

MicroShop – Softwarespezifikation

Fachhochschule Bielefeld, Campus Minden, Sommersemester 2021
Spezielle Gebiete zum Software Engineering, Prof. Dr. Jörg Brunsmann

Autoren: Lennart Dümke, David Nickel, Marc Schwettmann, Yannic Döll, Michael Nickel

Inhaltsverzeichnis

| | |
|--|-----------|
| 1 Einführung | 4 |
| 1.1 Beschreibung und Ziele | 4 |
| 2 Anforderungen | 5 |
| 2.1 Stakeholder | 5 |
| 2.1.1 Interne Stakeholder | 5 |
| 2.1.2 Externe Stakeholder | 6 |
| 2.2 Funktionale Anforderungen | 7 |
| 2.2.1 Akteure | 7 |
| 2.2.2 User Management Service | 8 |
| 2.2.3 Catalog Service | 9 |
| 2.2.4 Cart Service | 10 |
| 2.2.5 Order Service | 12 |
| 2.2.6 Payment Service | 13 |
| 2.3 Nicht-funktionale Anforderungen | 14 |
| 2.3.1 Für das gesamte System | 14 |
| 2.3.2 Speziell für die einzelnen Microservices | 15 |
| 2.4 Graphische Benutzerschnittstellen | 16 |
| 2.4.1 User Management Service | 16 |
| 2.4.2 Catalog Service | 20 |
| 2.4.3 Cart Service | 23 |
| 2.4.4 Order Service | 24 |
| 2.4.5 Payment Service | 26 |
| 2.5 Anforderungen im Detail | 27 |
| 2.5.1 User Management Service | 27 |
| 2.5.2 Catalog Service | 30 |
| 2.5.3 Cart Service | 32 |
| 2.5.4 Order Service | 35 |
| 2.5.5 Payment Service | 37 |
| 3 Technische Beschreibung | 38 |

| | |
|--|-----------|
| 3.1 Systemübersicht | 38 |
| 3.2 Softwarearchitektur | 39 |
| 3.2.1 User Management Service | 39 |
| 3.2.2 Catalog Service | 40 |
| 3.2.3 Cart Service | 41 |
| 3.2.4 Order Service..... | 42 |
| 3.2.5 Payment Service | 42 |
| 3.3 Schnittstellen | 43 |
| 3.4 Datenmodell..... | 43 |
| 3.4.1 Catalog Service | 43 |
| 3.4.2 Cart Service | 44 |
| 3.4.3 Order Service..... | 44 |
| 3.4.4 Payment Service..... | 49 |
| 3.5 Abläufe..... | 56 |
| 3.5.1 User Management Service | 56 |
| 3.5.2 Catalog Service | 59 |
| 3.5.3 Cart Service | 62 |
| 3.5.4 Order Service..... | 64 |
| 3.5.5 Payment Service | 64 |
| 3.6 Entwurf..... | 66 |
| 3.6.1 Catalog Service | 66 |
| 3.6.2 Payment Service..... | 66 |
| 3.7 Fehlerbehandlung | 67 |
| 3.7.1 User Management Service..... | 67 |
| 3.7.2 Catalog Service | 67 |
| 3.7.3 Cart Service | 67 |
| 3.7.4 Order Service..... | 68 |
| 3.7.5 Payment Service | 68 |
| 3.8 Validierung..... | 68 |
| 3.8.1 User Management Service | 68 |
| 3.8.2 Catalog Service..... | 69 |
| 3.8.3 Cart Service | 70 |
| 3.8.4 Order Service | 70 |
| 3.8.5 Payment Service..... | 71 |
| 4 Projektorganisation | 72 |
| 4.1 Verantwortlichkeiten..... | 72 |
| 4.1.1 Zuordnung der Personen zu den Microservices und Technologien | 72 |
| 4.1.2 Rollen..... | 73 |
| 4.1.3 Rollenzuordnung..... | 73 |
| 4.2 Grober Projektplan | 74 |

| | |
|-------------------------------------|-----------|
| 5 Anhänge | 75 |
| 5.1 Schnittstellen | 75 |
| 5.1.1 User Management Service | 75 |
| 5.1.2 Catalog Service | 75 |
| 5.1.3 Cart Service | 79 |
| 5.1.4 Order Service | 83 |
| 5.1.5 Payment Service | 87 |

1 Einführung

1.1 Beschreibung und Ziele

Im Zuge des Moduls „Spezielle Gebiete zum Software Engineering“ wird ein Online-Shop mit dem Namen MicroShop realisiert. MicroShop ist ein Verkäufer von Tontechnik für Endanwender. Zu dem Sortiment gehören unter anderem Mikrofone, Lautsprecher, Kopfhörer und Headsets. Der Online-Shop zielt auf junge, gamingaffine Menschen im Alter zwischen 13 und 30 Jahren ab. Durch das breite Produktspektrum richtet sich MicroShop sowohl an Einsteiger als auch an erfahrene Kunden.

Die Ziele dieses Projekts teilen sich dabei in zwei Gruppen auf: die Sicht des Kunden sowie die technische Sicht. Der Online-Shop soll für den Kunden optisch ansprechend sein und eine hohe Usability aufweisen. Durch ein gutes und verständliches Design möchte sich der Online-Shop von der Konkurrenz abheben. Optimierte Ladezeiten sollen dazu führen, Kunden ein gutes Shopperlebnis zu garantieren. Der Bezahlvorgang wird durch verschiedene Zahlungsdienstleister ermöglicht werden, um dem Kunden größtmögliche Freiheit zu gewähren. Auch der Versand der Waren soll durch verschiedene Versanddienstleister ermöglicht werden.

Es gibt etliche Online-Shops, deren Usability nicht nur sehr niedrig ist, sondern die auch langsam und optisch nicht ansprechend sind. MicroShop soll diese Defizite nicht aufweisen und den Kunden ein gutes Gefühl während des Einkaufens geben. Im Kontext der Digitalisierung ergeben sich häufige Wechsel der Anforderungen an ein System. Um die bestmögliche User Experience zu garantieren, muss es möglich sein, die Technik einfach auf dem neusten Stand zu halten. Durch die Möglichkeit der ständigen Verbesserung und der damit einhergehenden guten Nutzbarkeit, kann MicroShop überzeugen und sich von anderen Online-Shops unterscheiden. MicroShop versteht sich selbst als Prototyp und ist weder ein Forum noch ein Marketplace.

2 Anforderungen

2.1 Stakeholder

2.1.1 Interne Stakeholder

| Funktion / Relevanz | Name | Kontakt / Verfügbarkeit | Wissen | Interessen / Ziele |
|--|----------------|---|--|---|
| Eigentümer, Besitzer des Onlineshops | Julius Brauner | Per E-Mail und telefonisch tagsüber erreichbar, Verfügbarkeit 20%, Minden | Kennt den Markt | Gewinnerhöhung, Entscheidungsfreiheit, Einfluss |
| Manager, für die Planung und Organisation im Unternehmen zuständig | Anton Olten | Per E-Mail und telefonisch erreichbar, Verfügbarkeit 100%, Porta Westfalica | BWL Studium, kennt das Unternehmen und die Mitarbeiter | Hohes Einkommen, Entscheidungsfreiheit, Status |
| Mitarbeiterin, pflegt den Produktkatalog | Magda Krone | Per E-Mail und telefonisch erreichbar, Verfügbarkeit 80%, Minden | Ausbildung Handelskauffrau, kennt die Artikel und Preise | Arbeitsplatz, faires Gehalt, gutes Arbeitsklima, soziale Sicherheit |
| Mitarbeiter, IT-Abteilung, kümmert sich um einen Microservice | Axel Belz | Per E-Mail und telefonisch erreichbar, Verfügbarkeit 100%, Minden | Master Informatik, Cloud Computing, Cloud Native, Vue.js, Django, Python | Arbeitsplatz, faires Gehalt, gutes Arbeitsklima, soziale Sicherheit |

2.1.2 Externe Stakeholder

| Funktion / Relevanz | Name | Kontakt / Verfügbarkeit | Wissen | Interessen / Ziele |
|--|--------------------|---|--|---|
| Kundin, verwendet den Onlineshop | Erica Krimly | Per E-Mail erreichbar, Verfügbarkeit 20%, Köln | Gamerin, kennt sich ein wenig mit Mikrofonen aus | Kauf eines neuen Mikrofons |
| Lieferant, liefert mehrere Artikel aus dem Produktkatalog | Civan Gul | Telefonisch erreichbar, Verfügbarkeit 100%, Herford | Ausbildung zum Industriekaufmann, kennt seine Artikel | Stabile Geschäftsbeziehung, günstige Konditionen, Verlässlichkeit |
| Konkurrent, besitzt selbst einen Onlineshop für Tontechnik | Siegfried Feuchter | Potenziell über E-Mail und Telefon erreichbar, Verfügbarkeit 20%, Hamburg | Kennt den Markt | Gewinnerhöhung, Entscheidungsfreiheit, Einfluss |
| Journalistin, schreibt einen Artikel über den Onlineshop | Monique Wehnert | Per E-Mail und telefonisch erreichbar, Verfügbarkeit 100%, Gütersloh | Kennt die neusten Technologie-Trends, kennt die Konkurrenz | Verfassen eines interessanten Artikels, der mehrere Onlineshops für Tontechnik vergleicht |

2.2 Funktionale Anforderungen

2.2.1 Akteure

Kunde

Das Ziel des Kunden ist es, den Onlineshop MicroShop zu verwenden, um sich Tontechnik zu kaufen.

Zu den Tätigkeiten des Kunden gehört das Durchsuchen des Produktkataloges, um schnell einen gewünschten oder kostengünstigen Artikel zu finden. Mithilfe des Warenkorbs kann er Artikel zwischenlagern, die dieser potenziell kaufen möchte. Will der Kunde sich einen Artikel merken, den dieser aber nicht sofort kaufen möchte, so fügt er diesen Artikel der Merkliste hinzu. Vor dem Kauf muss sich der Kunde registrieren und anmelden. Beim Bestellvorgang kann sich der Kunde für verschiedene Versandarten und Bezahlarten entscheiden. Getätigte Bestellungen sind durch den Kunden einsehbar.

Mitarbeiter

Das Ziel des Mitarbeiters ist es, den Onlineshop zu verwalten.

Zu den Tätigkeiten des Mitarbeiters gehört das Managen des Produktkataloges. Er kann Artikel hinzufügen, entfernen und deren Informationen und Preise ändern. Er sorgt dafür, dass der Kunde immer aktuelle Informationen und Preise sieht und ein möglichst angenehmes Käuferlebnis hat. Ferner kümmert sich der Mitarbeiter um jegliche Fragen des Kunden bezüglich der Artikel und des Onlineshops.

Administrator

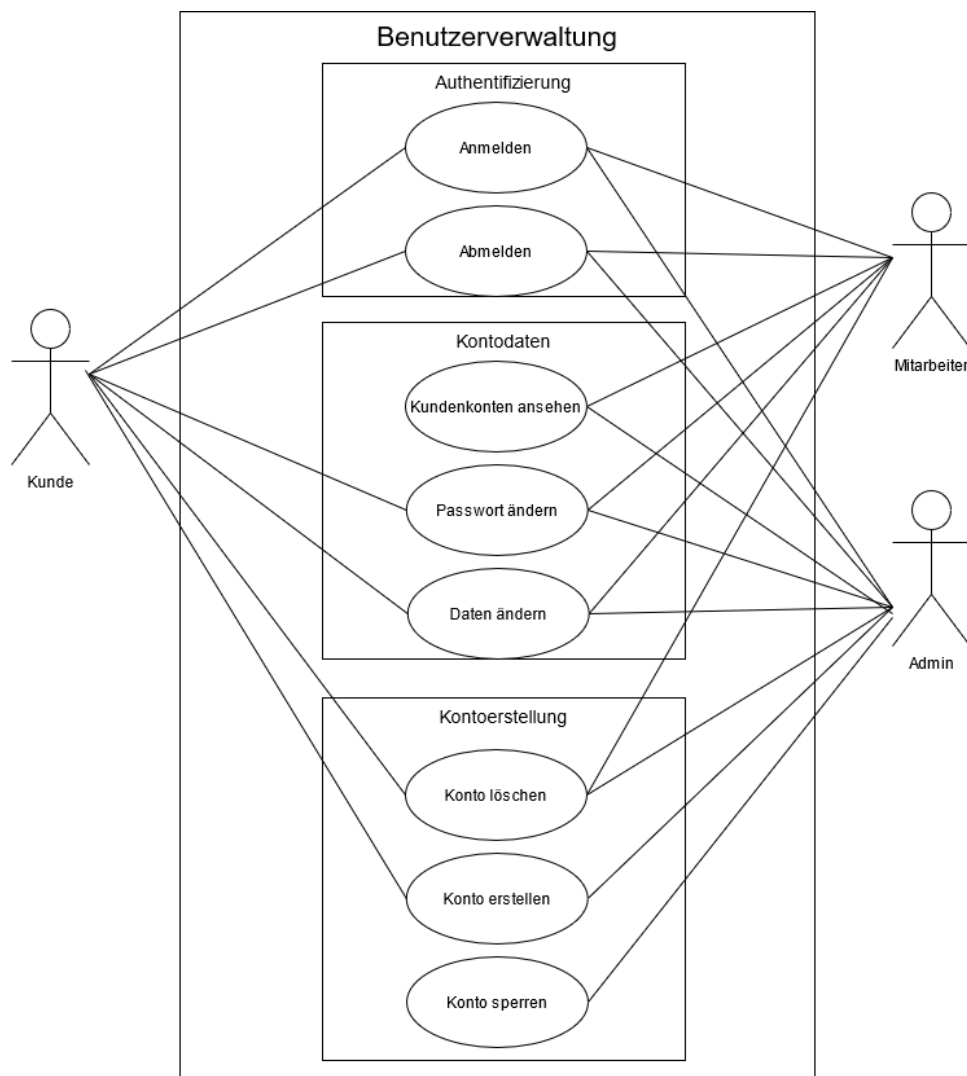
Das Ziel des Administrators ist es, die Accounts der Mitarbeiter zu verwalten. Ferner kann dieser Kundenkonten sperren.

2.2.2 User Management Service

2.2.2.1 Textuelle Anforderungen

| Bezeichnung | Beschreibung |
|----------------------|--|
| Konto erstellen | Ein Kunde muss ein Konto erstellen können. |
| Konto löschen | Ein Kunde soll sein Konto löschen können. |
| Anmelden | Ein Kunde muss sich anmelden können. |
| Abmelden | Ein Kunde muss sich abmelden können. |
| Passwort ändern | Ein Kunde soll sein Passwort ändern können. |
| Daten ändern | Ein Kunde soll seine Daten ändern können. |
| Kundenkonten ansehen | Ein Mitarbeiter soll sich Kundenkonten ansehen können. |
| Konto sperren | Der Administrator soll ein Konto sperren können. |

2.2.2.2 Use Case Diagramm

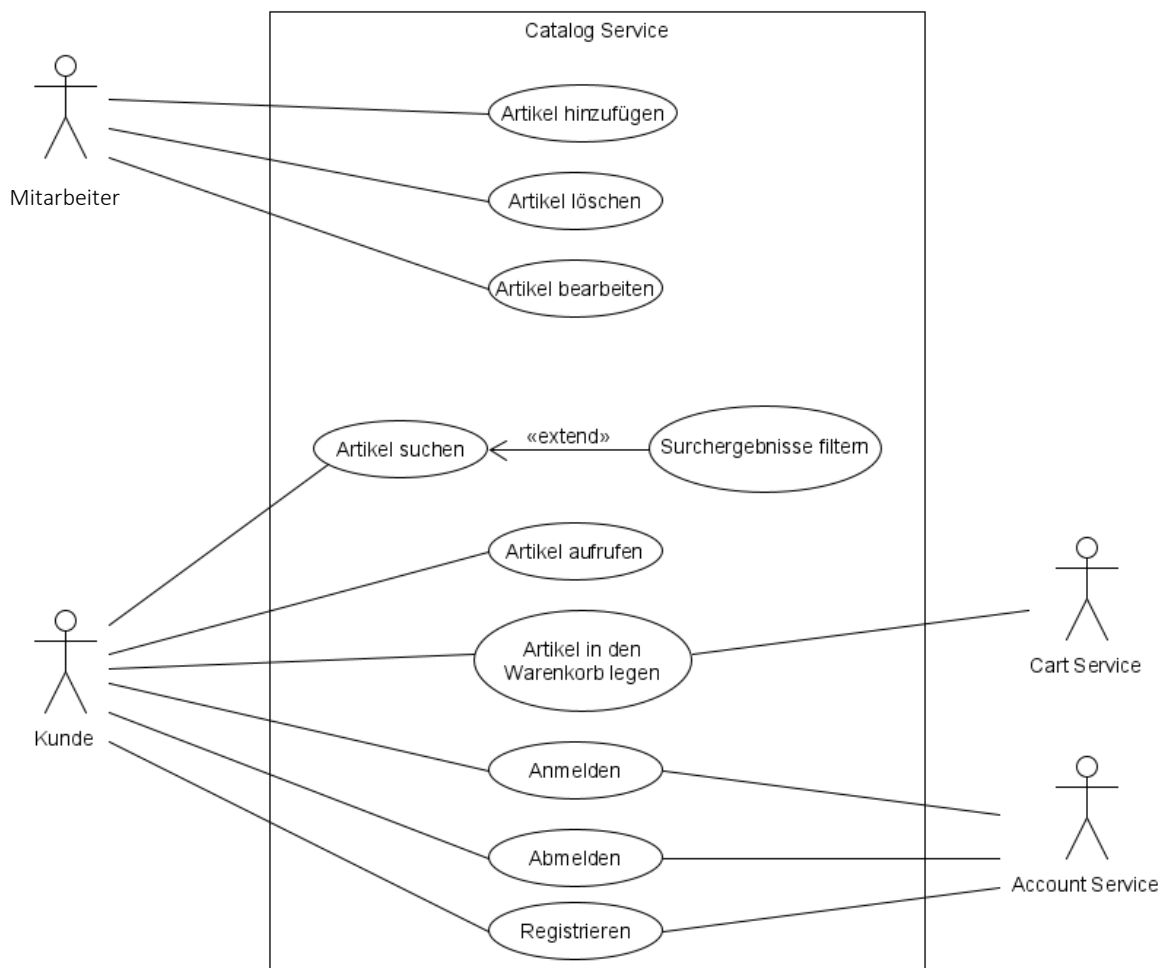


2.2.3 Catalog Service

2.2.3.1 Textuelle Anforderungen

| Bezeichnung | Beschreibung |
|----------------------------------|---|
| Artikel suchen | Ein Kunde muss nach einem Artikel suchen können. |
| Artikel filtern | Ein Kunde muss die angezeigten Artikel nach bestimmten Kriterien (Bsp. Preis) filtern können. |
| Artikel zum Warenkorb hinzufügen | Ein Kunde muss Artikel in den Warenkorb legen können. |
| Detailansicht | Ein Kunde muss durch Klicken auf einen Artikel auf die Detailansicht des Artikels kommen. |
| Artikelinformationen ändern | Ein Mitarbeiter muss Artikelinformationen ändern können. |
| Artikel hinzufügen | Ein Mitarbeiter muss neue Artikel hinzufügen können. |
| Artikel löschen | Ein Mitarbeiter muss Artikel löschen können. |

2.2.3.2 Use Case Diagramm

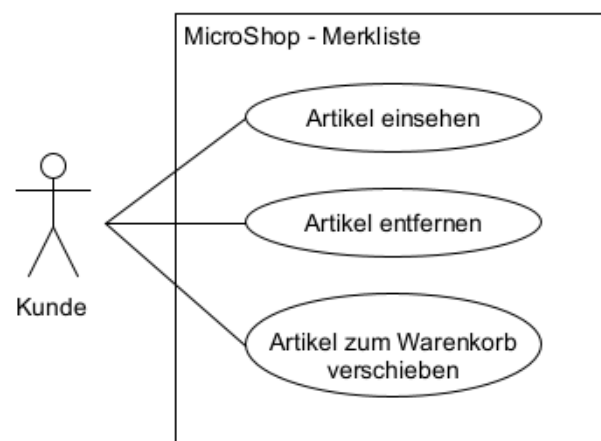
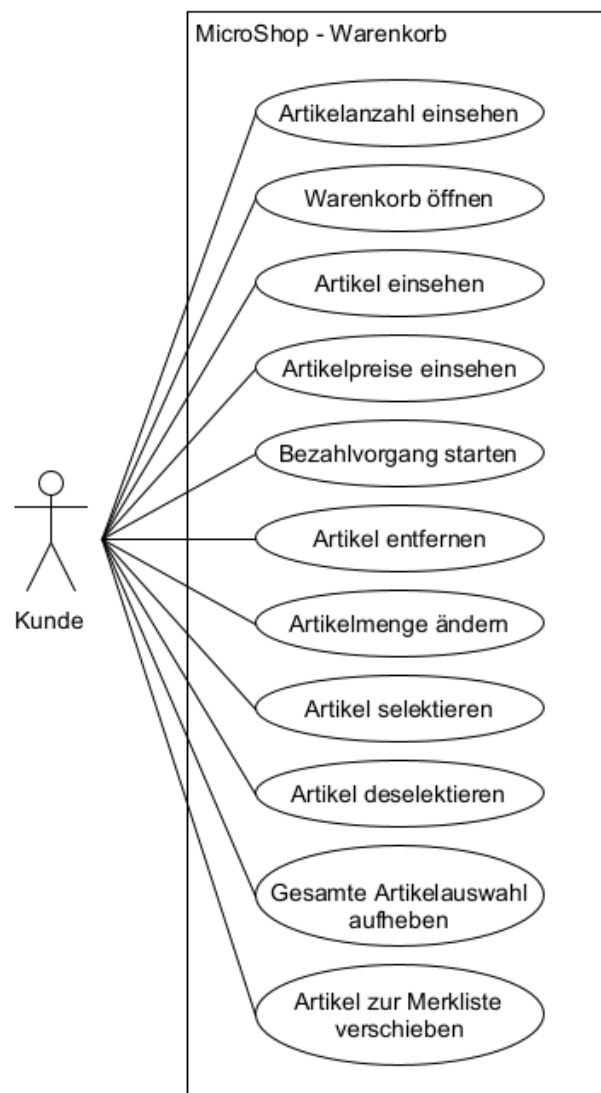


2.2.4 Cart Service

2.2.4.1 Textuelle Anforderungen

| Bezeichnung | Anforderung |
|---|---|
| Artikelanzahl einsehen | Nach dem erfolgreichen Einloggen soll das System dem Kunden die Möglichkeit bieten, die Anzahl der Artikel im Warenkorb einzusehen. |
| Warenkorb öffnen | Nach dem erfolgreichen Einloggen muss das System dem Kunden die Möglichkeit bieten, jederzeit den Warenkorb zu öffnen. |
| Artikel einsehen | Im Warenkorb muss das System alle durch den Kunden hinzugefügten Artikel übersichtlich darstellen. |
| Artikelpreise einsehen | Im Warenkorb muss das System die Preise, der durch den Kunden hinzugefügten Artikel, hervorheben und deren Summe ausrechnen. |
| Bezahlvorgang starten | Im Warenkorb muss das System dem Kunden die Möglichkeit bieten, den Bezahlvorgang für alle ausgewählten Artikel zu starten. |
| Artikel entfernen | Im Warenkorb muss das System dem Kunden die Möglichkeit bieten, einen Artikel aus dem Warenkorb zu entfernen. |
| Artikelanzahl ändern | Im Warenkorb muss das System dem Kunden die Möglichkeit bieten, die Artikelanzahl jedes Artikels zu ändern. |
| Artikel nicht auswählen | Im Warenkorb muss das System dem Kunden die Möglichkeit bieten, einen Artikel nicht auszuwählen, damit dieser Artikel nicht bei der Bestellung berücksichtigt wird. |
| Auswahl aller Artikel aufheben | Im Warenkorb soll das System dem Kunden die Möglichkeit bieten, die Auswahl aller Artikel gleichzeitig aufzuheben. |
| Artikel zur Merkliste hinzufügen | Im Warenkorb soll das System dem Kunden die Möglichkeit bieten, einen Artikel auf die Merkliste zu setzen. |
| Artikel einsehen | In der Merkliste soll das System alle durch den Kunden hinzugefügten Artikel übersichtlich darstellen. |
| Artikel entfernen | In der Merkliste soll das System dem Kunden die Möglichkeit bieten, einen Artikel aus der Merkliste zu entfernen. |
| Artikel zurück in den Warenkorb verschieben | In der Merkliste soll das System dem Kunden die Möglichkeit bieten, einen Artikel zum Warenkorb hinzuzufügen. |

2.2.4.2 Use Case Diagramme

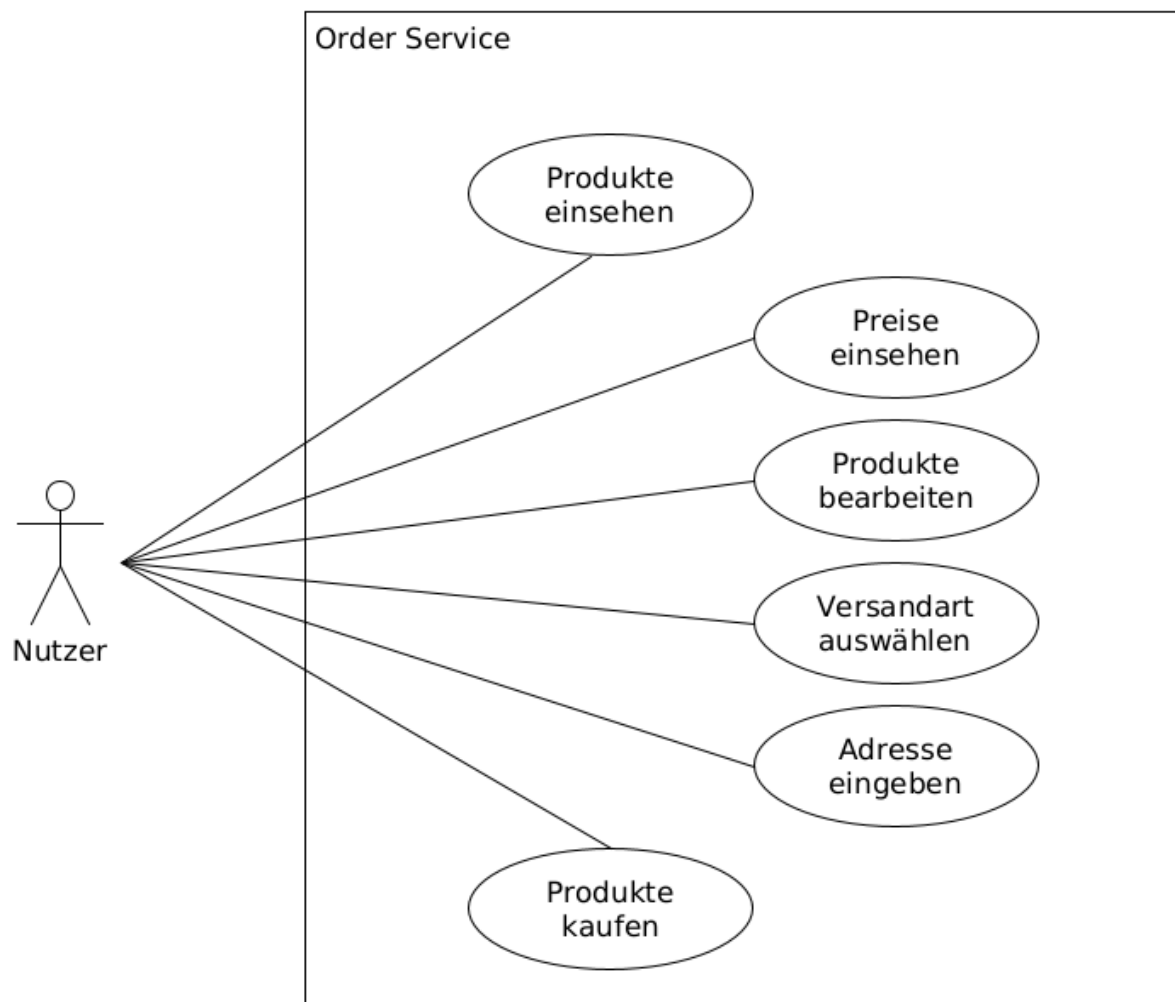


2.2.5 Order Service

2.2.5.1 Textuelle Anforderungen

| Bezeichnung | Beschreibung |
|-----------------------|---|
| Artikel einsehen | Das System muss dem Kunden die Möglichkeit bieten, die Artikel einzusehen, die dieser kaufen wird. |
| Preise einsehen | Das System muss dem Kunden anzeigen, wie viel dieser für welchen Artikel bezahlen muss. |
| Artikel entfernen | Das System muss dem Kunden die Möglichkeit bieten, Artikel aus der Bestellung zu entfernen. |
| Artikelanzahl ändern | Das System muss dem Kunden die Möglichkeit bieten, die Stückzahl der Artikel anzugeben, die dieser kaufen möchte. |
| Gesamtpreis anzeigen | Das System muss dem Kunden den Gesamtpreis der Bestellung anzeigen. |
| Lieferadresse angeben | Das System muss dem Kunden die Möglichkeit bieten, eine Lieferadresse anzugeben. |
| Versandart wählen | Das System muss dem Kunden die Möglichkeit bieten, die Versandart auszuwählen. |

2.2.5.2 Use Case Diagramm

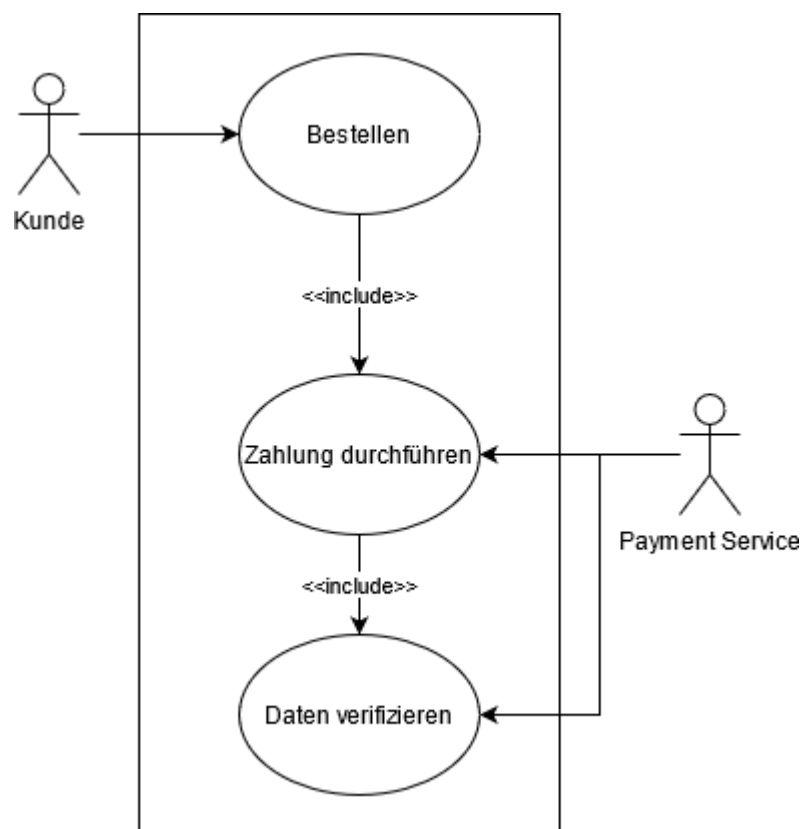


2.2.6 Payment Service

2.2.6.1 Textuelle Anforderungen

| Bezeichnung | Beschreibung |
|-------------------------------------|---|
| Bezahlung abwickeln | Der Payment Service muss die Bezahlung der Produkte abwickeln. |
| Meldung bei erfolgreicher Bezahlung | Der Payment Service muss den Kunden über eine erfolgreiche Zahlung benachrichtigen. |
| Daten verifizieren | Der Payment Service muss die angegebenen Daten verifizieren können. |
| Meldung bei gescheiterter Bezahlung | Der Payment Service muss den Kunden informieren, wenn die Zahlung nicht durchgeführt werden konnte. |

2.2.6.2 Use Case Diagramm



2.3 Nicht-funktionale Anforderungen

2.3.1 Für das gesamte System

| ID | Bezeichnung | Anforderung |
|------|---|--|
| N_01 | (Sicherheit) Rest-Schnittstelle | Die Rest-Schnittstelle soll nach dem OpenAPI-Standard implementiert werden. |
| N_02 | (Sicherheit) Datenübertragung | Die Datenübertrag zu den anderen Microservices soll geschützt erfolgen. |
| N_03 | (Sicherheit) Datenumgang | Mit den Daten soll sicher umgegangen werden. |
| N_04 | (Usability) Intuitive GUI | Die GUI soll intuitiv zu bedienen sein. |
| N_05 | (Usability) Moderne GUI | Die GUI soll modern und übersichtlich gestaltet sein. |
| N_06 | (Zuverlässigkeit) Korrektheit Microservices | Es muss sichergestellt werden, dass die Korrektheit der Microservices gewährleistet ist. |
| N_07 | (Zuverlässigkeit) Korrektheit Daten | Es muss sichergestellt werden, dass die Korrektheit der Daten gewährleistet ist. |
| N_08 | (Zuverlässigkeit) Ausfall Microservice | Fällt ein Microservice aus, sollen die anderen dennoch ansprechbar und bedienbar sein. |
| N_09 | (Effizienz) Schnelle Reaktion | Die Anwendung soll möglichst schnell auf Benutzereingaben reagieren. |
| N_10 | (Portierbarkeit) Ausführung auf Cloud-Umgebung | Die Applikation soll auf einer beliebigen Cloud-Umgebung ausgeführt werden können. |
| N_11 | (Skalierbarkeit) Horizontales Skalieren | Die Anwendung muss horizontal skalierbar sein. |
| N_12 | (Benutzbarkeit) Dokumentation | Der Programmcode und die APIs müssen ausreichend dokumentiert sein. |

2.3.2 Speziell für die einzelnen Microservices

2.3.2.1 User Management Service

| ID | Bezeichnung | Anforderung |
|--------|------------------------|--|
| N_U_01 | Autorisierung | Als Autorisierungsprotokoll soll der Standard von OAuth2 verwendet werden. |
| N_U_02 | Schnelle An-/Abmeldung | Die An- bzw. Abmeldung soll schnell geschehen. |

2.3.2.2 Cart Service

| ID | Bezeichnung | Anforderung |
|---------|-----------------------------|---|
| N_CT_01 | Obergrenze Warenkorbartikel | Es gibt eine Obergrenze an Warenkorbartikeln von 400 Stück. |
| N_CT_02 | Obergrenze Artikelmenge | Es gibt eine Obergrenze bei der Artikelmenge von 999. |

2.3.2.3 Order Service

| ID | Bezeichnung | Anforderung |
|--------|----------------------|---|
| N_O_01 | Versandmöglichkeiten | Eine gewisse Flexibilität muss gegeben sein, damit der Service um weitere Versandmöglichkeiten erweitert werden kann. |

2.3.2.4 Payment Service

| ID | Bezeichnung | Anforderung |
|--------|-------------------------|--|
| N_P_01 | Bezahlungsmöglichkeiten | Eine gewisse Flexibilität muss gegeben sein, damit der Service um weitere Bezahlungsmöglichkeiten erweitert werden kann. |

2.4 Graphische Benutzerschnittstellen

2.4.1 User Management Service

2.4.1.1 Konto-Erstellen-Maske

Page 1

https://www.draw.io

Neues Konto erstellen

Name:

Vorname:

Email:

Passwort:

Passwort wiederholen:

Telefonnummer:

Wohnort:

PLZ:

Straße:

In dieser Maske kann ein Kunde ein neues Konto anlegen. Nachdem die persönlichen Daten eingegeben wurden, wird durch das Drücken des Speicher-Buttons das Konto erstellt.

Abgehandelte User Stories: 1-3, 5, 10

2.4.1.2 Anmeldungs-Maske

The login mask is a vertical form with a light gray border. At the top, the title 'Anmelden' is followed by a horizontal line. Below this, there are two input fields: 'Email-Adresse' containing the text 'johndoe' and 'Passwort' containing seven asterisks. A blue button labeled 'Anmelden' is positioned below the password field. Underneath the button is a blue link labeled 'Forgot Password?'. At the bottom, the text 'Neues Konto' is followed by a blue button labeled 'Erstellen'.

Diese Maske erlaubt es einem Kunden oder Mitarbeiter sich anzumelden. Hierfür muss die E-Mail-Adresse und das Benutzerpasswort eingegeben und auf den Anmelden-Button gedrückt werden. Falls noch kein Konto existiert, kann über den Erstellen-Button die „Konto-Erstellen-Maske“ aufgerufen werden.

Abgehandelte User Stories: 6, 11

2.4.1.3 Kunde-Benutzerverwaltungs-Maske

The user management mask is a web page with a light gray border. At the top left, there is a tab labeled 'Page 1'. The browser address bar shows 'https://www.draw.io'. In the top right corner, there are three window control buttons. Below the browser bar, the title 'Benutzerverwaltung' is displayed on the left, and a blue 'Logout' button is on the right. The main content area is divided into two columns. The left column contains a box titled 'Meine Daten' with a blue 'bearbeiten' link. Inside this box, there are labels for 'Name: Name', 'Vorname: Vorname', 'Geburtsdatum: ---', and 'Adresse: -----' followed by two lines of dots. The right column contains a stack of four buttons: 'Meine Bestellungen' (blue), 'Meine Zahlungseinstellungen' (gray), 'Mein Passwort ändern' (gray), and 'Mein Konto löschen' (gray).

Nachdem sich ein Kunde erfolgreich angemeldet hat, kann die dargestellte Maske geöffnet werden. Sie erlaubt es dem Kunden, seine persönlichen Daten einzusehen und zu bearbeiten. Ferner kann der Kunde seine Bestellungen einsehen, das Passwort ändern oder sein Konto löschen.

Abgehandelte User Stories: 4, 7, 8, 12

2.4.1.4 Kunde-Daten-Ändern-Maske

Page 1

https://www.draw.io

Benutzerverwaltung Logout

Name:

Vorname:

Telefonnummer:

Wohnort:

PLZ:

Straße:

Speichern Abbrechen

Diese Maske erlaubt es dem Kunden, seine persönlichen Daten zu bearbeiten.

Abgehandelte User Stories: 9, 10, 12

2.4.1.5 Mitarbeiter-Benutzerverwaltungs-Maske

Page 1

https://www.draw.io

Benutzerverwaltung Logout

Meine Daten [bearbeiten](#)

Name: Name

Vorname: Vorname

Geburtsdatum: ---

Personalnummer

Bestellungen

Kundenkonten

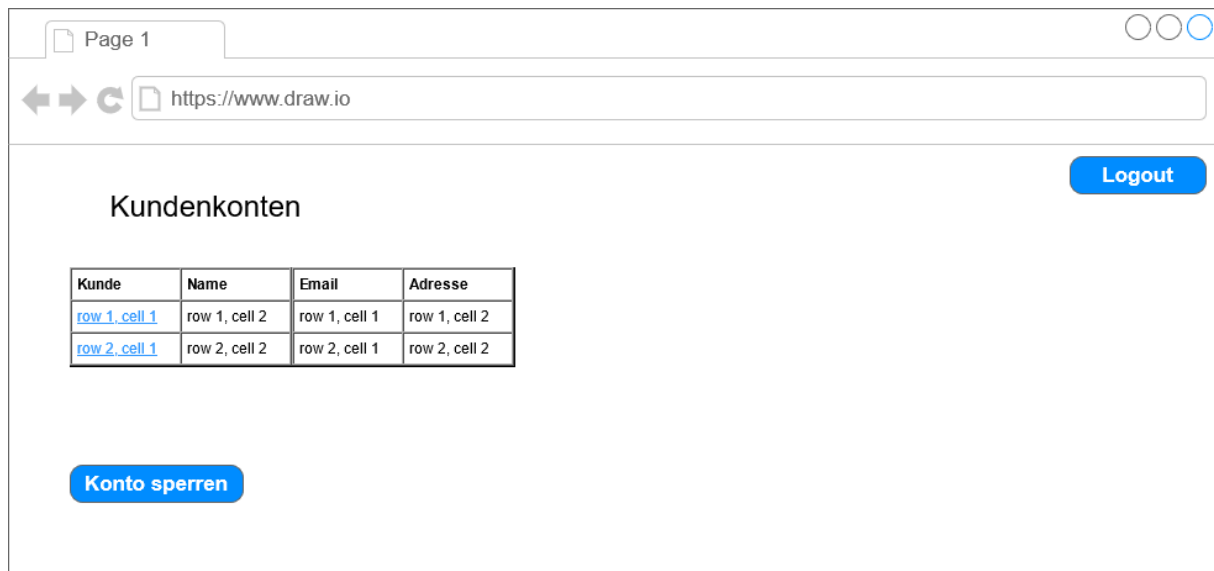
Mitarbeiterkonten

Mein Passwort ändern

Nachdem sich ein Mitarbeiter erfolgreich angemeldet hat, kann die dargestellte Maske geöffnet werden. Sie erlaubt es dem Mitarbeiter, die Kundenkonten einzusehen.

Abgehandelte User Stories: 4, 8, 12

2.4.1.6 Mitarbeiter-Kundenkonten-Maske



Diese Maske gibt dem Mitarbeiter eine Übersicht über die Kundenkonten.

Abgehandelte User Stories: 12, 15, 22

2.4.1.7 Mitarbeiter-Bestellungen-Maske



Diese Maske gibt dem Mitarbeiter eine Übersicht über die Kundenbestellungen.

Abgehandelte User Stories: 12, 17, 19

2.4.1.8 Mitarbeiterkonten-Maske

| Mitarbeiter | Name | Email | Adresse |
|-------------------------------|---------------|---------------|---------------|
| row 1, cell 1 | row 1, cell 2 | row 1, cell 1 | row 1, cell 2 |
| row 2, cell 1 | row 2, cell 2 | row 2, cell 1 | row 2, cell 2 |

Diese Maske kann durch den Administrator aufgerufen werden. Hiermit kann dieser die Konten der Mitarbeiter verwalten.

Abgehandelte User Stories: 12, 20, 21

2.4.2 Catalog Service

2.4.2.1 Produktkatalog-Maske

MicroShop

Registrieren Login

< Warenkorb

Main Page Sales Info

q search

Kategorie 1

Product Name Product Name Product Name

Kategorie 2

Product Name Product Name Product Name

In diese Maske werden Artikel sortiert nach ihrer Kategorie dargestellt.

Abgehandelte User Stories: 1

2.4.2.2 Artikel-Filterungs-Maske

Window Name

MicroShop

Registrieren

Login

< Warenkorb

Main Page

Sales

Info

Marke:

☐ Marke 1

☒ Marke 2

☒ Marke 3

☐ Marke 4

Preis von

bis

Product Name

Product Name

Product Name

Product Name

Product Name

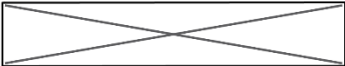
Product Name

Diese Maske stellt dar, wie die Artikel des Onlineshops durch den Kunden gefiltert werden können. Zum einen können nur bestimmte Marken angezeigt werden und zum anderen kann man nach dem Preis eines Artikels filtern.

Abgehandelte User Stories: 1, 2

2.4.2.3 Produktseiten-Maske

Window Name



MicroShop

Registrieren

Login

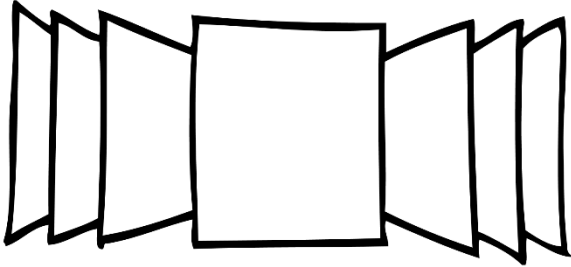
< Warenkorb

Main PageSalesInfo

search

Produkt#1

Produktbeschreibung



Preis: 100,00€

In den Warenkorb legen

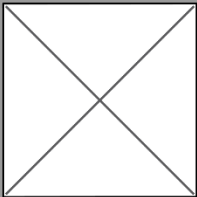
Details

Marke: ...

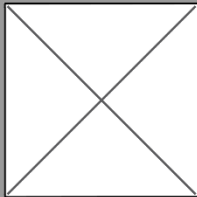
Größe: ...

Stromverbrauch: ...

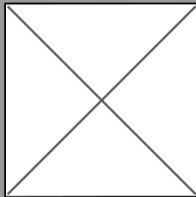
Weitere Produkte



Produkt 2



Produkt 3



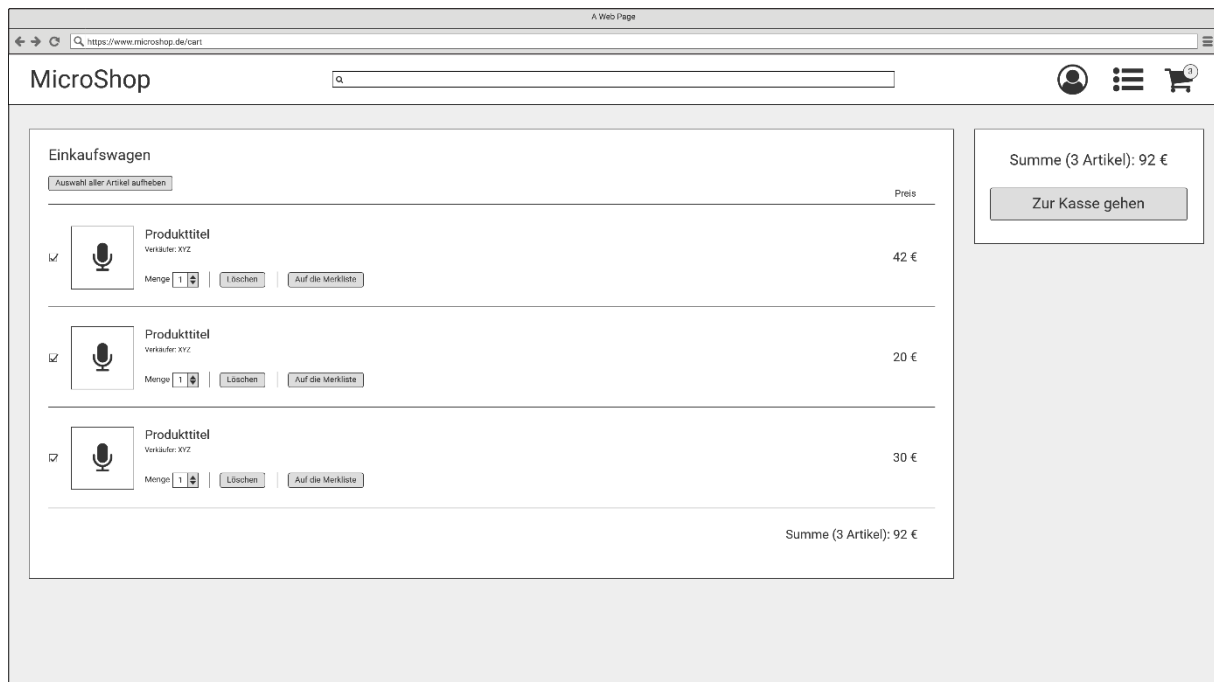
Produkt 4

Diese Maske wird geöffnet, wenn ein Kunde auf einen Artikel draufklickt. Hier werden Informationen über den Artikel und seinen Preis dargestellt. Zu einem Artikel können mehrere Bilder gehören. Kunden können den Artikel durch Klicken des In-Den-Warenkorb-Legen-Buttons dem Warenkorb hinzufügen. Ferner werden ähnliche Produkte vorgeschlagen.

Abgehandelte User Stories: 1, 3, 4

2.4.3 Cart Service

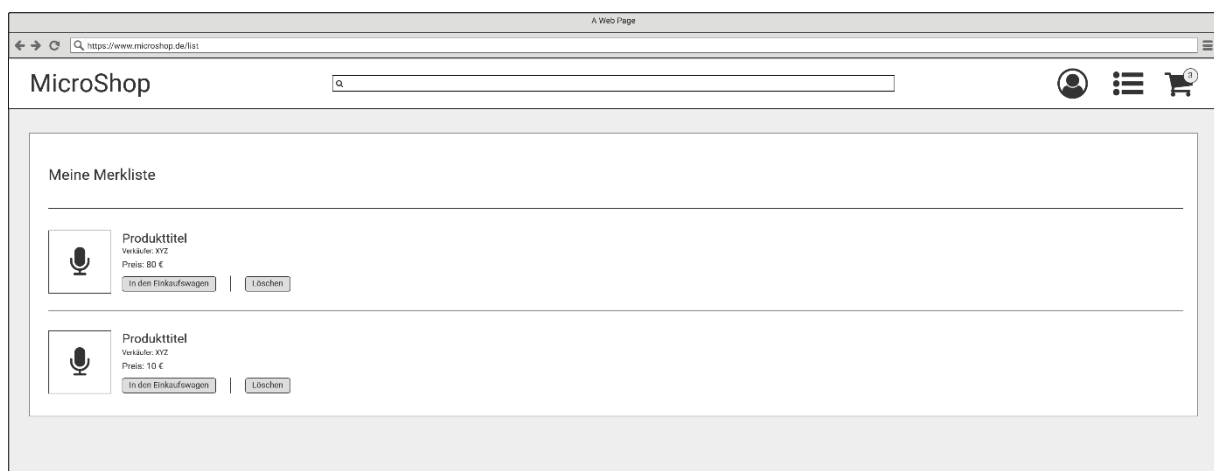
2.4.3.1 Warenkorb-Maske



Diese Maske stellt den Warenkorb dar. Der Kunde kann die hinzugefügten Artikel löschen oder deren Anzahl verändern. Durch Klicken des Zur-Kasse-Gehen-Buttons wird der Bestellvorgang gestartet. Nur markierte Artikel (Checkbox neben dem Produktbild) werden im Bestellvorgang berücksichtigt. Die Warenkorbartikel können zur Merkliste hinzugefügt werden.

Abgehandelte User Stories: 2-10

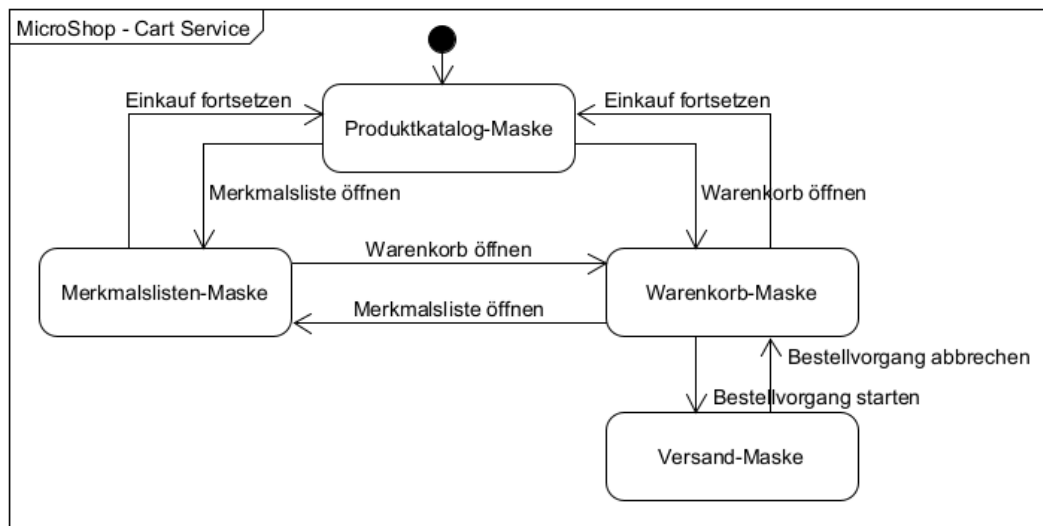
2.4.3.2 Merkmalslisten-Maske



Diese Maske stellt die Merkliste dar. Hier werden Artikel gesichert, die der Kunde später einmal kaufen möchte. Die Artikel können gelöscht werden oder wieder zum Warenkorb verschoben werden.

Abgehandelte User Stories: 1, 11-15

2.4.3.3 Zustandsdiagramm



Die Warenkorb- und Merkmalslisten-Masken können von überall aus mithilfe des Menüs geöffnet werden (hier die Produktkatalog-Maske). Wird der Bestellvorgang gestartet, so wird die Versand-Maske geöffnet.

2.4.4 Order Service

2.4.4.1 Versand-Maske

| 1. Versand | 2. Übersicht | 3. Zahlung | | | | | | |
|---|---|------------|--|---|---|---|--|--|
| <div style="text-align: center;"><h3>Versand</h3><div style="display: flex; justify-content: space-between;"><div><input checked="" type="checkbox"/> DHL Versand <small>2-4 Werktage</small></div><div><input type="checkbox"/> DHL Express Versand <small>1-2 Werktage</small></div></div><div style="margin-top: 20px;"><h3>Adresse</h3><table style="width: 100%; border: none;"><tr><td style="width: 50%;"><small>Vorname</small> <input style="width: 90%;" type="text"/></td><td style="width: 50%;"><small>Nachname</small> <input style="width: 90%;" type="text"/></td></tr><tr><td><small>Straße</small> <input style="width: 90%;" type="text"/></td><td><small>Hausnummer</small> <input style="width: 90%;" type="text"/></td></tr><tr><td><small>PLZ</small> <input style="width: 90%;" type="text"/></td><td><small>Stadt</small> <input style="width: 90%;" type="text"/></td></tr></table><div style="text-align: center; margin-top: 10px;"><input type="button" value="Weiter"/></div></div></div> | | | <small>Vorname</small> <input style="width: 90%;" type="text"/> | <small>Nachname</small> <input style="width: 90%;" type="text"/> | <small>Straße</small> <input style="width: 90%;" type="text"/> | <small>Hausnummer</small> <input style="width: 90%;" type="text"/> | <small>PLZ</small> <input style="width: 90%;" type="text"/> | <small>Stadt</small> <input style="width: 90%;" type="text"/> |
| <small>Vorname</small> <input style="width: 90%;" type="text"/> | <small>Nachname</small> <input style="width: 90%;" type="text"/> | | | | | | | |
| <small>Straße</small> <input style="width: 90%;" type="text"/> | <small>Hausnummer</small> <input style="width: 90%;" type="text"/> | | | | | | | |
| <small>PLZ</small> <input style="width: 90%;" type="text"/> | <small>Stadt</small> <input style="width: 90%;" type="text"/> | | | | | | | |

Nachdem der Kunde den Bestellvorgang gestartet hat, öffnet sich diese Maske. Hier kann die Versandart ausgewählt werden und die Lieferadresse eingegeben werden.

Abgehandelte User Stories: 4-7

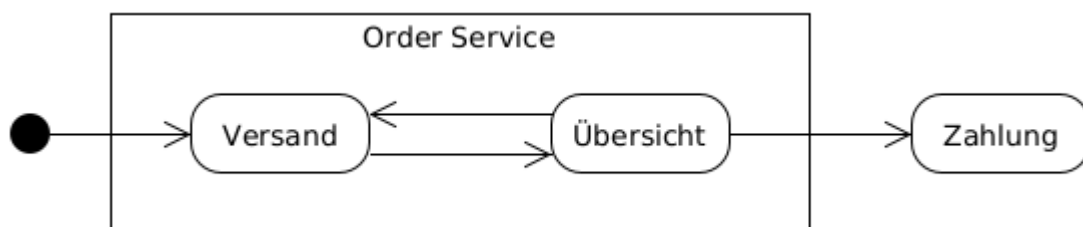
2.4.4.2 Bestellungen-Übersicht-Maske

| 1. Versand | | 2. Übersicht | | 3. Zahlung | | | | | | | | | | | | | | | | | |
|---|--------|--------------|-----------|------------|--|---------|--------|-------|--|-----------------------------------|---|----------|-----------|--------------------------------------|---|---------|-----------|--------------------------------|---|---------|-----------|
| <table border="1"><thead><tr><th>Produkt</th><th>Anzahl</th><th>Preis</th><th></th></tr></thead><tbody><tr><td><input type="checkbox"/> Mikrofon</td><td>1</td><td>109.99 €</td><td>Entfernen</td></tr><tr><td><input type="checkbox"/> Mikrofonarm</td><td>1</td><td>40.00 €</td><td>Entfernen</td></tr><tr><td><input type="checkbox"/> Kabel</td><td>2</td><td>23.00 €</td><td>Entfernen</td></tr></tbody></table> | | | | | | Produkt | Anzahl | Preis | | <input type="checkbox"/> Mikrofon | 1 | 109.99 € | Entfernen | <input type="checkbox"/> Mikrofonarm | 1 | 40.00 € | Entfernen | <input type="checkbox"/> Kabel | 2 | 23.00 € | Entfernen |
| Produkt | Anzahl | Preis | | | | | | | | | | | | | | | | | | | |
| <input type="checkbox"/> Mikrofon | 1 | 109.99 € | Entfernen | | | | | | | | | | | | | | | | | | |
| <input type="checkbox"/> Mikrofonarm | 1 | 40.00 € | Entfernen | | | | | | | | | | | | | | | | | | |
| <input type="checkbox"/> Kabel | 2 | 23.00 € | Entfernen | | | | | | | | | | | | | | | | | | |
| Gesamtpreis: 172.99 € | | | | | | | | | | | | | | | | | | | | | |
| <div>Weiter</div> | | | | | | | | | | | | | | | | | | | | | |

Bevor der Kunde sich für eine Bezahlungsart entscheiden muss, werden nochmal die zu kaufenden Artikel aufgelistet. Hier kann der Kunde erneut Artikel entfernen oder deren Anzahl ändern.

Abgehandelte User Stories: 1–3, 8, 9

2.4.4.3 Zustandsdiagramm



Wird im Warenkorb der Bestellvorgang gestartet, so öffnet sich die Versand-Maske. Nachdem die Versandinformationen ausgewählt und eingegeben wurden, öffnet sich die Bestellungen-Übersichts-Maske. Ist der Kunde zufrieden mit der Bestellung, drückt dieser auf den Weiter-Button und die Zahlungs-Maske öffnet sich.

2.4.5 Payment Service

| 1. Versand | 2. Übersicht | 3. Zahlung |
|---|--------------|------------|
| <div><div>PayPal</div><div>Kreditkarte</div><div>Überweisung</div></div> <div><div>Kartennummer</div><div>CVC</div></div> <div><div>Monat</div><div>Jahr</div></div> <div>Kostenpflichtig bestellen</div> | | |

In dieser Maske kann der Kunde eine Bezahlart auswählen und die dazugehörigen Daten eingeben.

Abgehandelte User Stories: 1-4

2.5 Anforderungen im Detail

2.5.1 User Management Service

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|------|----------------------------|---------------------------------------|---|---|--|-----------|
| U_01 | Konto erstellen | Kunde | Ein Konto erstellen | Ich meine Daten nicht immer neu angeben muss | es die Möglichkeit gibt ein neues Kundenkonto zu erstellen | Muss |
| U_02 | Pflichtfelder kennzeichnen | Kunde | Dass Pflichtfelder bei der Kontoerstellung gekennzeichnet sind | Ich diese erkennen kann | Es eine Kennzeichnung gibt | Kann |
| U_03 | Email verifizieren | Kunde | Dass meine E-Mail verifiziert wird | Niemand ein Konto mit meiner Email-Adresse erstellen kann | Die E-Mail-Adresse verifiziert werden muss | Soll |
| U_04 | Passwort ändern | Kunde, Mitarbeiter oder Administrator | Mein Passwort ändern können | ich immer ein sicheres Passwort habe | Es einen Dialog gibt, um das Passwort zu ändern | Soll |
| U_05 | Passwort Güte | Kunde, Mitarbeiter oder Administrator | Dass mir angezeigt wird, wie sicher mein gewähltes Passwort ist | ich meine Daten sicher schützen kann | Bei der Eingabe optisch die Güte dargestellt wird | Kann |
| U_06 | Passwort zurücksetzen | Kunde, Mitarbeiter oder Administrator | Mein Passwort zurücksetzen können | Ich, wenn ich mein Passwort vergessen habe, mein Konto weiter benutzen kann | Es einen Dialog „Passwort vergessen“ gibt | Soll |

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|-----------|--------------------|---------------------------------------|--|--|--|------------------|
| U_07 | Konto löschen | Kunde | Mein Konto löschen können | Meine Daten entfernt werden | Wenn es die Möglichkeit gibt, ein Konto endgültig zu löschen | Soll |
| U_08 | Kontoübersicht | Kunde, Mitarbeiter oder Administrator | Eine Übersicht über meine angegebenen Daten bekommen | Ich erkennen kann, welche Daten von mir gespeichert sind | Es eine Kontoübersicht in der Benutzerverwaltung gibt | Soll |
| U_09 | Kontodaten ändern | Kunde, Mitarbeiter oder Administrator | Meine Daten ändern können | Ich bei Änderungen mein Konto aktualisieren kann | Die Daten geändert werden können | Soll |
| U_10 | Eingabe validieren | Kunde, Mitarbeiter oder Administrator | Dass meine Eingaben während der Eingabe validiert werden | Ich sofort erkennen kann, wenn meine Daten nicht valide sind | Nicht valide Eingaben markiert werden | Kann |
| U_11 | Anmelden | Kunde, Mitarbeiter oder Administrator | Mich anmelden können | Ich auf den geschützten Bereich zugreifen kann | Eine Anmeldung möglich ist | Muss |
| U_12 | Abmelden | Kunde, Mitarbeiter oder Administrator | Mich abmelden können | Niemand auf meine Daten zugreifen kann | Ich auf allen Tabs im Browser abgemeldet werde | Muss |
| U_13 | Angemeldet bleiben | Kunde, Mitarbeiter oder Administrator | Angemeldet bleiben | Ich mich nicht jedes Mal neu anmelden muss | Die Anmeldung auch beim Schließen des Browsers bestehen bleibt | Kann |

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|-----------|-------------------------------------|--------------------------------|---|--|---|------------------|
| U_14 | Amazon-Konto | Kunde | Mich mit meinem Amazon-Konto anmelden können | Ich nicht ein neues Konto erstellen muss, um etwas zu bestellen | Eine Anmeldung mit meinem Amazon-Konto möglich ist | Kann |
| U_15 | Kundenübersicht | Mitarbeiter oder Administrator | Eine Übersicht über alle Kundenkonten haben | Ich diese schnell überblicken kann | Die Übersicht intuitiv ist und alle Daten beinhaltet | Soll |
| U_16 | Kunden filtern | Mitarbeiter oder Administrator | Die Kunden nach bestimmten Kriterien filtern | Ich nicht lange suchen muss | Es eine Suchfunktion in der Kundenübersicht gibt | Soll |
| U_17 | Bestellungsübersicht | Mitarbeiter oder Administrator | Eine Übersicht über alle Bestellungen haben | Ich diese schnell überblicken kann | Die Übersicht intuitiv ist und alle Daten beinhaltet | Soll |
| U_18 | Bestellungen filtern | Mitarbeiter oder Administrator | Die Bestellungen nach bestimmten Kriterien filtern | Ich nicht lange suchen muss | Es eine Suchfunktion in der Kundenübersicht gibt | Soll |
| U_19 | Link zur Bestellung oder zum Kunden | Mitarbeiter oder Administrator | In der Übersicht eine Bestellung oder einen Kunden auswählen können | Ich direkt auf weitere Informationen weitergeleitet werde | Es einen direkten Link gibt | Soll |
| U_20 | Mitarbeiterkonto erstellen | Administrator | Neue Mitarbeiterkonten erstellen können | Weitere Mitarbeiter auf das System mit Mitarbeiterrechten zugreifen können | Wenn nur der Administrator Mitarbeiterkonten erstellen kann | Muss |

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|------|--------------------------|------------------------|------------------------------------|---|--|-----------|
| U_21 | Mitarbeiterkonto löschen | Administrator | Mitarbeiterkonten entfernen können | Ausscheidenden Mitarbeitern die Rechte genommen werden | Mitarbeiterkonten endgültig gelöscht werden und kein Zugriff auf das System mehr gegeben ist | Muss |
| U_22 | Konto sperren | Administrator | Kundenkonten sperren können | Diese bei auffälligem Verhalten keinen Schaden anrichten können | Mit der entsprechenden Email-Adresse kein neues Konto erstellt werden kann und vom Kunden das alte Konto nicht reaktiviert werden kann | Soll |

2.5.2 Catalog Service

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|-------|-----------------|------------------------|--|--|---|-----------|
| CG_01 | Artikel suchen | Kunde | Nach bestimmten Artikeln suchen können | Ich die gesuchten Produkte schnell im Shop finde | Das Suchen nach Artikelnamen möglich ist und angezeigte Produkte eingegrenzt werden | Muss |
| CG_02 | Artikel filtern | Kunde | Die angezeigten Artikel nach Preis, Verfügbarkeit und Marke filtern können | Ich schneller meinen gesuchten Artikel finde | Filtern der Artikel nach Preis, Verfügbarkeit oder Marke möglich ist | Muss |

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|-----------|--------------------------------|-------------------------------|---|--|---|------------------|
| CG_03 | Artikel anschauen | Kunde | Mir interessante Artikel detailliert anschauen können | Ich alle wichtigen Informationen über den Artikel zusammen mit einer Produktbeschreibung in einer Übersicht sehe | Die Produktseite, die alle Informationen zum Artikel anzeigt, für jeden Artikel aufrufbar ist | Muss |
| CG_04 | Artikel in den Warenkorb legen | Kunde | Einen oder mehrere Artikel in den Warenkorb legen | Ich die Artikel im Anschluss kaufen kann | Ein Artikel in den Warenkorb gelegt werden kann | Muss |
| CG_05 | Artikel hinzufügen | Mitarbeiter | Weitere Artikel hinzufügen | Diese anschließend im Shop verkauft werden können | Es möglich ist, weitere Artikel im Shop hinzuzufügen und diese zu verkaufen | Muss |
| CG_06 | Artikel löschen | Mitarbeiter | Einen Artikel löschen | Dieser nicht mehr im Shop zum Verkauf angeboten wird | Es möglich ist, einen Artikel zu löschen, der dann nicht mehr angezeigt wird | Muss |
| CG_07 | Artikel bearbeiten | Mitarbeiter | Einen Artikel bearbeiten | Ich die Produktbeschreibung oder den Preis ändern kann | Es möglich ist, die Beschreibung oder den Preis eines Artikels zu ändern | Muss |

2.5.3 Cart Service

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|-------|------------------------|------------------------|--|---|--|-----------|
| CT_01 | Warenkorb öffnen | Kunde | Nachdem ich mich eingeloggt habe, jederzeit den Warenkorb öffnen können | Ich die aufgelisteten Artikel verwalten und anschließend kaufen kann | Es einen Warenkorb-Button gibt, der den Warenkorb öffnet | Muss |
| CT_02 | Artikel einsehen | Kunde | Dass mir die Artikel im Warenkorb übersichtlich dargestellt werden | Ich einen Überblick über alle Artikel bekomme | Die Warenkorbartikel übersichtlich aufgelistet werden | Muss |
| CT_03 | Artikelpreise einsehen | Kunde | Dass die Preise der Artikel im Warenkorb hervorgehoben werden und dass die Summe der Preise berechnet wird | Ich die Preise nicht selbst zusammenrechnen muss | Alle Artikelpreise hervorgehoben werden und deren Summe dargestellt wird | Muss |
| CT_04 | Bezahlvorgang starten | Kunde | Den Bezahlvorgang für alle ausgewählten Artikel starten können | Ich die Artikel kaufen kann | Es einen Zur-Kasse-Gehen-Button gibt, der die Versand-Maske öffnet | Muss |
| CT_05 | Artikel entfernen | Kunde | Artikel aus dem Warenkorb entfernen können | Ich Artikel aus dem Warenkorb nehmen kann, die ich nicht mehr kaufen möchte | Artikel aus dem Warenkorb entfernt werden können | Muss |

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|-----------|----------------------------------|---|--|--|---|------------------|
| CT_o6 | Artikelanzahl ändern | Kunde | Die Artikelanzahl eines Warenkorbartikels ändern können | Ich einen Artikel in einer größeren Menge kaufen kann | Es ein Formularfeld gibt, mit welchem die Anzahl eines Artikels verändert werden kann | Muss |
| CT_o7 | Artikel deselektieren | Kunde | Einen Warenkorbartikel deselektieren können | Der Artikel bei der Bestellung nicht berücksichtigt wird | Jeder Artikel mit einer Checkbox selektiert und deselektiert werden kann | Muss |
| CT_o8 | Artikelanzahl einsehen | Kunde | Nachdem ich mich eingeloggt habe, jederzeit die Anzahl der Artikel im Warenkorb sehen können | Ich darüber informiert werde, wie viele Artikel im Warenkorb sind | Neben dem Warenkorb-Button die Anzahl der Artikel angezeigt wird | Soll |
| CT_o9 | Auswahl aller Artikel aufheben | Kunde | Die Auswahl aller Warenkorbartikel aufheben können | Ich nicht manuell alle Artikel deselektieren muss, falls viele Artikel im Warenkorb sind | Es einen Button gibt, der die Auswahl aller Artikel aufhebt | Soll |
| CT_10 | Artikel zur Merkliste hinzufügen | Kunde | Artikel aus dem Warenkorb auf meine Merkliste verschieben können | Ich Artikel speichern kann, die ich vielleicht später kaufen werde | Artikel in die Merkliste verschoben werden können | Soll |

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|-----------|---|-------------------------------|---|---|--|------------------|
| CT_11 | Artikel einsehen | Kunde | Dass mir die Artikel meiner Merklste übersichtlich dargestellt werden | Ich einen Überblick über alle Artikel bekomme | Die Artikel der Merklste übersichtlich aufgelistet werden | Soll |
| CT_12 | Artikel entfernen | Kunde | Artikel von der Merklste entfernen können | Ich Artikel von der Merklste nehmen kann, an denen ich kein Interesse mehr habe | Artikel von der Merklste entfernt werden können | Soll |
| CT_13 | Artikel zurück in den Warenkorb verschieben | Kunde | Artikel der Merklste in den Warenkorb zurückverschieben können | Ich den Artikel anschließend kaufen kann | Artikel der Merklste in den Warenkorb verschoben werden können | Soll |
| CT_14 | Benutzerdefinierte Merklsten | Kunde | Mehrere benutzerdefinierte Merklsten erstellen können | Ich meine gewünschten Artikel sortieren kann | Weitere Merklsten angelegt und benannt werden können | Nicht |
| CT_15 | Merklsten teilen | Kunde | Meine Merklsten mit anderen Benutzern teilen können | Meine Freunde sehen, welche Artikel ich mir wünsche | Merklsten mit anderen Benutzern geteilt werden können | Nicht |

2.5.4 Order Service

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., damit ich ... | Erfüllt, wenn... | Priorität |
|------|------------------------|------------------------|--|--|--|-----------|
| O_01 | Artikel einsehen | Kunde | Die Produkte einsehen können, die ich kaufen werde | Sicherstellen kann, dass ich keine falschen Produkte kaufe | Eine Übersicht der Produkte angezeigt wird | Muss |
| O_02 | Stückzahl einsehen | Kunde | Die Stückzahl der Produkte einsehen können | Nicht zu viel kaufe | Die Stückzahl der Artikel angezeigt wird | Muss |
| O_03 | Preise einsehen | Kunde | Den Gesamtpreis einsehen können | Weiß, wie viel ich bezahlen muss | Der Gesamtpreis angezeigt wird | Muss |
| O_04 | Versandart wählen | Kunde | Die Versandart auswählen können | Entscheiden kann, auf welche Weise mein Paket versendet wird | Der Kunde verschiedene Versandarten auswählen kann | Muss |
| O_05 | Versandkosten einsehen | Kunde | Die Versandkosten einsehen können | Weiß, wie viel ich zusätzlich bezahlen muss | Die Versandkosten einsehbar sind | Muss |

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., damit ich ... | Erfüllt, wenn... | Prior ität |
|-----------|-------------------------|---|---|---|--|-----------------------|
| O_06 | Versandadresse eingeben | Kunde | Meine Versandadresse eingeben können | Meine Ware am gewünschten Ort empfangen | Eine Versandadresse eingegeben werden kann | Muss |
| O_07 | Lieferzeiten einsehen | Kunde | Die voraussichtlichen Lieferzeiten der einzelnen Versandarten einsehen können | Weiß, ob das Paket rechtzeitig ankommt | Die Lieferzeiten einsehbar sind | Muss |
| O_08 | Artikelanzahl ändern | Kunde | Die Menge der Produkte bearbeiten können | Fehlerhafte Eingaben korrigieren kann | Die Artikelanzahl geändert werden kann | Soll |
| O_09 | Artikel entfernen | Kunde | Produkte aus der Bestellung entfernen können | Nur die Produkte kaufen, die ich haben möchte | Artikel aus der Bestellung entfernt werden können | Soll |
| O_10 | Zurück zum Shop | Kunde | Zurück zum Shop gelangen können | Produkte hinzufügen kann, falls ich welche vergessen habe | Der Kunde einfach zum Produktkatalog zurückkehren kann | Soll |

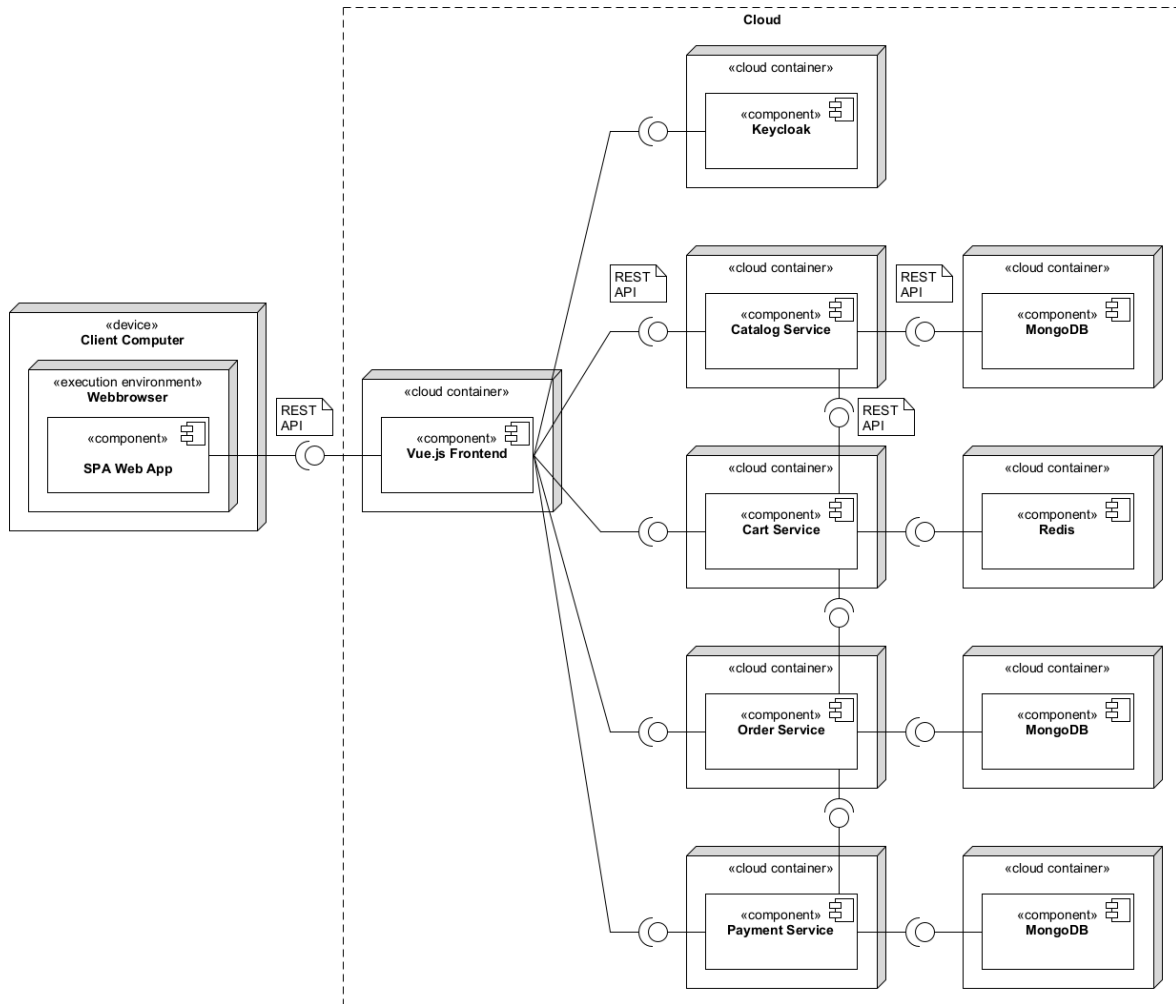
2.5.5 Payment Service

| ID | Name | In meiner Rolle als... | ...möchte ich... | ..., so dass... | Erfüllt, wenn... | Priorität |
|------|-------------------------------------|------------------------|---|--|--|-----------|
| P_o1 | Bezahlverfahren auswählen | Kunde | Verschiedene Bezahlverfahren wählen können | Ich mehr Auswahl habe in der Wahl des Verfahrens | Der Kunde verschiedene Bezahlverfahren auswählen kann | Muss |
| P_o2 | Bezahlung abwickeln | Kunde | Die Zahlung erfolgreich durchführen | Der Kauf erfolgreich abgeschlossen werden kann | Die Zahlung erfolgreich ist | Muss |
| P_o3 | Meldung bei erfolgreicher Bezahlung | Kunde | Darüber benachrichtigt werden, wenn die Zahlung erfolgreich war | Ich eine Rückmeldung habe | Die Zahlung erfolgt ist und eine Benachrichtigung an den Nutzer gesendet wurde | Muss |

3 Technische Beschreibung

3.1 Systemübersicht

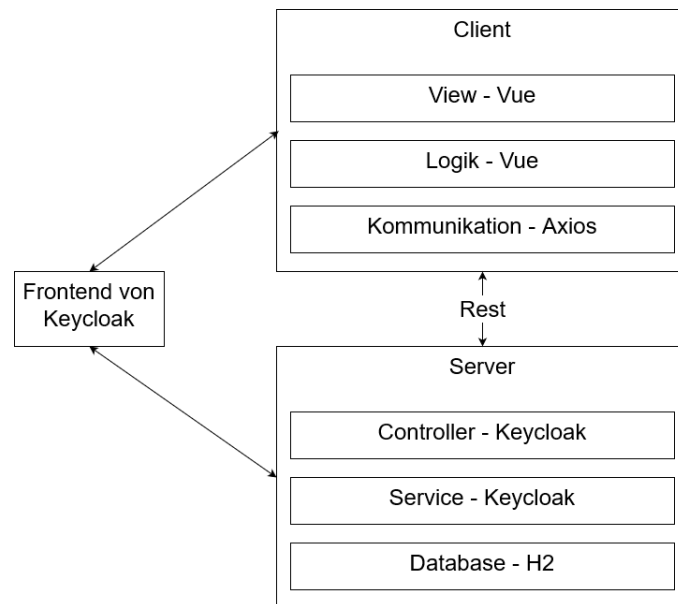
UML-Deployment-Diagramm



Dieses UML-Deployment-Diagramm visualisiert die Microservice-Architektur. Die Kunden verwenden eine Single-Page-Webanwendung. Diese wird durch den Frontend-Server bereitgestellt. Die gesamte Kommunikation zwischen dem Frontend-Server und den einzelnen Microservices läuft mittels einer REST-API. Der Frontend-Server, die Microservices und die Datenbanken werden in Docker-Containern auf einer Cloud-Umgebung ausgeführt. Jeder Microservice hat seine eigene Datenbank. Um die Benutzeraccounts zu verwalten wurde ein Keycloak-Authentifizierungs-Server eingesetzt. Die Microservices kommunizieren direkt miteinander über die REST-API.

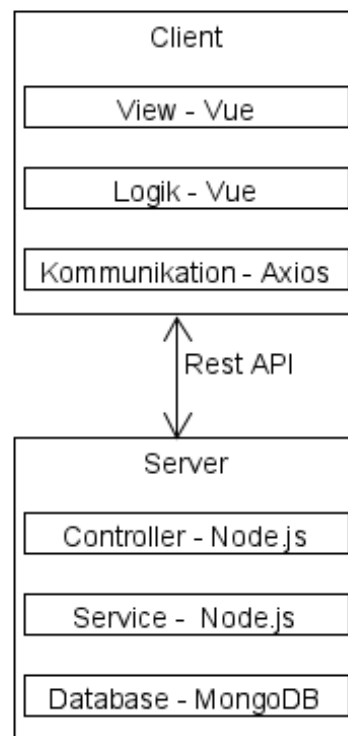
3.2 Softwarearchitektur

3.2.1 User Management Service



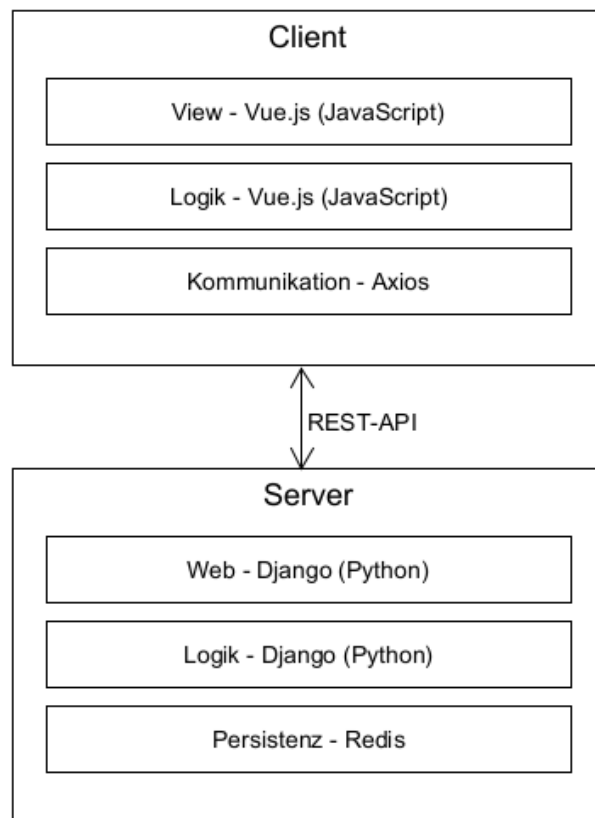
Für das User-Management wird Keycloak mit der Datenbank H2 verwendet. Für das Login wird auch das Frontend von Keycloak genutzt. Teilweise werden jedoch REST-Anfragen an Keycloak von dem Vue-Frontend gesandt.

3.2.2 Catalog Service



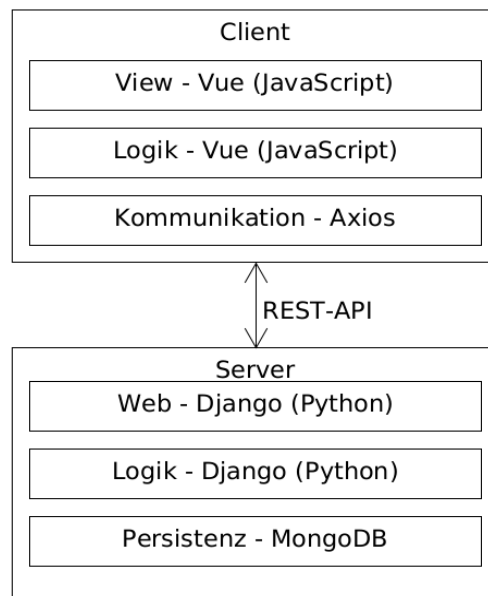
Der Catalog Service verwendet als Backend die JavaScript-Laufzeitumgebung Node.js. Um Daten zu persistieren, wird MongoDB benutzt. Als Frontend wird das JavaScript-Webframework Vue.js eingesetzt. Die Kommunikation zwischen Backend und Frontend geschieht über eine REST-API. Es wird der HTTP-Client Axios verwendet.

3.2.3 Cart Service



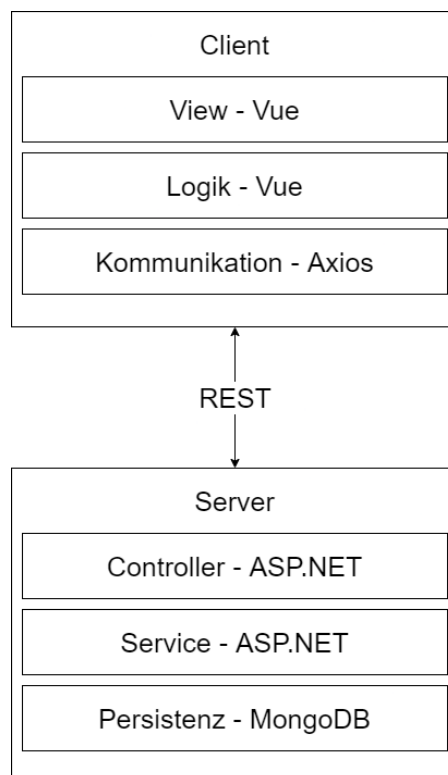
Der Cart Service verwendet als Backend das Python-Webframework Django. Um Daten zu persistieren, wird die Key-Value-Datenbank Redis benutzt. Als Frontend wird das JavaScript-Webframework Vue.js eingesetzt. Die Kommunikation zwischen Backend und Frontend geschieht über eine REST-API. Es wird der HTTP-Client Axios verwendet.

3.2.4 Order Service



Der Order Service verwendet als Backend das Python-Webframework Django. Um Daten zu persistieren, wird MongoDB benutzt. Als Frontend wird das JavaScript-Webframework Vue.js eingesetzt. Die Kommunikation zwischen Backend und Frontend geschieht über eine REST-API. Es wird der HTTP-Client Axios verwendet.

3.2.5 Payment Service



Der Payment Service verwendet als Backend das Web-Application-Framework ASP.NET mit der Programmiersprache C#. Um Daten zu persistieren, wird MongoDB benutzt. Als Frontend wird das JavaScript-Webframework Vue.js eingesetzt. Die Kommunikation zwischen Backend und Frontend geschieht über eine REST-API. Es wird der HTTP-Client Axios verwendet.

3.3 Schnittstellen

Um die REST-konformen Programmierschnittstellen zu beschreiben, wird die OpenAPI Spezifikation verwendet. Alle Teammitglieder haben den Swagger-Editor verwendet. Die Schnittstellendefinitionen finden sich im Anhang wieder. Um sich die API-Dokumentation interaktiv anzuschauen, müssen die Schnittstellendefinitionen in den Swagger-Editor kopiert werden.

3.4 Datenmodell

3.4.1 Catalog Service

JSON-Schema

```
{
  "Article": {
    "type": "object",
    "properties": {
      "id": {
        "type": "string"
      },
      "articlename": {
        "type": "string"
      },
      "description": {
        "type": "string"
      },
      "price": {
        "type": "string"
      },
      "Details": {
        "type": "array",
        "items": {
          "key": {
            "type": "string"
          },
          "value": {
            "type": "string"
          }
        }
      },
      "pictures": {
        "type": "array",
        "items": {
          "type": "number"
        }
      }
    }
  }
}

{
  "Picture": {
    "type": "object",
    "properties": {
      "id": {
        "type": "string"
      },
      "data": {
```

```

        "type": "string",
        "format": "base64"
    }
}
}
}

```

3.4.2 Cart Service

Der Cart Service setzt die NoSQL-Datenbank Redis ein. Die Warenkorbartikel werden in Hashmaps gespeichert. Dieses sind Schlüssel-Wert-Maps. Der Inhalt der Hashmaps wird in der folgenden Tabelle abgebildet. Die Hashmap-Schlüssel sehen folgendermaßen aus: „user:<test>:cart-item:o“.

Für jeden Nutzer werden die Hashmap-Schlüssel in einem Sorted Set gespeichert. Dieses sind sortierte Sammlungen an einzigartigen Strings. Das Sorted Set eines Kunden entspricht seinem Warenkorb. Jeder Kunde hat zwei Sorted Sets: Eine für den Warenkorb und eine für die Merkliste. Der Schlüssel eines Sorted Sets sieht wie folgt aus: „user:<test>:cart“.

| Name | Beschreibung |
|--------------------|---------------------------------|
| article_name | Titel des Artikels |
| article_vendor | Verkäufer des Artikels |
| article_price | Preis des Artikels |
| article_count | Anzahl des Artikels |
| article_image | Bild des Artikels |
| article_catalog_id | Artikel-ID des Produktkataloges |

3.4.3 Order Service

JSON-Schema

```

{
    "$schema": "http://json-schema.org/draft-06/schema#",
    "$ref": "#/definitions/Order",
    "definitions": {
        "Order": {
            "type": "object",
            "additionalProperties": true,
            "properties": {
                "date": {
                    "type": "string",
                    "format": "date-time"
                },
                "user": {
                    "$ref": "#/definitions/User"
                },
                "products": {
                    "type": "array",

```

```

        "items": {
            "$ref": "#/definitions/Product"
        }
    },
    "address": {
        "$ref": "#/definitions/Address"
    },
    "shippingAddress": {
        "$ref": "#/definitions/Address"
    },
    "shippingMethod": {
        "$ref": "#/definitions/ShippingMethod"
    }
},
"required": [
    "address",
    "date",
    "products",
    "shippingAddress",
    "shippingMethod",
    "user"
],
"title": "Order"
},
"Address": {
    "type": "object",
    "additionalProperties": true,
    "properties": {
        "firstName": {
            "type": "string",
            "format": "integer"
        },
        "lastName": {
            "type": "string",
            "format": "integer"
        },
        "street": {

```

```

        "type": "string",
        "format": "integer"
    },
    "number": {
        "type": "string",
        "format": "integer"
    },
    "postCode": {
        "type": "string",
        "format": "integer"
    },
    "city": {
        "type": "string",
        "format": "integer"
    }
},
"required": [
    "city",
    "firstName",
    "lastName",
    "number",
    "postCode",
    "street"
],
"title": "Address"
},
"Product": {
    "type": "object",
    "additionalProperties": true,
    "properties": {
        "article_price": {
            "type": "number"
        },
        "article_name": {
            "type": "string"
        },
        "article_count": {

```

```

        "$ref": "#/definitions/ArticleCount"
    },
    "article_catalog_id": {
        "type": "string"
    },
    "article_vendor": {
        "type": "string"
    },
    "article_image": {
        "type": "string"
    },
    "article_id": {
        "type": "string"
    },
    "checkbox_value": {
        "type": "boolean"
    }
},
"required": [
    "article_catalog_id",
    "article_count",
    "article_id",
    "article_image",
    "article_name",
    "article_price",
    "article_vendor",
    "checkbox_value"
],
"title": "Product"
},
"ShippingMethod": {
    "type": "object",
    "additionalProperties": true,
    "properties": {
        "name": {
            "type": "string"
        },

```

```

        "description": {
            "type": "string"
        },
        "price": {
            "type": "number"
        }
    },
    "required": [
        "description",
        "name",
        "price"
    ],
    "title": "ShippingMethod"
},
"User": {
    "type": "object",
    "additionalProperties": true,
    "properties": {
        "id": {
            "type": "string",
            "format": "uuid"
        }
    },
    "required": [
        "id"
    ],
    "title": "User"
},
"ArticleCount": {
    "anyOf": [
        {
            "type": "integer"
        },
        {
            "type": "string",
            "format": "integer"
        }
    ]
}

```



```

    ],
    "title": "ArticleCount"
  }
}
}

```

3.4.4 Payment Service

JSON-Schema

```

{
  "definitions": {},
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://example.com/object1622807775.json",
  "title": "Invoice",
  "type": "object",
  "required": [
    "invoiceDetails",
    "recipient",
    "items"
  ],
  "properties": {
    "invoiceDetails": {
      "$id": "#root/invoiceDetails",
      "title": "Invoicedetails",
      "type": "object",
      "required": [
        "invoiceNumber",
        "invoiceDate",
        "currencyCode",
        "note",
        "dueDate"
      ],
      "properties": {
        "invoiceNumber": {
          "$id": "#root/invoiceDetails/invoiceNumber",
          "title": "Invoicenummer",
          "type": "string",
          "default": "",
          "examples": [
            "string"
          ],
          "pattern": "^.*$"
        },
        "invoiceDate": {
          "$id": "#root/invoiceDetails/invoiceDate",
          "title": "Invoicedate",
          "type": "string",
          "format": "date",
          "default": "",
          "examples": [
            "2021-06-04"
          ],
          "pattern": "^.*$"
        },
        "currencyCode": {
          "$id": "#root/invoiceDetails/currencyCode",
          "title": "Currencycode",

```

```

        "type": "string",
        "default": "",
        "examples": [
            "string"
        ],
        "pattern": "^.*$"
    },
    "note": {
        "$id": "#root/invoiceDetails/note",
        "title": "Note",
        "type": "string",
        "default": "",
        "examples": [
            "string"
        ],
        "pattern": "^.*$"
    },
    "dueDate": {
        "$id": "#root/invoiceDetails/dueDate",
        "title": "Duedate",
        "type": "string",
        "format": "date",
        "default": "",
        "examples": [
            "2021-06-04"
        ],
        "pattern": "^.*$"
    }
}

,
"recipient": {
    "$id": "#root/recipient",
    "title": "Recipient",
    "type": "object",
    "required": [
        "firstName",
        "surname",
        "address",
        "emailAddress",
        "phoneNumber",
        "shippingInfo"
    ],
    "properties": {
        "firstName": {
            "$id": "#root/recipient/firstName",
            "title": "Firstname",
            "type": "string",
            "default": "",
            "examples": [
                "string"
            ],
            "pattern": "^.*$"
        },
        "surname": {
            "$id": "#root/recipient/surname",
            "title": "Surname",
            "type": "string",
            "default": "",
            "examples": [
                "string"
            ],

```

```

        "pattern": "^.*$"
    },
    "address": {
        "$id": "#root/recipient/address",
        "title": "Address",
        "type": "object",
        "required": [
            "street",
            "postalCode",
            "city",
            "countryCode"
        ],
        "properties": {
            "street": {
                "$id":
"#root/recipient/address/street",
                "title": "Street",
                "type": "string",
                "default": "",
                "examples": [
                    "string"
                ],
                "pattern": "^.*$"
            },
            "postalCode": {
                "$id":
"#root/recipient/address/postalCode",
                "title": "Postalcode",
                "type": "string",
                "default": "",
                "examples": [
                    "string"
                ],
                "pattern": "^.*$"
            },
            "city": {
                "$id":
"#root/recipient/address/city",
                "title": "City",
                "type": "string",
                "default": "",
                "examples": [
                    "string"
                ],
                "pattern": "^.*$"
            },
            "countryCode": {
                "$id":
"#root/recipient/address/countryCode",
                "title": "Countrycode",
                "type": "string",
                "default": "",
                "examples": [
                    "string"
                ],
                "pattern": "^.*$"
            }
        }
    }
},
{
    "emailAddress": {
        "$id": "#root/recipient/emailAddress",

```

```

        "title": "Emailaddress",
        "type": "string",
        "default": "",
        "examples": [
            "string"
        ],
        "pattern": "^.*$"
    },
    "phoneNumber": {
        "$id": "#root/recipient/phoneNumber",
        "title": "Phonenumber",
        "type": "string",
        "default": "",
        "examples": [
            "string"
        ],
        "pattern": "^.*$"
    },
    "shippingInfo": {
        "$id": "#root/recipient/shippingInfo",
        "title": "Shippinginfo",
        "type": "object",
        "required": [
            "firstName",
            "surname",
            "address"
        ],
        "properties": {
            "firstName": {
                "$id":
"#root/recipient/shippingInfo/firstName",
                "title": "Firstname",
                "type": "string",
                "default": "",
                "examples": [
                    "string"
                ],
                "pattern": "^.*$"
            },
            "surname": {
                "$id":
"#root/recipient/shippingInfo/surname",
                "title": "Surname",
                "type": "string",
                "default": "",
                "examples": [
                    "string"
                ],
                "pattern": "^.*$"
            },
            "address": {
                "$id":
"#root/recipient/shippingInfo/address",
                "title": "Address",
                "type": "object",
                "required": [
                    "street",
                    "postalCode",
                    "city",
                    "countryCode"
                ],
                "properties": {

```

```

        "street": {
            "$id":
"#root/recipient/shippingInfo/address/street",
            "title": "Street",
            "type": "string",
            "default": "",
            "examples": [
                "string"
            ],
            "pattern": "^.*$"
        },
        "postalCode": {
            "$id":
"#root/recipient/shippingInfo/address/postalCode",
            "title": "Postalcode",
            "type": "string",
            "default": "",
            "examples": [
                "string"
            ],
            "pattern": "^.*$"
        },
        "city": {
            "$id":
"#root/recipient/shippingInfo/address/city",
            "title": "City",
            "type": "string",
            "default": "",
            "examples": [
                "string"
            ],
            "pattern": "^.*$"
        },
        "countryCode": {
            "$id":
"#root/recipient/shippingInfo/address/countryCode",
            "title":
"Countrycode",
            "type": "string",
            "default": "",
            "examples": [
                "string"
            ],
            "pattern": "^.*$"
        }
    },
    "items": {
        "$id": "#root/items",
        "title": "Items",
        "type": "array",
        "default": [],
        "items": {
            "$id": "#root/items/items",
            "title": "Items",

```

```

"type": "object",
"required": [
    "name",
    "description",
    "quantity",
    "discount",
    "tax",
    "amount"
],
"properties": {
    "name": {
        "$id": "#root/items/items/name",
        "title": "Name",
        "type": "string",
        "default": "",
        "examples": [
            "string"
        ],
        "pattern": "^.*$"
    },
    "description": {
        "$id": "#root/items/items/description",
        "title": "Description",
        "type": "string",
        "default": "",
        "examples": [
            "string"
        ],
        "pattern": "^.*$"
    },
    "quantity": {
        "$id": "#root/items/items/quantity",
        "title": "Quantity",
        "type": "integer",
        "examples": [
            0
        ],
        "default": 0
    },
    "discount": {
        "$id": "#root/items/items/discount",
        "title": "Discount",
        "type": "integer",
        "examples": [
            0
        ],
        "default": 0
    },
    "tax": {
        "$id": "#root/items/items/tax",
        "title": "Tax",
        "type": "integer",
        "examples": [
            0
        ],
        "default": 0
    },
    "amount": {
        "$id": "#root/items/items/amount",
        "title": "Amount",
        "type": "object",
        "required": [

```

```

        "currencyCode",
        "value"
    ],
    "properties": {
        "currencyCode": {
            "$id":
"#root/items/items/amount/currencyCode",
            "title": "Currencycode",
            "type": "string",
            "default": "",
            "examples": [
                "string"
            ],
            "pattern": "^.*$"
        },
        "value": {
            "$id":
"#root/items/items/amount/value",
            "title": "Value",
            "type": "integer",
            "examples": [
                0
            ],
            "default": 0
        }
    }
}

}

}

}

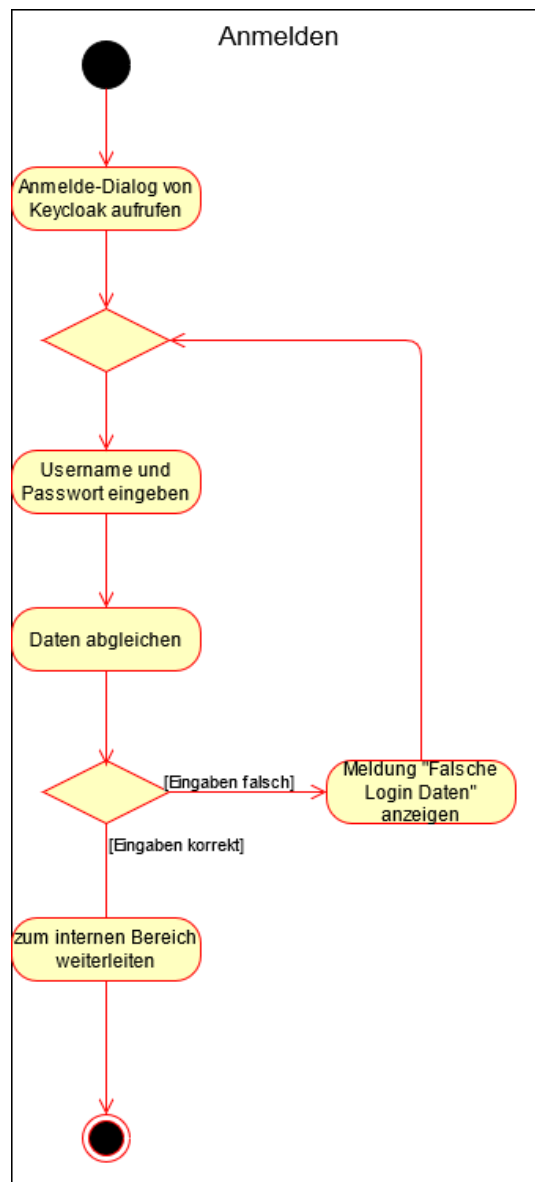
}

```

3.5 Abläufe

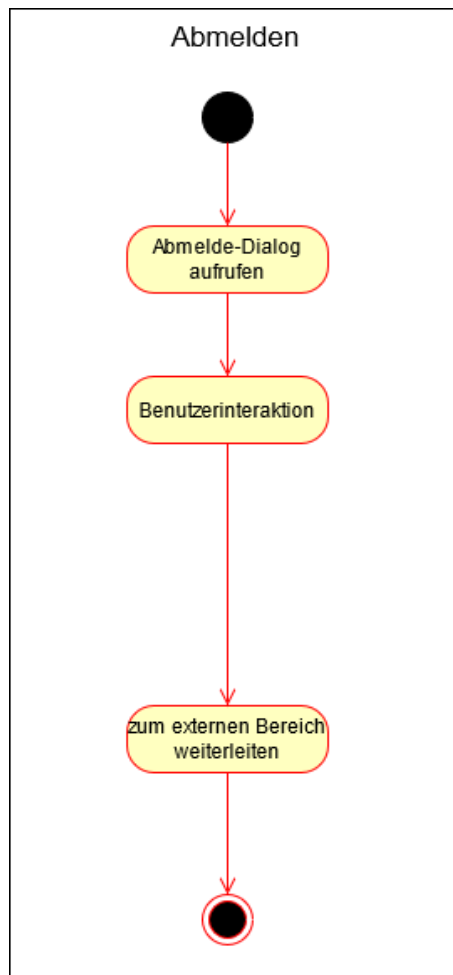
3.5.1 User Management Service

3.5.1.1 Anmelden



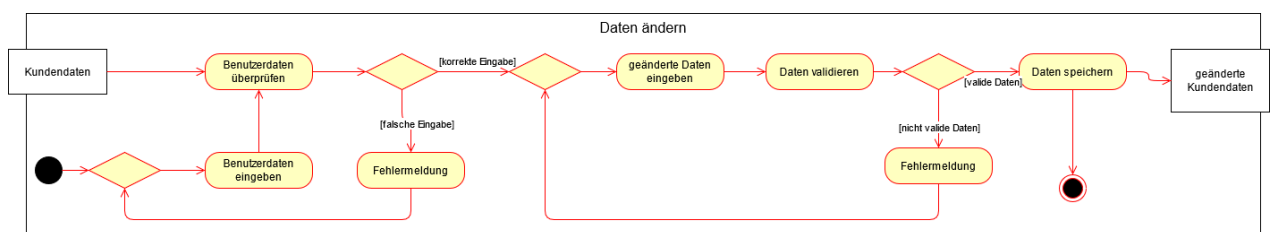
Der Kunde wird aufgefordert, seine Anmeldedaten anzugeben. Sind diese korrekt, wird er zum internen Bereich weitergeleitet.

3.5.1.2 Abmelden



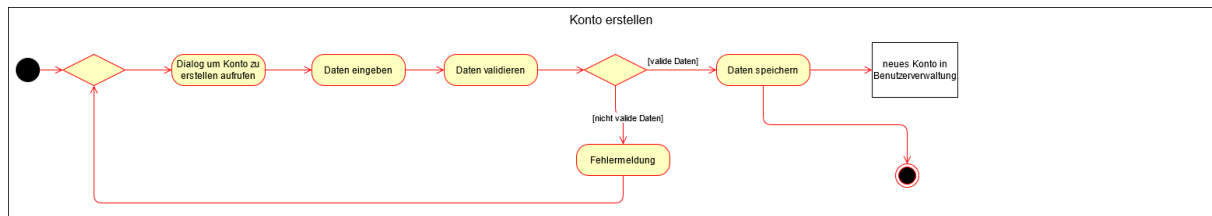
Wenn der Kunde sich abmeldet, wird er zum externen Bereich weitergeleitet.

3.5.1.3 Daten ändern



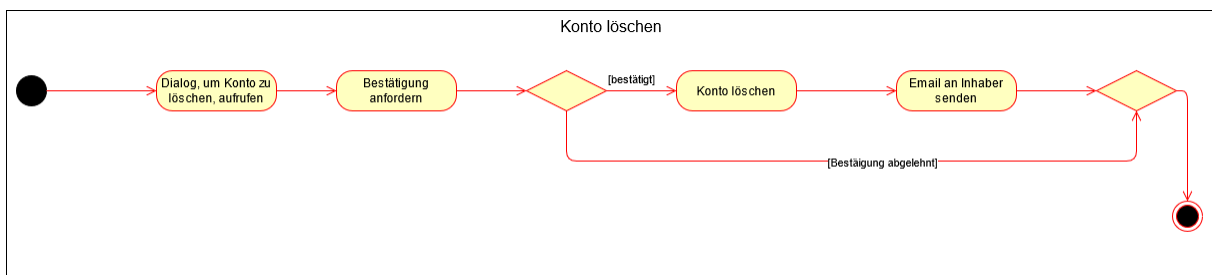
Der Kunde muss sich mit korrekten Benutzerdaten anmelden. Anschließend kann der Kunde die geänderten Daten eingeben. Sind diese valide, werden sie in der Datenbank gespeichert.

3.5.1.4 Konto erstellen



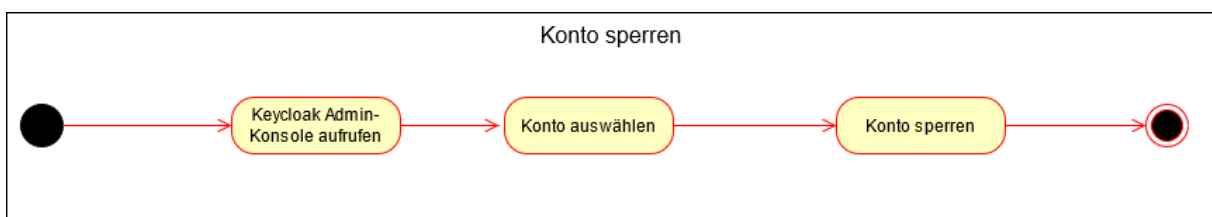
Um ein Konto zu erstellen, muss der Kunde valide Daten eingeben. Geschieht dies, wird das Konto in der Datenbank gespeichert.

3.5.1.5 Konto löschen



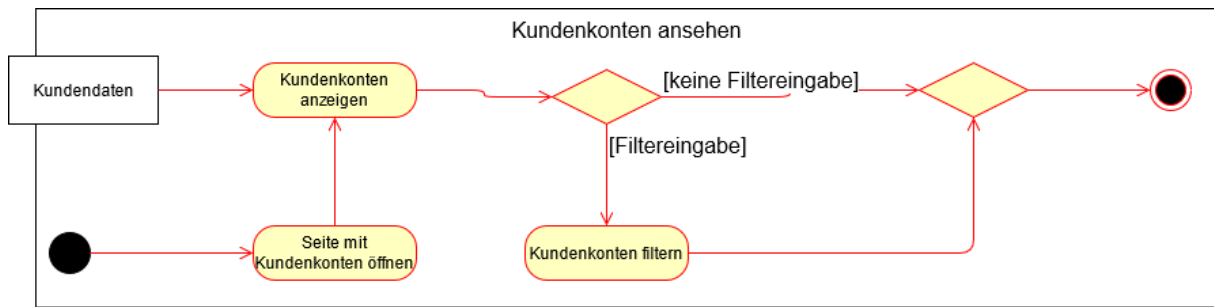
Um ein Kundenkonto zu löschen, wird der Kunde zur Bestätigung aufgefordert. Das Konto wird gelöscht und eine E-Mail an den Inhaber gesendet.

3.5.1.6 Konto sperren



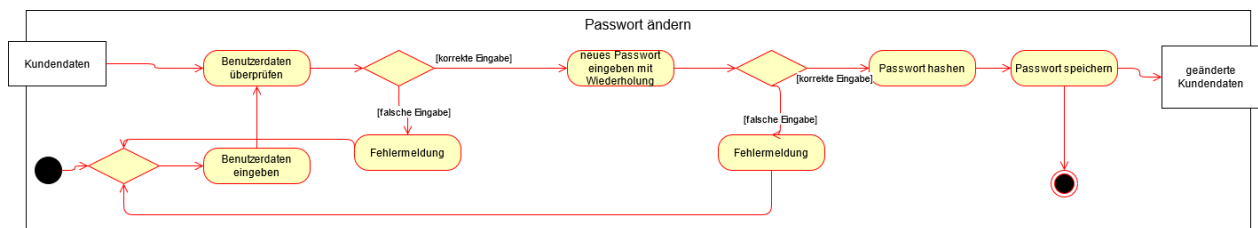
Der Administrator kann ein Kundenkonto auswählen, welches er sperren möchte.

3.5.1.7 Kundenkonten ansehen



Auf der Seite mit den Kundenkonten können Filter eingestellt werden.

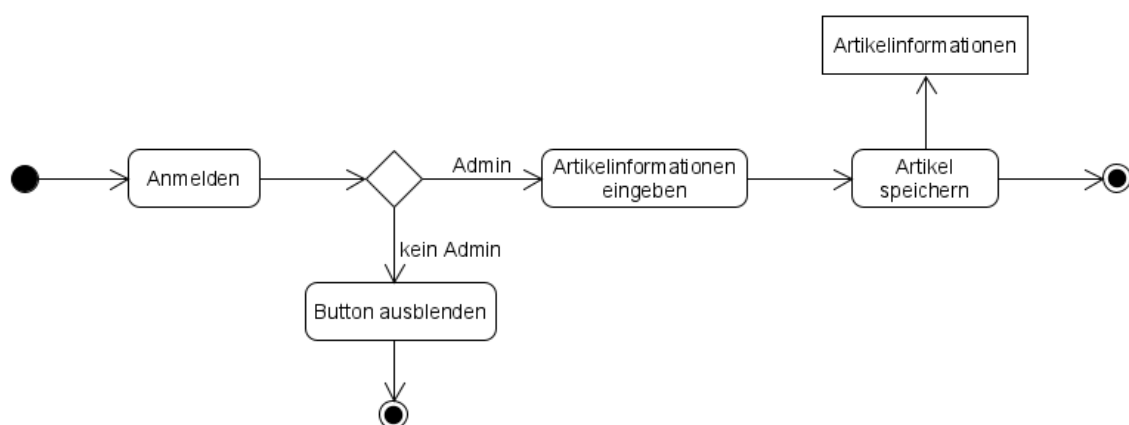
3.5.1.8 Passwort ändern



Wenn der Kunde sich anmeldet, kann er ein neues Passwort eingeben. Das Passwort wird gehasht in der Datenbank gespeichert.

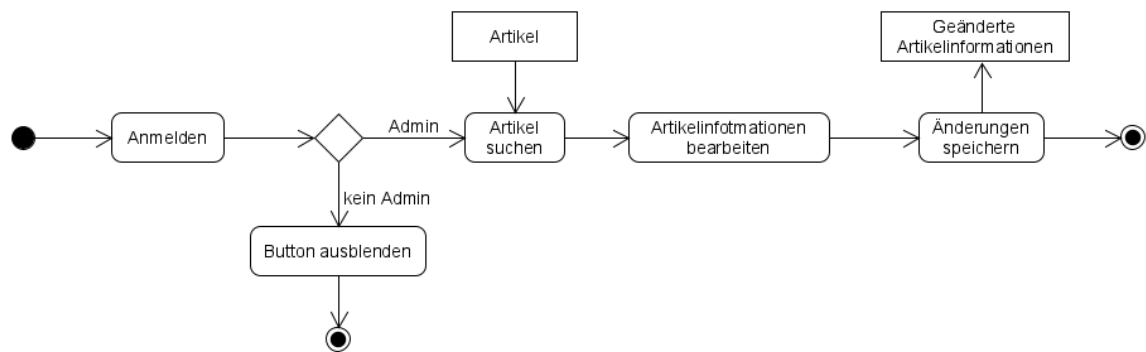
3.5.2 Catalog Service

3.5.2.1 Artikel anlegen



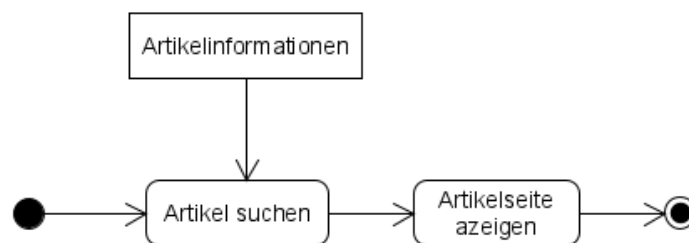
Beschreibung: Um einen Artikel anzulegen muss sich der Nutzer zuerst anmelden. Ist der Nutzer kein Admin, so wird der Button zum Artikel anlegen nicht angezeigt. Die Artikelinformationen werden anschließend eingegeben und der neue Artikel wird in der Datenbank gespeichert.

3.5.2.2 Artikel ändern



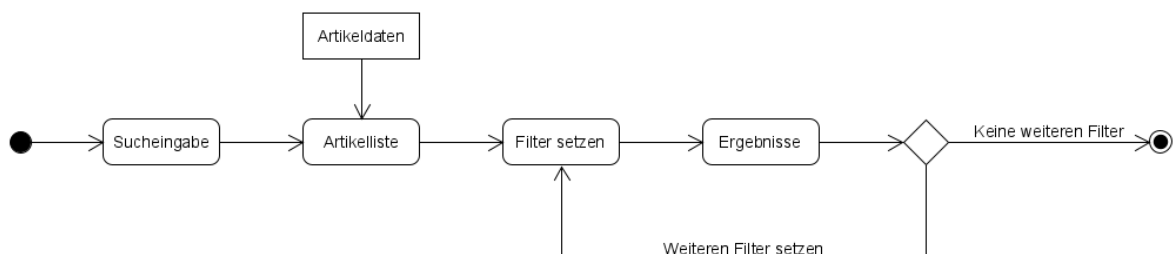
Beschreibung: Um einen Artikel zu ändern muss sich der Nutzer zuerst anmelden. Ist der Nutzer kein Admin, so wird der Button zum ändern nicht angezeigt. Anschließend wird der zu bearbeitende Artikel herausgesucht. Die Artikelinformationen werden dann aktualisiert und die geänderten Informationen werden in der Datenbank gespeichert.

3.5.2.3 Artikel aufrufen



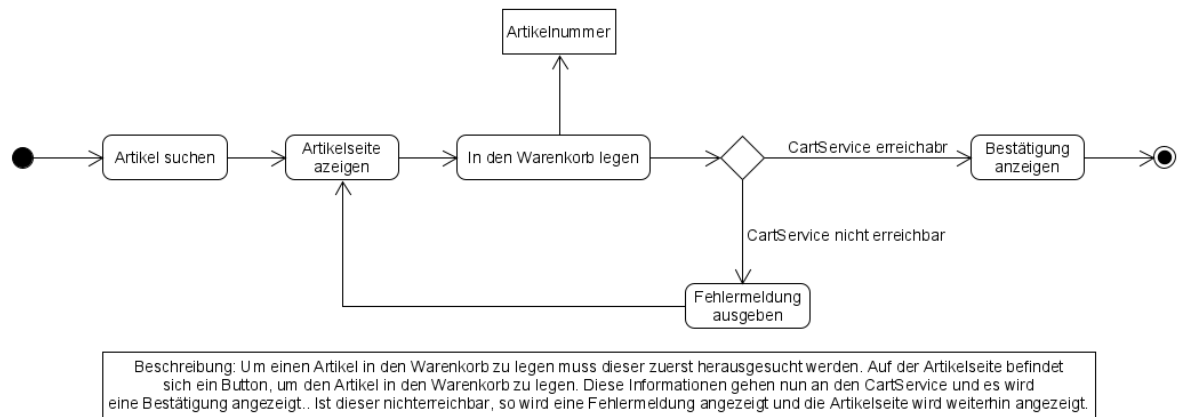
Beschreibung: Ein Nutzer kann einen Artikel über die Suchleiste suchen. Zu allen gefundenen Artikeln kann anschließend eine Seite mit allen Informationen zum Artikel angezeigt werden.

3.5.2.4 Artikel filtern

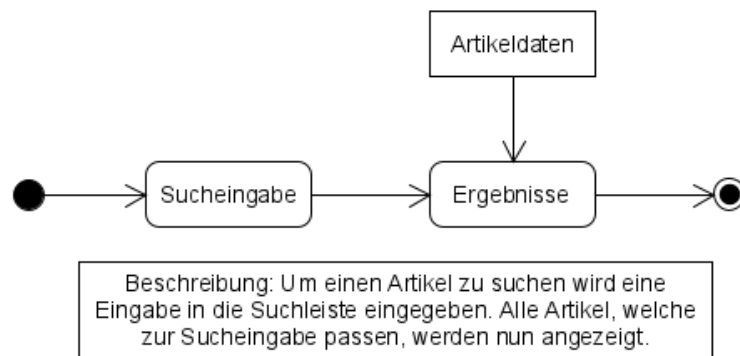


Beschreibung: Ein Nutzer kann die Suchergebnisse filtern. Dazu wird zuerst ein Suchbegriff eingegeben. Nun werden alle Artikel angezeigt die dem Suchbegriff entsprechen. Diese Artikelliste kann nun mit Hilfe von weiteren Filtern wie Marke oder Preis weiter eingeschränkt werden. Die Artikelliste aktualisiert sich mit jedem neuen gesetzten Filter.

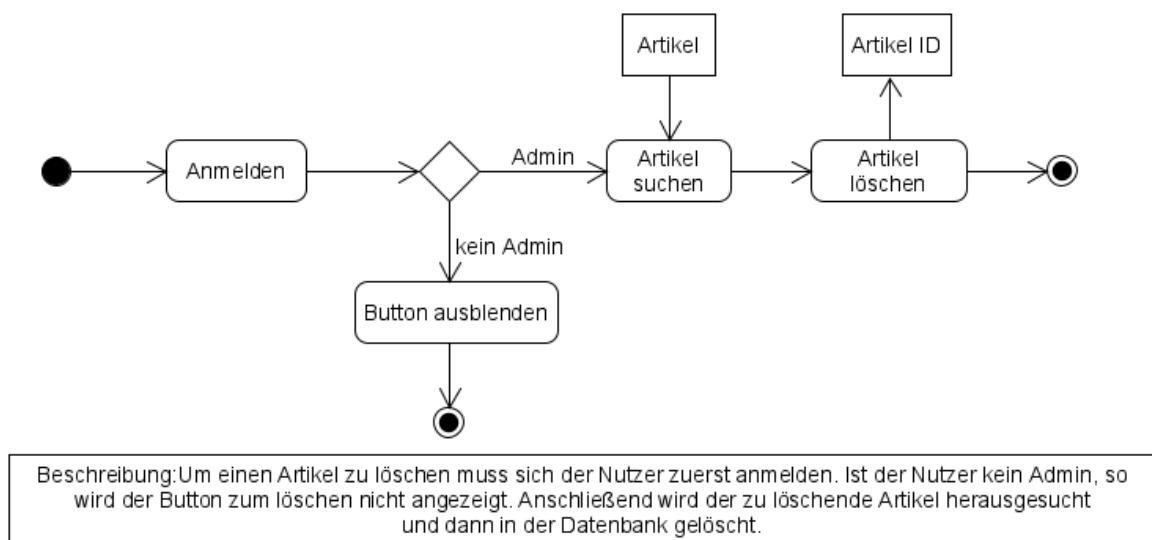
3.5.2.5 Artikel in den Warenkorb legen



3.5.2.6 Artikel suchen

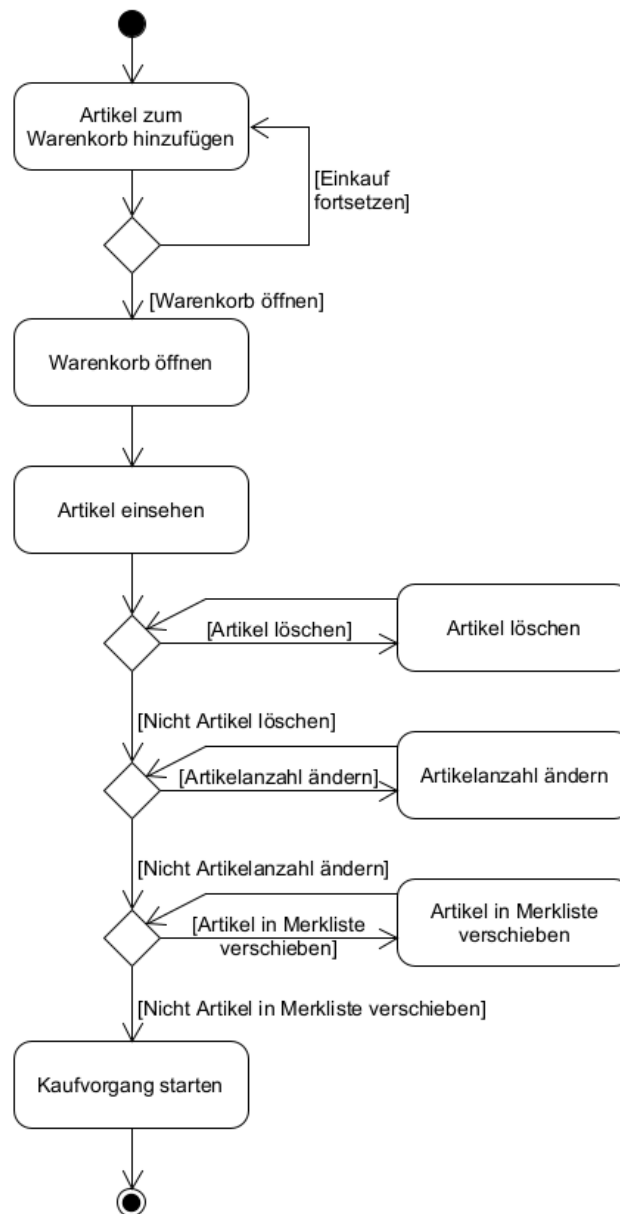


3.5.2.7 Artikel löschen



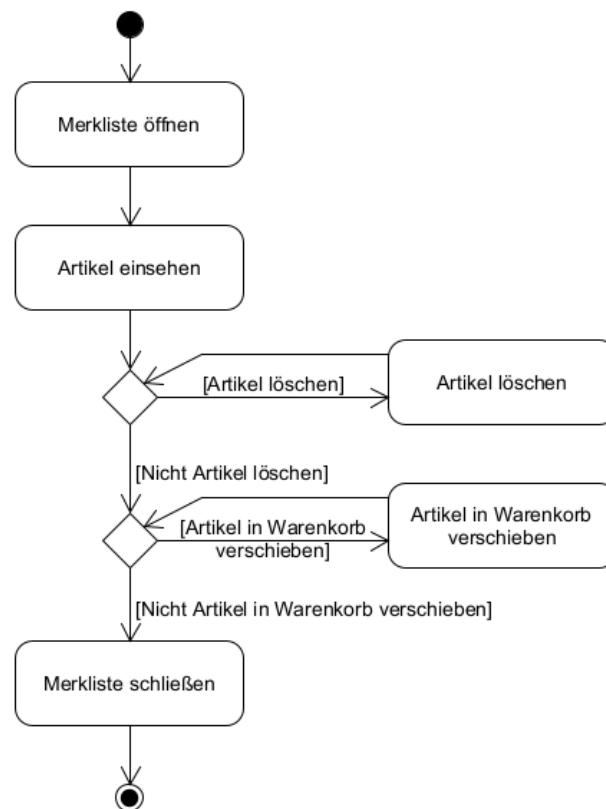
3.5.3 Cart Service

3.5.3.1 Warenkorb



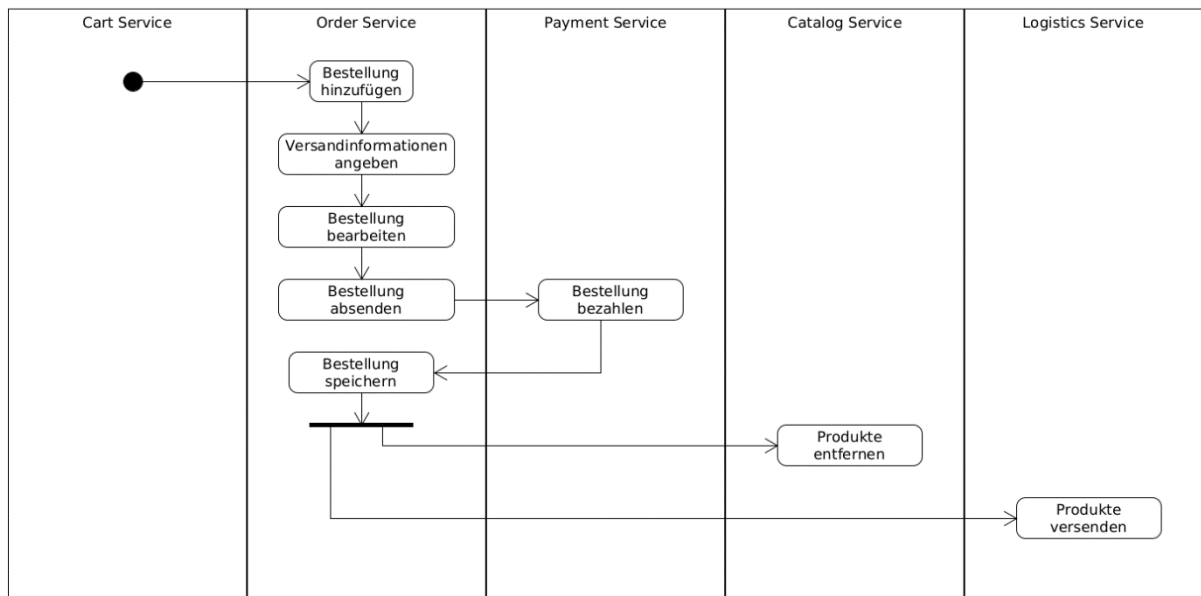
Der Kunde kann Artikel zum Warenkorb hinzufügen. Im Warenkorb können die Artikel eingesehen werden, gelöscht werden, es kann die Artikelanzahl geändert werden oder Artikel in die Merklste verschoben werden. Will der Kunde die Artikel kaufen, startet er durch Drücken eines Buttons den Kaufvorgang.

3.5.3.2 Merkliste



In der Merkliste können Artikel eingesehen, gelöscht werden und zurück in den Warenkorb verschoben werden.

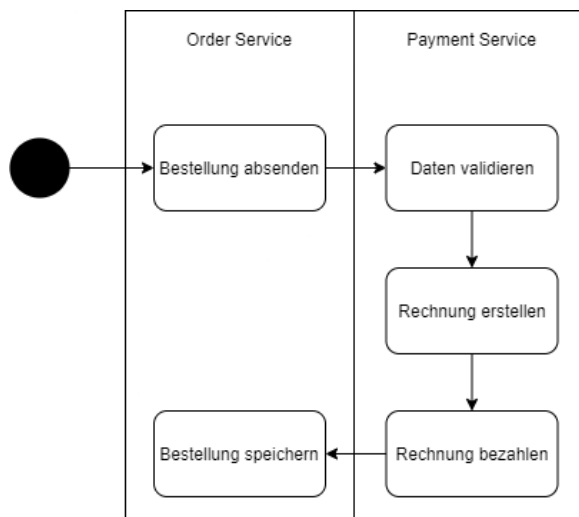
3.5.4 Order Service



Sobald ein Kunde die Artikel in seinem Warenkorb kaufen möchte, wird durch den Cart Service eine neue Bestellung erstellt. Diese beinhaltet die Artikel und den Kunden, der diese kaufen möchte. Im Order Service kann der Nutzer Versandinformationen angeben. Anschließend wird dem Nutzer eine Übersicht über die Artikel angezeigt, in welcher dieser noch einmal die Chance erhält, Artikel aus der Bestellung zu entfernen. Um den Kauf abzuschließen, muss der Kunde die Bestellung bestätigen. Nun wird dieser zum Payment Service weitergeleitet, in welchem die Bestellung bezahlt werden kann. Wenn die Zahlung erfolgreich war, wird der Order Service darüber benachrichtigt. Dieser speichert die Bestellung ab und sendet alle nötigen Informationen an den Catalog- und den Logistics Service, damit die Artikel versendet und aus dem Produktkatalog entfernt werden (Der Logistics Service existiert nicht in unserem Projekt).

3.5.5 Payment Service

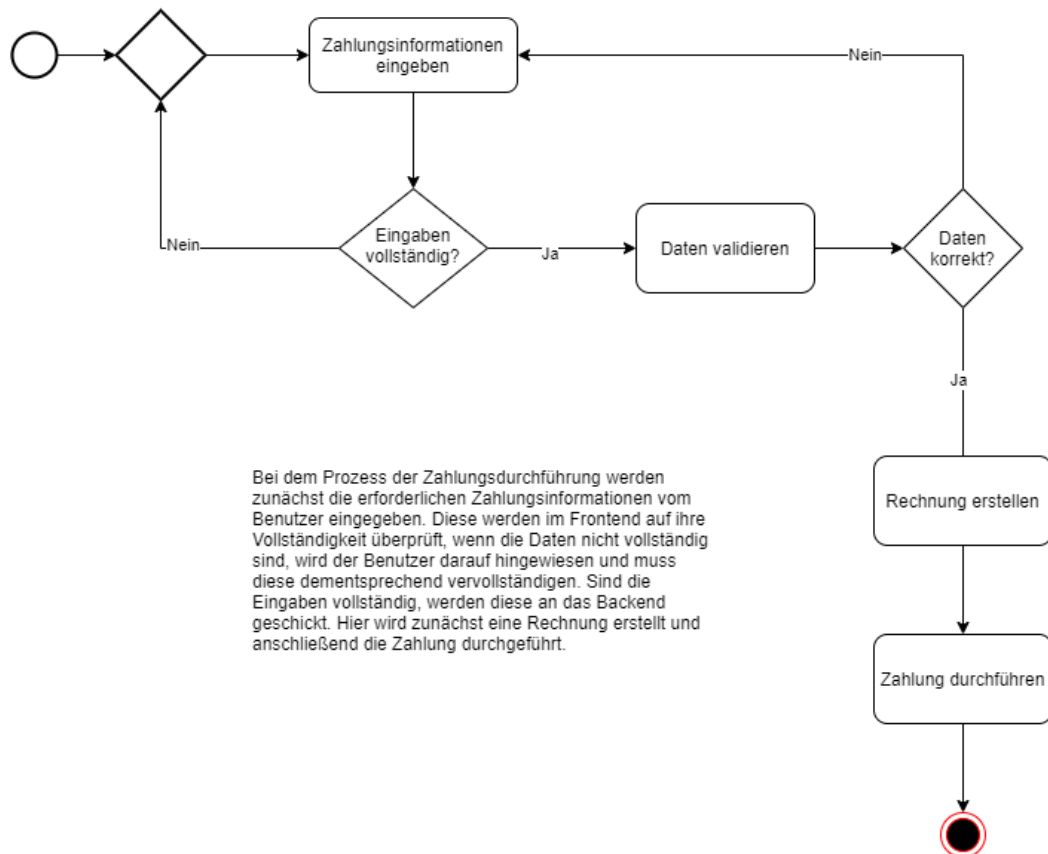
3.5.5.1 Bestellung absenden



Der Payment Service erwartet vom Order Service eine abgeschickte Bestellung, damit mit den vorhandenen Daten eine Zahlung durchgeführt werden kann. Hierzu werden zunächst die Daten validiert, anschließend eine Rechnung erstellt und schließlich diese Rechnung bezahlt. Darauf folgt die Rückmeldung an den Order Service, damit hier die Bestellung gespeichert werden kann.

3.5.5.2 Zahlung durchführen

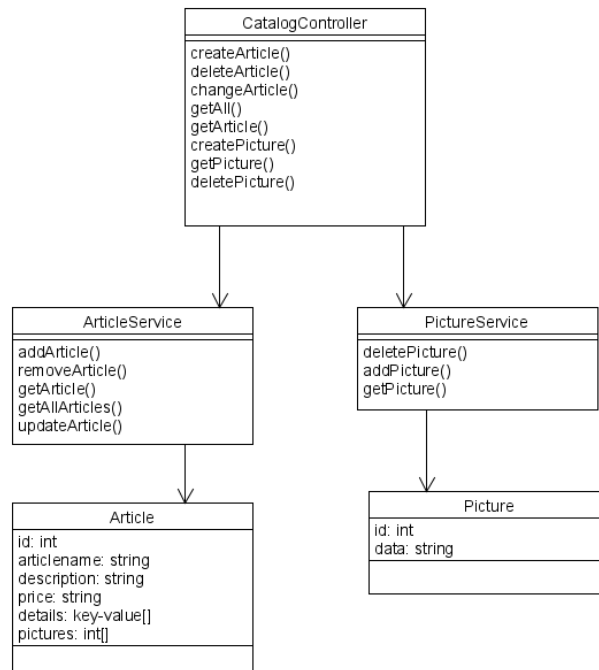
Zahlung durchführen



3.6 Entwurf

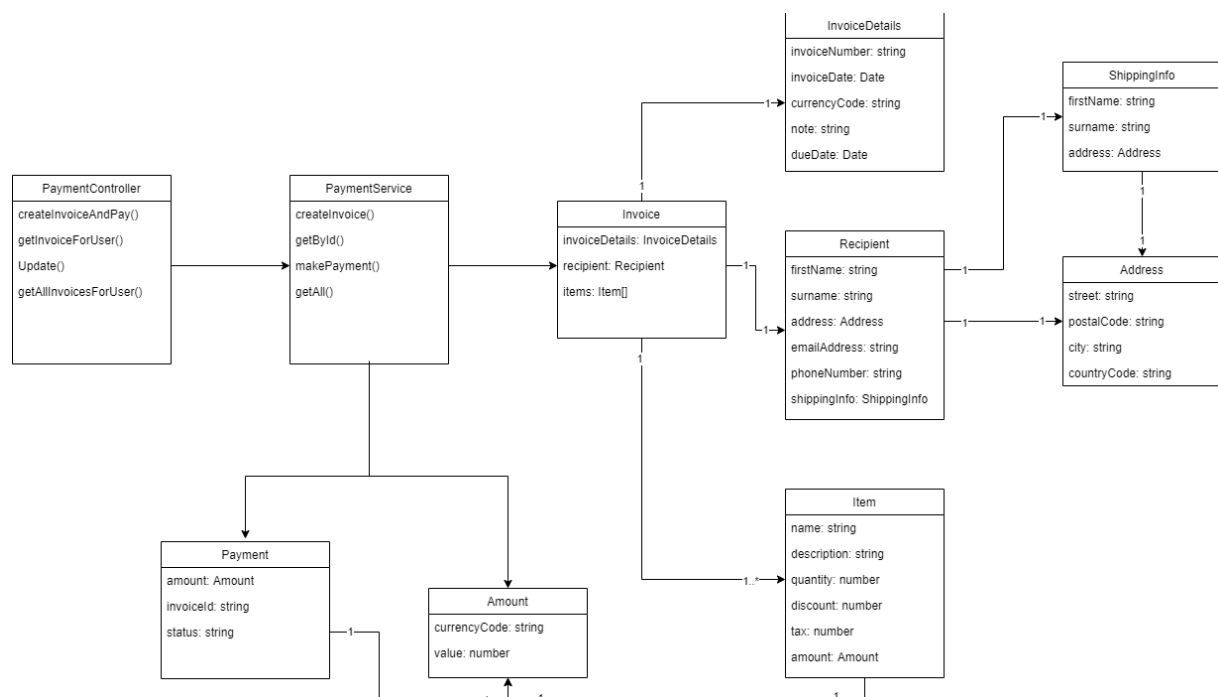
3.6.1 Catalog Service

Klassendiagramm



3.6.2 Payment Service

Klassendiagramm



3.7 Fehlerbehandlung

3.7.1 User Management Service

| ID | Beschreibung |
|--------|---|
| F_U_01 | Datenbankserver nicht erreichbar. |
| F_U_02 | Kunde nicht gefunden. |
| F_U_03 | Benutzername oder Passwort sind nicht korrekt. |
| F_U_04 | Adresse nicht gefunden. |
| F_U_05 | Emailadresse wurde abgelehnt. |
| F_U_06 | Kundenkonto mit Emailadresse ist schon vorhanden. |
| F_U_07 | Konto ist gesperrt und kann nicht verwendet werden. |
| F_U_08 | Es sind nicht alle Pflichtfelder ausgefüllt. |

3.7.2 Catalog Service

| ID | Beschreibung |
|---------|-----------------------------------|
| F_CG_01 | Datenbankserver nicht erreichbar. |
| F_CG_02 | Artikel nicht vorhanden. |
| F_CG_03 | Cart Service nicht erreichbar. |
| F_CG_04 | Account Service nicht erreichbar. |

3.7.3 Cart Service

| ID | Beschreibung |
|---------|---|
| F_CT_01 | Redis-Datenbank nicht erreichbar. |
| F_CT_02 | Artikel bereits im Warenkorb. |
| F_CT_03 | Artikel bereits in der Merkliste. |
| F_CT_04 | Maximale Artikelanzahl im Warenkorb erreicht. |
| F_CT_05 | Maximale Artikelmenge erreicht. |
| F_CT_06 | Maximale Artikelanzahl in der Merkliste erreicht. |
| F_CT_07 | Order Service nicht erreichbar. |
| F_CT_08 | Artikelinformationen fehlen. |

3.7.4 Order Service

| ID | Beschreibung |
|--------|--|
| F_O_01 | /placeOrder/ → 406: Data not acceptable |
| F_O_02 | /placeOrder/ → 500: Could not handle query |
| F_O_03 | /getOrders/{userId} → 404: User not found |
| F_O_04 | /getOrders/{userId} → 500: Could not handle query |
| F_O_05 | /getOrder/{userId}/{orderId} → 404: Order not found |
| F_O_06 | /getOrder/{userId}/{orderId} → 500: Could not handle query |

3.7.5 Payment Service

| ID | Beschreibung |
|--------|---|
| F_P_01 | Zahlung konnte nicht durchgeführt werden (500 Error). |
| F_P_02 | Eingegebene Zahlungsinformationen sind inkorrekt. |

3.8 Validierung

3.8.1 User Management Service

| ID | Use-Case | Beschreibung | Erwartetes Ergebnis |
|--------|----------------------|--|---|
| V_U_01 | Anmelden | Korrekte Benutzername und Passwort an Server senden | Weiterleitung zum internen Bereich |
| V_U_02 | | Falsches Passwort eingeben bei korrektem Benutzernamen | Fehlermeldung |
| V_U_03 | | Falschen Benutzernamen eingeben | Fehlermeldung |
| V_U_04 | Abmelden | Logout | Weiterleitung auf externen Bereich |
| V_U_05 | Kundenkonten ansehen | Übersicht aufrufen | Kundenkonten werden zurückgegeben |
| V_U_06 | Passwort ändern | Passwort ändern | Passwort wird in der Datenbank geändert |
| V_U_07 | Daten ändern | Änderung des Geburtsdatums | Kundenkonto wird mit neuem Geburtsdatum zurückgegeben |

| | | | |
|--------|-----------------|--|--|
| V_U_08 | Konto löschen | Funktion zum Konto löschen wird aufgerufen | Konto wird endgültig aus der Datenbank entfernt |
| V_U_09 | Konto erstellen | Neues Konto wird erstellt | Kundenkonto wird in der Datenbank angelegt und zurückgegeben |
| V_U_10 | Konto sperren | Kundenkonto sperren wird aufgerufen | In dem Kundenkonto wird active auf false gesetzt |

3.8.2 Catalog Service

| ID | Use-Case | Beschreibung | Erwartetes Ergebnis |
|---------|--------------------------------|---|---|
| V_CG_01 | Artikel hinzufügen | Ein Artikel wird in die Datenbank aufgenommen. | Datenbank wurde um den Artikel erweitert. |
| V_CG_02 | Artikel löschen | Artikel wurde aus der Datenbank gelöscht. | Artikel in der Datenbank gelöscht. |
| V_CG_03 | | Artikel, der bereits durch einen anderen Admin gelöscht wurde, wird gelöscht. | Fehlermeldung |
| V_CG_04 | Artikel bearbeiten | Artikelinformationen werden geändert. | Aktualisierte Artikelinformationen in der Datenbank. |
| V_CG_05 | Artikel suchen | Artikel werden nach Suchbegriff gefiltert. | Übersicht passend zum Suchbegriff wird angezeigt. |
| V_CG_06 | Suchergebnisse filtern | Angezeigte Artikel werden durch weitere Filter eingegrenzt. | Anzeigen aller Artikel entsprechend der eingestellten Filter. |
| V_CG_07 | Artikel in den Warenkorb legen | Artikel wird in den Warenkorb gelegt. | Artikel ist im Warenkorb. |
| V_CG_08 | Anmelden | Mit bestehendem Account anmelden. | Nutzer ist angemeldet. |
| V_CG_09 | Abmelden | Abmelden. | Nutzer ist abgemeldet. |
| V_CG_10 | Registrieren | Nutzer kann einen Account anlegen. | Neuer Account angelegt. |

3.8.3 Cart Service

| ID | Use-Case | Beschreibung | Erwartetes Ergebnis |
|---------|---|---|---|
| V_CT_01 | Warenkorb öffnen | Der Warenkorb eines Kunden wird geöffnet. | Die Warenkorbartikel des Kunden werden angezeigt. |
| V_CT_02 | Bezahlvorgang starten | Der Zur-Kasse-Gehen-Button wird gedrückt. | Die Versand-Maske des Order Services öffnet sich. |
| V_CT_03 | Artikel entfernen (CT_05) | Ein Artikel wird aus dem Warenkorb entfernt. | Der Artikel muss aus der Redis-Datenbank gelöscht werden und nicht mehr angezeigt werden. |
| V_CT_04 | Artikelanzahl ändern | Die Artikelanzahl wird bei einem Artikel erhöht. | Die neue Anzahl wird korrekt in der Datenbank gespeichert und visualisiert. |
| V_CT_05 | Artikel deselektieren | Es werden mehrere Artikel im Warenkorb deselektiert und anschließend der Bezahlvorgang gestartet. | Es dürfen nur markierte Artikel in den Masken des Order Services berücksichtigt und angezeigt werden. |
| V_CT_06 | Artikelanzahl einsehen | Dem Warenkorb wird ein weiterer Artikel hinzugefügt. | Die Anzahl beim Warenkorbicon muss sich um eins erhöhen. |
| V_CT_07 | Artikel zur Merkliste hinzufügen | Ein Warenkorbartikel wird in die Merkliste verschoben. | Der Artikel muss aus dem Warenkorb entfernt werden und in die Merkliste verschoben werden. |
| V_CT_08 | Artikel einsehen (CT_11) | Die Merkliste eines Kunden wird geöffnet. | Die Merklistenartikel des Kunden werden angezeigt. |
| V_CT_09 | Artikel entfernen (CT_12) | Ein Artikel wird aus der Merkliste gelöscht. | Der Artikel muss aus der Redis-Datenbank gelöscht werden und nicht mehr angezeigt werden. |
| V_CT_10 | Artikel zurück in den Warenkorb verschieben | Ein Artikel wird von der Merkliste in den Warenkorb verschoben. | Der Artikel muss aus der Merkliste entfernt werden und in den Warenkorb verschoben werden. |

3.8.4 Order Service

| ID | Use-Case | Beschreibung | Erwartetes Ergebnis |
|--------|-----------------------------|--|---|
| V_O_01 | Bestellung absenden | Bestellung wird in der Datenbank gespeichert | Alle erforderlichen Felder sind vorhanden |
| V_O_02 | Adressinformationen angeben | Adresse wird ausgefüllt | Alle Textfelder wurden ausgefüllt |
| V_O_03 | Versandart wählen | Versandart wird ausgewählt | Es wurde eine Versandart ausgewählt |
| V_O_04 | Bestellung absenden | Nutzer klickt auf Bestellung absenden | Die Bestellung enthält mindestens ein Produkt |

| | | | |
|--------|-----------------------|-------------------------------|--|
| V_O_05 | Bestellung platzieren | Neue Bestellung wird erstellt | Es wurden alle für den Bestellvorgang erforderlichen Daten korrekt übergeben |
|--------|-----------------------|-------------------------------|--|

3.8.5 Payment Service

| ID | Use-Case | Beschreibung | Erwartetes Ergebnis |
|--------|--------------------------------|--|---|
| V_P_01 | Zahlungsinformationen eingeben | Der Nutzer gibt die Zahlungsinformationen an | Es wurden alle erforderlichen Zahlungsinformationen angegeben |
| V_P_02 | | Zahlungsinformationen wurden angegeben | Die angegebenen Daten können weiterverarbeitet werden |
| V_P_03 | Bezahlvorgang abschließen | Testzahlung wird durchgeführt | Zahlung war erfolgreich |

4 Projektorganisation

4.1 Verantwortlichkeiten

4.1.1 Zuordnung der Personen zu den Microservices und Technologien

| Microservice | Verantwortung | Technologien |
|-------------------------|------------------|--|
| User Management Service | David Nickel | Frontend: Vue.js (JavaScript) Backend: Keycloak Datenbank: H2 |
| Catalog Service | Marc Schwettmann | Frontend: Vue.js (JavaScript) Backend: Node.js (JavaScript) Datenbank: MongoDB |
| Cart Service | Lennart Dümke | Frontend: Vue.js (JavaScript) Backend: Django (Python) Datenbank: Redis |
| Order Service | Yannic Döll | Frontend: Vue.js (JavaScript) Backend: Django (Python) Datenbank: MongoDB |
| Payment Service | Michael Nickel | Frontend: Vue.js (JavaScript) Backend: ASP.NET (C#) Datenbank: MongoDB |

4.1.2 Rollen

4.1.2.1 Scrum-Master (Projektleiter)

Der Scrum-Master sorgt für die organisatorische Prozess-Qualität. Er leitet die Teambildung und Auswahl der Kommunikationsplattformen ein. Außerdem stellt er die Qualität der Abgaben sicher. Der Scrum-Master ermittelt Termine und deren Einhaltung (Zuständig für das Kanban-Board).

4.1.2.2 DevOps-Engineer

Der DevOps-Engineer sorgt für die Infrastruktur-Qualität. Er erstellt GitHub-Repos, CI/CD-Pipelines, Schablonen für Dockerfiles etc. Außerdem ist er Ansprechpartner für Docker, Kubernetes, die Cloud und Deployments.

4.1.2.3 Cloud-Software-Engineer

Der Cloud-Software-Engineer sorgt für die technische Qualität. Er implementiert einen Microservice mit Frontend, Backend und DB. Außerdem entwickelt er die Software-Architektur, Spezifikation, Implementation und Testfälle.

4.1.3 Rollenzuordnung

| Name | Rolle |
|------------------|---|
| Lennart Dümke | Cloud-Software-Engineer + Scrum-Master |
| David Nickel | Cloud-Software-Engineer + DevOps-Engineer |
| Marc Schwettmann | Cloud-Software-Engineer |
| Yannic Döll | Cloud-Software-Engineer |
| Michael Nickel | Cloud-Software-Engineer |

4.2 Grober Projektplan

Meilensteine

| KW | Datum | Meilenstein |
|----|-------|---|
| 23 | 07.06 | Abgabe Spezifikation |
| 24 | 17.06 | 1. Meilenstein (Microservice hat Test-GUI + Kann in der Cloud deployt werden) |
| 25 | 24.06 | 2. Meilenstein (Logik des Microservices implementiert + Kommuniziert mit seinem Datenspeicher) |
| 26 | 01.07 | 3. Meilenstein (Kommunikation der Microservices untereinander + Tests) |
| 26 | 04.07 | Eigene Deadline |
| 27 | | Pufferzone |
| 28 | 13.07 | Abgabe Softwareprojekt + Abschlusspräsentation |

5 Anhänge

5.1 Schnittstellen

5.1.1 User Management Service

Für das User Management wird Keycloak verwendet. Die REST-API Dokumentation ist zu finden unter: <https://www.keycloak.org/docs-api/14.0/rest-api/index.html>.

5.1.2 Catalog Service

```
openapi: 3.0.0
info:
  version: '1'
  title: Catalog Service Api
  description: ''

paths:
  /createArticle/:
    post:
      tags:
        - CatalogService
      requestBody:
        $ref: '#/components/requestBodies/Article'
      responses:
        200:
          $ref: '#/components/responses/OK'
        400:
          $ref: '#/components/responses/invalid_data'
        500:
          $ref: '#/components/responses/server_error'

  /deleteArticle/{id}/:
    parameters:
      - $ref: '#/components/parameters/Id'
    delete:
      tags:
        - CatalogService
      responses:
        200:
          $ref: '#/components/responses/OK'
        404:
          $ref: '#/components/responses/not_found'
        500:
          $ref: '#/components/responses/server_error'

  /changeArticle/{id}/:
    parameters:
      - $ref: '#/components/parameters/Id'
    put:
      tags:
        - CatalogService
      responses:
        200:
          $ref: '#/components/responses/OK'
        404:
          $ref: '#/components/responses/not_found'
```

```

        500:
          $ref: '#/components/responses/server_error'
/getAll/:
  get:
    tags:
      - CatalogService
    responses:
      200:
        $ref: '#/components/responses/OK'
      404:
        $ref: '#/components/responses/no_data'
      500:
        $ref: '#/components/responses/server_error'
/getArticle/{id}/:
  parameters:
    - $ref: '#/components/parameters/Id'
  get:
    tags:
      - CatalogService
    responses:
      200:
        $ref: '#/components/responses/OK'
      404:
        $ref: '#/components/responses/not_found'
      500:
        $ref: '#/components/responses/server_error'
/createPicture/:
  post:
    tags:
      - CatalogService
    requestBody:
      $ref: '#/components/requestBodies/Picture'
    responses:
      200:
        $ref: '#/components/responses/OK'
      500:
        $ref: '#/components/responses/server_error'

/getPicture/{id}/:
  parameters:
    - $ref: '#/components/parameters/Id'
  get:
    tags:
      - CatalogService
    responses:
      200:
        $ref: '#/components/responses/OK'
      404:
        $ref: '#/components/responses/not_found'
      500:
        $ref: '#/components/responses/server_error'
/deletePicture/{id}/:
  parameters:
    - $ref: '#/components/parameters/Id'
  delete:
    tags:
      - CatalogService
    responses:
      200:
        $ref: '#/components/responses/OK'
      404:
        $ref: '#/components/responses/not_found'

```

```

        500:
          $ref: '#/components/responses/server_error'
/search/{searchterm}/:
  parameters:
    - $ref: '#/components/parameters/Searchterm'
  post:
    tags:
      - CatalogService
    requestBody:
      $ref: '#/components/requestBodies/Search'
    responses:
      200:
        $ref: '#/components/responses/OK'
      404:
        $ref: '#/components/responses/no_data'
      500:
        $ref: '#/components/responses/server_error'
/getManufacturer/:
  get:
    tags:
      - CatalogService
    responses:
      200:
        $ref: '#/components/responses/OK'
      404:
        $ref: '#/components/responses/not_found'
      500:
        $ref: '#/components/responses/server_error'
/getCategory/:
  get:
    tags:
      - CatalogService
    responses:
      200:
        $ref: '#/components/responses/OK'
      404:
        $ref: '#/components/responses/not_found'
      500:
        $ref: '#/components/responses/server_error'

components:
  parameters:
    Id:
      name: id
      in: path
      required: true
      schema:
        type: string
    Searchterm:
      name: searchterm
      in: path
      required: true
      schema:
        type: string

  requestBodies:
    Article:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Article'
    Picture:

```

```

    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Picture'
  Search:
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Search'

responses:
  OK:
    description: Ok
  not_found:
    description: Item not found
  no_data:
    description: No Articles found
  invalid_data:
    description: Invalid Data
  server_error:
    description: Server Error

schemas:
  Article:
    type: object
    properties:
      articlename:
        type: string
        example: microphone
      description:
        type: string
        example: my microphone
      price:
        type: string
        example: 10,99€
      hersteller:
        type: string
        example: Test_Hersteller
      kategorie:
        type: string
        example: Test_Kategorie
      details:
        type: array
        items:
          $ref: '#/components/schemas/key-value'
      pictures:
        type: array
        items:
          $ref: '#/components/schemas/pictures'
  Picture:
    type: object
    properties:
      data:
        type: string
        format: base64
        example: JJ7/D9bCfmdF1qkYNhtk/P5uvZ0N2zAUsiScDJA==XXuR

  key-value:
    type: object
    properties:
      key:
        type: string

```

```

    example: key
  value:
    type: string
    example: value
pictures:
  type: integer
  example: 123
Search:
  type: object
  properties:
    hersteller:
      type: array
      items:
        $ref: '#/components/schemas/hersteller'
    kategorie:
      type: array
      items:
        $ref: '#/components/schemas/kategorie'
    preisMin:
      type: integer
      example: 111
    preisMax:
      type: integer
      example: 999
  hersteller:
    type: string
    example: Test_Hersteller
  kategorie:
    type: string
    example: Test_Kategorie

```

5.1.3 Cart Service

```

openapi: 3.0.1
info:
  title: SGSE - Cart Service
  description: Cart API Description
  version: v1
paths:
  /cart/getArticleCount/{userID}:
    get:
      summary: 'Returns the shopping cart article count for a given customer'
      parameters:
        - name: "userID"
          in: "path"
          description: "ID of the user"
          required: true
          schema:
            type: string
      responses:
        '200':
          description: 'Article count'
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/articlecountcart'
        '500':
          description: 'Could not handle query'
  /cart/getArticles/{userID}:

```

```

get:
  summary: 'Returns the shopping cart articles for a given customer'
  parameters:
    - name: "userID"
      in: "path"
      description: "ID of the user"
      required: true
      schema:
        type: string
  responses:
    '200':
      description: 'Cart articles'
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/articlescart'
    '500':
      description: Could not handle query
/cart/addArticle/{userID}:
post:
  summary: 'Adds an article to the cart'
  parameters:
    - name: "userID"
      in: "path"
      description: "ID of the user"
      required: true
      schema:
        type: string
  requestBody:
    $ref: '#/components/requestBodies/article'
  responses:
    '200':
      description: 'OK'
/cart/deleteArticle/{userID}/{articleID}:
delete:
  summary: 'Deletes a cart article'
  parameters:
    - name: "userID"
      in: "path"
      description: "ID of the user"
      required: true
      schema:
        type: string
    - name: "articleID"
      in: "path"
      description: "ID of the article"
      required: true
      schema:
        type: string
  responses:
    '200':
      description: 'OK'
    '500':
      description: 'User not found'
/cart/deleteArticles/{userID}:
delete:
  summary: 'Deletes all cart articles'
  parameters:
    - name: "userID"
      in: "path"
      description: "ID of the user"
      required: true

```



```

    schema:
      type: string
  responses:
    '200':
      description: 'OK'
    '500':
      description: 'User not found'
/cart/deletePassedArticles/{userID}:
  delete:
    summary: 'Deletes the passed cart articles'
    parameters:
      - name: "userID"
        in: "path"
        description: "ID of the user"
        required: true
        schema:
          type: string
    requestBody:
      $ref: '#/components/requestBodies/deletearticle'
    responses:
      '200':
        description: 'OK'
      '500':
        description: 'User not found'
/cart/updateArticleQuantity:
  put:
    summary: 'Changes the quantity of an article in the cart'
    requestBody:
      $ref: '#/components/requestBodies/articlequantity'
    responses:
      '200':
        description: 'OK'
/list/getArticles/{userID}:
  get:
    summary: 'Returns the wish list articles for a given customer'
    parameters:
      - name: "userID"
        in: "path"
        description: "ID of the user"
        required: true
        schema:
          type: string
    responses:
      '200':
        description: 'List articles'
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/articlescart'
      '500':
        description: 'Could not handle query'
/list/addArticle/{userID}:
  post:
    summary: 'Adds an article to the wish list'
    parameters:
      - name: "userID"
        in: "path"
        description: "ID of the user"
        required: true
        schema:
          type: string
    requestBody:

```

```

    $ref: '#/components/requestBodies/article'
  responses:
    '200':
      description: 'OK'
/list/deleteArticle/{userID}/{articleID}:
  delete:
    summary: 'Deletes a wish list article'
    parameters:
      - name: "userID"
        in: "path"
        description: "ID of the user"
        required: true
        schema:
          type: string
      - name: "articleID"
        in: "path"
        description: "ID of the article"
        required: true
        schema:
          type: string
    responses:
      '200':
        description: 'OK'
      '500':
        description: 'User not found'
components:
  schemas:
    deletearticles:
      type: array
      items:
        type: string
    articleuser:
      type: object
      properties:
        article_name:
          type: string
        article_vendor:
          type: string
        article_price:
          type: number
        article_catalog_id:
          type: string
        article_image:
          type: string
    article:
      type: object
      properties:
        article_id:
          type: string
        article_name:
          type: string
        article_vendor:
          type: string
        article_price:
          type: number
        article_catalog_id:
          type: string
        article_image:
          type: string
        article_count:
          type: number
    articlequantity:

```

```

    type: object
    properties:
      article_id:
        type: string
      new_quantity:
        type: number
    articlecountcart:
      type: object
      properties:
        message:
          type: string
        count:
          type: number
    articlescart:
      type: array
      items:
        oneOf:
          - $ref: '#/components/schemas/article'
    requestBodies:
      deletearticle:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/deletearticles'
      article:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/articleuser'
      articlequantity:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/articlequantity'

```

5.1.4 Order Service

```

openapi: 3.0.1
info:
  title: SGSE - Order Service
  description: Order API Description
  version: v1
paths:
  /placeOrder/:
    post:
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/NewOrder'

```

```

    summary: 'Place new order'
    responses:
      '200':
        description: OK
      '406':
        description: Data not acceptable
  /getOrders/{userId}:
    get:
      parameters:
        - in: path
          name: userId
          schema:
            type: string
          required: true
      summary: 'Get all orders of a user'
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Orders'
        '404':
          description: User not found
  /deleteOrder/{orderId}:
    delete:
      parameters:
        - in: path
          name: orderId
          schema:
            type: string
          required: true
      summary: 'Delete an order by id'
      responses:
        '200':
          description: OK
          content:

```

```

        application/json:
            schema:
                $ref: '#/components/schemas/Order'
components:
    schemas:
        Orders:
            type: object
            properties:
                orders:
                    type: array
                    items:
                        $ref: '#/components/schemas/User'
        NewOrder:
            type: object
            properties:
                date:
                    type: string
                    format: date
                user:
                    $ref: '#/components/schemas/User'
                products:
                    type: array
                    items:
                        $ref: '#/components/schemas/Product'
                address:
                    $ref: '#/components/schemas/Address'
                shippingMethod:
                    $ref: '#/components/schemas/ShippingMethod'
        Order:
            type: object
            properties:
                id:
                    type: string
                date:
                    type: string
                    format: date
                user:

```

```

    $ref: '#/components/schemas/User'
  products:
    type: array
    items:
      $ref: '#/components/schemas/Product'
  address:
    $ref: '#/components/schemas/Address'
  shippingMethod:
    $ref: '#/components/schemas/ShippingMethod'
User:
  type: object
  properties:
    id:
      type: string
Product:
  type: object
  properties:
    article_price:
      type: number
    article_name:
      type: string
    article_count:
      type: number
    article_catalog_id:
      type: string
    article_vendor:
      type: string
    article_image:
      type: string
    article_id:
      type: string
    checkbox_value:
      type: boolean
Address:
  type: object
  properties:
    firstName:

```

```

    type: string
  lastName:
    type: string
  street:
    type: string
  number:
    type: string
  postCode:
    type: number
  city:
    type: number
ShippingMethod:
  type: object
  properties:
    name:
      type: string
    description:
      type: string
    price:
      type: string

```

5.1.5 Payment Service

```

openapi: 3.0.1
info:
  title: SGSE - Payment Service
  description: Payment API Description
  version: v1
paths:
  /createInvoiceAndPay/:
    post:
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Invoice'

      summary: ''
      description: ''
      operationId: ''
      responses:
        '200':
          description: OK
        '500':
          description: Internal Server Error
        '400':
          description: Wrong information
  /getInvoiceForUser/{invoiceId}/{userId}:

```

```

get:
  parameters:
    - in: path
      name: invoiceId
      schema:
        type: string
      required: true
    - in: path
      name: userId
      schema:
        type: string
      required: true
  summary: ''
  description: ''
  operationId: ''
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/InvoiceDetails'
    '500':
      description: Internal Server Error
    '400':
      description: Wrong information
/getAllInvoicesForUser/{userId}:
get:
  parameters:
    - in: path
      name: userId
      schema:
        type: string
      required: true
  summary: ''
  description: ''
  operationId: ''
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/InvoiceDetails'
    '500':
      description: Internal Server Error
    '400':
      description: Wrong information
/showDetailsForPayment/{paymentId}:
get:
  parameters:
    - in: path
      name: paymentId
      schema:
        type: string
      required: true
  summary: ''
  description: ''
  operationId: ''
  responses:

```



```

    '200':
      description: OK
      content:
        application/json:
          schema:
            type: array
            items:
              $ref: '#/components/schemas/Payment'
    '500':
      description: Internal Server Error
    '400':
      description: Wrong information
components:
  schemas:
    Invoice:
      type: object
      properties:
        invoiceDetails:
          $ref: '#/components/schemas/InvoiceDetails'
        recipient:
          $ref: '#/components/schemas/Recipient'
        items:
          type: array
          items:
            $ref: '#/components/schemas/Item'
    InvoiceDetails:
      type: object
      properties:
        invoiceNumber:
          type: string
        invoiceDate:
          type: string
          format: date
        currencyCode:
          type: string
        note:
          type: string
        dueDate:
          type: string
          format: date
    Recipient:
      type: object
      properties:
        firstName:
          type: string
        surname:
          type: string
        address:
          $ref: '#/components/schemas/Address'
        emailAddress:
          type: string
        phoneNumber:
          type: string
        shippingInfo:
          $ref: '#/components/schemas/ShippingInfo'
    Item:
      type: object
      properties:
        name:
          type: string
        description:
          type: string

```

```

    quantity:
      type: number
    discount:
      type: number
    tax:
      type: number
    amount:
      $ref: '#/components/schemas/Amount'
Address:
  type: object
  properties:
    street:
      type: string
    postalCode:
      type: string
    city:
      type: string
    countryCode:
      type: string

ShippingInfo:
  type: object
  properties:
    firstName:
      type: string
    surname:
      type: string
    address:
      $ref: '#/components/schemas/Address'
Amount:
  type: object
  properties:
    currencyCode:
      type: string
    value:
      type: number
Payment:
  type: object
  properties:
    amount:
      $ref: '#/components/schemas/Amount'
    invoiceId:
      type: string
    status:
      type: string

```