie eine responsive Gestaltung deutlich verbessert. Das Projekt bildet eine solide Grundlage für zukünftige Entwicklungen und bietet zahlreiche Potenziale zur weiteren Optimierung und Erweiterung.

Literaturverzeichnis

- Bel25. BELL, DONALD: *An introduction to the Unified Modeling Language*. <https://developer.ibm.com/articles/an-introduction-to-uml/>, 2025. Veröffentlicht am 25. Juni 2023, Download am 20.07.2025.
- Boo25. BOOTSTRAP AUTHORS: *Get started with Bootstrap*. <https://getbootstrap.com/docs/5.3/getting-started/introduction/>, 2025. Download am 22. Juli 2025.
- Bra25. BRATSLAVSKY, PAUL: *Top 6 Benefits of Implementing TypeScript*. <https://strapi.io/blog/benefits-of-typescript>, 2025. Veröffentlicht am 22. Februar 2025, Download am 22. Juli 2025.
- Clo25. CLOUDFLARE: *What is HTTPS?* <https://www.cloudflare.com/learning/ssl/what-is-https/>, 2025. Download am 02. Juli 2025.
- Cor25a. CORPORATION, MICROSOFT: *TypeScript for JavaScript Programmers*. <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>, 2025. Download am 21. Juli 2025.
- Cor25b. CORPORATION, ORACLE: *MySQL Workbench Manual - General Information*. <https://dev.mysql.com/doc/workbench/en/wb-intro.html>, 2025. Download am 04. Juli 2025.
- Cor25c. CORPORATION, ORACLE: *What is MySQL?* <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>, 2025. Download am 04. Juli 2025.
- Fac25. FACEBOOK: *Create React App - Getting Started*. <https://create-react-app.dev/docs/getting-started/>, 2025. Download am 03.08.2025.
- Gmb25a. GMBH, THALIA BÜCHER: *Das ist Thalia*. <https://unternehmen.thalia.de/unternehmen/>, 2025. Download am 14. August 2025.

- Gmb25b. GMBH, THALIA BÜCHER: *Fremdsprachige Bücher*. <https://www.thalia.de/kategorie/fremdsprachige-buecher-880/>, 2025. Download am 14. August 2025.
- Gmb25c. GMBH, THALIA BÜCHER: *Lesen & hören so flexibel wie nie*. <https://www.thalia.de/vorteile/thalia-lesen-und-hoeren-app>, 2025. Download am 14. August 2025.
- GT25a. GIT-TEAM: *Git About - Branching and Merging*. <https://git-scm.com/about/branching-and-merging>, 2025. Download am 01. August 2025.
- GT25b. GITHUB-TEAM: *About GitHub*. <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git#about-github>, 2025. Download am 01. August 2025.
- Gup25. GUPTA, LOKESH: *What is REST?* <https://restfulapi.net/>, 2025. Veröffentlicht am 1. April 2025, Download am 19. Mai 2025.
- HT25. HIBERNATE-TEAM: *Idiomatic persistence for Java and relational databases*. <https://hibernate.org/orm/>, 2025. Download am 06.08.2025.
- HTM25a. HTML.COM: *HTML5 Basics For Everyone Tired Of Reading About Deprecated Code*. <https://html.com/html5/>, 2025. Download am 02.06.2025.
- HTM25b. HTML.COM: *Intimidated By CSS? The Definitive Guide To Make Your Fear Disappear*. <https://html.com/css/>, 2025. Download am 02.06.2025.
- i1825. i18NEXT ORGANISATION: *i18next documentation*. <https://www.i18next.com/>, 2025. Download am 2. August 2025.
- IBM25. IBM: *What is a relational database?* <https://www.ibm.com/think/topics/relational-databases>, 2025. Veröffentlicht am 20. Oktober 2021, Download am 04. Juli 2025.
- IRM⁺20. ISLAM, MAZHARUL, SAZZADUR RAHAMAN, NA MENG, BEHNAZ HASSANSHAH, PADMANABHAN KRISHNAN und DANFENG DAPHNE YAO: *Coding Practices and Recommendations of Spring Security for Enterprise Applications*. Seiten 49–57, 09 2020.
- JT25a. JUNIT-TEAM: *Junit 5 User Guide*. <https://docs.junit.org/current/user-guide/>, 2025. Download am 02. August 2025.
- JT25b. JWT-TEAM: *Introduction to JSON Web Tokens*. <https://jwt.io/introduction>, 2025. Download am 19. Juli 2025.

-
- MP25a. META PLATFORMS, INC.: *Build a React app from Scratch*. <https://react.dev/learn/build-a-react-app-from-scratch#step-2-build-common-application-patterns>, 2025. Download am 22. Juli 2025.
- MP25b. META PLATFORMS, INC.: *Describing the UI*. <https://react.dev/learn/describing-the-ui>, 2025. Download am 22. Juli 2025.
- MP25c. META PLATFORMS, INC.: *React — A JavaScript library for building user interfaces*. <https://legacy.reactjs.org/>, 2025. Download am 22. Juli 2025.
- MT25a. MEDIUM-TEAM: *Stripe Integration Using Flutter Web and NestJS*. <https://medium.com/%40akshelar.18119/stripe-integration-using-flutter-web-and-nestjs-4e94ccd33b9f>, 2025. Download am 15. August 2025.
- MT25b. MOCKITO-TEAM: *Tasty mocking Framework for unit tests in Java*. <https://site.mockito.org/>, 2025. Download am 02. August 2025.
- Ora25. ORACLE CORPORATION: *What is Java technology and why do I need it?* https://www.java.com/en/download/help/whatis_java.html, 2025. Download am 19. Mai 2025.
- OT25a. OKTA-TEAM: *Oauth 2.0 and OpenID Connect overview*. <https://developer.okta.com/docs/concepts/oauth-openid/>, 2025. Download am 15. August 2025.
- OT25b. OKTA-TEAM: *What is Okta and what does Okta do?* https://support.okta.com/help/s/article/what-is-okta?language=en_US, 2025. Download am 19. Mai 2025.
- OT25c. OPENID-TEAM: *How OpenId Connect Works*. <https://openid.net/developers/how-connect-works/>, 2025. Download am 19. Juli 2025.
- PT25. POSTMAN-TEAM: *What is Postman?* <https://www.postman.com/product/what-is-postman/>, 2025. Download am 02. August 2025.
- Pur25. PUROHIT, RAKESH: *A Step-by-Step Guide to Implementing React Spring Boot in Your Application*. <https://www.dhiwise.com/post/a-step-by-step-guide-to-implementing-react-spring-boot>, 2025. Veröffentlicht am 17. Juni 2025, Download am 23. Juli 2025.
- RHHH20. RAHMAN, SHAWON, NAZMUL HOSSAIN, MD. ALAM HOSSAIN und MD. ZOBAYER HOSSAIN: *OAuth 2.0: A Framework to Secure the OAuth-Based Service for Packaged Web Application*. Seiten 92–139, 01 2020.

- Spr25a. SPRING-TEAM: *Spring Boot*. <https://spring.io/projects/spring-boot>, 2025. Download am 15. Mai 2025.
- Spr25b. SPRING-TEAM: *Spring Boot*. <https://spring.io/projects/spring-boot>, 2025. Download am 15. Mai 2025.
- ST25a. SPRING-TEAM: *Spring Data JPA*. <https://spring.io/projects/spring-data-jpa>, 2025. Download am 06.08.2025.
- ST25b. SPRING-TEAM: *Spring Data Rest*. <https://spring.io/projects/spring-data-rest>, 2025. Download am 06.08.2025.
- ST25c. STRIPE-TEAM: *Stripe API Documentation*. <https://docs.stripe.com/api>, 2025. Download am 03. Juli 2025.
- Tri25. TRIER, STADTBÜCHEREI: *Informationen für Sie*. <https://www.stadtbumpe-trier.de/informationen-fuer-sie/>, 2025. Download am 14. August 2025.

A

Systemanforderungen

ID	Anforderung (Beschreibung)
FR1	Nutzer können Bücher nach Titel oder Kategorie suchen.
FR2	Nutzer können die Verfügbarkeit eines Buches in Echtzeit einsehen.
FR3	Nutzer können bis zu fünf Bücher gleichzeitig ausleihen.
FR4	Nutzer können ihre ausgeliehenen Bücher einsehen.
FR5	Nutzer können ihre Ausleihhistorie einsehen.
FR6	Nutzer können ein Buch bewerten.
FR7	Nutzer können eine Rezension verfassen.
FR8	Nutzer können Rezensionen anderer Nutzer lesen.
FR9	Das System zeigt eine Übersicht der verfügbaren Bücher im Katalog an.
FR10	Administratoren können neue Bücher zum Katalog hinzufügen.
FR11	Administratoren können die Anzahl der Exemplare eines vorhandenen Buches erhöhen oder verringern.
FR12	Administratoren können Bücher aus dem Katalog löschen.
FR13	Nutzer können sich in ihr persönliches Konto einloggen.
FR14	Nutzer können die Leihfrist eines Buches um bis zu 7 Tage verlängern.
FR15	Nutzer können ausgeliehene Bücher zurückgeben.
FR16	Das System zeigt an, wenn ein Buch nicht verfügbar ist.
FR17	Für kostenpflichtige Dienstleistungen ist eine Zahlungsabwicklung über Stripe integriert.
FR18	Nutzer müssen eingeloggt sein, um ein Buch auszuleihen.
FR19	Nutzer müssen eingeloggt sein, um eine Rezension verfassen zu können.
FR20	Nutzer müssen eingeloggt sein, um ein Buch bewerten zu können.
FR21	Das System zeigt an, wie viele Bücher von den maximal fünf gleichzeitig ausgeliehen sind.
FR22	Das System zeigt die Anzahl der Suchergebnisse basierend auf eingegebenen Suchbegriffen und gewählten Kategorien an.
FR23	Das System zeigt Bücherlisten (Suchergebnisse, ausgeliehene Bücher, Ausleihhistorie) paginiert an.

FR24	Das System muss sicherstellen, dass Nutzer sowohl einen Nachrichtentitel als auch den Nachrichtentext angeben, bevor sie eine Anfrage absenden können.
FR25	Das System muss sicherstellen, dass alle erforderlichen Felder beim Anlegen eines neuen Buches ausgefüllt sind.
FR26	Das System muss sicherstellen, dass Administratoren das Antwortfeld ausfüllen, bevor sie eine Antwort absenden können.
FR27	Das System muss im Ausleihverlauf eines Nutzers sowohl das Ausleihdatum als auch das Rückgabedatum jedes Buches anzeigen.
FR28	Bei verspäteter Rückgabe von Büchern soll das System die fälligen Gebühren automatisch berechnen und über die Stripe-API zur Zahlung auffordern.
FR29	Für jede Buchbeschreibung muss ein Button vorhanden sein, mit dem der Nutzer die Beschreibung zwischen Deutsch und Englisch umschalten kann.
FR30	Wenn ein Buch gelöscht wurde, darf der Benutzer die Ausleihe dieses Buches nicht mehr verlängern können.
FR31	Wenn ein Buch gelöscht wird, muss der Benutzer in der Ausleihliste darüber informiert werden, dass dieses Buch gelöscht wird.

Tabelle A.1: Funktionale Anforderungen von LibraNova

ID	Anforderung (Beschreibung)
NFR1	Das System muss eine responsive Benutzeroberfläche bieten, die auf verschiedenen Geräten und Bildschirmgrößen funktioniert.
NFR2	Die Anwendung muss eine sichere Kommunikation über HTTPS (TLS) gewährleisten, um die Datenübertragung zwischen Frontend und Backend zu schützen (Ports 8443 und 3000).
NFR3	Das System muss sicherstellen, dass Benutzer nur mit gültigen und nicht abgelaufenen Tokens Zugriff auf geschützte Ressourcen erhalten.
NFR4	Die Reaktionszeit der Such- und Filterfunktionen im Frontend darf 2 Sekunden nicht überschreiten, um ein flüssiges Nutzererlebnis zu gewährleisten.
NFR5	Die Anwendung muss grundlegende Barrierefreiheitsanforderungen gemäß WCAG 2.1 (Level AA) erfüllen, um auch Nutzern mit Einschränkungen einen uneingeschränkten Zugang und eine barrierefreie Nutzung zu ermöglichen.
NFR6	Die REST-APIs müssen robust gegen fehlerhafte Eingaben sein und validierte Anfragen verarbeiten, um die Stabilität des Systems zu gewährleisten.
NFR7	Die Backend-Services sollen modular aufgebaut sein, um einfache Wartbarkeit und Erweiterbarkeit zu ermöglichen.
NFR8	Die API-Dokumentation muss aktuell und entwicklerfreundlich sein.

ID	Anforderung (Beschreibung)
NFR9	Die Anwendung muss eine klare Trennung zwischen Benutzerrollen (Nutzer, Administrator) ermöglichen.
NFR10	Die Anwendung muss auf allen gängigen Webbrowersn wie Firefox, Chrome und Safari ohne Einschränkungen lauffähig sein.
NFR11	Der Anmeldevorgang eines Nutzers (ab dem Klick auf den Login-Button bis zur erfolgreichen Authentifizierung und Weiterleitung zur Startseite) darf nicht länger als 3 Sekunden dauern.
NFR12	Die Benutzeroberfläche muss auf allen Seiten konsistente Navigationselemente, Schaltflächen-Designs und Layouts verwenden, um eine einheitliche Nutzererfahrung sicherzustellen.
NFR13	Die Benutzerführung muss intuitiv gestaltet sein, sodass typische Nutzeraktionen wie Buchsuche, Ausleihe oder Bewertung ohne zusätzliche Anleitung verständlich und ausführbar sind.
NFR14	Das System muss sicherstellen, dass nur autorisierte Clients auf geschützte Endpunkte wie <code>/api/books/secure/**</code> oder <code>/api/admin/secure/**</code> zugreifen können.
NFR15	Die Anwendung muss vollständig auf Deutsch und Englisch verfügbar sein, wobei alle Seiten und Benutzeroberflächen entsprechend übersetzt werden.

Tabelle A.2: Nicht-funktionale Anforderungen von LibraNova

ID	Anforderung (Beschreibung)
TR1	Das Backend muss HTTPS-Verbindungen über Port 8443 ermöglichen, mithilfe eines selbstsignierten SSL-Zertifikats.
TR2	Der Zugriff auf geschützte API-Endpunkte muss über JWT-Token abgesichert werden (z., B. <code>/api/books/secure/**</code>).
TR3	CSRF-Schutz muss für REST-APIs deaktiviert werden, da die Authentifizierung stateless über JWT erfolgt.
TR4	Das Backend muss eine RESTful API zur Verfügung stellen, um Daten zwischen Frontend und Backend zu übertragen.
TR5	Die API-Endpunkte müssen CORS-konform konfiguriert sein, sodass ausschließlich Anfragen von der vertrauenswürdigen Frontend-Domain <code>https://localhost:3000</code> zugelassen werden.
TR6	Die Authentifizierung und Autorisierung erfolgt über Okta als Identity Provider unter Verwendung von OAuth2, OpenID Connect und JWT, um sichere und standardisierte Benutzerzugriffe zu gewährleisten.
TR7	Die REST-API muss mit Swagger dokumentiert sein und über Swagger UI erreichbar sein.
TR8	Die Anwendung muss HTTP-Methoden wie POST, PUT, DELETE und PATCH für bestimmte Ressourcen (z. B. Book, Review, Message) deaktivieren, wenn sie nicht benötigt werden.

ID	Anforderung (Beschreibung)
TR9	Die Anwendung muss eine relationale MySQL-Datenbank verwenden (libranova_db), angebunden über JDBC.
TR10	Die Anwendung muss Spring Data JPA zur Datenpersistenz verwenden.
TR11	Die Anwendung muss Entities wie Book, Review und Message als JPA-Entitäten modellieren und mittels Repository-Schnittstellen verfügbar machen.
TR12	Die Anwendung muss Spring Data REST verwenden, um Repository-Schnittstellen automatisch als RESTful API verfügbar zu machen.
TR13	Lombok wird zur Reduzierung von Boilerplate-Code in Entities und anderen Klassen verwendet.
TR14	Die Anwendung muss Spring Boot Web Starter verwenden, um Web-Server-Funktionalitäten und REST-Controller bereitzustellen.
TR15	Die Anwendung muss Unit-Tests mit JUnit und Mocking mit Mockito implementieren, um die Funktionalität des Backends abzusichern.
TR16	Das Frontend muss in React umgesetzt sein und via HTTPS über Port 3000 laufen.
TR17	Das Frontend muss mit dem Backend über REST-APIs kommunizieren.
TR18	Das Frontend muss die vom Backend bereitgestellten Endpunkte nutzen und Authentifizierung über Tokens abwickeln.
TR19	Die Internationalisierung der Anwendung wird im Frontend mit React unter Verwendung der Bibliothek i18next umgesetzt.

Tabelle A.3: Technische Anforderungen von LibraNova

B

Abkürzungsverzeichnis

SQL	Structured Query Language
API	Application Programming Interface
JPA	Java Persistence API
MVC	Model-View-Controller
OAuth2	Open Authorization 2
JWT	JSON Web Token
CORS	Cross-Origin Resource Sharing
CSRF	Cross-Site Request Forgery
ACID	Atomicity, Consistency, Isolation, Durability
CRUD	Create, Read, Update, Delete
ORM	Object-Relational Mapping
JDBC-URL	Java Database Connectivity Uniform Resource Locator
IDE	Integrated Development Environment
SCM	Source Code Management
CVS	Concurrent Versions System
HTTPS	Hypertext Transfer Protocol Secure
UML	Unified Modeling Language
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
SPA	Single-Page Application
UI	User Interface
RxJs	Reactive Extensions for JavaScript
HTTP	Hypertext Transfer Protocol
URI	Uniform Resource Identifier
JSON	JavaScript Object Notation
REST API	Representational State Transfer
DTO	Data Transfer Object

C

Selbstständigkeitserklärung

- Diese Arbeit wurde als Gruppenarbeit angefertigt. Meinen Anteil habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Namen der Mitverfasser:

Imad-Eddine ABDESSAMI, Mohammed Salih MEZRAOUI

Datum

Unterschrift der Kandidatin/des Kandidaten

Datum

Unterschrift der Kandidatin/des Kandidaten