JEE Webshop Dokumentation

Luca Malisan

2025

Einführung und Ziele

Aufgabenstellung

Bei dem zu entwickelnden System handelt es sich um einen Webshop. Dieser besteht aus den folgenden Komponenten:

- K: Digitaler Katalog aller Produkte
- D: Detailansicht jedes produkts
- W: Warenkorb
- L: Login
- R: Rest-API zur Anbindung externer Systeme

Im Produktkatalog müssen Einträge per Volltextsuche gefunden werden können (K1). Zudem muss die Menge der Resultate nach Haupt- und Subkategorie eingeschränkt werden können (K2). Jeder Eintrag in der Liste muss Artikelbild, Titel, Preis, Rabatt und Verfügbarkeit anzeigen (K3). Pro Seite muss die Anzahl Einträge limitiert werden, Navigation zwischen den Seiten muss möglich sein (K4). Durch Klick auf einen Eintrag muss die Detailseite geöffnet werden. (K5)

Auf der Detailansicht müssen zusätzlich zum Listeneintrag alle anderen Bilder und die Beschreibung sowie die verfügbare Anzahl angezeigt werden (D1). Zudem muss ein Button angezeigt werden, um den Artikel zum Warenkorb zu hinzufügen. Die Anzahl muss hierbei einstellbar sein (D2).

Der Warenkorb muss einen Überblick über alle hinzugefügten Artikel bieten (W1). Die Anzahl jedes Artikels muss angepasst werden können bzw. Artikel müssen entfernt werden können (W2). Der Gesamtpreis mit und ohne Mwst. sowie Rabatte müssen ebenfalls angezeigt werden (W3).

Benutzer müssen über einen Login-Button auf die Login-Page gelangen (L1). Unautorisierte Benutzer können alle Artikel ansehen, um etwas zum Warenkorb hinzuzufügen wird aber ein Login verlangt (L2). Das Login muss dabei in einem separaten Login-Formular stattfinden (L3). Die eingegebene E-Mail muss durch eine Bestätigungsmail validiert werden (L4).

Für das Hinzufügen, Updaten und Löschen von Artikeln und Kategorien muss eine REST-API zur Verfügung gestellt werden (R1). Diese darf nur von authentifizierten Maschinen angesteuert werden (R2).

Qualitätsziele

Für eine gute Usability muss der Webshop in Tests mit Testkunden mindestens 90% Zufriedenheit erreichen. Um eine optimale Performance zu garantieren, darf jede Operation im Webshop (Website laden, Artikel zum Warenkorb hinzufügen, ...) maximal 500ms dauern. Zudem müssen alle Methoden, die Business-Logik enthalten, mit mindestens einem Unit-Test abgesichert werden.

Unautorisierte Zugriffe müssen verhindert werden. Sowohl der Code als auch die verwendeten Dependencies dürfen keine Sicherheitslücken aufweisen. Für die Verständlichkeit des Codes müssen sämtliche Methoden und REST-Schnittstellen dokumentiert sein. Bei den redundanten Daten im Webshop, die aus angebundenen Systemen stammen, muss der Ursprung eindeutig nachvollziehbar sein.

Vision

Mit dem Webshop soll eine Lösung vermarktet werden, die eigenständig und technologieunabhängig betrieben werden kann. Durch die einfache Wartbarkeit und hohe Anpassbarkeit können Kundenprojekte vergleichsweise günstig umgesetzt werden. Durch die Verwendung einer eigenen Datenbank lässt es sich zudem nahtlos in bestehende Umgebungen inregrieren. Dadurch bildet das Produkt eine sichere, universelle Standardlösung für Unternehmen aller Art und Grösse.

Kontextabgrenzung

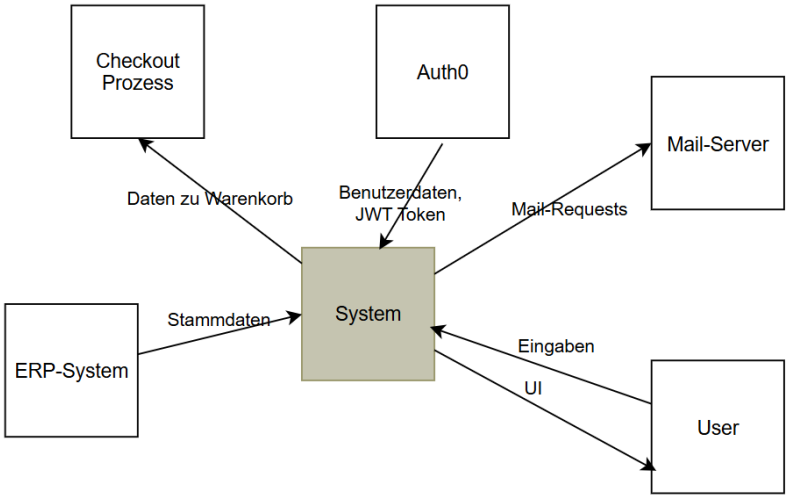

Figure 1: Technischer Kontext

Lösungsstrategie

Inhalt

Für die Entwicklung des Systems wurden die folgenden zentralen Entscheidungen getroffen:

- Da das Projekt aus Sicht der Funktionalität relativ schlank ist, soll die Projektgrösse auf das nötige Minimum reduziert werden. So kann eine niedrige Komplexität und eine hohe Performance erreicht werden.

- Um die korrekte Funktionalität der Applikation sicherzustellen, muss die Business-Logik nach jeder Änderung getestet werden.
- Um Sicherheitslücken zu verhindern und den Datenschutz einzuhalten, darf es zu keinen unautorisierten Zugriffen auf Daten kommen. Benutzer dürfen nur auf öffentliche und ihre eigenen Daten zugreifen, Änderungen an den Stammdaten nur durch berechtigte Applikationen erfolgen.
- Login- und Checkout-Prozess des Webshops sind hinsichtlich Datenschutz und Sicherheit am kritischsten, weshalb diese an eine externe Software delegiert wird. Es gibt hierfür bereits umfangreiche, standardisierte Lösungen.

Bausteinsicht

Das System besteht aus den folgenden physischen und logischen Komponenten:

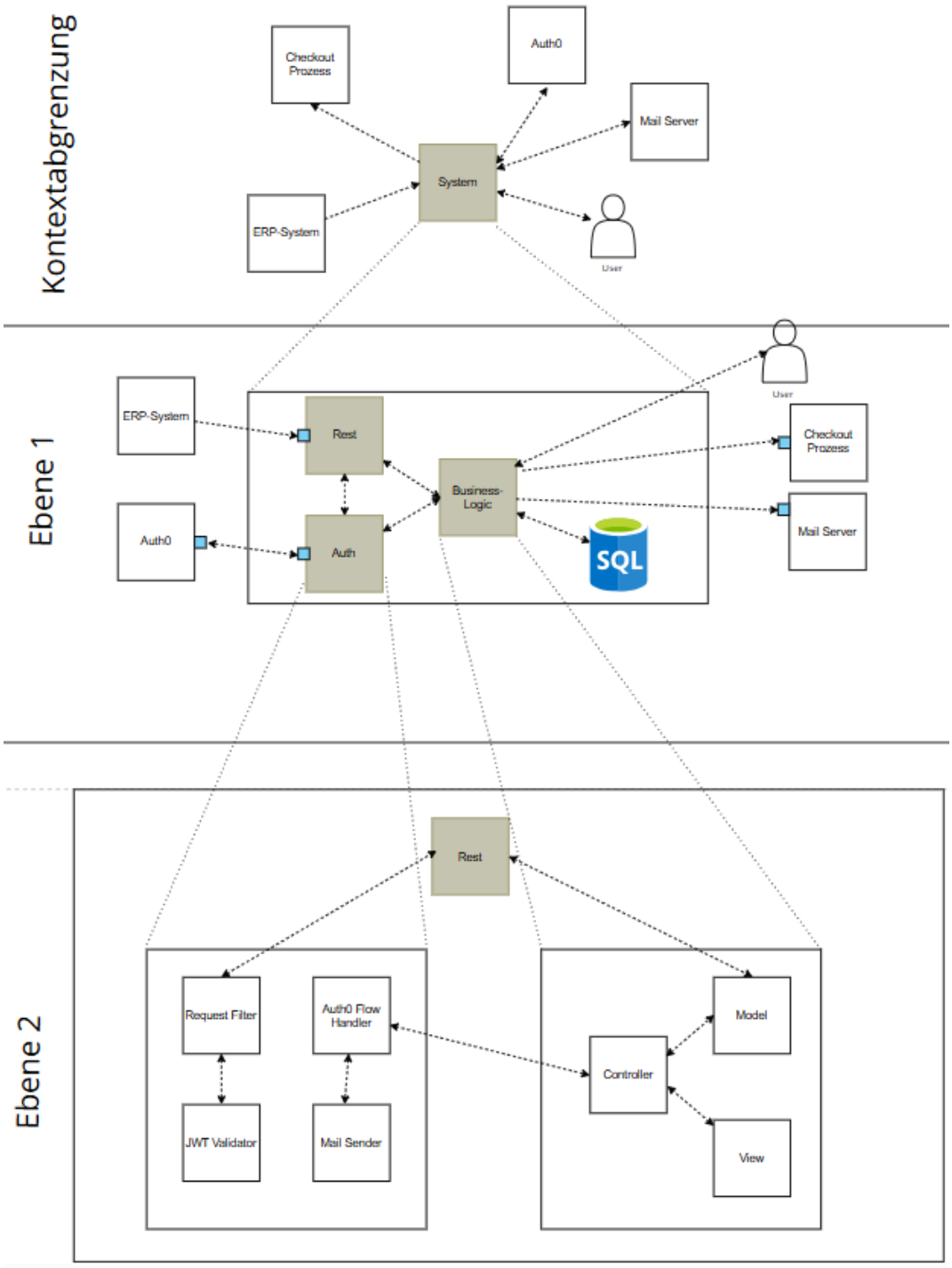

Figure 2: Systemaufbau

Kontextabgrenzung

- ERP-System: Im ERP-System werden die Artikeldaten und Kategorien verwaltet. Die Daten werden regelmässig mit dem Webshop synchronisiert.

- Checkout Prozess: Der Kunde wird vom Webshop aus auf das Checkout-System weitergeleitet
- Auth0: Der Kunde wird vom Webshop auf den Login-Screen von Auth0 weitergeleitet. Nach erfolgreicher Eingabe seiner Daten gelangt er wieder zurück und die Daten werden verarbeitet
- Der Benutzer interagiert mit dem GUI des Webshops

Ebene 1

- Rest-Komponente: REST-Schnittstelle zur Synchronisation der Datenbasis mit angebundenen Systemen
- Auth-Komponente: Sicherstellung autorisierter Zugriffe
- Business-Logic-Komponente: Setzt die eigentliche Funktion des Webshops um

Ebene 2

- Request-Filter: überprüft die Authentifizierung bei Zugriff auf REST-Schnittstellen
- JWT-Validator: validiert JWT
- Auth0 Flow Handler: koordiniert die nötigen Aktionen für das Auth0 Login
- Mail Sender: erstellt ein Bestätigungsmail und sendet es ab
- Model - View - Controller: Umsetzung des MVC Patterns

Datenbankmodell

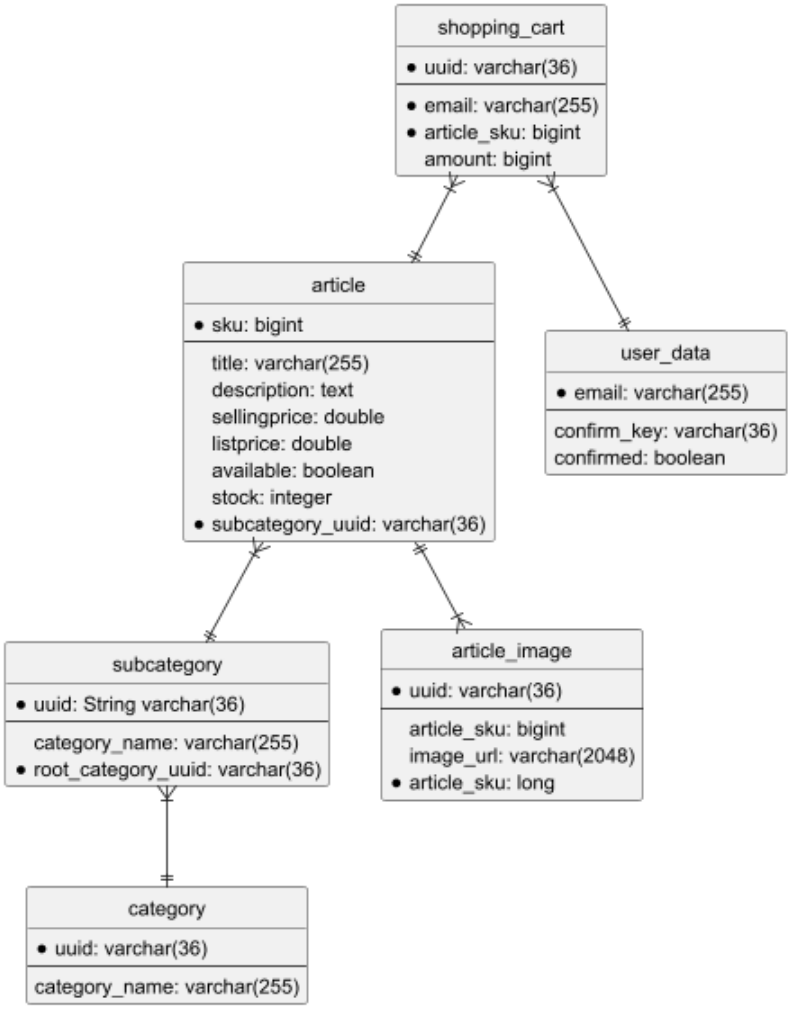

Figure 3: ERD Diagramm der Datenbank

Laufzeitsicht

Login Flow

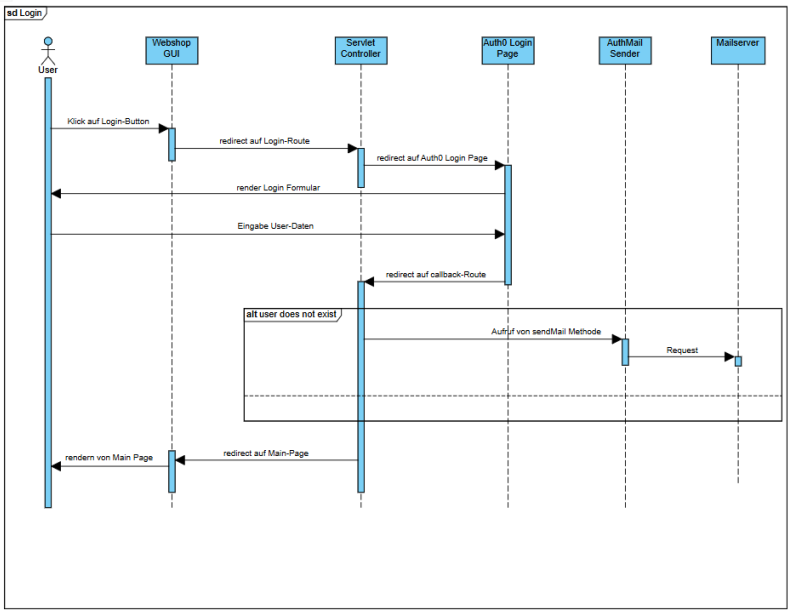

Figure 4: Login-Flow Sequenzdiagramm

REST-API Call

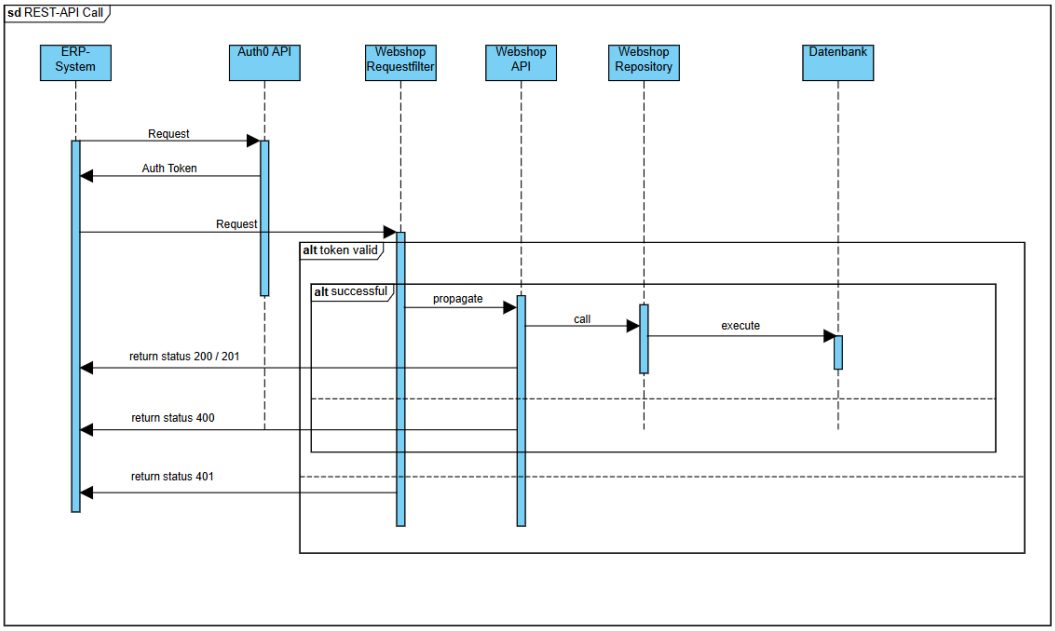

Figure 5: REST-API Call Sequenzdiagramm

Rendern der Artikeldetail Ansicht

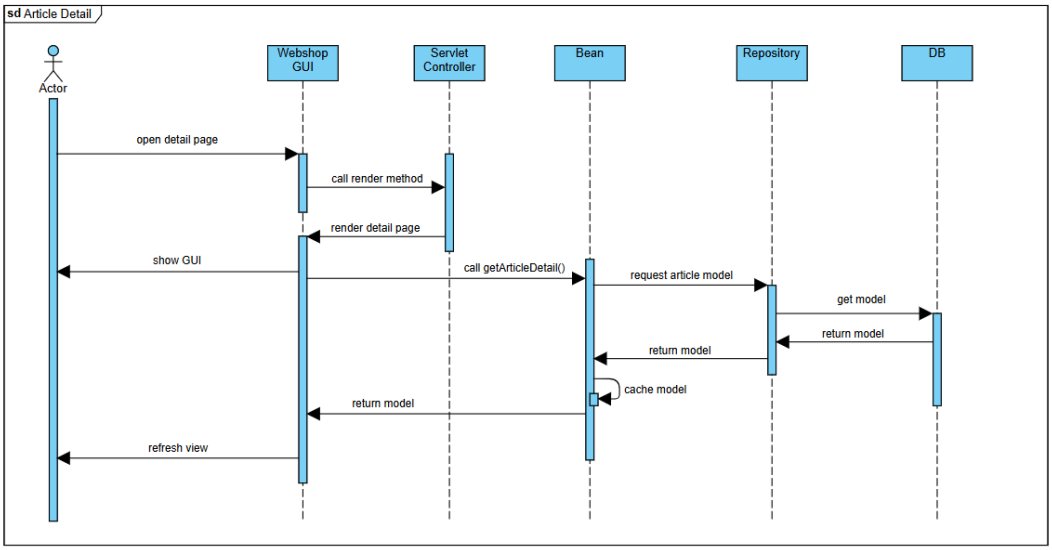

Figure 6: Sequenzdiagramm Artikeldetail Ansicht

Verteilungssicht

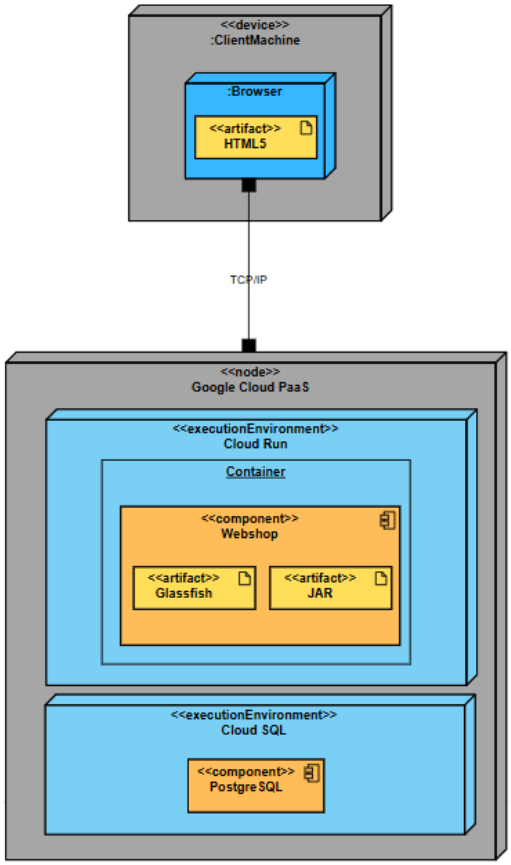

Figure 7: Verteilungsdiagramm Webshop

Querschnittliche Konzepte

Architektur & Patterns

- Das Projekt wird mit reinem Jakarta EE und Java Server Faces umgesetzt. Sämtliche verwendete Dependencies müssen framework-unabhängig sein.
- Aufgrund der kleinen Grösse des System wird ein klassischer monolithischer Ansatz gewählt, um die Komplexität niedrig zu halten und Ladezeiten zu minimieren.
- Gemäss dem Jakarta-EE Standard wird eine Three-Tier-Architektur aus Datenbank, Backend und Frontend gewählt.
- Die Daten im Frontend werden nach dem MVC-Pattern berechnet und angezeigt

Entwicklung

- Mit Mockito und JUnit werden die Methoden mit Business-Logik durch Unit-Tests abgesichert.

Sicherheit

- Um die REST-API abzusichern, wird M2M-Authentifizierung von Auth0 verwendet. Das Login von Benutzern wird ebenfalls mit Auth0 umgesetzt
- Da Benutzer nur auf ihre eigenen Daten zugreifen können, müssen die Daten des Warenkorbs eindeutig einem Benutzer zugeordnet werden können. Deshalb wird dafür ein Login verlangt.

Deployment

- Als Build-Tool wird Maven verwendet. Damit wird die Applikation als JAR-File gebaut.
- Die Applikation läuft als Container auf der Google Cloud. Als Webserver kommt Glassfish zum Einsatz, der das JAR-File als Web-Anwendung zur Verfügung stellt.
- Als Datenbank wird PostgreSQL verwendet, die in der CloudSQL-Umgebung läuft.

Qualitätsanforderungen

Qualitätsbaum

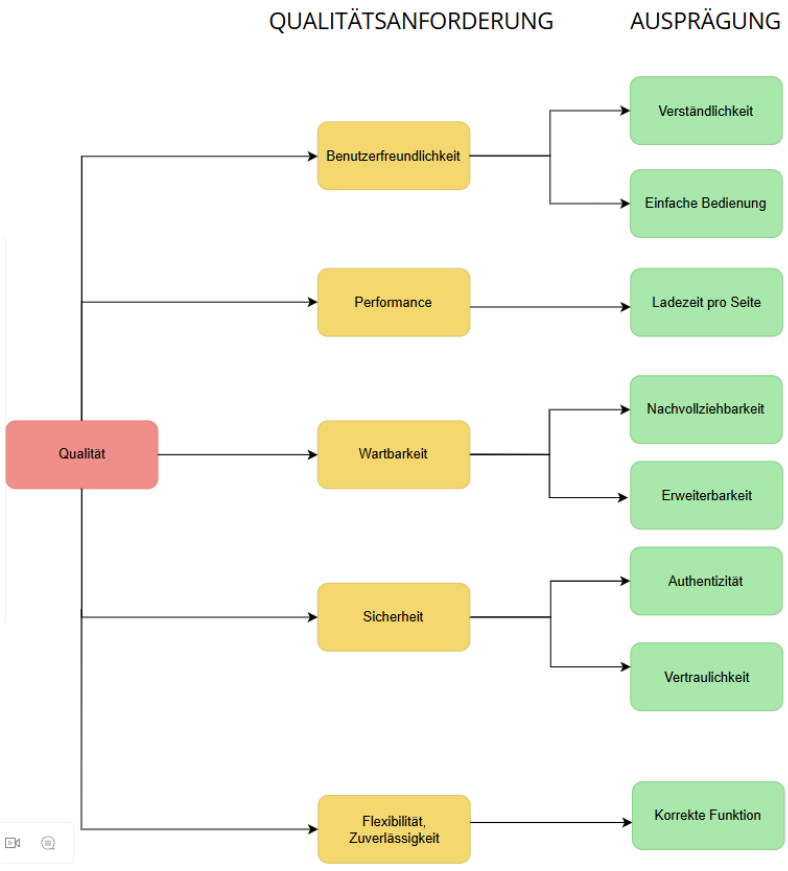

Figure 8

Qualitätsszenarien

- Der Kunde führt eine beliebige Operation im Webshop aus. Die Wartezeit, bis die Operation vollständig abgeschlossen ist, darf maximal 500ms betragen.
- Ein Benutzer versucht, manuell über die REST-API des Webshops Daten zu verändern. Die Anfrage muss in jedem Fall abgelehnt werden und die Aktion darf nicht ausgeführt werden.
- Ein Benutzer versucht, auf die Warenkorbdaten eines anderen Benutzers zuzugreifen. Dies darf ohne Logindaten des entsprechenden Benutzers nicht möglich sein.
- Ein Entwickler muss eine Änderung an der Navigation zwischen den Artikelseiten umsetzen. Durch die ausführliche Dokumentation geschieht hat er eine niedrige Lernkurve und implementiert die Anforderung korrekt.
- Eine neue Quelle für Stammdaten muss an den Webshop angeschlossen werden. Durch die ausführliche Dokumentation der API kann dies mit niedriger Lernkurve umgesetzt werden.
- Ein unerfahrener Mitarbeiter setzt eine Änderung am Webshop um und führt dabei einen Bug ein. Die entsprechenden Unit-Tests schlagen fehl und sorgen dafür, dass der fehlerhafte Code nicht in der Produktion landet.

Risiken und technische Schulden

Folgende Risiken und technische Schulden existieren im Projekt, geordnet nach Priorität:

- Anbindung an geeignetes Checkout-System: Aktuell ist der Webshop nur bis zum Warenkorb umgesetzt, für einen produktiven Einsatz muss der Checkout-Prozess noch angebunden werden
- Skalierbarer Maildienst: Aktuell verwendet der Webshop den normalen SMTP-Server von Gmail. Für einen flächendeckenden Einsatz sollte ein Email Service Provider wie MailJet eingesetzt werden.
- Verwendung von Frontend-Libraries: Aktuell ist das Frontend eher einfach gehalten und basiert auf standardisierten HTML-Elementen. In einer späteren Version kann das GUI durch den Einsatz von Frontend-Libraries modernisiert werden.
- Skalierung: Mit der Zunahme der Datenmengen und der gleichzeitigen Benutzer muss das Deployment Load Balancing unterstützen. Aktuell gibt es nur eine Instanz der Datenbank und Applikation.

Testing & Abnahmekriterien

Die Business-Logik des Webshops wird mithilfe von Unit-Tests getestet. Für die REST-API wird ein Postman Script verwendet. Als Abnahmekriterium gilt, dass 100% der Unit Tests erfolgreich sein müssen und alle Postman-Anfragen mit Status 200/201 erfolgen müssen

Glossar

Begriff	Definition
Anbindung externer Systeme	eine andere Applikation interagiert greift autorisiert auf die REST-API des Webshops zu
Volltextsuche	es wird nach Artikel gesucht, deren Titel den Suchbegriff enthalten; am Anfang, in der Mitte oder am Ende

Begriff	Definition
Bestätigungsmail	eine E-Mail, die einen Link enthält, den man öffnen muss; damit bestätigt man, dass man die E-Mail erhalten und damit Zugriff auf das E-Mail Postfach hat.
Business-Logik	Funktionen des Webshops die nicht standardisiert sind. Nicht zur Business-Logik zählen beispielsweise Mathematik- oder CRUD-Funktionen.
Mail-Server	Server, der angefragt werden kann, um Mails zu verschicken
Checkout-Prozess	umfasst Eingabe von Lieferadresse und Zahlungsmethode
Login Callback	Bevor auf das Login-Formular von Auth0 umgeleitet wird, wird definiert, was ausgeführt werden soll, wenn der Request anschliessend wieder zurück auf den Webshop geht. Dies ist die Callback-Funktion.
Servlet-Route	Route, die verwendet werden, um Seiten oder deren Daten zu laden und anzuzeigen
Request Filter	ein Filter, der bei jedem Request überprüft, ob dieser zugelassaen werden soll