

ShopNook: E-Commerce Warenkorbanwendung mit Spring
Boot

ShopNook: Development of an e-commerce shopping cart application
with Spring Boot

Imad-Eddine Abdessami, Mohammed Salih Mezraoui

Bachelor-Projektarbeit

Betreuer: Prof. Dr. Andreas Lux

Trier, den 29.08.2024

Kurzfassung

Die Arbeit befasst sich mit der Entwicklung einer E-Commerce Warenkorbanwendung, die mit Spring Boot entwickelt wurde.

Ziel des Projekts ist es, einen benutzerfreundlichen Online-Marktplatz zu schaffen, der es Kunden ermöglicht, Produkte zu durchsuchen, Artikel in den Warenkorb zu legen und sichere Zahlungen durchzuführen. Die Anwendung integriert Front-End-Technologien wie Angular mit Back-End-Lösungen in Spring Boot und nutzt eine MySQL-Datenbank zur Verwaltung von Benutzer-, Produkt- und Bestelldaten. Besondere Schwerpunkte liegen auf der sicheren Zahlungsabwicklung über die Stripe-API, der HTTPS-Verschlüsselung und der JWT-basierten Authentifizierung, um ein zuverlässiges und sicheres Einkaufserlebnis zu gewährleisten.

Diese Arbeit liefert wertvolle Einblicke in die Herausforderungen und Lösungen bei der Entwicklung effizienter Online-Shopping-Anwendungen.

Inhaltsverzeichnis

1	Einleitung und Problemstellung	1
1.1	Überblick	1
1.2	Globale Anforderungen	1
1.2.1	Funktionale Anforderungen	1
1.2.2	Nicht-funktionale Anforderungen	2
1.2.3	Technische Anforderungen	2
2	Methodologie und Systementwurf	3
2.1	Entwurf und Konzeption	3
2.1.1	UML	3
2.1.2	Klassendiagramm	3
2.1.3	Sequenzdiagramm	5
2.2	Front-End Technologien	6
2.2.1	HTML/CSS	6
2.2.2	Typescript	6
2.2.3	Angular	6
2.3	Back-End Technologien	8
2.3.1	Java	8
2.3.2	REST-API	8
2.3.3	HTTPS und SSL/TLS	9
2.3.4	Spring Boot	11
2.3.5	Stripe API	13
2.4	Datenbankstruktur	15
2.4.1	Übersicht über relationale Datenbanken und Auswahl von MySQL	15
2.4.2	Datenbankschemaentwurf mit Spring Data JPA	16
2.4.3	Datenpersistenz und Transaktionen	16
2.4.4	MySQL-Workbench und SQL	16
2.4.5	Datenbankverbindung und -konfiguration in Spring Boot	17
2.5	Authentifizierungs- und Autorisierungsprotokolle	17
2.5.1	Okta	17
2.5.2	JWT	17
2.5.3	OAuth2	18

2.5.4	OpenID Connect	18
2.6	Entwicklungsumgebung und Versionskontrolle	18
2.6.1	Entwicklungsumgebung	18
2.6.2	Versionskontrolle	19
3	Implementierung	21
3.1	Startseite	21
3.1.1	header	21
3.1.2	Menu	23
3.2	Online-Shopping	24
3.2.1	Darstellung von Produkten auf der Produktseite	24
3.2.2	Auswahl und Hinzufügen von Produkten zum Warenkorb	24
3.2.3	Warenkorbverwaltung	25
3.2.4	Paginierung und Suchfunktionalität	25
3.3	Bezahlprozess	27
3.3.1	Frontend	27
3.3.2	Backend	28
4	Ergebnisse und Analyse	30
4.1	Startseite	30
4.1.1	Linke Seite	31
4.1.2	Header	31
4.1.3	Hauptinhalt	31
4.1.4	Fußzeile	32
4.2	Warenkorb-Details	32
4.3	Checkout-Seite	33
4.4	Login	36
4.4.1	Auftragsverlauf	37
4.4.2	Mitgliederseite	37
5	Zusammenfassung und Ausblick	39
	Literaturverzeichnis	40
	Abkürzungsverzeichnis	42
	Selbstständigkeitserklärung	43

Abbildungsverzeichnis

2.1	Klassendiagramm für die E-commerce Anwendung ShopNook	4
2.2	Sequenzdiagramm der Anwendung ShopNook mit OAuth 2.0-Autorisierung	5
2.3	Beispielhafte Dateistruktur einer Angular-Anwendung mit verschiedenen Komponenten	7
4.1	Startseite	31
4.2	Warenkorb-Details	32
4.3	Checkout-1	34
4.4	Checkout-2	35
4.5	Purchase-Button	36
4.6	Login	36
4.7	Login-Leiste	37
4.8	Auftragsverlauf	37
4.9	Mitgliederseite	38

Einleitung und Problemstellung

Hier werden folgende Aspekte berücksichtigt:

- Ein kleiner Überblick über die Anwendung ShopNook
- und deren funktionale, nicht funktionale und technische Anforderungen.

1.1 Überblick

ShopNook ist ein internetbasierter Marktplatz, der ein nahtloses und angenehmes Einkaufserlebnis bieten soll. Käufer können ganz einfach eine breite Palette von Produkten erkunden, sie in den Warenkorb legen und ihre Einkäufe sicher abschließen. Die Plattform verfügt über ein attraktives und reaktionsfähiges Design, das eine konsistente und ansprechende Benutzererfahrung auf allen Gerätegrößen gewährleistet.

Neben der benutzerfreundlichen Oberfläche verfügt ShopNook über robuste Funktionen wie ein Bestandsverwaltungssystem, Benutzerkonten mit verschiedenen Berechtigungen und erweiterte Suchoptionen, die den Käufern helfen, genau das zu finden, was sie brauchen. Die Anwendung ist in der Lage, verschiedene administrative Aufgaben zu erledigen und bietet Käufern einen sicheren und effizienten Checkout-Prozess.

1.2 Globale Anforderungen

1.2.1 Funktionale Anforderungen

Die App wird über alle notwendigen Funktionen verfügen, um ein reibungsloses Einkaufserlebnis zu gewährleisten. Die Kunden werden die Möglichkeit haben, sich sicher zu registrieren, einzuloggen und einen Produktkatalog zu erkunden. Zu jedem Produkt werden vollständige Details wie Name, Beschreibung, Preis und Bilder angezeigt. Die Kunden können schnell Artikel in ihren Einkaufskorb legen, den Inhalt ändern und zur Zahlung übergehen. Sichere Kreditkartenzahlungen werden während des Kaufvorgangs über die Stripe-API abgewickelt. Administratoren können die Waren überwachen und die Bestellungen im Auge behalten, um sicherzustellen, dass das Online-Geschäft gut funktioniert.

1.2.2 Nicht-funktionale Anforderungen

Die App wird effizient mit gleichzeitigen Nutzern umgehen und die Ladezeiten der Seite sind minimal. HTTPS-Verschlüsselung und JWT-basierte Authentifizierung stellen die Sicherheit in den Vordergrund. Die Einheitlichkeit der Geräte wird durch eine reaktionsschnelle und einfach zu bedienende Schnittstelle gewährleistet.

1.2.3 Technische Anforderungen

Die vollwertige E-Commerce-App wird mit npm für JavaScript-Abhängigkeiten und Maven für die Java Abhängigkeiten und Build-Prozesse entwickelt. Im Frontend wird Angular verwendet, um eine dynamische und reaktionsschnelle Schnittstelle für die Benutzer bereitzustellen, während im Backend das Spring Boot-Framework verwendet wird, um RESTful APIs zu erstellen. MySQL wird die Datenbank sein, die Benutzer-, Produkt- und Bestelldaten speichert. Git und GitHub werden für die Versionskontrolle verwendet, um Änderungen zu verwalten und die Teamarbeit zu erleichtern.

Methodologie und Systementwurf

Hier werden über folgende Punkte diskutiert:

- Die Entwicklungskonzeption, die wir während der Implementation der Anwendung benutzt haben, sowie die Tools, Technologien und Methodologien.
- Überblick über Spring Boot und seine wichtigsten Funktionalitäten, die auch in dem Projekt angewendet sein werden.
- Detaillierte Erklärung der Struktur und Design der Anwendung (Klassendiagramm, Sequenzdiagramm)
- Datenbanken, UI-Design und unterschiedliche Komponenten des Systems.
- Wie Spring Boot die Entwicklung und Implementation der verschiedenen Module vereinfacht.

2.1 Entwurf und Konzeption

In der Softwareentwicklung spielen Entwurf und Konzeption eine entscheidende Rolle. Diese Phase des Entwicklungsprozesses befasst sich mit der Strukturierung und Planung von Systemen, bevor die eigentliche Implementierung beginnt.

Ziel ist es, ein klares und verständliches Modell zu erstellen, das die Anforderungen und Funktionen eines Systems abbildet. In diesem Kapitel werden UML und einige Diagrammtypen vorgestellt.

2.1.1 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 [Bel23a].

2.1.2 Klassendiagramm

In diesem Abschnitt stellen wir das Klassendiagramm für die Anwendung vor. Das Diagramm (siehe 2.1) veranschaulicht die wichtigsten beteiligten Klassen, ihre Attribute und die Beziehungen zwischen ihnen. Das Klassendiagramm dient

der Darstellung der Struktur des eCommerce-Systems. Es umfasst die folgenden Klassen:

- Customer
- Order
- OrderItem
- Address
- ProductCategory
- Product
- State
- Country

Die Beziehungen zwischen diesen Klassen sind wie folgt:

- Ein **Kunde** kann mehrere **Bestellungen** aufgeben.
- Eine **Bestellung** kann mehrere **Bestellpositionen** enthalten.
- Jede **Bestellposition** ist einem **Produkt** zugeordnet.
- Eine **Bestellung** hat eine **Lieferadresse** und eine **Rechnungsadresse**.
- Ein **Produkt** gehört zu einer **Produktkategorie**.
- Ein **Bundesstaat** gehört zu einem **Land**.

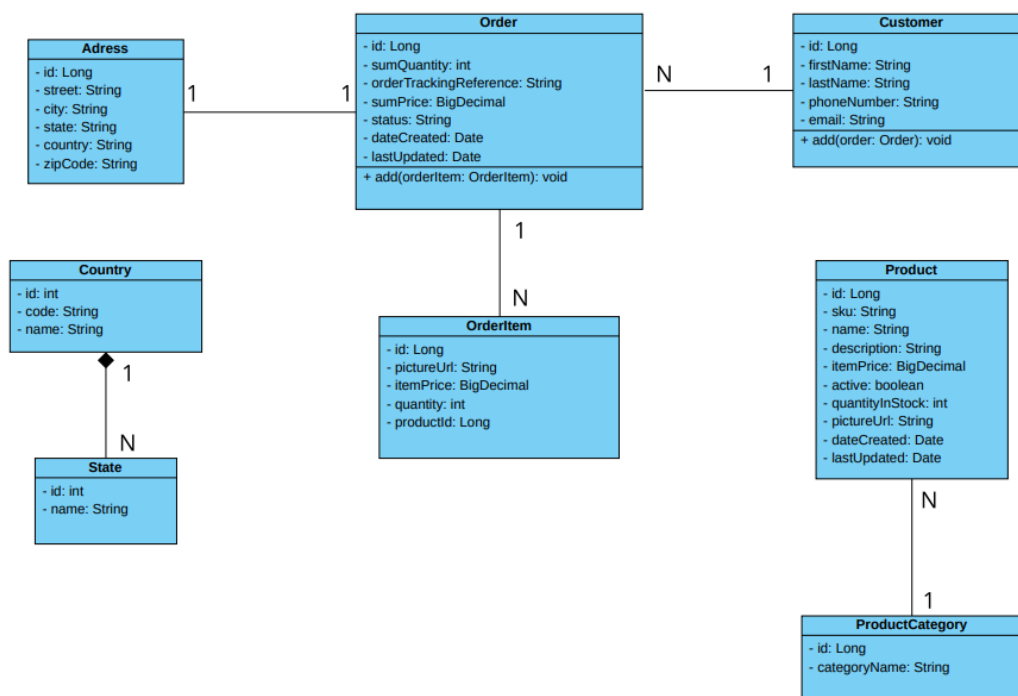


Abbildung 2.1: Klassendiagramm für die E-commerce Anwendung ShopNook

2.1.3 Sequenzdiagramm

Ein Sequenzdiagramm ist eine Art von UML-Diagramm, das den Austausch von Nachrichten zwischen verschiedenen Objekten in einem System im zeitlichen Verlauf darstellt. Es zeigt, wie die verschiedenen Komponenten eines Systems miteinander interagieren, um eine bestimmte Funktion oder einen bestimmten Prozess auszuführen. Sequenzdiagramme sind besonders nützlich, um die Kommunikation und die Reihenfolge der Nachrichten in einem Prozess zu visualisieren. Sie helfen dabei, die Dynamik eines Systems zu verstehen und die Interaktion zwischen den Systemkomponenten zu analysieren [Cor21].

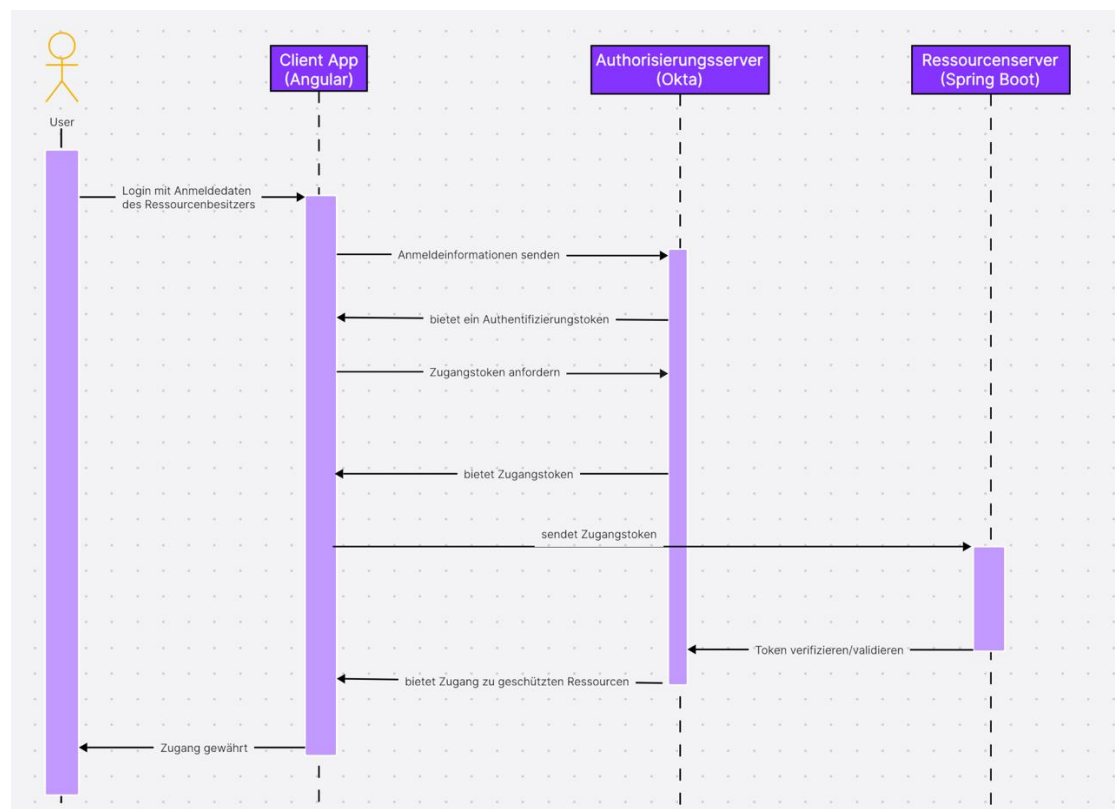


Abbildung 2.2: Sequenzdiagramm der Anwendung ShopNook mit OAuth 2.0-Autorisierung

In dem vorliegenden Sequenzdiagramm 2.2 wird der Ablauf einer Autorisierungsanfrage gezeigt. Es stellt dar, wie ein Benutzer über eine Client-Anwendung mit einem Autorisierungsserver (Okta) und einem Ressourcenserver interagiert, um Zugriff auf geschützte Ressourcen zu erhalten.

Das Diagramm zeigt die verschiedenen Schritte, wie die Anmeldung des Benutzers, die Überprüfung der Anmeldeinformationen durch den Autorisierungsserver, die Ausgabe eines Tokens und die anschließende Validierung dieses Tokens durch den Ressourcenserver, bevor dem Benutzer der Zugang zu den gewünschten Ressourcen gewährt wird.

2.2 Front-End Technologien

Dieses Kapitel befasst sich mit Front-End-Technologien wie HTML, CSS, TypeScript und Angular, die für die Erstellung interaktiver, reaktionsfähiger und visuell ansprechender Benutzeroberflächen in unserer E-Commerce-Anwendung unerlässlich sind.

2.2.1 HTML/CSS

HTML and CSS sind die grundlegenden Technologien zur Erstellung und Gestaltung von Webseiten. HTML liefert die Struktur und den Inhalt einer Webseite, während CSS für die visuelle Darstellung, Layout, Farben und Schriftarten, verwendet wird. Sie ermöglichen es optisch ansprechende und gut strukturierte Webseiten zu erstellen [HTM24].

2.2.2 Typescript

TypeScript¹ ist eine stark typisierte Programmiersprache, die auf JavaScript aufbaut und statische Typdefinitionen hinzufügt. Sie soll die Entwicklung umfangreicher Anwendungen überschaubarer machen, indem sie es den Entwicklern ermöglicht, Fehler schon früh im Entwicklungsprozess zu erkennen, wodurch Fehler reduziert und die Codequalität verbessert werden. [Shu22].

Die wichtigsten Vorteile von TypeScript sind:

- **Statische Typisierung:** Die statische Typisierung ermöglicht es Entwicklern, Variablentypen explizit zu definieren, was zu einer besseren Tooling-Unterstützung führt, z. B. bei der Code-Vervollständigung und beim Refactoring. Es hilft auch bei der Identifizierung von typbezogenen Fehlern während der Kompilierungszeit und nicht zur Laufzeit.
- **Wartbarkeit:** Durch die Hinzufügung von Typdefinitionen wird der Code selbstdokumentierender und leichter verständlich, was in großen Codebasen, in denen mehrere Entwickler zusammenarbeiten, entscheidend ist.

Durch die Wahl von TypeScript für das ShopNook-Projekt wurde es sichergestellt, dass das Frontend der Anwendung robust, wartbar und skalierbar ist und die hohen Anforderungen an eine professionelle E-Commerce-Plattform erfüllt.

2.2.3 Angular

Angular² ist ein Open-Source-Framework für Webanwendungen, das von Google entwickelt und gepflegt wird. Es wird für die Erstellung dynamischer, einseitiger Anwendungen (SPAs) mit TypeScript und HTML verwendet. Das Framework bietet eine robuste Plattform für die Entwicklung komplexer Anwendungen, indem es einen umfassenden Satz von Tools und Funktionen wie Datenbindung, Dependency Injection und eine modulare Architektur bietet [Ang24a].

¹ <https://www.typescriptlang.org>

² <https://angular.io>

Eine der Hauptstärken von Angular ist seine komponentenbasierte Struktur, die es Entwicklern ermöglicht, wiederverwendbare UI-Komponenten zu erstellen, die die Wartbarkeit und Skalierbarkeit verbessern. Darüber hinaus bietet Angular leistungsstarke Funktionen wie reaktive Programmierung mit RxJS³, Zustandsverwaltung und ein reichhaltiges Ökosystem von Bibliotheken, was es zu einer idealen Wahl für Anwendungen auf Unternehmensebene macht.

2.2.3.1 Komponentenbasierte Architektur

Die komponentenbasierte Architektur von Angular gliedert die Anwendung in kleine, in sich geschlossene Einheiten, die als Komponenten bezeichnet werden. Jede Komponente kapselt einen bestimmten Teil der Benutzeroberfläche (UI) und die damit verbundene Logik [Ang24b].

Komponenten sind die wichtigsten Bausteine für Angular-Anwendungen. Wie es in der Abbildung 2.3 bezeichnet wurde besteht jede Komponente aus:

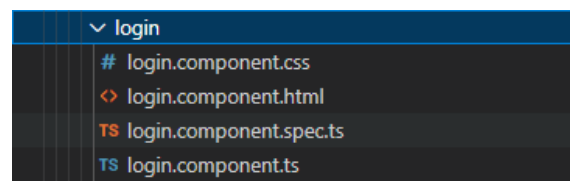


Abbildung 2.3: Beispielhafte Dateistruktur einer Angular-Anwendung mit verschiedenen Komponenten

- einer Komponentendatei `<component-name>.component.ts`
- einer Vorlagedatei `<component-name>.component.html`
- einer CSS-Datei `<component-name>.component.css`
- Einer Datei mit Prüfvorschriften `<component-name>.component.spec.ts`

Dieser modulare Ansatz bringt mit sich mehrere Vorteile wie Wiederverwendbarkeit und Wartbarkeit.

³ <https://rxjs.dev>

2.2.3.2 Single Page Application (SPA)

Angular eignet sich besonders gut für die Erstellung von Single Page Applications (SPAs). In einer SPA lädt die gesamte Anwendung eine einzige HTML-Seite und aktualisiert den Inhalt dynamisch, wenn der Benutzer mit der Anwendung interagiert, ohne dass eine komplette Seite neu geladen werden muss [Ang24c].

Dieser Ansatz bietet mehrere Vorteile:

- **Verbesserte Benutzerfreundlichkeit:** SPAs bieten eine reibungslosere und schnellere Benutzererfahrung, da nur die notwendigen Teile der Seite aktualisiert werden, anstatt die gesamte Seite neu zu laden.
- **Geringere Serverlast:** Sie entlasten den Server, da weniger Ganzseitenanfragen gestellt werden, was zu einer effizienteren Nutzung der Serverressourcen führt.

Durch die Nutzung der komponentenbasierten Architektur und der SPA-Funktionen von Angular kann die Anwendung ShopNook eine nahtlose und reaktionsschnelle Benutzererfahrung bieten, die für die Kundenbindung und -zufriedenheit entscheidend ist.

2.3 Back-End Technologien

In diesem Kapitel werden Back-End-Technologien untersucht, wobei der Schwerpunkt auf Spring Boot liegt, um robuste, skalierbare und sichere serverseitige Logik und APIs für unsere E-Commerce-Anwendung zu erstellen.

2.3.1 Java

Java ist eine weit verbreitete Programmiersprache, die für ihre Plattformunabhängigkeit, Stabilität und umfassende Bibliotheken bekannt ist. Aufgrund ihrer Vielseitigkeit und Leistung wird Java häufig für die Entwicklung von Back-End-Anwendungen verwendet [Jav24]. In der E-Commerce-Anwendung (ShopNook) wird Java genutzt, um eine robuste und skalierbare serverseitige Logik zu gewährleisten.

2.3.2 REST-API

REST ist ein Architekturstil für die Entwicklung vernetzter Anwendungen. Er basiert auf einem zustandslosen, Client-Server- und cachefähigen Kommunikationsprotokoll, und in fast allen Fällen wird das HTTP-Protokoll verwendet. REST-APIs sind so konzipiert, dass sie einfach, leichtgewichtig und skalierbar sind, was sie zu einer beliebten Wahl für Webdienste macht [RES24].

2.3.2.1 Einführung in REST

Angelehnt an [RES24] bieten REST-APIs eine Möglichkeit, über HTTP auf die Funktionalität und Daten einer Anwendung zuzugreifen. Sie folgen den Prinzipien

von REST, die auf Ressourcen und deren Repräsentationen basieren. Jede Ressource wird durch eine eindeutige URI identifiziert und kann durch standardisierte HTTP-Methoden manipuliert werden:

- GET: Abrufen von Ressourcen
- POST: Erstellen neuer Ressourcen
- PUT: Aktualisieren bestehender Ressourcen
- DELETE: Löschen von Ressourcen

REST-APIs sind also aufgrund ihrer Architektur und Funktionsweise weit verbreitet und bieten eine Reihe von Vorteilen:

- **Skalierbarkeit:** Durch das stateless Design können REST-APIs leicht skaliert werden. Jeder HTTP-Request enthält alle notwendigen Informationen, um ihn zu verarbeiten, ohne dass der Server den vorherigen Zustand kennen muss.
- **Flexibilität:** REST-APIs sind flexibel und können mit verschiedenen Datenformaten arbeiten. JSON ist das gebräuchlichste Format, da es leichtgewichtig und gut lesbar ist.
- **Interoperabilität:**⁴ REST-APIs nutzen standardisierte HTTP-Methoden, was ihre Interoperabilität mit verschiedenen Clients und Plattformen gewährleistet.
- **Leichte Integration:** REST-APIs lassen sich leicht in bestehende Systeme integrieren, da sie auf bekannten Webstandards basieren.

Um die Sicherheit von REST-APIs in der ShopNook-Plattform weiter zu gewährleisten, spielt die Implementierung von HTTPS in Kombination mit SSL/TLS eine entscheidende Rolle, da diese Technologien eine sichere und verschlüsselte Kommunikation zwischen dem Client und dem Server ermöglichen.

2.3.3 HTTPS und SSL/TLS

HTTPS ist die sichere Version von HTTP, dem Protokoll, über das die Daten zwischen dem Browser des Kunden und der Website, mit der er verbunden ist, gesendet werden. Das „S“ am Ende von HTTPS steht für „Secure“, was bedeutet, dass die gesamte Kommunikation zwischen dem Kunden und dem Server verschlüsselt ist [HTT24].

SSL/TLS sind allerdings Verschlüsselungsprotokolle, die eine sichere Kommunikation über ein Computernetz ermöglichen. SSL war das ursprüngliche Protokoll, aber es wurde zugunsten von TLS, das sicherer und effizienter ist, veraltet [Bel23b]. Diese Protokolle bieten mehrere wichtige Sicherheitsfunktionen, die im Folgenden näher erläutert werden:

- **Verschlüsselung der Daten:** SSL und TLS verschlüsseln die zwischen dem Client und dem Server übertragenen Daten und stellt sicher, dass die Daten, selbst wenn sie abgefangen werden, nicht von Unbefugten gelesen werden können.

⁴ Interoperabilität ist die Fähigkeit verschiedener Systeme oder Software, zusammenzuarbeiten und Informationen nahtlos auszutauschen[wik].

- **Server-Authentifizierung:** Diese Protokolle überprüfen auch die Identität des Servers und stellen sicher, dass die Clients mit einem legitimen Server und nicht mit einem Betrüger kommunizieren.
- **Datenintegrität:** Sie bieten auch Datenintegrität, indem es sicherstellt, dass die Daten während der Übertragung nicht manipuliert wurden.

In der Anwendung wurde eine HTTPS Konfiguration (siehe 2.1) erstellt, um eine sichere Kommunikation zwischen Clients und dem Server zu gewährleisten. Dazu wurde der Server so eingestellt, dass er auf Port 8443 auf HTTPS-Datenverkehr wartet.

Der Konfigurationsprozess beinhaltet die Aktivierung der SSL-Unterstützung, die für die Verschlüsselung der über das Netzwerk ausgetauschten Daten unerlässlich ist.

Eine Keystore-Datei im PKCS12-Format `shopnook-keystore.p12` wurde verwendet, in der die für die Verschlüsselung erforderlichen SSL/TLS-Zertifikate sicher gespeichert sind. Der Keystore ist durch ein Passwort geschützt (`server.ssl.keystore-password=secret`), so dass der Zugriff auf die kryptografischen Schlüssel nur autorisierten Prozessen vorbehalten ist.

Innerhalb des Schlüsselspeichers wird das Zertifikat durch den Alias **shopnook** identifiziert, auf den in der Konfiguration verwiesen wird, um die SSL-Vorgänge des Servers mit dem richtigen Zertifikat zu verknüpfen. Diese Einrichtung stellt sicher, dass alle zwischen dem Client und dem Server übertragenen Daten verschlüsselt werden, was einen robusten Schutz gegen Abhör- und Man-in-the-Middle-Angriffe bietet.

Darüber hinaus ermöglicht die Konfiguration eine einfache Anpassung an unterschiedliche Umgebungen. So kann beispielsweise der Port der Anwendung je nach Einsatzumgebung geändert werden, indem die Eigenschaft `server.port` angepasst wird. Für die QA-Demo-Umgebung ist der Port auf **9898** eingestellt, der durch Umschalten der Kommentarseiten aktiviert werden kann.

Durch die sorgfältige Konfiguration dieser SSL-Einstellungen stellen wir sicher, dass ShopNook den Industriestandards für Sicherheit entspricht.

```
1      #####
2      #
3      # HTTPS configuration
4      #
5      #####
6      #Define the web server's listening port for HTTPS traffic
7      server.port=8443
8
9      #QA Demo Port
10     #server.port=9898
11
12     #Activate SSL support to secure the application with HTTPS
13     server.ssl.enabled=true
14
15     #Specify the key alias used within the keystore for SSL
16     server.ssl.key-alias=shopnook
17
18     #Keystore path
19     server.ssl.key-store=classpath:shopnook-keystore.p12
20
21     #Password to unlock the keystore and access the SSL keys
22     server.ssl.key-store-password=secret
23
24     #format of the keystore file
25     server.ssl.key-store-type=PKCS12
```

Listing 2.1: Implementierung von HTTPS und SSL Konfiguration

Durch die Implementierung von HTTPS mit SSL/TLS hält sich ShopNook an die Industriestandards für Datensicherheit, um seine Nutzer zu schützen und das Vertrauen zu erhalten. Dies ist ein entscheidender Aspekt des Backends der Anwendung, insbesondere für die Wahrung der Vertraulichkeit und Integrität sensibler Daten bei Transaktionen.

2.3.4 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 und Microservices [Sprd].

2.3.4.1 Warum wurde Spring Boot ausgewählt ?

Spring Boot wurde im Rahmen dieses Projekts aufgrund der breiten Unterstützung in der Entwicklergemeinschaft ausgewählt. Hier sind einige der Hauptgründe:

- **Produktivität:** Das Framework ermöglicht es auf das Schreiben von Programmlogik zu konzentrieren, anstatt sich um die Konfiguration von Technologien zu kümmern.
- **Microservices:** Es eignet sich hervorragend für die Entwicklung der Architekturen von Microservices, indem es Server wie Tomcat bietet, sodass die Anwendungen ohne externen Server laufen können.
- **Integration:** Es lässt sich nahtlos in andere Spring-Projekte und eine Vielzahl von Datenbanken integrieren.

Das Framework bietet auch viele Funktionalitäten, die es von anderen abheben. Es hat viele weitere Vorteile, die hier genannt werden:

- **Auto-Konfiguration:** Die Autokonfigurationsfunktion von Spring Boot konfiguriert Ihre Anwendung automatisch anhand der Abhängigkeiten, die Sie dem Projekt hinzugefügt haben. Dadurch wird die Notwendigkeit einer manuellen Konfiguration minimiert und die Entwicklung beschleunigt.
- **Eigenständige Anwendungen:** Spring Boot-Anwendungen können als eigenständige Java-Anwendungen verpackt werden, was die Bereitstellung einfacher und konsistenter macht.
- **Produktionstaugliche Funktionen:** Das Framework umfasst zahlreiche produktionsreife Funktionen wie Zustandsprüfungen, Metriken und externalisierte Konfiguration.

2.3.4.2 Wie funktioniert Spring Boot ?

1. **Initial Setup:** Mit Spring Initializr kann man schnell ein neues Spring Boot-Projekt mit allen erforderlichen Abhängigkeiten und Konfigurationen initialisieren.
2. **Main Application Class:** Jede Spring Boot-Anwendung hat eine Hauptklasse, die mit `@SpringBootApplication` annotiert ist. Diese Annotation ist eine Kombination aus `@Configuration`, `@EnableAutoConfiguration` und `@ComponentScan`.

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

Listing 2.2: Main-Application-Class-Implementierung in Java

3. **Application Properties:** Spring Boot ermöglicht eine einfache Konfiguration durch `application.properties`-Datei. Dadurch wird die Konfiguration externalisiert, was die Verwaltung verschiedener Umgebungen erleichtert.

```
1 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
2 spring.datasource.url=jdbc:mysql://localhost:3306/full-stack-ecommerce?
3 useSSL=false&useUnicode=yes&characterEncoding=UTF-8&allowPublicKeyRetrieval=
4 true&serverTimezone=UTC
5 spring.datasource.username=username
6 spring.datasource.password=password
7 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
8 spring.data.rest.base-path=/api
```

Listing 2.3: Implementierung von Application.properties Datei

2.3.4.3 Spring Security

Spring Security ist ein robustes und anpassungsfähiges Framework, das für die Authentifizierung und Zugriffskontrolle entwickelt wurde und sich zum Standard für die Absicherung von mit Spring erstellten Anwendungen entwickelt hat.

Spring Security ist ein Framework, das sowohl die Authentifizierung als auch die Autorisierung von Java-Anwendungen übernimmt. Seine Stärke liegt in der Flexibilität, die es erlaubt, es leicht zu erweitern, um spezifische Anforderungen zu erfüllen.

Das Framework bietet eine Reihe von Funktionen, darunter umfassende Unterstützung für Authentifizierung und Autorisierung, Schutz vor verschiedenen Angriffen wie Session Fixation, Clickjacking und Cross-Site Request Forgery, Integration mit der Servlet API und optionale Kompatibilität mit Spring Web MVC [Spre].

Zum Schutz der Endpunkte für Kundenbestellungen wurde Spring Security mit OAuth2-Authentifizierung implementiert, wobei JWT zur Gewährleistung eines sicheren Zugriffs verwendet wird. Die Einrichtung umfasste die Konfiguration von CORS zur Verwaltung von Herkunftsübergreifenden Anfragen und die Anpassung von Sicherheitseinstellungen wie die Deaktivierung des CSRF-Schutzes und die Handhabung der Inhaltsaushandlung. Dieser Ansatz gewährleistet robuste Sicherheit, während er verschiedene Inhaltstypen unterstützt und die gemeinsame Nutzung von Ressourcen über verschiedene Ursprünge hinweg effektiv handhabt.

2.3.5 Stripe API

In diesem Teil wird die Stripe-API näher beleuchtet, beginnend mit einem Überblick über ihre grundlegenden Funktionen und Hauptmerkmale. Anschließend wird die Integration der API in unsere Anwendung beschrieben, gefolgt von spezifischen Anwendungsfällen, die die Vielseitigkeit von Stripe in unserem Projekt veranschaulichen. Diese Analyse zeigt, wie Stripe eine nahtlose und sichere Zahlungsabwicklung ermöglicht.

2.3.5.1 Überblick

Die Stripe-API bietet ein umfassendes Paket von Tools für Entwickler zur Integration und Interaktion mit den Diensten von Stripe. Sie wurde nach REST-Prinzipien mit vorhersehbaren, ressourcenorientierten URLs entwickelt. Anfragen werden mit verschlüsseltem Form Körpern gesendet, und Antworten werden im JSON-Format zurückgegeben. Je nach verwendetem API-Schlüssel kann die Stripe-API entweder im Testmodus oder im Live-Modus betrieben werden. Verschiedene Stripe-Konten können je nach Version und benutzerdefinierten Funktionen unterschiedliche API-Funktionen aufweisen. Die API-Referenz enthält Codebeispiele für die Integration in verschiedenen Programmiersprachen[ST Ja].

2.3.5.2 Hauptmerkmale der Stripe API

Zu den wichtigsten Funktionen der Stripe-API gehören:

- Die API bearbeitet Anfragen einzeln, ohne Unterstützung für Massenaktualisierungen, pro Anfrage kann nur ein Objekt aktualisiert werden.
- Die Funktionalität kann sich mit jeder neuen Version der API ändern, wenn Updates veröffentlicht werden.
- Die Stripe-API unterstützt eine breite Palette von Zahlungsmethoden, darunter Karten, digitale Geldbörsen, Bankabbuchungen und -überweisungen sowie Bankumleitungen, zusammen mit Optionen wie „Sofort kaufen, später bezahlen“ und bargeldbasierten Gutscheinen.
- Sie bietet Funktionen zur Umsatzoptimierung, einschließlich Tools für Authentifizierung, Autorisierung, Betrugsprävention, Streitfallmanagement und Abgleich.
- Stripe bietet auch Berichts- und Analysetools wie Stripe Sigma für detaillierte Einblicke und das Stripe Dashboard für das Zahlungsmanagement[ST Ja].

2.3.5.3 Stripe API Integration für nahtlose Zahlungsabwicklung

Integration:

In der Anwendung wurde die Stripe-API sowohl in das Frontend als auch in das Backend integriert, um eine sichere und effiziente Zahlungsabwicklung zu ermöglichen. Auf dem Frontend wurde die Stripe API Version „8.179“ in der Visual Studio Code Umgebung installiert. Der `stripePublishableKey` wurde eingebunden, um die Funktionalität von Stripe im Frontend zu aktivieren. Außerdem wurde die Stripe.js-Bibliothek durch Hinzufügen des folgenden Skripts eingebunden:

```
1 <script src="https://js.stripe.com/v3" async></script>
```

sodass die Anwendung direkt über den Browser mit der API von Stripe interagieren kann.

Am Backend wurde der geheime Schlüssel von Stripe(`stripe.key.secret`) sicher in der Datei `application.properties` der Spring Boot-Anwendung gespeichert. Dieser Schlüssel ist für die sichere Handhabung sensibler Vorgänge wie die Erstellung von Zahlungsintentionen und die Verwaltung von Transaktionen unerlässlich. Die Payment Intent API wurde im Backend zur Abwicklung des Zahlungsprozesses verwendet, um sicherzustellen, dass die Transaktionen authentifiziert, autorisiert und effizient verarbeitet werden[ST Jb].

Anwendungsfälle:

Die Stripe API spielt eine entscheidende Rolle in verschiedenen zahlungsbezogenen Szenarien innerhalb der Anwendung. Sie wird hauptsächlich verwendet für:

- **Zahlungsabwicklung:** Die Stripe-API wird verwendet, um Zahlungen sicher zu verarbeiten. Dazu gehört das Erstellen von Zahlungsabsichten im Backend mit Java in Spring Boot, das den gesamten Zahlungslebenszyklus verwaltet, von der Zahlungsauslösung bis zur Bestätigung.
- **E-Mail-Bestätigung:** Nach einem erfolgreichen Kauf wird die Stripe-API so konfiguriert, dass sie Bestätigungs-E-Mails an die Benutzer sendet, um ihnen eine Aufzeichnung ihrer Transaktion zu liefern.
- **Frontend-Zahlungsabwicklung:** Das Frontend interagiert für die Zahlungsabwicklung mit Stripe und nutzt die Stripe.js-Bibliothek, um die Zahlungsinformationen sicher zu sammeln und mit Token zu versehen, bevor sie zur weiteren Verarbeitung an das Backend gesendet werden.

Diese umfassende Integration der Stripe-API sowohl in das Frontend als auch in das Backend stellt sicher, dass alle zahlungsrelevanten Vorgänge sicher, effizient und mit einer nahtlosen Benutzererfahrung abgewickelt werden.

2.4 Datenbankstruktur

In diesem Abschnitt wird die Datenbankstruktur der Webanwendung untersucht, um ein umfassendes Verständnis für die Organisation, den Zugriff und die Verwaltung der Daten zu vermitteln. Der Schwerpunkt liegt auf dem relationalen Datenbankmodell und den spezifischen Technologien und Frameworks, die zur Handhabung der Datenpersistenz und der Transaktionen verwendet werden. Dabei wird MySQL als das gewählte Datenbanksystem zusammen mit Spring Data JPA für eine nahtlose Dateninteraktion untersucht. In den folgenden Unterabschnitten werden diese Aspekte im Detail behandelt, wobei relationale Datenbanken, die Auswahl von MySQL, Datenpersistenz und Transaktionsmanagement sowie die Integration von MySQL mit Spring Boot behandelt werden.

2.4.1 Übersicht über relationale Datenbanken und Auswahl von MySQL

Die Daten in einer relationalen Datenbank sind in Tabellen mit Spalten und Zeilen organisiert, und auch die Primär- und Fremdschlüssel werden verwendet, um die Beziehungen zwischen den Tabellen herzustellen. Dieser Ansatz ist ideal für E-Commerce-Anwendungen, da er komplexe Abfragen und aussagekräftige Daten verarbeiten kann. Produktlieferungen, Auftragstransaktionen und Kundendaten werden alle ordnungsgemäß verwaltet. Die Möglichkeit, Tabellen zu verknüpfen, erleichtert umfassende Analysen, einschließlich der Nachverfolgung von Kaufdatensätzen und der Bestandskontrolle, während die Einhaltung von ACID-Eigenschaften eine zuverlässige Transaktionsverarbeitung gewährleistet, die für die Auftragsabwicklung und das Zahlungsmanagement entscheidend ist. Relationale Datenbanken wie MySQL bieten die Robustheit und Konsistenz, die für einen sicheren und produktiven Online-Shopping-Betrieb erforderlich sind [IBM].

2.4.2 Datenbankschemaentwurf mit Spring Data JPA

Ziel dieses Abschnitts ist es, zu erörtern, wie Spring Data JPA den Entwurf und die Verwaltung von Datenbankschemata vereinfacht. Es wird gezeigt, wie Spring Data JPA es Entwicklern ermöglicht, Entitäten und Beziehungen innerhalb eines Datenbankschemas abzubilden und zu verwalten, und wie es die Komplexität des Datenzugriffs durch seine Repository-Schicht abstrahiert.

Spring Data JPA vereinfacht den Entwurf von Datenbankschemata, indem es Entwicklern ermöglicht, Java-Klassen mithilfe von Annotationen wie `@Entity`, `@Id` und `@OneToMany` auf Datenbanktabellen abzubilden. Diese Annotationen übernehmen automatisch die Schemaerstellung und -verwaltung und stellen sicher, dass die Datenbankstruktur mit dem Domänenmodell der Anwendung übereinstimmt [Sprc, Bae].

Die Repository-Schicht in Spring Data JPA abstrahiert die Komplexität des Datenzugriffs und bietet integrierte CRUD-Operationen über Schnittstellen wie `JpaRepository`. Dies macht manuelle SQL-Abfragen überflüssig und rationalisiert die Datenbankinteraktionen, so dass sich die Entwickler auf die Geschäftslogik konzentrieren können [Sprb, Spra].

2.4.3 Datenpersistenz und Transaktionen

In diesem Abschnitt wird erläutert, wie Spring Boot und Hibernate die Datenpersistenz handhaben. Der Schwerpunkt liegt dabei auf CRUD-Operationen, der Rolle des `EntityManager` und der Interaktion des Hibernate ORM mit MySQL.

CRUD-Operationen werden mit Spring Boot und Hibernate verwaltet. Der `EntityManager` ist in diesem Prozess von zentraler Bedeutung und ermöglicht das Erstellen, Abrufen, Aktualisieren und Löschen von Entitäten. Er verwaltet auch den Lebenszyklus von Entitäten innerhalb von Transaktionen und stellt sicher, dass Änderungen bei Bedarf ordnungsgemäß übertragen oder zurückgenommen werden [Bae24].

Hibernate ORM wird verwendet, um Java-Objekte auf Datenbanktabellen abzubilden, was die Interaktion mit der MySQL-Datenbank vereinfacht. Dieses ORM-Framework macht manuelles SQL überflüssig und bietet einen objektorientierten Ansatz für Datenbankoperationen [Hib].

2.4.4 MySQL-Workbench und SQL

In diesem Abschnitt wird die MySQL Workbench als Werkzeug für den Entwurf und die Verwaltung von Datenbanken und SQL als Sprache für die Erstellung von Abfragen und die Verwaltung der Datenbank vorgestellt.

MySQL Workbench wurde zur Visualisierung des Datenbankschemas eingesetzt und ermöglichte eine intuitive Gestaltung und Verwaltung der Datenbankstrukturen. Sie erleichterte die Erstellung und Änderung von Tabellen, Beziehungen und Indizes. Darüber hinaus wurde SQL verwendet, um Abfragen für Tests und die Datenbankverwaltung auszuführen, einschließlich Aufgaben wie Datenmanipulation und Schemaaktualisierungen [myS].

2.4.5 Datenbankverbindung und -konfiguration in Spring Boot

In diesem Abschnitt wird erläutert, wie die Datenbank innerhalb der Spring Boot-Anwendung verbunden und konfiguriert wird.

Die Datenbankverbindung wird in der Datei `application.properties` konfiguriert. Zu den wichtigsten Eigenschaften gehören die JDBC-URL, der Name der Treiberklasse und die Anmeldeinformationen, wie in Listing 2.2 dargestellt.

2.5 Authentifizierungs- und Autorisierungsprotokolle

In der modernen digitalen Landschaft ist die Gewährleistung eines sicheren Zugriffs auf Ressourcen von entscheidender Bedeutung. In diesem Abschnitt werden die wichtigsten Protokolle und Standards zur Verwaltung der Benutzerauthentifizierung und -autorisierung untersucht, darunter JWT, OAuth2 und OpenID Connect.

2.5.1 Okta

Okta verbindet Benutzer mit jeder Anwendung über alle Geräte hinweg. Es ist ein Cloud-basierter Identitätsmanagement-Service, der mit verschiedenen lokalen Systemen kompatibel ist. Okta bietet der IT-Abteilung die Möglichkeit, den Zugriff auf Anwendungen und Geräte über eine sichere, zuverlässige und gut geprüfte Cloud-Plattform zu verwalten, mit einer tiefen Integration in lokale Systeme und Verzeichnisse.

Okta bietet Funktionen wie Provisioning, Single Sign-On (SSO), Active Directory (AD) und LDAP-Integration, zentrales Nutzer-Deprovisioning, Multi-Faktor-Authentifizierung (MFA), mobiles Identitätsmanagement und anpassbare Sicherheitsrichtlinien. Diese Funktionen werden durch das Okta Integration Network (OIN) vereinheitlicht, das umfangreiche Integrationsoptionen bietet und SSO für alle Anwendungen ermöglicht, die Ihre Benutzer benötigen[OTb].

2.5.2 JWT

JWT ist ein offener Standard (RFC 7519) für die sichere Übertragung von Informationen in Form eines JSON-Objekts. Diese Informationen sind aufgrund der digitalen Unterzeichnung überprüfbar und vertrauenswürdig. JWTs können mit einem geheimen Schlüssel mit HMAC oder mit einem öffentlichen/privaten Schlüsselpaar mit RSA oder ECDSA signiert werden. Hier sind einige Situationen, in denen JSON-Web-Token von Vorteil sind:

- **Autorisierung:** JWTs werden in der Regel zur Authentifizierung verwendet. Nach der Anmeldung enthält jede Anfrage das JWT, das den Zugriff auf autorisierte Routen, Dienste und Ressourcen ermöglicht.
- **Informationsaustausch:** Mit JWTs werden Informationen sicher zwischen Parteien übertragen. Signierte JWTs bestätigen die Identität des Absenders und stellen sicher, dass der Inhalt unverändert bleibt, da die Signatur sowohl aus der Kopfzeile als auch aus der Nutzlast abgeleitet wird[JT].

2.5.3 OAuth2

OAuth 2.0, kurz für „Open Authorization“, ist ein Standard, der es einer Website oder Anwendung ermöglicht, im Namen eines Benutzers auf Ressourcen anderer Webanwendungen zuzugreifen. Er löste 2012 OAuth 1.0 ab und hat sich seitdem zum Industriestandard für die Online-Autorisierung entwickelt. OAuth 2.0 erleichtert den autorisierten Zugriff und schränkt gleichzeitig die Aktionen ein, die eine Client-Anwendung für den Benutzer auf Ressourcen durchführen kann, ohne die Anmeldedaten des Benutzers preiszugeben[OTa].

2.5.4 OpenID Connect

OpenID Connect ist ein Authentifizierungsprotokoll, das auf dem OAuth 2.0 Framework (IETF RFC 6749 und 6750) aufbaut. Es rationalisiert die Überprüfung der Benutzeridentität, indem es die von einem Autorisierungsserver durchgeführte Authentifizierung nutzt und Benutzerprofilinformationen in einer RESTful und interoperablen Weise bereitstellt.

OpenID Connect fördert ein Internet-Identitäts-Ökosystem durch einfache Integration, Sicherheits- und Datenschutzfunktionen, breite Client- und Gerätehilfe und die Möglichkeit für jede Einrichtung, als OpenID Provider (OP) zu agieren[OTc].

2.6 Entwicklungsumgebung und Versionskontrolle

In diesem Abschnitt werden die integrierten Entwicklungsumgebungen (IDEs) und Versionskontrollsysteme beschrieben, die im Rahmen des Projekts eingesetzt wurden. Er bietet einen Überblick über die für die Entwicklung ausgewählten IDEs und die Versionskontrollwerkzeuge, die zur effektiven Verwaltung von Codeänderungen eingesetzt wurden.

2.6.1 Entwicklungsumgebung

Dieser Unterabschnitt beschreibt die im Projekt verwendeten Entwicklungsumgebungen, wobei der Schwerpunkt auf IntelliJ IDEA und Visual Studio Code (VS Code) liegt. Er hebt die Funktionen und Vorteile der einzelnen IDEs bei der Unterstützung des Entwicklungsprozesses und der Steigerung der Produktivität hervor.

2.6.1.1 IntelliJ IDEA

IntelliJ IDEA ist bekannt für seinen leistungsstarken Code-Editor, der sich durch eine umfassende Anfangsindizierung beim Verstehen und Verwalten von Code auszeichnet. Diese Funktion ermöglicht es ihm, Fehler in Echtzeit zu erkennen, kontextbezogene Vorschläge zur Code-Vervollständigung anzubieten und sicheres Code-Refactoring durchzuführen, neben anderen Funktionen. Es ist mit einer umfangreichen Suite integrierter Entwicklerwerkzeuge ausgestattet und bietet robuste Unterstützung für verschiedene Spring-Frameworks sowohl in Java als auch in

Kotlin. Dazu gehören Spring MVC, Spring Boot, Spring Integration, Spring Security und Spring Cloud. Zu den bemerkenswerten Vorteilen gehören intelligente Codierungshilfe, sofortige Navigation innerhalb des Spring-Codes, integrierte Entwicklungswerkzeuge und erweiterte Visualisierungsfunktionen [Jet].

2.6.1.2 Visual Studio Code

Visual Studio Code ist ein vielseitiger und effizienter Quellcode-Editor, der mit Windows, macOS und Linux kompatibel ist. Er bietet native Unterstützung für JavaScript, TypeScript und Node.js und verfügt über eine breite Palette von Erweiterungen für weitere Sprachen und Plattformen [Tead].

Da die Front-End-Entwicklung Angular, ein auf TypeScript basierendes Framework, nutzt, wurde Visual Studio Code als Entwicklungsumgebung gewählt. Die starke native Unterstützung für TypeScript erleichtert die effiziente und optimierte Projektentwicklung und macht zusätzliche Erweiterungen überflüssig. [Teac].

2.6.2 Versionskontrolle

In diesem Unterabschnitt werden die im Projekt eingesetzten Versionskontrollsysteme beschrieben, wobei der Schwerpunkt auf Git und GitHub liegt. Es wird erläutert, wie diese Tools die Codeverwaltung, die Zusammenarbeit und die Verfolgung von Änderungen während des gesamten Entwicklungsprozesses erleichtert haben.

2.6.2.1 Git

Git ist ein kostenloses und quelloffenes, verteiltes Versionskontrollsystem, das entwickelt wurde, um Projekte jeder Größe schnell und effizient zu verwalten. Es hat einen minimalen Platzbedarf und liefert eine außergewöhnlich schnelle Leistung. Git übertrifft herkömmliche SCM-Tools wie Subversion, CVS, Perforce und ClearCase mit Funktionen wie kostengünstiger lokaler Verzweigung, benutzerfreundlichen Staging-Bereichen und flexiblen Arbeitsabläufen.

Eines der herausragenden Merkmale von Git ist sein Verzweigungsmodell, das es von fast allen anderen SCM-Tools unterscheidet. Git ermöglicht und fördert die Erstellung mehrerer lokaler Zweige, von denen jeder unabhängig funktionieren kann. Das Erstellen, Zusammenführen und Löschen von Zweigen geht schnell, nahtlos vonstatten und dauert oft nur Sekunden [Teaa].

2.6.2.2 GitHub

GitHub ist eine cloudbasierte Plattform zum Speichern, Freigeben und gemeinsamen Bearbeiten von Code. Die von Git unterstützten Kollaborationsfunktionen von GitHub ermöglichen es uns:

- Projekte zu präsentieren oder zu verbreiten.
- Verfolgen und Verwalten von Codeänderungen im Laufe der Zeit.

-
- Andere einladen, den Code zu überprüfen und Verbesserungen vorzuschlagen.
 - Zusammenarbeit an Projekten ohne das Risiko, dass unbeabsichtigte Änderungen die Arbeit der anderen beeinträchtigen, bis die Änderungen integriert werden können. [Teab].

Implementierung

In diesem Kapitel wird die Implementierung der prinzipiellen Funktionsweise der Anwendung beschrieben. Als Grundlage für alle Seiten in der Anwendung wird ein globales Layout `app.component.html` angelegt, welches für alle übersetzten Seiten verwendet wird. Darin wird definiert, welche Komponenten auf jeder Seite angezeigt werden sollen.

3.1 Startseite

Für dieses Projekt sind standardmäßig für alle Seiten die Komponenten Header, Menu, Footer und Hauptinhalt definiert. Im Header ist eine Navigation zwischen verschiedenen Seiten, das Einloggen mit Benutzerkonto und das Suchen nach bestimmten Produkten. Der Footer zeigt weiterführende Links mit Information zum About-us, Contact-us und Help. Dazu werden als Komponenten im Ordner `/components` Komponenten wie z.B `/about-us`, `/search` oder `/product-category-menu` angelegt, in dem diese Funktionalitäten eingebaut werden.

3.1.1 header

In der Datei `/app.component.html` werden drei Komponenten im Header, wie man in 3.1 sehen kann, aufgerufen.

```
1 <app-search></app-search>
2 <app-login-state></app-login-state>
3 <app-cart-state></app-cart-state>
```

Listing 3.1: Aufruf von Header-Komponenten

Das Komponent `/app-search` ist dafür zuständig, den Benutzern eine Suchfunktion bereitzustellen, mit der sie schnell und einfach nach Produkten suchen können. In 3.2 kann man eine HTML-Struktur der Komponente `/app-search` sehen:

```
1 <div class="search-container">
2 <input #searchInput class="search-input"
3 type="text" placeholder="Search for products..."
4 (keyup.enter)="searchProducts(searchInput.value)"/>
5 <button (click)="searchProducts(searchInput.value)" class="au-btn-submit">
6 <a href="#" class="search-btn">
7 <i class="fas fa-search"></i>
8 </a>
9 </button>
10 </div>
```

Listing 3.2: HTML-Struktur der app-search-Komponente

Das Eingabefeld ist mit der Angular-Komponente über die Template Referenzvariable `#searchInput` verbunden. Benutzer können hier ihre Suchanfragen eingeben. Die Suche kann auf zwei Arten ausgelöst werden:

1. Durch Drücken der Enter-Taste: Das Ereignis `keyup.enter` ist mit der Methode `searchProducts()` verbunden, sodass die Suche gestartet wird, wenn der Benutzer die Enter-Taste drückt.
2. Durch Klicken auf die Suchschaltfläche: Die Suche kann auch durch Klicken auf die Suchschaltfläche gestartet werden, die ebenfalls an die Methode `searchProducts()` gebunden ist.

Die `/cart-state` bietet den Benutzern Echtzeit-Updates über den Zustand ihres Einkaufswagens, einschließlich der Gesamtanzahl der Artikel und des Gesamtpreises. Diese Komponente ermöglicht es den Benutzern, problemlos auf die Warenkorbdetails zuzugreifen und zur Kasse zu gehen oder den Inhalt ihres Warenkorbs zu ändern. Die Html-Struktur dieser Komponente wird in 3.3 hervorgehoben:

```
1 <div class="cart-area">
2 <a routerLink = "/cart-details" class="cart-link">
3 <div class="cart-content">
4 <div class="cart-icon">
5 <i class="fas fa-shopping-cart"></i>
6 <span class="cart-quantity">{{ totalQuantity }}</span>
7 </div>
8 <div class="total-price">{{ totalPrice | currency: 'EUR' }}</div>
9 </div>
10 </a>
11 </div>
```

Listing 3.3: HTML-Struktur der cart-state-Komponente

Die Komponente zeigt die Gesamtanzahl der Artikel im Warenkorb (`totalQuantity`) und den Gesamtpreis (`totalPrice`) an, die dynamisch aktualisiert werden, wenn der Benutzer mit dem Warenkorb interagiert. Sie dient auch als Link zur Warenkorbdetailseite `/cart-details`, sodass Benutzer den Inhalt ihres Warenkorbs anzeigen und ändern können.

Zusätzlich zu den Komponenten `/app-search` und `/app-cart-state` ist die Komponente `/app-login-state`, wie in 3.1 dargestellt, für die Verwaltung des Benutzerauthentifizierungsstatus zuständig. Diese Komponente ändert ihre Anzeige dynamisch, je nachdem, ob ein Benutzer angemeldet ist oder nicht. Die Komponente `/login-state` wurde entwickelt, um eine optimierte Benutzererfahrung

zu bieten, indem sie je nach Authentifizierungsstatus des Benutzers verschiedene Optionen anzeigt. Wenn ein Benutzer nicht angemeldet ist, zeigt die Komponente eine einfache Schaltfläche „Login“ an. Wenn der Benutzer auf diese Schaltfläche klickt, wird er auf die Anmeldeseite weitergeleitet.

Sobald sich der Benutzer jedoch erfolgreich anmeldet, verhält sich die Komponente dynamisch, um die Benutzerinteraktion zu verbessern. Die folgenden Elemente werden angezeigt:

- **Willkommensnachricht** : Eine personalisierte Nachricht heißt den Benutzer willkommen und zeigt seinen vollständigen Namen an, der vom Okta-Authentifizierungsdienst abgerufen wird.
- **Schaltfläche „Logout“** : Es wird eine Schaltfläche „Logout“ angezeigt, mit der Benutzer ihre Sitzung beenden und sich sicher abmelden können. Diese Schaltfläche ruft die in der Komponente definierte Funktion `logout()` in `login-state.component.ts` auf, die den OktaAuth-Dienst verwendet, um den Benutzer abzumelden und die vorhandenen Token zu löschen.
- **Navigations-Schaltflächen** : Zwei zusätzliche Schaltflächen, „Member“ und „Orders“, werden angezeigt, sobald der Benutzer authentifiziert ist. „Member“ leitet den Benutzer zur `only-members-page.component.html`, wo er auf mitgliedsspezifische Funktionen und Informationen zugreifen kann. „Orders“ führt den Nutzer zur Seite `order-log.component.html`, wo er einen Überblick über seine bisherigen Bestellungen erhält.

3.1.2 Menu

`app-product-category-menu` implementiert die Seitenleiste der Anwendung, die als dynamisches Navigationsmenü fungiert. Sie ermöglicht es den Benutzern, schnell auf verschiedene Produktkategorien zuzugreifen. Die Komponente ist auf der linken Seite platziert und bleibt in der Desktop-Ansicht immer sichtbar, um eine einfache Navigation zu gewährleisten. In der 3.4 wurde die HTML-Struktur der Komponente `app-product-category-menu.html` dargestellt:

```
1 <section class="app">
2 <aside class="sidebar">
3 <header>
4   Menu
5 </header>
6 <nav class="sidebar-nav">
7 <ul class="list-unstyled navbar-list">
8 <li *ngFor="let tempProductCategory of productCategories">
9   <a routerLink="/category/{{tempProductCategory.id}}"
10  routerLinkActive="active-link"> {{tempProductCategory.categoryName}}
11 </a>
12 </li>
13 </ul>
14 </nav>
15 </aside>
16 </section>
```

Listing 3.4: HTML-Struktur der `app-product-category-menu`-Komponente

Die Seitenleiste `sidebar` zeigt eine Liste von Produktkategorien an, die dynamisch aus dem Backend abgerufen werden. Dies wird durch die Verwendung der Angular-Direktive `*ngFor` erreicht, die über die `productCategories`-Liste iteriert und für jede Kategorie ein Navigationslink erstellt.

Jeder Link im Navigationsmenü verwendet also die Angular-Direktive `routerLink`, um die Benutzer zu einer Seite weiterzuleiten, die Produkte in der ausgewählten Kategorie anzeigt. Der `routerLinkActive`-Attributwert „active-link“ wird verwendet, um den aktiven Status des Links hervorzuheben, wenn die Kategorie aktiv ist.

3.2 Online-Shopping

Der Online-Shopping-Prozess in der Anwendung besteht aus mehreren Hauptfunktionen, die alle darauf abzielen, den Benutzern ein nahtloses Einkaufserlebnis zu bieten. Dieser Prozess umfasst das Durchsuchen von Produkten, das Auswählen von Produkten, das Hinzufügen von Produkten zum Warenkorb und das Verwalten des Warenkorbs.

3.2.1 Darstellung von Produkten auf der Produktseite

Durch das Aufrufen von der Datei `product-list-grid.component.html`, wird eine Liste der verfügbaren Produkte in einer grid-basierten oder tabellenbasierten Ansicht gezeigt. Dies ermöglicht es den Benutzern, verschiedene Produkte schnell zu durchsuchen und auszuwählen.

Die Produktliste bietet eine visuelle Darstellung der Produkte mit einem Bild, dem Produktnamen, dem Preis und einer Schaltfläche zum Hinzufügen zum Warenkorb.

3.2.2 Auswahl und Hinzufügen von Produkten zum Warenkorb

Benutzer können Produkte auswählen, indem sie auf das Produktbild oder den Namen klicken, um zur Produktdetailseite zu navigieren. Alternativ können sie direkt das Produkt durch einen Klick auf „Add to cart“ zur Einkaufsliste hinzufügen. `addItemToCart()` (siehe 3.5) in der Komponente `product-list.component.ts` wird verwendet, um ein Produkt zum Warenkorb hinzuzufügen.

```
1 addItemToCart(myProduct: Product) {  
2     console.log('Adding to the shopping cart: ${myProduct.name},  
3     ${myProduct.itemPrice}');  
4     const cartItem = new CartBasket(myProduct);  
5     this.cartService.addToCart(cartItem);  
6 }
```

Listing 3.5: Funktion zum Hinzufügen von Produkten zum Warenkorb

Die Funktion nimmt ein `Product`-Objekt als Parameter, das das ausgewählte Produkt repräsentiert und erstellt ein neues `CartBasket`-Objekt, das die Produktdetails enthält, und ruft `addToCart()` aus der Datei `cart.service.ts` auf, um das Produkt zum Warenkorb hinzuzufügen.

3.2.3 Warenkorbverwaltung

Nach dem Hinzufügen von Produkten zum Warenkorb können Benutzer ihren Warenkorb verwalten, indem sie Produkte entfernen, die Menge ändern oder zur Kasse gehen.

- **Produkt entfernen:** Die Funktion `removeItemFromCart`, die in 3.6 dargestellt wurde, entfernt ein bestimmtes Produkt aus dem Warenkorb.
- **Menge ändern:** Die Benutzeroberfläche kann auch Steuerelemente bereitstellen, um die Menge eines bestimmten Produkts zu ändern. Diese Änderungen werden ebenfalls über die Funktion `decreaseQuantity` in `cart-service.ts` verarbeitet (siehe 3.6).

```
1  remove(cartItem: CartBasket) {
2      const index = this.cartItems.findIndex(cartItem =>
3          cartItem.id === cartItem.id);

5      if(index > -1)
6          this.cartItems.splice(index, 1)

8          this.calculateCartTotals();
9  }

11 decreaseQuantity(cartItem: CartBasket) {
12     cartItem.quantity--;

14     if(cartItem.quantity == 0)
15     {
16         this.remove(cartItem);
17     }
18     else
19     {
20         this.calculateCartTotals();
21     }
22 }
```

Listing 3.6: Warenkorb Service Funktionen

3.2.4 Paginierung und Suchfunktionalität

Die Produktliste unterstützt die Paginierung und eine Suchfunktionalität, um eine große Anzahl von Produkten effizient zu handhaben.

- **Paginierung:** Die Paginierung ermöglicht es den Benutzern, die Produktliste seitenweise zu durchsuchen. Die Paginierungssteuerung `ngb-pagination` aktualisiert die Anzeige basierend auf der ausgewählten Seite und der Anzahl der Elemente pro Seite.
- **Suchfunktionalität:** wie es in 3.1 erwähnt wurde, können die Benutzer die Produktsuche verwenden, um spezifische Artikel zu finden. Die Anwendung verarbeitet die Suchanfragen, ruft die entsprechenden Produkte ab und zeigt sie auf der Seite an.

In den Folgenden wird die implementierung der Suchfunktionalität hervorgehoben:

```
1  manageSearchProducts() {
2      const myKeyword: string = this.route.snapshot.paramMap.get('keyword')!;

4      if (this.formerKeyword !== myKeyword) {
5          this.pageNumber = 1;
6      }

8      this.formerKeyword = myKeyword;

10     console.log('keyword=${myKeyword}, pageNumber=${this.pageNumber}');

12     this.productService.searchProductsPaginated(this.pageNumber - 1,
13     this.pageSize, myKeyword)
14     .subscribe(this.processProductData());
15 }

18 manageProductList() {
19     const hasCategoryId: boolean = this.route.snapshot.paramMap.has('id');

21     if (hasCategoryId) {
22         this.currentCategoryId = +this.route.snapshot.paramMap.get('id')!;
23     } else {
24         this.currentCategoryId = null;
25     }

27     if (this.previousCategoryId !== this.currentCategoryId) {
28         this.pageNumber = 1;
29     }

31     this.previousCategoryId = this.currentCategoryId;

33     console.log('currentCategoryId=${this.currentCategoryId},
34     pageNumber=${this.pageNumber}');

36     if (this.currentCategoryId !== null) {
37         this.productService.getProductListPaginated(this.pageNumber - 1,
38         this.pageSize, this.currentCategoryId)
39         .subscribe(this.processProductData());
40     } else {
41         this.productService.getAllProductsPaginated(this.pageNumber - 1,
42         this.pageSize).subscribe(this.processProductData());
43     }
44 }
```

Listing 3.7: Verwaltung der Produktliste und Suche

Das Online-Shopping-Modul der Anwendung ist so konzipiert, dass es eine benutzerfreundliche Oberfläche zum Durchsuchen, Auswählen und Verwalten von Produkten bietet.

Durch die Verwendung von Komponenten wie `product-list-grid.component.html` und `product-list-table.component.html` wird eine dynamische und interaktive Benutzeroberfläche geschaffen, die die Benutzererfahrung verbessert und das Einkaufen erleichtert.

3.3 Bezahlprozess

Der Checkout-Prozess ist ein zentraler Bestandteil des Online-Shopping-Erlebnisses in der ShopNook-Anwendung. In diesem Prozess werden die Benutzer durch die Eingabe ihrer Lieferinformationen und Zahlungsdetails geführt, um eine Bestellung abzuschließen und zu bezahlen.

Der Checkout-Prozess besteht aus mehreren Schritten, die durch das Frontend (Angular) und das Backend (Spring Boot) unterstützt werden. Im Folgenden wird der Checkout-Prozess detailliert beschrieben.

3.3.1 Frontend

Der Checkout-Prozess beginnt auf der Frontend-Seite, die in Angular mit der Komponente `checkout.component.ts` in `/checkout` implementiert ist. Diese Komponente bietet eine benutzerfreundliche Oberfläche für die Eingabe der notwendigen Informationen zur Bestellung und zur Zahlung.

Die wichtigsten Abschnitte und Funktionen der Frontend-Seite sind wie folgt definiert:

- **Formulare für Benutzerinformationen:** `checkout.component.html` nutzt Angular FormBuilder, um ein Formular `formGroupCheckout` zu erstellen, das verschiedene Abschnitte für Kundeninformationen, Lieferadresse, Zahlungsadresse und Kreditkartendetails enthält.
- **Integration mit Stripe für Zahlungsabwicklung:**
 - Die Stripe-Bibliothek wird initialisiert, um eine sichere Zahlungsabwicklung zu gewährleisten. Stripe-Elemente (z.B. für die Eingabe von Kreditkartendetails) werden in die Checkout-Seite eingebettet.
 - Die Methode `initializeStripePaymentForm()`, wie sie in 3.8 dargestellt ist, konfiguriert die Stripe-Zahlungskomponente und integriert Validierungslogik für die Kreditkarteneingabe.
 - Die Methode `submitPurchase()` wird aufgerufen, wenn der Benutzer seine Bestellung abschickt. Diese Methode überprüft die Formularvalidierung und erstellt eine Zahlungsabsicht (Payment Intent) über den CheckoutService im Backend.


```
1  initializeStripePaymentForm() {
2      // Obtain a reference to the Stripe Elements instance
3      var elements = this.stripe.elements();

5      // Create a card input field without showing the postal code field
6      this.cardElement = elements.create('card', { hidePostalCode: true });

8      // Mount the card input component into the 'card-element' container
9      this.cardElement.mount('#card-element');

11     // Bind an event listener to handle changes in the card input field
12     this.cardElement.on('change', (event: any) => {
13         // card-errors element
14         this.displayError = document.getElementById('card-errors');

16         if (event.complete) {
17             this.displayError.textContent = "";
18         }
19         else if (event.error) {
20             // Display validation errors to the customer
21             this.displayError.textContent = event.error.message;
22         }
23     });
24 }
```

Listing 3.8: Integration von Stripe für Zahlungsabwicklung

3.3.2 Backend

Das Backend, implementiert in Spring Boot, stellt die notwendige Logik zur Abwicklung von Bestellungen und Zahlungen bereit.

Die Klasse **CheckoutServiceImplementation** implementiert das **CheckoutService**-Interface und enthält Methoden zur Bearbeitung der Bestellung und zur Erzeugung einer Zahlungsabsicht mit Stripe, die im folgenden beschrieben werden:

- **submitOrder**: angelehnt an 3.9 verarbeitet die Methode eine eingehende Bestellung und führt dabei mehrere Schritte durch. Zunächst extrahiert sie die Bestellinformationen aus dem Purchase-DTO. Anschließend wird eine eindeutige Bestellverfolgungsnummer generiert. Die Methode fügt dann die Bestellartikel zur Bestellung hinzu. Es wird überprüft, ob der Kunde bereits in der Datenbank vorhanden ist. Ist dies der Fall, wird der bestehende Kundeneintrag aktualisiert, andernfalls wird ein neuer Kundeneintrag erstellt. Schließlich speichert die Methode die Bestellung in der Datenbank und gibt eine Bestellbestätigung zurück.
- **generatePaymentIntent**: Diese Methode, wie in 3.9 zu sehen ist, erzeugt eine Zahlungsabsicht **PaymentIntent** über die Stripe-API. Sie nimmt Zahlungsinformationen entgegen und erstellt die entsprechenden Parameter für Stripe, um eine sichere Zahlungstransaktion zu initiieren.
- **generateOrderTrackingReference**: Generiert eine eindeutige Bestellverfolgungsnummer, die dem Kunden zur Nachverfolgung der Bestellung dient.

```
1  public PurchaseResult submitOrder(Purchase purchase) {
2      // get the order info from the dto
3      Order order = purchase.getOrder();

4
5      // generate tracking reference
6      String orderTrackingReference = generateOrderTrackingReference();
7      order.setOrderTrackingReference(orderTrackingReference);

8
9      // populating the order with orderItems
10     Set<OrderItem> orderItemSet = purchase.getOrderItemSet();
11     if (orderItemSet != null) {
12         orderItemSet.forEach(order::add);
13     }
14     // populating order with the Address of delivery and payment
15     order.setPaymentAddress(purchase.getPaymentAddress());
16     order.setDeliveryAddress(purchase.getDeliveryAddress());

17
18     // populating the customer with the order
19     Customer customer = purchase.getCustomer();

20
21     // check if this customer already made a purchase
22     String c_email = customer.getEmail();

23
24     Customer customerAlreadyInTheDb=customerRepository.findByEmail(c_email);

25
26     if(customerAlreadyInTheDb != null)
27     {
28         // we found our customer
29         customer = customerAlreadyInTheDb;
30     }
31     customer.add(order);

32
33     // saving into the DB
34     customerRepository.save(customer);

35
36     // return a response
37     return new PurchaseResult(orderTrackingReference);
38 }

39
40 public PaymentIntent generatePaymentIntent
41     (PaymentInformation paymentInformation) throws StripeException {
42     List<String> supportedPaymentMethods = new ArrayList<>();
43     supportedPaymentMethods.add("card");

44
45     Map<String, Object> params = new HashMap<>();
46     params.put("amount", paymentInformation.getAmount());
47     params.put("currency", paymentInformation.getCurrency());
48     params.put("payment_method_types", supportedPaymentMethods);
49     params.put("description", "Shopnook□purchase");
50     params.put("receipt_email", paymentInformation.getReceiptEmail());

51
52     return PaymentIntent.create(params);
53 }
```

Listing 3.9: Java-Code für die Übermittlung von bestellungen und die Generierung von Zahlungsanweisungen

Ergebnisse und Analyse

Dieses Kapitel bietet einen umfassenden Überblick über die Funktionalität und die Benutzerfreundlichkeit der E-Commerce-Webanwendung, die anhand einer Reihe detaillierter Screenshots und Analysen vorgestellt wird. Es werden die wichtigsten Funktionen der Anwendung untersucht, beginnend mit dem Anmeldeprozess, der einen sicheren Benutzerzugang gewährleistet, gefolgt von den Produktsuchfunktionen, die es dem Benutzer ermöglichen, Artikel nach Schlüsselwort oder Kategorie zu finden. Der Prozess des Hinzufügens von Produkten in den Einkaufswagen, das Fortschreiten zur Kasse und das Abschließen von Zahlungen wird veranschaulicht, um die nahtlose Integration dieser wesentlichen E-Commerce-Funktionen zu demonstrieren. Darüber hinaus wird in diesem Kapitel die Funktion „Bestellhistorie“ erläutert, mit der Benutzer frühere Einkäufe überprüfen können. Auch die Seiten mit eingeschränktem Zugriff, die nur angemeldeten Benutzern zur Verfügung stehen, werden besprochen, wobei die verbesserte Benutzerfreundlichkeit und die implementierten Sicherheitsmaßnahmen hervorgehoben werden. Durch diese Elemente soll das Kapitel die Effektivität und Effizienz des Designs und der Funktionalität der Anwendung aufzeigen.

4.1 Startseite

Die Homepage der E-Commerce-Anwendung zeichnet sich durch ein schlankes und benutzerfreundliches Layout aus (siehe Abbildung 4.1).

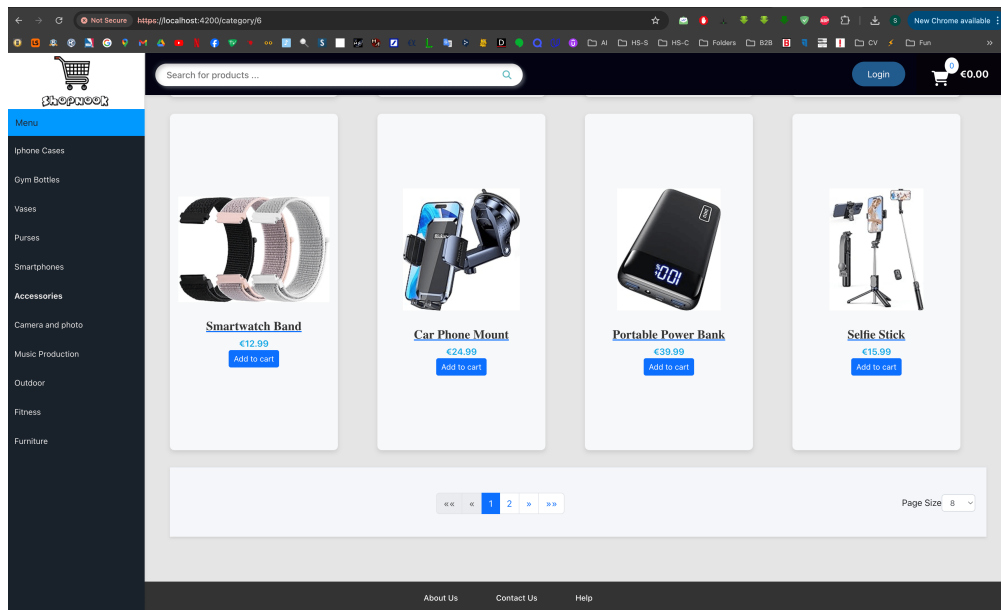


Abbildung 4.1: Startseite

4.1.1 Linke Seite

Auf der linken Seite befindet sich:

- **App-Logo:** Deutlich sichtbar auf der linken Seite, um eine gute Sichtbarkeit der Marke und eine einfache Navigation zurück zur Homepage zu gewährleisten.
- **Navigationsmenü:** Dieses Menü befindet sich unterhalb des Logos und bietet klare Produktkategorien, die das Surfen vereinfachen und einen schnellen Zugang zu den verschiedenen Produktbereichen ermöglichen.

4.1.2 Header

In der Kopfzeile steht:

- **Suchleiste:** Diese Suchleiste befindet sich am oberen Rand und ermöglicht es den Nutzern, Produkte durch die Eingabe von Schlüsselwörtern schnell zu finden, was die Effizienz der Produktsuche erhöht.
- **Anmeldeschaltfläche:** Diese Schaltfläche befindet sich in der Nähe der Suchleiste und bietet den Nutzern eine unkomplizierte Möglichkeit, sich in ihr Konto einzuloggen.
- **Warenkorb-Symbol:** Dieses Symbol befindet sich neben der Login-Schaltfläche und zeigt die Anzahl der Artikel im Warenkorb zusammen mit dem Gesamtpreis an, sodass die Nutzer einen schnellen Überblick über ihre Einkaufsaktivitäten erhalten.

4.1.3 Hauptinhalt

Innerhalb des Hauptinhalts der Homepage gibt es:

- **Produkt-Raster:** Zeigt Produkte in einem Rasterformat an, wobei jeder Eintrag gekennzeichnet ist:
 - **Bild:** Bietet eine visuelle Vorschau des Produkts.
 - **Preis:** Wird zur Transparenz deutlich angezeigt.
 - **Titel:** Identifiziert das Produkt.
 - **Link zu Details:** Führt den Nutzer zu einer Seite mit weiteren Informationen.
 - **Schaltfläche „In den Warenkorb“:** Ermöglicht das schnelle Hinzufügen von Artikeln zum Einkaufswagen.
- **Paginierung-Schaltflächen:** Diese Schaltflächen befinden sich am unteren Rand des Produkt-Rasters und ermöglichen es den Benutzern, durch mehrere Produktseiten zu navigieren.

4.1.4 Fußzeile

Unten auf der Homepage gibt es Links zu

- **About us:** Ein Link mit Informationen über Shopnook und sein Team.
- **Contact us:** Ein Link, über den Benutzer mit dem Kundendienst oder Support in Kontakt treten können.
- **Help:** Ein Link zu häufig gestellten Fragen oder Hilfsressourcen, die den Nutzern bei allgemeinen Fragen helfen.

4.2 Warenkorb-Details

Die Abbildung 4.2 zeigt eine typische Schnittstelle für einen Einkaufswagen.

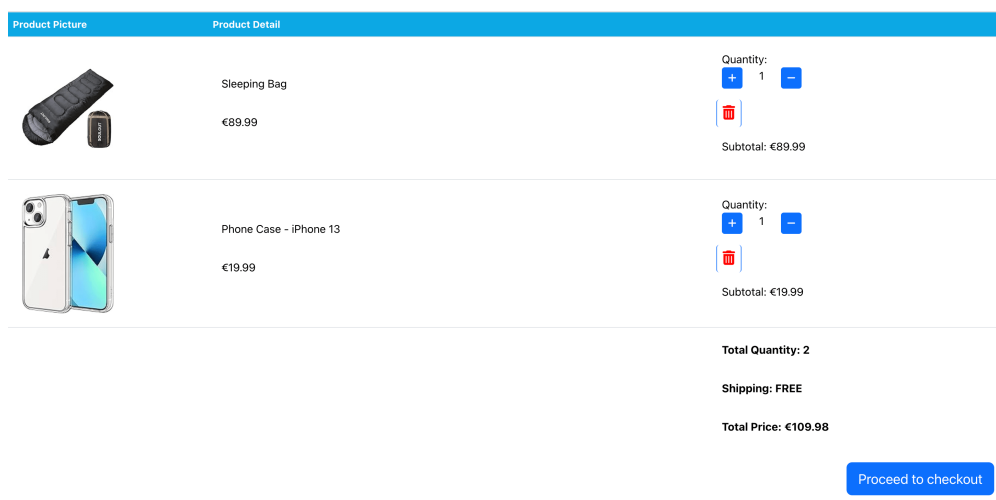


Abbildung 4.2: Warenkorb-Details

Die Detailseite des Warenkorbs bietet eine umfassende und benutzerfreundliche Schnittstelle, die mehrere Schlüsselemente für ein optimales Einkaufserlebnis

enthält. In der Produktliste werden alle Artikel angezeigt, die in den Warenkorb gelegt wurden, jeweils mit einem Bild und detaillierten Informationen. Der Benutzer kann seinen Warenkorb mit der Mengensteuerung leicht verwalten, indem er die Menge jedes Produkts über Schaltflächen zum Erhöhen und Verringern anpasst oder Artikel nach Bedarf entfernt. Die Produktpreise werden übersichtlich dargestellt, einschließlich des Preises pro Einheit und der Zwischensummen, die auf der Grundlage der ausgewählten Menge berechnet werden. Die Bestellübersicht bietet einen Überblick über die Gesamtmenge der Artikel, wobei die Versandkosten als kostenlos gekennzeichnet sind und der endgültige Gesamtpreis deutlich angezeigt wird. Zur Erleichterung der nächsten Schritte im Kaufprozess ist eine auffällige Schaltfläche „Zur Kasse gehen“ enthalten, die den Benutzer zum Fortfahren mit der Bestellung anleitet.

4.3 Checkout-Seite

Nach einem Klick auf die Schaltfläche „Zur Kasse gehen“ leitet die Anwendung den Kunden zur Kassenseite weiter. Auf dieser Seite werden die Benutzer aufgefordert, ihre persönlichen Daten einzugeben, um den Kauf abzuschließen. Dazu gehören wichtige Angaben wie Name, Adresse und Zahlungsinformationen. Die Kassenseite ist so gestaltet, dass sie einen reibungslosen und sicheren Transaktionsprozess gewährleistet und den Benutzer durch die letzten Schritte zur Bestätigung seiner Bestellung und zum Abschluss des Kaufs führt.

Der erste Teil der Checkout-Seite (siehe Abbildung 4.3) enthält ein Kundeninformationsformular, in dem die für die Auftragsabwicklung erforderlichen Angaben erfasst werden. Das Formular enthält Felder für grundlegende persönliche Informationen wie z. B.:

- **Name:** Zur Identifizierung des Kunden.
- **Telefonnummer:** Für Kontaktzwecke.
- **E-Mail:** Für Auftragsbestätigungen und Aktualisierungen.

Darüber hinaus werden in dem Formular Informationen zur Lieferadresse erfasst:

- **Land:** Zur Bestimmung der Versandregion.
- **Straße:** Die spezifische Lieferadresse.
- **Stadt:** Die Stadt für die genaue Zustellung.
- **Bundesland:** Das Bundesland oder die Provinz.
- **Postleitzahl:** Zur genauen Identifizierung des Standorts.

The image shows a checkout form with two main sections: 'Customer' and 'Delivery Address'. The 'Customer' section contains four input fields: 'First Name *', 'Last Name *', 'Phone Number *', and 'Email *'. The 'Delivery Address' section contains five input fields: 'Country *' (a dropdown menu), 'Street *', 'City *', 'State *' (a dropdown menu), and 'Zip Code *'. All input fields are white with rounded ends and are set against a light gray background.

Abbildung 4.3: Checkout-1

Der Rest des Checkout-Formulars, wie in Abbildung 4.4 gezeigt, besteht aus:

- Zahlungsadresse: Dieselbe wie die Lieferadresse.
- Zahlungsmethode: Akzeptiert Debit- oder Kreditkarten; erfordert Kartennummer, Gültigkeitsdatum und CVC-Code.
- Bestellübersicht: Zeigt die Gesamtzahl der Artikel, die Versandkosten und den Gesamtpreis an.
- Schaltfläche Kaufen: Ermöglicht es dem Benutzer, seinen Einkauf abzuschließen.

The screenshot displays a checkout interface with three main sections: 'Payment Address', 'Debit or Credit Card', and 'Order Summary'. The 'Payment Address' section contains five input fields: 'Country *' (a dropdown menu), 'Street *', 'City *', 'State *' (a dropdown menu), and 'Zip Code *'. The 'Debit or Credit Card' section features a large input field for the 'Card number' and a smaller field for 'MM / YY CVC'. The 'Order Summary' section lists 'Total Quantity: 2', 'Shipping: FREE', and 'Total Price: €109.98'. At the bottom center, there is a blue 'Purchase' button.

Abbildung 4.4: Checkout-2

Nach dem Klicken auf die Schaltfläche „Kaufen“ wird eine Benachrichtigung von localhost:4200 ausgelöst, die bestätigt, dass die Bestellung eingegangen ist (siehe Abbildung 4.5). Die Nachricht enthält eine Referenznummer zur Auftragsverfolgung. Nach dieser Bestätigung wird der Kunde zur Homepage zurückgeleitet, und der Warenkorb wird geleert.

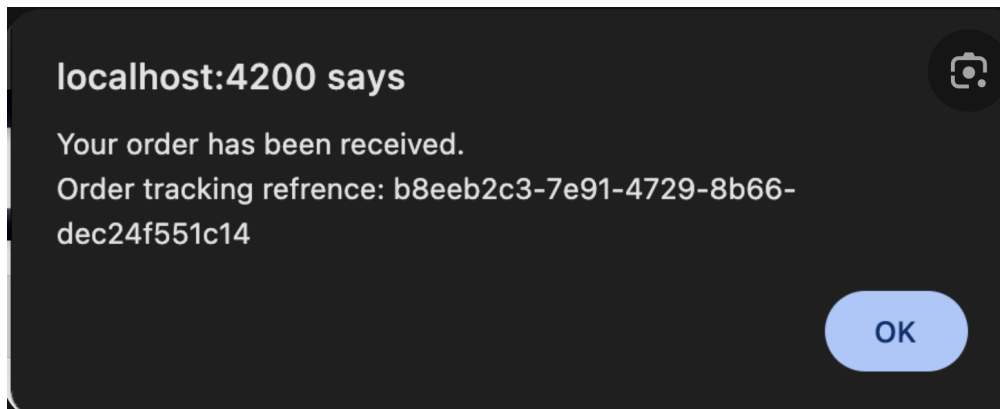


Abbildung 4.5: Purchase-Button

4.4 Login

Wie auf der Startseite zu sehen ist, gibt es eine Leiste mit einer Login-Schaltfläche, mit der sich Kunden authentifizieren können, um auf Seiten zuzugreifen, die nur für Mitglieder zugänglich sind. Die untenstehende Abbildung 4.6 zeigt die Schnittstelle, die nach dem Klicken auf die Schaltfläche „Login“ erscheint.

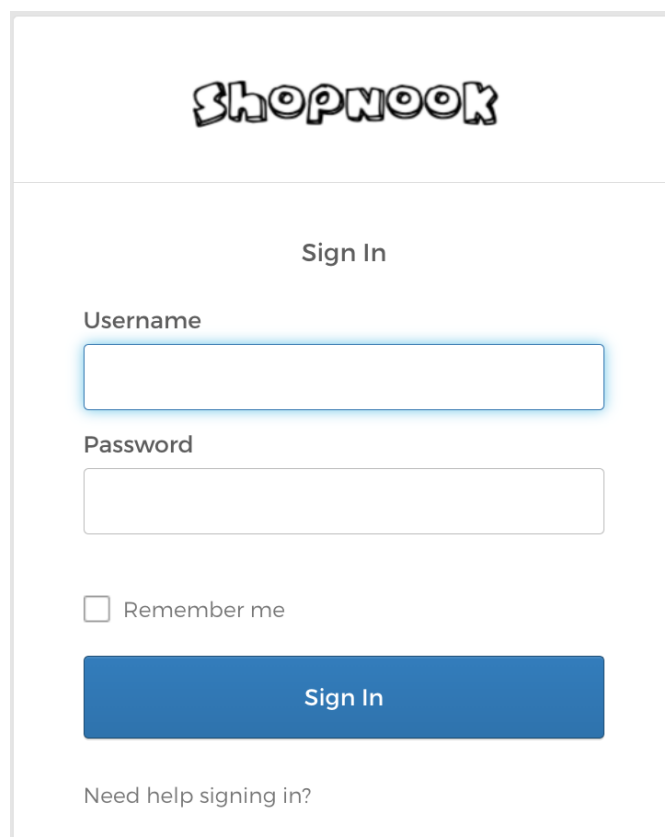
A login form for "SHOPNOOK". The logo "SHOPNOOK" is at the top in a stylized, outlined font. Below it is the heading "Sign In". There are two input fields: "Username" and "Password". Below the password field is a checkbox labeled "Remember me". At the bottom is a blue button labeled "Sign In". Below the button is a link that says "Need help signing in?".

Abbildung 4.6: Login

Nach Eingabe der korrekten E-Mail und des Passworts leitet die App den Kunden zur Homepage weiter. Wie unten dargestellt (siehe Abbildung 4.7), sieht der Kunde zwei Links, die nur eingeloggten Benutzern zur Verfügung stehen: Bestellungen, Mitgliederseiten und eine Schaltfläche „Logout“ zum Ausloggen.

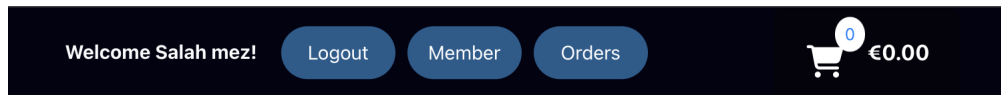


Abbildung 4.7: Login-Leiste

4.4.1 Auftragsverlauf

Nach dem Klicken auf „Orders“ zeigt das Bild eine Liste der letzten Bestellungen mit Details wie Bestellungsreferenz, Gesamtpreis, Gesamtmenge und Datum, wie unten (siehe Abbildung 4.8) gezeigt.

Your Orders			
Order Tracking Reference	Total Price	Total Quantity	Date
b8eeb2c3-7e91-4729-8b66-dec24f551c14	€109.98	2	Aug 22, 2024, 11:18:21 PM
4a0663b3-ac1f-4441-b5a9-98882cbd542f	€62.97	3	Aug 19, 2024, 1:20:06 AM
8e6c8a4d-85e3-4fa4-aad8-67dec17182ae	€20.99	1	Aug 17, 2024, 12:29:07 PM
1052720d-272b-4620-a319-093485f4218d	€83.96	4	Aug 17, 2024, 2:52:52 AM
2d2645b0-6b94-4f15-9597-4ba15e2f01b9	€20.99	1	Aug 17, 2024, 2:31:56 AM
9291f3b6-fe90-4d63-9227-9d0975966729	€23.99	1	Aug 16, 2024, 1:01:28 PM
3620d23c-2ab6-499f-9123-3f46fa0d7a3e	€20.99	1	Aug 16, 2024, 12:03:55 PM
30b2993e-8ff0-4a61-b082-825e909f1073	€13.99	1	Aug 15, 2024, 9:02:12 PM
7d5fec38-4994-4dc5-b8c3-ad686d0c9dad	€33.98	2	Aug 15, 2024, 8:38:37 PM
ce51bdf2-7228-471f-9f85-75a6388702fa	€14.99	1	Aug 15, 2024, 5:03:36 PM
9cd63ca6-856b-4483-ae1-30cfa8873ad	€83.96	4	Aug 14, 2024, 11:27:00 PM

Abbildung 4.8: Auftragsverlauf

4.4.2 Mitgliederseite

Die Mitgliederseite bietet eine Vielzahl von exklusiven Inhalten, die auf angemeldete Benutzer zugeschnitten sind. Sie enthält aktuelle Angebote und Rabatte, die besondere Einsparungen bei ausgewählten Artikeln ermöglichen. Außerdem werden auf der Seite Produktempfehlungen angezeigt, die auf den Vorlieben und dem Surfverhalten der Nutzer basieren. Um das Einkaufserlebnis zu verbessern, werden auch Rückmeldungen anderer Kunden angezeigt, die Aufschluss über die Produktqualität und -zufriedenheit geben. Diese Kombination aus personalisierten Empfehlungen, Werbeaktionen und Erfahrungsberichten hilft den Mitgliedern, fundierte Kaufentscheidungen zu treffen und sich über die neuesten Angebote und Produkte zu informieren.

Unten (siehe Abbildung 4.9) ist eine Illustration des Inhalts der Seite:

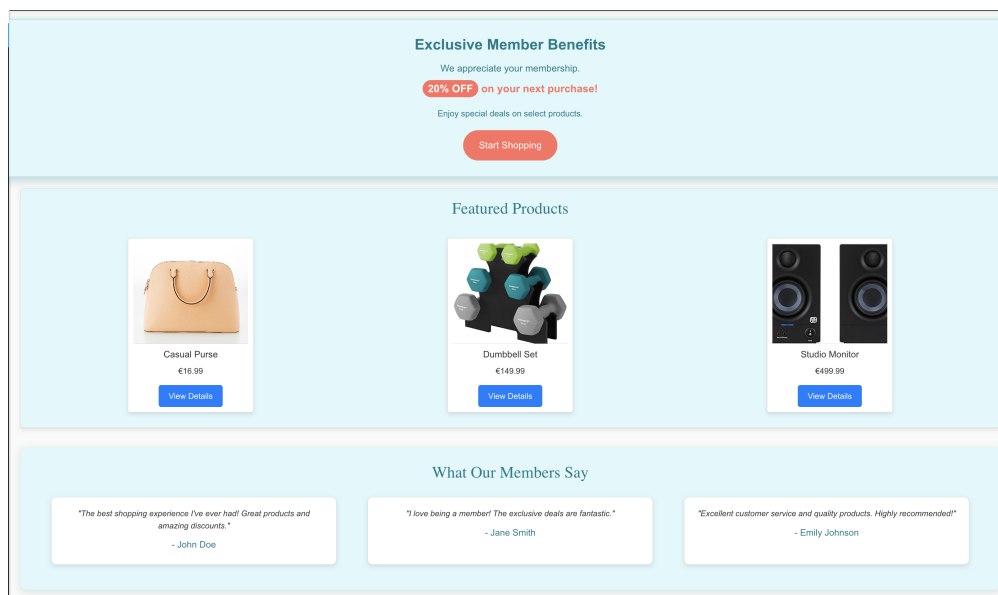


Abbildung 4.9: Mitgliederseite

Abschließend werden anhand dieser Beispiele einige der wichtigsten Funktionen der E-Commerce-Web-App „ShopNook“ vorgestellt. Sie veranschaulichen die Kernaspekte der Benutzererfahrung und konzentrieren sich auf die Funktionalität und die Navigation innerhalb der Plattform.

Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit werden mehrere Schwierigkeiten und Herausforderungen bei der Konzeptentwicklung und Umsetzung der E-Commerce-Warenkorbanwendung „ShopNook“ thematisiert. Zu den Herausforderungen gehören:

- **Technische Integration:** Die nahtlose Integration der verschiedenen Technologien (Angular für das Frontend, Spring Boot für das Backend und MySQL für die Datenbank) stellte eine Herausforderung dar, insbesondere in Bezug auf die Kommunikation zwischen den Komponenten und die Sicherstellung der Datenkonsistenz.
- **Sicherheitsaspekte:** Die Implementierung von Sicherheitsmaßnahmen wie HTTPS-Verschlüsselung und JWT-basierte Authentifizierung erforderte sorgfältige Planung und Tests, um sicherzustellen, dass die Benutzerdaten und Transaktionen geschützt sind.
- **Benutzererfahrung:** Die Gestaltung einer intuitiven und ansprechenden Benutzeroberfläche, die auf verschiedenen Geräten gut funktioniert, war eine weitere Herausforderung. Es war wichtig, ein Gleichgewicht zwischen Funktionalität und Benutzerfreundlichkeit zu finden.

Allerdings sind offene Punkte geblieben, die noch weiter untersucht werden müssen, und sie umfassen:

- **Leistungsoptimierung:** Es gilt herauszufinden, wie die Anwendung unter hoher Last performt und welche Maßnahmen zur Optimierung der Ladezeiten und der Reaktionsfähigkeit ergriffen werden können.
- **Erweiterte Funktionen:** Die Implementierung zusätzlicher Funktionen, wie z.B. personalisierte Empfehlungen oder ein verbessertes Bestandsmanagement, könnte die Benutzererfahrung weiter verbessern.

Literaturverzeichnis

- Ang24a. *Angular - Landing Page*. <https://angular.io>, 2024. Abgerufen am 02.06.2024.
- Ang24b. *Angular components overview*. <https://v17.angular.io/guide/component-overview>, 2024. Abgerufen am 18.08.2024.
- Ang24c. *Routing in single-page applications*. <https://angular.dev/guide/routing/router-tutorial>, 2024. Abgerufen am 18.08.2024.
- Bae. BAELDUNG: *JPA Entities*. <https://www.baeldung.com/jpa-entities/>. Abgerufen am 14.08.2024.
- Bae24. BAELDUNG: *EntityManager*. <https://www.baeldung.com/hibernate-entitymanager/>, 2024. Abgerufen am 14.08.2024.
- Bel23a. BELL, DONALD: *An introduction to the Unified Modeling Language*. <https://developer.ibm.com/articles/an-introduction-to-uml/>, 2023. Abgerufen am 26.07.2024.
- Bel23b. BELL, DONALD: *What is SSL/TLS: An In-Depth Guide*. <https://www.ssl.com/article/what-is-ssl-tls-an-in-depth-guide/>, 2023. Abgerufen am 19.08.2024.
- Cor21. CORPORATION, IBM: *Sequenzdiagramme*. <https://www.ibm.com/docs/de/radfw/9.6.1?topic=diagrams-sequence>, 2021. abgerufen am 28.08.2024.
- Hib. HIBERNATE: *Hibernate*. <https://hibernate.org/orm/what-is-an-orm/>. Abgerufen am 14.08.2024.
- HTM24. *HTML For Beginners*. <https://html.com>, 2024. Abgerufen am 02.06.2024.
- HTT24. *An overview of HTTP*. <https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview>, 2024. Abgerufen am 19.08.2024.
- IBM. IBM: *What is a relational database?* <https://www.ibm.com/topics/relational-databases/>. Abgerufen am 14.08.2024.
- Jav24. *What is Java technology and why do I need it?* https://www.java.com/en/download/help/whatis_java.html, 2024. Abgerufen am 19.08.2024.
- Jet. JETBRAINS: *The IDE for Spring developers*. <https://www.jetbrains.com/idea/spring/>. Abgerufen am 14.08.2024.
- JT. JWT-TEAM: *Introduction to JSON Web Tokens*. <https://jwt.io/introduction>. Abgerufen am 19. August 2024.

- myS. MySQL: *MySQL Workbench Manual*. <https://dev.mysql.com/doc/workbench/en/wb-intro.html>. Abgerufen am 14.08.2024.
- OTa. OAUTH2-TEAM: *What is OAuth 2.0?* <https://auth0.com/intro-to-iam/what-is-oauth-2>. Abgerufen am 19. August 2024.
- OTb. OKTA-TEAM: *What is Okta and what does Okta do?* https://support.okta.com/help/s/article/what-is-okta?language=en_US. Abgerufen am 19. August 2024.
- OTc. OPENID-TEAM: *How OpenId Connect Works*. <https://openid.net/developers/how-connect-works/>. Abgerufen am 19. August 2024.
- RES24. *What is REST?* <https://www.codecademy.com/article/what-is-rest>, 2024. Abgerufen am 18.08.2024.
- Shu22. SHUBEL, MEREDITH: *Typescript*. <https://thenewstack.io/what-is-typescript/>, 2022. Abgerufen am 18.08.2024.
- Spra. SPRING: *JPA Query Methods*. <https://docs.spring.io/spring-data/jpa/reference/jpa/query-methods.html/>. Abgerufen am 14.08.2024.
- Sprb. SPRING: *JPA Repositories*. <https://docs.spring.io/spring-data/jpa/docs/1.6.0.RELEASE/reference/html/jpa.repositories.html/>. Abgerufen am 14.08.2024.
- Sprc. SPRING: *Spring Data JPA*. <https://www.ibm.com/topics/relational-databases/>. Abgerufen am 14.08.2024.
- Sprd. SPRING: *Spring Framework*. <https://spring.io/projects/spring-framework>. Abgerufen am 28. August 2024.
- Spre. SPRING: *Spring Security*. <https://spring.io/projects/spring-security>. Abgerufen am 14. August 2024.
- ST Ja. STRIPE-TEAM: *Stripe Docs Overview*. <https://docs.stripe.com/get-started/> and <https://docs.stripe.com/api>, o. J. Accessed on August 19, 2024.
- ST Jb. STRIPEAPI-TEAM: *Stripe Payment Intents*. https://docs.stripe.com/api/payment_intents, o. J. Accessed on August 19, 2024.
- Teaa. TEAM, GIT: *Git About*. <https://git-scm.com/about>. Abgerufen am 14. August 2024.
- Teab. TEAM, GITHUB: *GitHub About*. <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git#about-github>. Abgerufen am 14. August 2024.
- Teac. TEAM, VISUAL STUDIO: *Using Angular in Visual Studio Code*. <https://code.visualstudio.com/docs/nodejs/angular-tutorial>. Abgerufen am 14. August 2024.
- Tead. TEAM, VISUAL STUDIO: *Visual Studio Code Overview*. <https://code.visualstudio.com/docs>. Abgerufen am 14. August 2024.
- wik. *Wikipedia: Interoperability*. <https://en.wikipedia.org/wiki/Interoperability>. Abgerufen am 27.07.2024.

A

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

B

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:

Datum

Unterschrift der Kandidatin/des Kandidaten

Datum

Unterschrift der Kandidatin/des Kandidaten