Proof of Concept:
Integrierte Webshoplösung für Abacus

Studienarbeit

Herbstsemester 2023

Autor: Ramon Ebneter

Betreuer: Stefan Kapferer

Industriepartner: Customize AG

Neuwiesenstrasse 20

8400 Winterthur

Inhaltsverzeichnis

Abstract

Der Softwarehersteller Abacus Research AG hat als Teil ihrer Enterprise-Resource-Planning (ERP) Lösung 'Abacus' bis anhin eine E-Commerce-Applikation namens 'AbaShop' entwickelt. Da die Wartung dieser Shop-Lösung per Ende des Jahres 2025 eingestellt wird, steht die Customize AG, ein Vertriebspartner von Abacus und Industriepartner dieser Arbeit, vor der Aufgabe eine Nachfolgelösung für ihre bestehenden und zukünftigen Shop-Kunden zu finden. Um dies zu ermöglichen, stellt die Abacus eine neue RESTful HTTP-Schnittstelle zur Verfügung, über welche Kunden in Zukunft ihre eigenen Shop-Lösungen an das ERP anbinden können.

In dieser Studienarbeit wurden die Anforderungen der Kunden der Customize AG an die Shop-Lösung in Form von Use Cases festgehalten. Diese beinhalten sowohl bestehende als auch neue, häufig nachgefragte Funktionen. Basierend auf diesen Use Cases erfolgte eine Analyse der neuen Abacus-Schnittstelle um die Machbarkeit eines integrierten Webshops zu beurteilen. Durch eine Nutzwertanalyse wurden verschiedene E-Commerce-Frameworks evaluiert, um zu entscheiden, ob die Customize AG auf eine Eigenentwicklung setzt oder ein bestehendes Shop-System in Abacus integrieren soll. Um die Machbarkeit des erarbeiteten Konzepts aufzuzeigen, wurde ein Proof of Concept (PoC) implementiert.

Der in dieser Arbeit entwickelte Proof of Concept basiert auf einem Plugin für das Shop-System 'nopCommerce'. Anhand einer Auswahl von Use Cases konnte aufgezeigt werden, dass die neue Abacus-Schnittstelle die Anforderungen der Kunden erfüllt. Notwendige API-Erweiterungen wurden identifiziert und dokumentiert. Mittels einem Synchronisationsdienst werden die Produkt-Stammdaten aus dem Abacus mit dem Shop-System abgeglichen. Im Warenkorb werden kundenspezifische Preise und Rabatte ausgegeben und die getätigten Bestellungen werden automatisch als Aufträge in Abacus erfasst. Ebenfalls wird aufgezeigt, wie die restlichen Anforderungen auf Basis des entwickelten Prototyps umgesetzt werden können.

Kapitel 1

Management Summary

1.1 Ausgangslage

Ein modernes ERP-System bietet seinen Kunden die Möglichkeit auf Basis der vorhandenen ERP-Stammdaten einen Webshop mit automatisiertem Datenaustausch zur Verfügung zu stellen. Die Abacus Research AG erfüllte diese Anforderung bisher mit der Lösung AbaShop. Aufgrund der kommunizierten Einstellung der Wartung von AbaShop per Ende 2025 steht der Industriepartner Customize vor der dringenden Aufgabe, für seine bestehenden und zukünftigen Kunden, eine neue, zukunftssichere, effiziente und flexibel integrierbare Webshop-Lösung zu finden.

Die jüngsten Entwicklungen im Bereich der HTTP RESTful Schnittstellen bei Abacus bieten neue Möglichkeiten für die Entwicklung eines in das ERP integrierten Webshops. Das Ziel dieser Arbeit ist es zu evaluieren, inwiefern die Schnittstellen die Entwicklung eines Webshops ermöglichen, der nahtlos in das Abacus ERP integriert werden kann.

1.2 Vorgehen

Im Rahmen dieser Studienarbeit wurde eine detaillierte Analyse der funktionalen Anforderungen an einen in Abacus ERP integrierten Webshop durchgeführt. Diese Anforderungen decken ab, was das bestehende Produkt AbaShop zur Verfügung stellt, evaluieren aber auch, welche Funktionalitäten von bestehenden Kunden häufig gewünscht werden und somit die bestehenden Funktionalitäten erweitern könnten. Basierend auf diesen Anforderungen wurde eine umfassende Analyse der neuen RESTful HTTP-Schnittstelle von Abacus durchgeführt. Ziel war es, dem Industriepartner eine objektive Einschätzung zu geben, ob ein Webshop auf Basis von Abacus ERP mit diesen Schnittstellen möglich ist und ob sie ihren bestehenden Kunden eine Nachfolgelösung anbieten können.

In dieser Arbeit wird zudem analysiert, ob ein bestehendes E-Commerce-System als Basis für eine Anbindung an Abacus verwendet werden kann. Basierend auf dieser Analyse wurden vier repräsentative Use Cases ausgewählt, um eine Proof of Concept (PoC) Implementation für den Webshop zu entwickeln. Repräsentativ bedeutet in diesem Fall, dass die Use Cases von einer möglichen breiten Nutzung der Abacus-Schnittstelle ausgehen und die wichtigsten Prozesse in einem Webshop abbilden.

1.3 Ergebnis

Die vorliegende Arbeit zeigt, dass die von Abacus entwickelte RESTful HTTP-Schnittstelle optimale Voraussetzungen für einen integrierten Webshop bietet. Die Analyse ergab nur wenige Verbesserungsmöglichkeiten der Schnittstelle, die dem Softwarehersteller mitgeteilt wurden. Einige dieser Verbesserungen befinden sich aufgrund der Erkenntnissen dieser Arbeit bereits in Entwicklung. Auf Basis der Schnittstellenanalyse wurde eine C#-Bibliothek (Client) entwickelt, welche die Authentifizierung und Interaktion mit der Schnittstelle über Services abstrahiert und sauber typisierte, strukturierte Objekte der abgefragten Entitäten zurückgibt. In Absprache mit dem Industriepartner wurde entschieden, die definierten Use Cases für den PoC mit dem E-Commerce System nopCommerce [29] umzusetzen. Zu diesem Zweck wurde ein Connector zwischen nopCommerce und Abacus ERP entwickelt, der eine Architektur aufzeigt, mit der die evaluierten Anforderungen an einen integrierten Webshop erfüllt werden können.

1.4 Ausblick

In der nächsten Projektphase steht die Umsetzung der im Kapitel 3.2 definierten Use Cases im Vordergrund. Ziel ist es, den Webshop auf den Status eines Minimum Viable Products (MVP) zu bringen und damit eine Basis für erstes Kundenfeedback und zukünftige Entwicklungen zu schaffen. Gleichzeitig ist eine detaillierte Evaluierung notwendig, um zu entscheiden, ob das System als headless angeboten werden soll [41]. Dies ermöglicht die Entwicklung eines personalisierten Frontends für jeden Kunden und erlaubt die Nutzung über verschiedene Kanäle.

Kapitel 2

Kontext

2.1 Dokumentation

In diesem Dokument dient die fiktive Firma 'Bürobedarfs AG' als Modellunternehmen, um konzeptionelle Themen anhand eines realen Beispiels zu veranschaulichen. Solche Beispiele werden mit folgender Box gekennzeichnet:

Abacus

Die Bürobedarfs AG nutzt das Abacus ERP um ihre Geschäftsprozesse digital abzubilden.

2.2 Ausgangslage

Die 1985 gegründete Abacus Research AG ist der führende Anbieter von ERP-Software in der Schweiz [34]. Industriepartner dieser Arbeit ist die Customize AG, einer der drei grössten Vertriebspartner von Abacus. Die Abacus Research AG kümmert sich ausschliesslich um die weiterentwicklung vom ERP System. Für die Implementation der Software besitzt sie ein Netzwerk von Vertriebspartnern, welche die Parametrierung beim Kunden vornehmen.

Bisher hat Abacus selbst ein Shopsystem namens AbaShop entwickelt. Damit konnten Kunden mit dem Modul Auftragsbearbeitung (ABEA) eine integrierte Webshop-Lösung mit automatischem Stammdatenaustausch auf Basis ihres ERP-Systems anbieten. Die Ankündigung von Abacus, den AbaShop per Ende 2025 einzustellen, stellt den Industriepartner vor eine Herausforderung, seinen bestehenden und zukünftigen Kunden eine Alternativlösung für das Shopsystem anzubieten. Damit alternative Lösungen aufgebaut werden können, stellt Abacus mit der Version 2023 eine RESTful HTTP-Schnittstelle zur Verfügung, um mit den notwendigen Entitäten für eine integrierten Webshop zu interagieren.

2.3 Abacus

Abacus ist ein modular aufgebautes ERP-System. Eine einzelne Applikation erfüllt selbstständig einen fachlichen Teilbereich vom ERP. Alle diese Applikationen sind innerhalb vom Abacus mittels Standardschnittstellen miteinander verbunden. Im Kontext einer Webshop-Lösung sind drei Applikationen aus dem Portofolio vom Abacus involviert (Auftragsbearbeitung, Customer Relationship Management (CRM) und E-Business). In der Abbildung 2.1 wird aufgezeigt, wie diese Applikationen zusammenspielen und wo die Schnittstellen innerhalb vom Abacus sowie zu einem Webshop sind. Innerhalb einer Gruppe sind die wichtigsten Domänenkonzepte im Kontext eines E-Commerce-Systems der jeweiligen Applikation als Klassen aufgeführt.

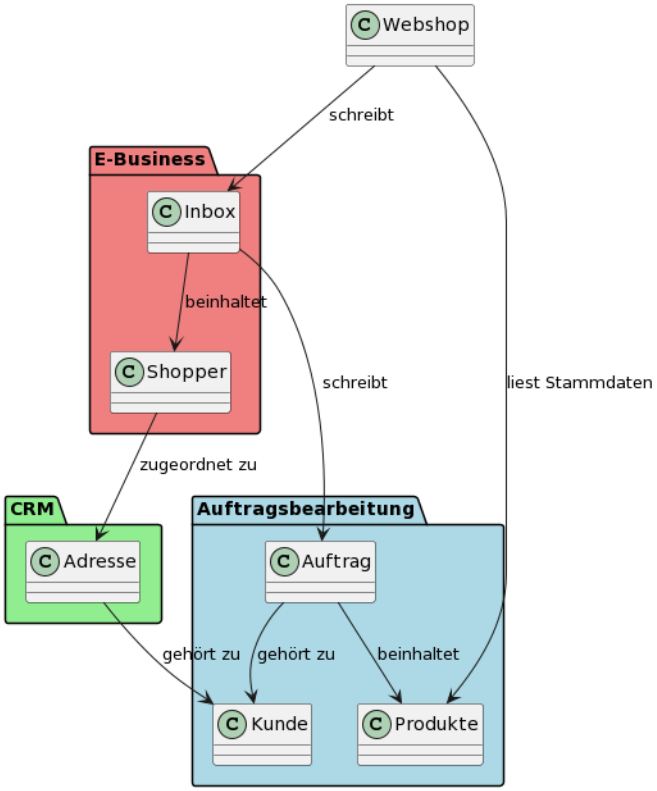

Abbildung 2.1: Applikationen Abacus (UML Klassendiagramm)

Die Abbildung der Schnittstellen zu einem Webshop ist vereinfacht dargestellt und wird im Abschnitt 4.1 detaillierter erläutert.

Das Kernstück ist die Auftragsbearbeitung, welche die gesamte Auftragsabwicklung und die Verwaltung von Produktstammdaten abdeckt. Der Webshop speichert die Verkaufsaufträge nicht direkt in der Auftragsbearbeitung. Sie werden in der Inbox des E-Business abgelegt und von den Sachbearbeitern des Kunden bearbeitet. Hierbei werden zum Beispiel neue Shopper einer Adresse im CRM zugeordnet. Die Adressen gehören zu einem Kunden, an welchem der Auftrag schlussendlich gestellt wird.

2.4 Domänenmodell

In der folgenden Abbildung 2.2 ist das Domänenmodell der im vorherigen Abschnitt 2.3 beschriebenen Applikationen vom Abacus ERP dargestellt. Der Webshop ist dabei als einzelne Entität modelliert, da dieses Modell die fachliche Domäne des ERP abbilden soll. Um die Zuordnung der Entitäten zu den Applikationen zu verdeutlichen, folgt die Farbgebung dem vorherigen Abschnitt.

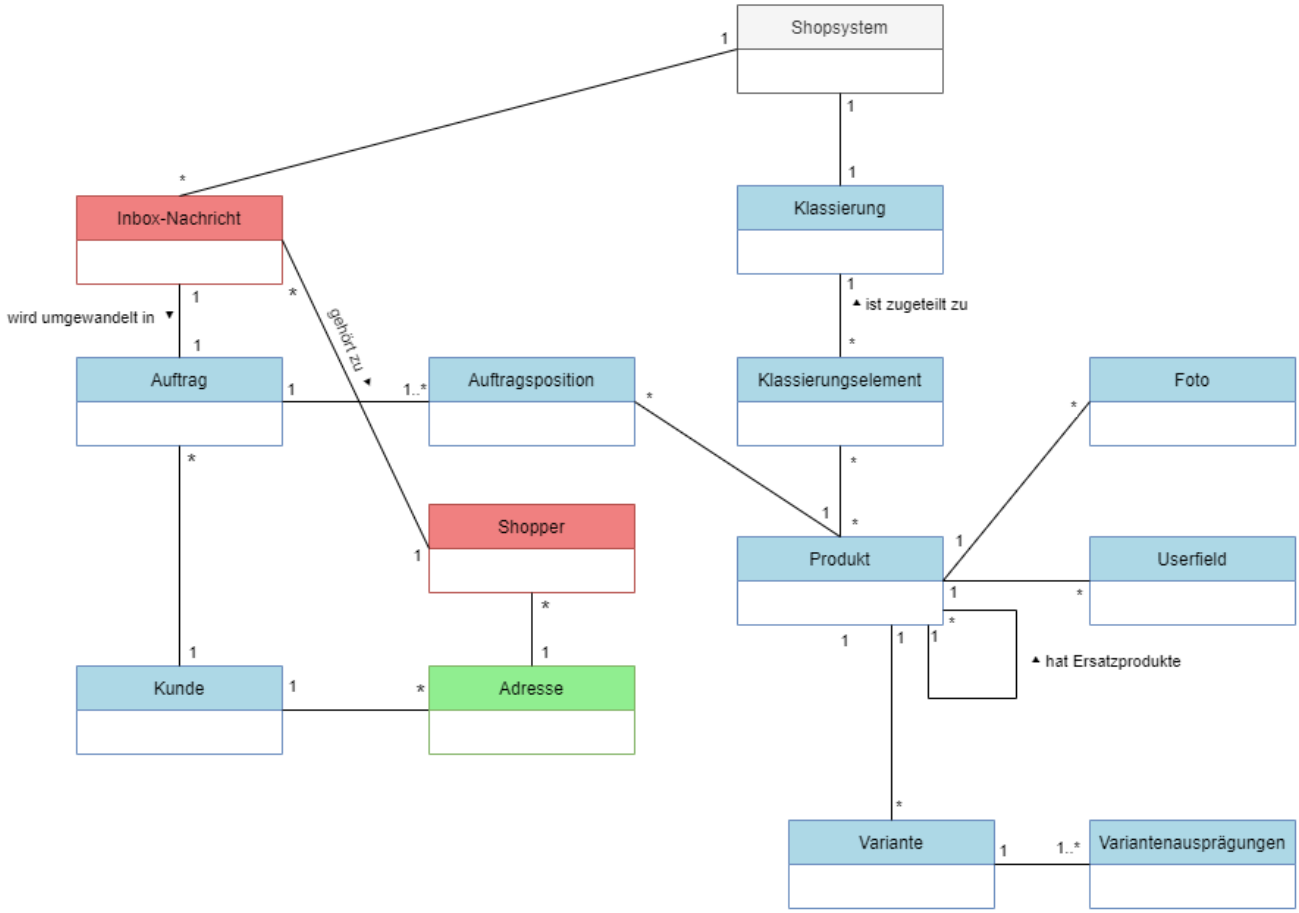

Abbildung 2.2: Domänenmodell Abacus ERP mit integriertem Shop

In den nachfolgenden Abschnitten erfolgen detaillierte Ausführungen zum Domänenmodell von Abacus, die für den Kontext dieser Arbeit von wesentlicher Bedeutung sind.

2.4.1 Klassierung

Es ist zu beachten, dass eine Instanz eines Webshops immer nur einer Klassierung zugeordnet ist, auch wenn in Abacus mehrere verwaltet werden können. Von besonderer Bedeutung ist die Beziehung zwischen einem Produkt und einem Klassierungselement, die eine Mehrfachzuordnung erfordern kann und im Webshop abgebildet werden muss. Im Abacus wird dies als Mehrfachklassierung bezeichnet, da ein Produkt unter verschiedenen Klassierungspfaden gefunden werden kann.

Klassierung

Die Bürobedarfs AG strukturiert ihr Sortiment mittels einer Baumstruktur, um eine klare Kategorisierung der Artikel zu gewährleisten und deren Auffindbarkeit zu optimieren. Abbildung 2.3 zeigt, wie eine solche Klassierung aussehen kann.

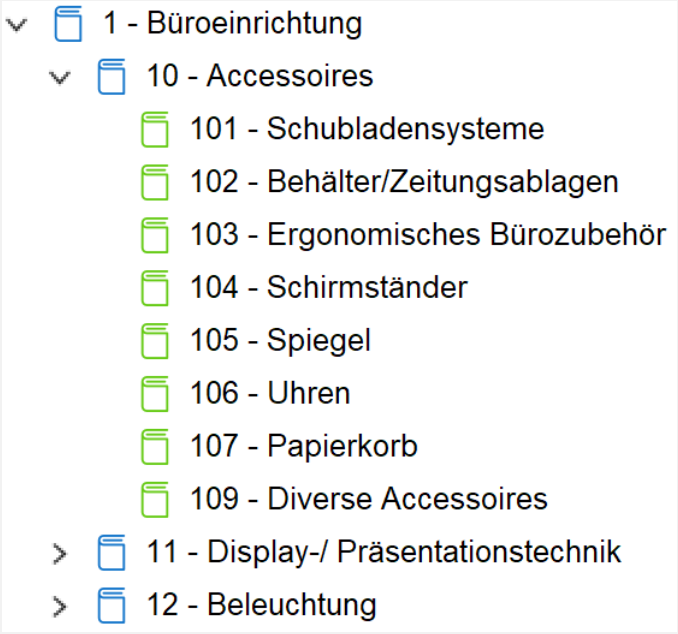

Abbildung 2.3: Beispiel Klassierung in Abacus

Jeder im Webshop angebotene Artikel ist einem oder mehreren Klassifikationselementen zugeordnet. Die Klassifikation wird im Webshop als Navigationsbaum dargestellt.

2.4.2 Userfields

Userfields ermöglichen, auf der Ebene von Entitäten eigene Attribute zu definieren. Diese individuell definierten Attribute sind insbesondere im Kontext eines Webshops relevant, da Kunden aus unterschiedlichen Branchen spezifische Eigenschaften für ihre Produkte benötigen. Da durch Userfields kundenspezifische Informationen abgebildet werden können, sind sie ein wichtiges Konzept in Abacus.

Userfields

Die Bürobedarfs AG führt für den Verkauf von Computern auf ihren Produkten ein Userfield 'Arbeitsspeicher', um die Kunden über die Grösse des Arbeitsspeichers des jeweiligen Computers zu informieren. Die Mitarbeiter können auf jedem gewünschten Artikel einen Wert für das Userfield 'Arbeitsspeicher' definieren. Durch die strukturierte Erfassung dieser Information im Userfield ist es möglich, einen Filter im Webshop zu implementieren. Dieser Filter ermöglicht es dem Kunden gezielt nach Computern mit den gewünschten Spezifikationen zu suchen, was die Benutzererfahrung erheblich verbessert.

2.4.3 Variante

Ein Artikel kann als Variantenartikel geführt werden. Dies bedeutet, dass der Kunde zwischen verschiedenen Ausprägungen eines Artikels wählen kann. Eine Variante definiert sich dabei durch ihre Variantenausprägungen.

Variante

Die Bürobedarfs AG bietet eine Ablagebox in verschiedenen Farben an. Jede Variante definiert sich dabei durch die beiden Variantenausprägungen Farbe (rot, grün, blau) und Grösse (S, M, L). Somit ergeben sich 6 Varianten des Artikels mit beispielsweise einer Variante mit Farbe = grün und Grösse = M.

Pro Variante können Lagerbestand und Preis individuell definiert werden. Deshalb wird ein Variantenartikel in Abacus neben der Artikelnummer zusätzlich durch die Variantennummer eindeutig identifiziert.

2.4.4 Sprache

Um die Komplexität des Domänenmodells zu minimieren, wurde nicht explizit modelliert, dass Produkte und Bezeichnungen von Klassifikationselementen übersetzt werden können. Auch Userfields können als übersetzte Attribute erfasst werden.

2.4.5 Inbox / Shopper

Die Applikation E-Business stellt die beiden Entitäten Inbox und Shopper zur Verfügung. Das E-Business agiert hierbei als Schnittstelle für den schreibenden Zugriff aus dem Webshop. Bei einer getätigten Bestellung im Webshop wird ein Eintrag in der Inbox erzeugt. Nebst den Auftragsdaten muss dort ein bestehender oder ein neuer Shopper mitgeschickt werden. Bei der Generierung eines Eintrages in der Inbox wird von Abacus eine Vielzahl von Validierungen durchgeführt, damit fehlerhafte Bestellungen von Mitarbeitern manuell überarbeitet werden können. Ein unbekannter Käufer wird beispielsweise in der Inbox als Fehler markiert und durch einen Mitarbeiter einer bestehenden oder neuen Adresse im CRM zugeordnet.
Es ist zu beachten, dass ein Shopper stets einer einzigen Adresse zugeordnet ist, während eine Adresse mehrere Shopper haben kann. Ein Kunde kann ausserdem mehrere Adressen haben. Mit diesem Konstrukt lässt sich darstellen, dass mehrere Shopper für denselben Kunden im ERP einkaufen können.

Shopper

Die Bürobedarfs AG hat einen grossen Kunden mit zwei Standorten. An jedem Standort arbeiten mehrere Mitarbeiter, die bei der Bürobedarfs AG ihr Material einkaufen. Im Abacus der Bürobedarfs AG gibt es einen Kunden mit zwei Adressen. Jeder Mitarbeiter des Kunden hat einen Shopper, der der Adresse seines Standorts zugeordnet ist. Die Bürobedarfs AG kann die Rechnung jeweils demselben Kunden zuordnen. Die Lieferung der Ware erfolgt dann an die entsprechende Adresse des Shoppers.

2.5 Umfang der Studienarbeit

Der folgende Abschnitt fasst das Ziel dieser Arbeit zusammen. Die vollständige Aufgabenstellung befindet sich im Anhang D.

In dieser Arbeit lag der Hauptfokus auf der Evaluation der neuen RESTful-Schnittstellen von Abacus und ihrer Eignung zur Abdeckung der E-Commerce-Bedürfnisse der Kunden von Customize AG. Hierbei wurde geprüft, ob es sinnvoll ist, ein Plugin für ein bestehendes E-Commerce-System zu entwickeln oder ob eine Eigenentwicklung die bessere Lösung darstellt.

Um dieses Hauptziel zu erreichen, wurden folgende Liefergegenstände verfolgt:

1. **Anforderungserhebung und -Analyse:** Es wurden detaillierte Anforderungen der Kunden von Customize an ihre E-Commerce-Lösungen ermittelt. Diese Anforderungen wurden repräsentativ und aussagekräftig evaluiert, um eine solide Grundlage für die weitere Arbeit zu schaffen.
2. **Produktevaluation:** Die verbreitetsten E-Commerce-Plattformen und -Lösungen (Minimum: 3, Maximum: 7) wurden analysiert und bewertet. Dabei wurde untersucht, wie gut sie die zuvor

identifizierten Anforderungen erfüllen können. Dieser Schritt war signifikant für die Entscheidung, ob ein Plugin oder eine Eigenentwicklung die bessere Wahl ist.

3. **Prototypentwicklung:** Im Rahmen des Projekts wurde ein Prototyp entwickelt, der einige der wichtigsten Use Cases für die Kunden von Customize abdeckt. Dieser Prototyp dient dazu, die Machbarkeit der Umsetzung der Anforderungen zu demonstrieren und erste praktische Erfahrungen mit den Schnittstellen zu sammeln. Hierbei wurde insbesondere Wert auf die Evaluation der Eignung der Abacus API zur Erstellung des besagten Prototyps gelegt.
4. **Migrationsfähigkeit:** Es wurde untersucht, wie eine mögliche Migration der bestehenden Abashop-Lösung auf die neue Webshop-Lösung durchgeführt werden kann.

Folgende Punkte wurden als kritische Erfolgsfaktoren erachtet:

1. Kundenanforderungen wurden repräsentativ und in einem ausreichenden Detaillierungsgrad erfasst. Es wurden 3-4 repräsentative Use Cases definiert und in einem Prototyp implementiert. Repräsentativ bedeutet in diesem Kontext, dass die Use Cases die Bedürfnisse der Nutzer und Kunden möglichst breit abdecken. Es wurde ausserdem eine zielgruppengerechte Dokumentation erstellt, die es dem Industriepartner ermöglicht, das Produkt selbstständig weiterzuentwickeln.

2. Es wurden Standard-Lösungen für E-Commerce-Systeme evaluiert und eine begründete Entscheidung getroffen, wie der Prototyp implementiert wird (Eigenbau vs. Integration von Standardprodukten).

3. Bei der Entwicklung des Prototyps wurde auf Benutzbarkeit, Erweiterbarkeit sowie Wartbarkeit geachtet. Es galten die üblichen Ansprüche an die Software: Die Software soll insbesondere hohe Kohäsion sowie geringe Kopplung umsetzen. Zur Erstellung dieser Software wurden industrieübliche Prozesse eingesetzt. Dies inkludiert insbesondere Software Engineering Hygienefaktoren wie automatisierte Builds, Tests und angemessene Versionierung mit Git.

4. Es wurde eine Aussage über die Eignung der Abacus API Schnittstelle gemacht sowie begründet.

Kapitel 3

Anforderungen

Dieses Kapitel beschreibt alle funktionalen und nicht-funktionalen Anforderungen einer Webshoplösung für Abacus. Für die Visualisierung der Priorisierung werden die Anforderungen mit folgendem Farbcode beschrieben.

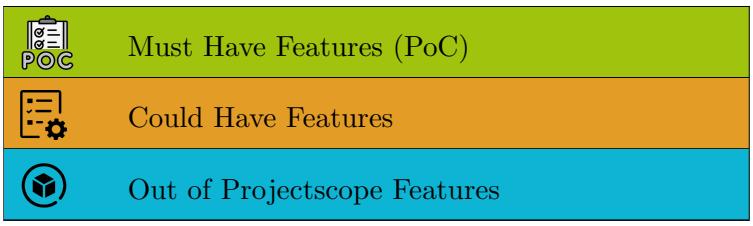

Tabelle 3.1: Use Cases - Farbcode

Must Have Features (PoC)	Definieren alle Anforderungen, welche bei Projektende im Prototypen vorhanden sein sollen.
Could Have Features	Optionale Anforderungen, die mit dem Prototypen umgesetzt werden, sollte noch Zeit übrig bleiben. Diese Features sollten für einen Minimum Viable Product (MVP) Webshop vorhanden sein.
Out of Projectscope Features	Anforderungen ausserhalb vom Projektscope. Die Architektur des Prototypen soll eine spätere Implementierung ermöglichen.

3.1 Akteure

Das System under Development (SuD) hat mehrere Akteure, welche mit dem System interagieren. In der nachfolgenden Tabelle werden diese beschrieben, weshalb in den kommenden Kapiteln auf eine weitere Ausführung ihrer Rolle verzichtet wird.

Akteur	**Beschreibung**
Shopper	Beschreibt den Endkunden, welcher im Shop Einkäufe tätigt. Dies kann eine Firma oder eine Privatperson sein, je nachdem, ob es sich um einen Business-to-Business (B2B) oder Business-to-Consumer (B2C) Shop handelt.
Shopbetreiber	Der Shopbetreiber ist eine Firma (Kunde von Customize AG), welcher ein Abacus ERP System besitzt und seinen Kunden (Akteur Shopper) einen Webshop anbieten möchte.
Consultant	Der Akteur Consultant ist ein Mitarbeiter der Customize AG, welcher den Akteur Shopbetreiber betreut. Der Consultant richtet bei Neuprojekten das Shopsystem ein und führt im Nachgang die weiterführende Betreuung durch.
Abacus	Das Abacus ERP agiert als Mastersystem der Stammdaten. Dieser Akteur beliefert das SuD mit Stammdaten und nimmt Verkaufsaufträge entgegen.
Bezahlprovider	Externer Service, welcher Bezahlvorgänge im Webshop durchführt.
Mailsystem	Im gesamten Lebenszyklus einer Shopinteraktion werden diverse Mails über den Akteur Mailsystem versendet.

Tabelle 3.2: Beschreibung Akteure

3.2 Funktionale Anforderungen

Die funktionalen Anforderungen an das System werden mittels Use Cases modelliert.
Alle erfassten Use Cases sind im Anhang A im Fully Dressed Format ausführlich definiert. Die Struktur für diese Beschreibung basiert auf der Vorlage von Craig Larman, welche um wenige Felder reduziert wurde:

- **Scope:** Ist im Kontext dieser Arbeit immer das neue Webshopsystem
- **Level:** Use Cases sind in jedem Fall User Goals
- **Special Requirements:** Werden unter Preconditions erfasst

Die Erhebung der Use Cases basiert auf den Funktionalitäten des AbaShops. Es sind alle Anforderungen enthalten, die der AbaShop erfüllt und im Rahmen der Priorisierung als 'Must Have' und 'Could Have' eingestuft wurden. Diese müssen für ein Minimum Viable Product (MVP) berücksichtigt werden, da das System under Development (SUD) auch darauf abzielt, bestehende Kunden des Industriepartners Customize zu übernehmen. Es wurden zusätzliche Use Cases aufgenommen, die von vielen Kunden als gewünschte Funktionalitäten im AbaShop identifiziert wurden.

Im Abschnitt 3.2.1 werden die spezifischen Anforderungen an den Proof of Concept (PoC) näher ausgeführt, während der Abschnitt 3.2.3 die übrigen Anforderungen umschreibt.

3.2.1 Anforderungen Proof of Concept

Im Folgenden werden die Use Cases beschrieben, welche an den PoC gestellt wurden. Die Gründe für die Auswahl dieser Use Cases sind im Abschnitt 3.2.2 zu finden.

UC01 - Konfiguration

Ein Consultant verbindet den Shop mit Abacus zur Synchronisierung von Stammdaten.

Der Akteur Consultat ist ein Mitarbeiter des Industriepartners, der jeweils einem Shopbetreiber als Berater zugeordnet ist. Dieser ist für die Einführung und spätere Betreuung des Webshops verantwortlich. Für den Berater soll eine einfache Möglichkeit geschaffen werden, die Konfiguration der Anbindung an Abacus vorzunehmen und zu kontrollieren. Auf Fehler in der Konfiguration sollte der Berater hingewiesen werden. Da sich die Verbindungsdaten zu einem späteren Zeitpunkt ändern können, muss eine laufende Anpassung der Konfiguration möglich sein.

UC02 - Produktsuche

Ein Shopper durchsucht das Produktangebot über die Produktklassierung.

Im Webshop soll der Shopper die Möglichkeit haben, das Angebot des Shopbetreibers zu durchsuchen. Die Detailansicht eines Produkts stellt die in Abacus gepflegten Stammdaten in der gewählten Shopsprache dar. Tabelle 3.3 zeigt die minimal erforderlichen Informationen, die ein Shopper für jedes Produkt in dieser Ansicht sehen muss.

Entität	**Beschreibung**	**Übersetzt**
Texte	Ein in HTML formatierter Produkttext und eine Produktbezeichnung	Ja
Preis	Listenpreis inklusive der korrekten Mehrwertsteuer	Nein
Produktinformation	Die externe Produktnummer und die EAN-Nummer	Ja
Bilder	Alle dem Artikel zugeordneten Produktbilder	Nein
Userfields	Die Attribute des Artikels, einschliesslich Attributname und zugeordnetem Wert	Ja
Lagerdaten	Der verfügbare Lagerbestand wie auch die minimale/maximale Bestellmenge	Nein

Tabelle 3.3: Minimale Informationen auf einem Produkt

Die Produktklassierung (siehe Abschnitt 2.4.1) ermöglicht dem Shopper eine gezielte Navigation zu den gewünschten Produkten. Jedes Klassierungselement wird in der Navigation mit seinem übersetzten Namen angezeigt. In Abacus kann jedem Klassierungselement ein Bild zugeordnet werden. Wenn der Shopper ein Klassierungselement auswählt, das untergeordnete Klassierungselemente enthält, wird das Bild des übergeordneten Klassierungselements neben dessen Namen dargestellt, um eine visuelle Unterstützung bei der weiteren Auswahl zu bieten.

UC03 - Warenkorb

Ein Shopper kann Produkte zum Warenkorb hinzufügen und bearbeiten.

Nachdem der Kunde seine Produktauswahl abgeschlossen hat, kann er die ausgewählten Produkte in den Warenkorb legen. Der Warenkorb ist editierbar und ermöglicht es dem Kunden, Produkte zu entfernen oder deren Menge anzupassen.

Dieser Use Case deckt einen kritischen Teil des Proof of Concept ab. Wenn Artikel zum Warenkorb hinzugefügt werden, muss der gesamte Warenkorb an die RESTful HTTP-Schnittstelle von Abacus gesendet werden, damit das ERP die Preisfindung durchführen kann. Dies ist notwendig, da Abacus über eine umfassende Preis- und Rabattfindung verfügt, die nicht eigenständig nachgebaut werden sollte. Auf diese Weise wird sichergestellt, dass im Webshop dieselben Konditionen gelten wie bei einer Bestellung, die direkt im ERP erfasst wurde. Hierbei sollen auch kundenspezifische Konditionen berücksichtigt werden. Treten bei der Durchführung der Preisfindung Fehler auf, soll dem Shopper der Listenpreis angezeigt werden.

UC04 - Bestellung

Ein Shopper kann eine Bestellung tätigen.

Sobald der Warenkorb gemäss UC03 fertiggestellt ist, kann der Shopper die Bestellung aufgeben. Dazu gibt er seine Adresse ein und wählt ein Zahlungsmittel. Im Rahmen der PoC-Implementierung wird die Option der Kreditkartenzahlung bewusst ausgelassen, da der Industriepartner zunächst festlegen muss, mit welchen Zahlungsanbietern zusammengearbeitet wird. Es wird daher ausschliesslich die Bezahlung per Rechnung implementiert.

Sobald eine Bestellung erfolgreich im Webshop platziert wurde, soll sie direkt in die Inbox von Abacus übermittelt werden. Es besteht die Möglichkeit, dass diese Übermittlung fehlschlägt. In einem solchen Fall ist es wichtig, dass die Bestellung für den Shopper dennoch verarbeitet wird. Dies stellt sicher, dass der Webshop auch dann funktionsfähig bleibt, wenn beispielsweise das Abacus ERP gewartet wird. Der Shopbetreiber oder Consultant sollte über solche fehlgeschlagenen Übermittlungen informiert werden, um eine erneute Übermittlung einleiten zu können.

3.2.2 Priorisierung

Die funktionalen Anforderungen wurden priorisiert, um einen repräsentativen Proof of Concept zu realisieren. Die Priorisierung zielte darauf ab, mit einer minimalen Anzahl von Use Cases eine möglichst breite Nutzung der Abacus Schnittstelle zu ermöglichen. Die Abdeckung wird im Abschnitt 4.1.7 dargestellt. Diese breite Nutzung war notwendig, um festzustellen, ob eine integrierte Webshop-Lösung für Abacus mit den derzeitigen Möglichkeiten realisierbar ist. In der Tabelle 3.4 werden die Überlegungen zur Auswahl der in Rahmen dieser Studienarbeit umgesetzten Use Cases ausgeführt.

Use Case	**Begründung**
UC01	Die Konfiguration (Verbindung zu Abacus) bildet die Basis vom ganzen System. Eine Ausprogrammierung von diesem Use Case wird nach Absprache mit dem Industriepartner angestrebt, da die Involvierung vom Akteur Consultant ein wichtiger Punkt ist.
UC02	Die Produktsuche und entsprechende Darstellung benötigt alle lesenden Schnittstellen der Produktstammdaten. Dies bildet das Fundament für einen Webshop.
UC03	Der Warenkorb bildet einen kritischen Punk vom PoC ab. Durch die komplexe Preisfindung innerhalb vom Abacus ERP muss der Warenkorb an Abacus übermittelt werden, um eine endgültige Preisfindung durchzuführen.
UC04	Die Bestellung deckt die wichtigste schreibende Schnittstelle ab, damit Bestellungen automatisiert im ERP erfasst werden.

Tabelle 3.4: Priorisierung funktionale Anforderungen

3.2.3 Weitere Anforderungen

Tabelle 3.5 bietet einen Überblick über alle Use Cases, die nicht zum Umfang des PoC gehören. Ein zentrales Ergebnis dieser Arbeit stellt die umfassende Anforderungsanalyse dar. Daher werden auch diese Use Cases im Anhang A, im Fully Dressed Format, ausführlich spezifiziert.

Could Have	
Use Case	**Beschreibung**
UC05 - Erweiterte Produktsuche	Userfields, die auf einem Artikel definiert sind, sollen durch den Shopbetreiber oder Consultant als filterbare Attribute gekennzeichnet werden können. Dadurch kann der Shopper neben der Navigation über die Produktklassifizierung zusätzlich mithilfe von Filtern auf den Produktattributen das Suchergebnis verfeinern.
UC06 - Registrierung	Ein Shopper soll sich im Webshop registrieren können und dabei seine Adressdaten eigenständig verwalten.
UC07 - Login	Ein gemäss UC06 registrierter Shopper soll sich für zukünftige Einkäufe einloggen können.
UC08 - Cross Selling	Einem Produkt können in Abacus mehrere Ersatzprodukte zugeordnet sein, welche dem Shopper auf der Produktseite als Cross-Selling-Vorschläge präsentiert werden sollen.
UC09 - Konditionen und Rabatte	Beim Durchstöbern des Produktkatalogs werden dem Shopper seine spezifischen Preise und Rabatte angezeigt. Bei Variantenartikeln wird der Preis für die ausgewählte Variante dynamisch dargestellt.
UC10 - Einkaufsverlauf anzeigen	Ein eingeloggter Shopper kann seine getätigten Einkäufe einsehen, wobei der aktuelle Status der Bestellung (zum Beispiel offen, versendet, abgeschlossen) angezeigt wird.
Out of Project Scope	
Use Case	**Beschreibung**
UC11 - Produktbewertungen	Nach Abschluss einer Bestellung soll ein Shopper in der Lage sein, Bewertungen für Produkte abzugeben. Es soll auch möglich sein, schriftliche Bewertungen zu verfassen, die anderen Shoppern angezeigt werden.

UC12 - Wunschliste	Ein eingeloggter Kunde kann eine Wunschliste mit Produkten führen, die später in den Warenkorb gelegt werden können.
UC13 - Rücksendung und Umtausch	Basierend auf UC10 soll ein Shopper eigenständig eine Rücksendung oder einen Umtausch einleiten können.
UC14 - Gutscheinverwaltung	Der Shopbetreiber kann Gutscheine im Adminbereich verwalten und zum Verkauf anbieten. Diese können vom Shopper erworben und als Zahlungsmittel verwendet werden.
UC15 - Content Management	Der Shopbetreiber soll in der Lage sein, statische Inhaltsseiten im Webshop ohne Programmierkenntnisse zu pflegen.

Tabelle 3.5: Beschreibung der Anforderungen 'Could Have' und 'Out of Project Scope'

3.2.4 Use Case Diagramm

Die Abbildung 3.1 zeigt ein UML Use Case Diagramm nach Larman [22]. Es gibt einen Überblick über alle Use Cases und die damit interagierenden Akteure.

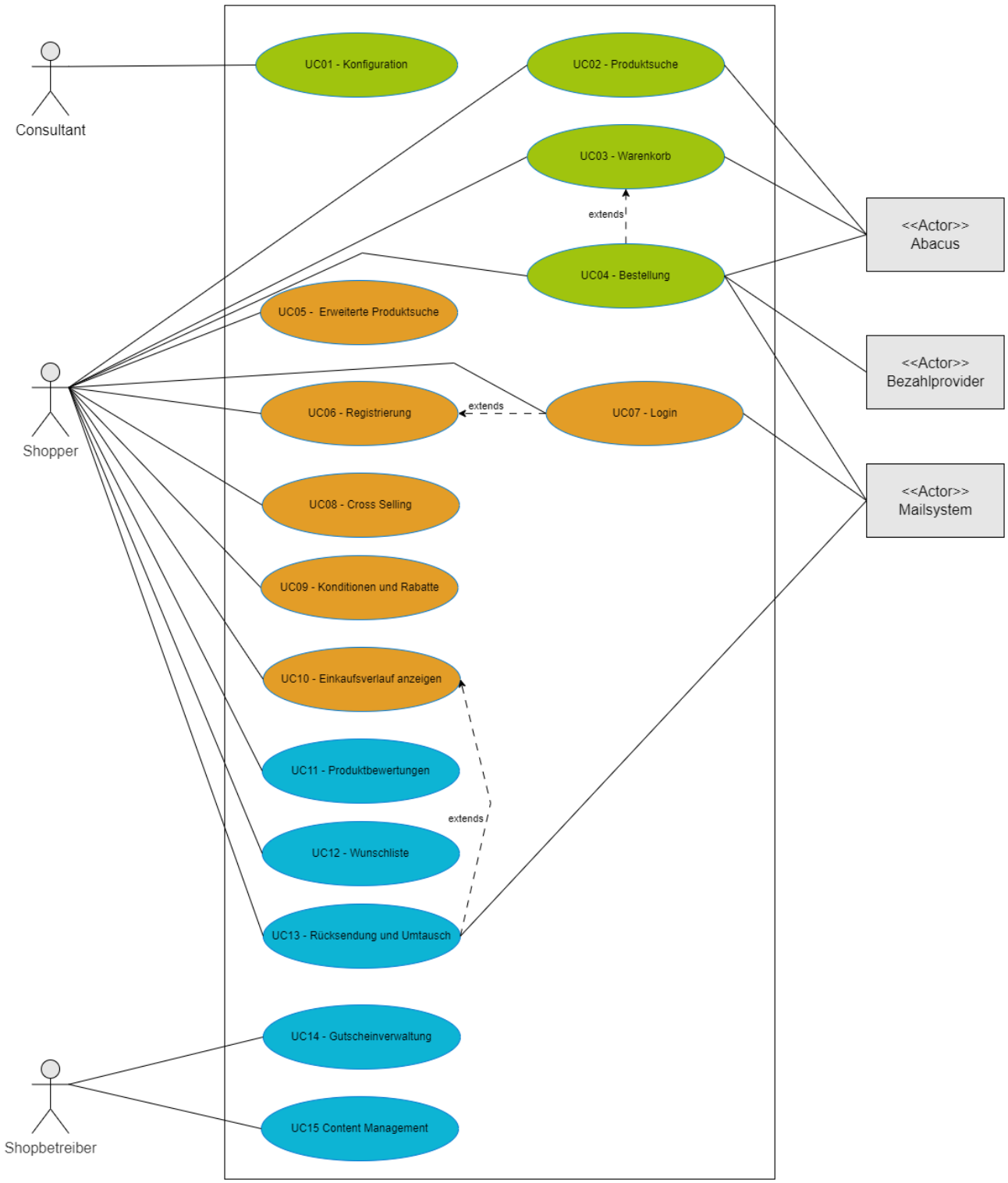

Abbildung 3.1: Use Case Diagramm (UML)

3.3 Nicht-Funktionale Anforderungen

Die folgenden nicht-funktionalen Anforderungen (NFR) legen spezifische Qualitätsstandards für den Webshop fest. Diese Anforderungen sind gemäss dem ISO Standard 25010 in verschiedene Kategorien und Unterkategorien unterteilt [19].

Für die Priorisierung der identifizierten NFR wird die nachstehende Skala verwendet:

1. Hohe Priorität (MUSS)
2. Mittlere Priorität (SOLLTE)
3. Niedrige Priorität (KÖNNTE)

Zusätzlich zu den oben genannten Kategorien und Unterkategorien wurden die nicht-funktionalen Anforderungen, basierend auf der Entscheidung, ob eine Eigenentwicklung durchgeführt oder ein bestehendes System genutzt wird, in drei Hauptgruppen unterteilt: In der Gruppe 'Allgemein' wurden die generell gültigen Anforderungen erfasst. Die Gruppe 'Eigenentwicklung' bezieht sich auf den selbst produzierten Code, und unter 'Fremdsystem' werden konkrete nicht-funktionale Anforderungen an ein potentielles Fremdsystem gestellt. Diese Gruppierung erleichtert die spezifische Betrachtung und Anpassung der Anforderungen je nach gewähltem Entwicklungsweg.

3.3.1 Allgemein

In den Tabellen 3.6 bis 3.8 sind Anforderungen aufgeführt, die sowohl bei einer Eigenentwicklung als auch bei der Integration einer bestehenden E-Commerce-Lösung gelten.

Performance

Die Performance eines Webshops ist ein wichtiges Qualitätsmerkmal für den Shopbetreiber, da sie direkten Einfluss auf den Umsatz hat [39]. Es ist schwierig, eine allgemeine Aussage über die Akzeptanzkriterien zu treffen, da die Kunden der Customize kleine bis sehr grosse Webshopshops mit tausenden von Artikeln betreiben. Die in Tabelle 3.6 definierten Messkriterien beziehen sich auf eine Webshop-Instanz, die auf Basis des Abacus-Mustermandanten erstellt wurde.

ID	NFR01
Kategorie	Qualität
Unterkategorie	Zeitliches Verhalten
Szenario	Der Shopper erwartet eine schnelle und flüssige Interaktion mit der Applikation, ohne lange Wartezeiten.
Auslöser	Der Shopper navigiert durch die Applikation.
Reaktion	Die Applikation antwortet prompt und ohne spürbare Verzögerung.
Messkriterium	Bei einem Anwender ohne Cache sollte ein Pageload nicht länger als 3 Sekunden dauern. Die Lösung sollte in der Google Light House Bewertung in der Kategorie 'Performance' mindestens 80 von 100 Punkten erreichen [14].
Priorität	Hoch

Tabelle 3.6: Beschreibung NFR01:Qualität (Zeitliches Verhalten)

Mobilfähigkeit und Responsivität in Webshops

Da ein bedeutender Anteil des Webshop-Traffics über mobile Endgeräte generiert wird [32], ist es essenziell, dass Webshops optimal für die Nutzung durch mobile Geräte ausgelegt sind. Dazu gehört die Gewährleistung einer reibungslosen Funktionalität und einer korrekten Darstellung des Webshops auf verschiedenen Gerätegrössen.

ID	NFR02
Kategorie	Qualität
Unterkategorie	Benutzerfreundlichkeit
Szenario	Shopper greifen über verschiedene Geräte, wie Smartphones oder Computer, auf den Shop zu.
Auslöser	Interaktion des Shoppers mit der Applikation.
Reaktion	Der Webshop passt seine Bedienelemente entsprechend der Bildschirmgrösse und Geräteart an.
Messkriterium	Verifikation mittels Usertests.
Priorität	Hoch

Tabelle 3.7: Beschreibung NFR02: Qualität (Benutzerfreundlichkeit)

Search Engine Optimization

Um eine Website in Suchmaschinen gut auffindbar zu machen, gibt es eine Vielzahl an Techniken und Best Practices für die Search Engine Optimization (SEO). Es ist nicht das Ziel dieser Arbeit, aufzuzeigen, wie eine optimale SEO erreicht werden kann, da dies auch stark vom Inhalt des Webshops abhängt. Dieses NFR soll jedoch die Basis für SEO legen, was hauptsächlich folgendes beinhaltet:

- **Markup-Struktur:** Die Verwendung von semantischen HTML-Elementen im Markup.
- **Lesbare URLs:** Der Uniform Resource Locator (URL) sollte lesbar sein.
  Beispielsweise ist die URL `https://buerobedarf.ch/ablagebox` besser für die SEO-Optimierung geeignet als `https://buerobedarf.ch/product/1`.

ID	NFR03
Kategorie	Benutzerfreundlichkeit
Unterkategorie	Leichter Zugang
Szenario	Ein Shopper sucht online nach einem Produkt. Der Webshop soll prominent in den Suchergebnissen erscheinen, um eine optimale Sichtbarkeit und Reichweite zu gewährleisten.
Auslöser	Online-Suche des Shoppers nach einem Produkt oder Webshop.
Reaktion	Der Webshop erscheint unter den Top-Suchergebnissen durch eine optimierte Search Engine Optimization (SEO).
Messkriterium	Das System sollte in der Google Light House [14] Auswertung in der Kategorie SEO mindestens 80/100 Punkte erzielen.
Priorität	Mittel

Tabelle 3.8: Beschreibung NFR03: Benutzerfreundlichkeit (Leichter Zugang)

3.3.2 Anforderungen an Eigenentwicklungs-Code

Der entwickelte Code unterliegt einer grösseren Kontrolle als externer Code. Aus diesem Grund gelten für den entwickelten Code die folgenden zusätzlichen NFR, die in den Tabellen 3.9 bis 3.10 dargestellt sind.

Testing
Um die zukünftige Erweiterbarkeit des entwickelten Codes zu erleichtern, ist es notwendig diesen durch Unit-Tests zu verifizieren. Diese Tests sollten so gestaltet sein, dass sie schnell ausgeführt werden können, um den Entwicklern zu ermöglichen, sie regelmässig durchzuführen. Dadurch wird sichergestellt, dass der Code kontinuierlich auf seine Funktionalität und Zuverlässigkeit hin überprüft wird. Die Testabdeckung soll dabei mit DotCover verifiziert werden [20].

ID	NFR04
Kategorie	Qualität
Unterkategorie	Prüfbarkeit
Szenario	Der Code soll stabil und zuverlässig sein.
Auslöser	Der Shop ist in Betrieb und wird weiterentwickelt.
Reaktion	Durch Unit Tests wird die korrekte Funktionalität des Codes sichergestellt.
Messkriterium	Mindestens 70% des Codes sind durch Unit Tests abgedeckt.
Priorität	Mittel

Tabelle 3.9: Beschreibung NFR04: Qualität (Prüfbarkeit)

Zukunftssicherheit
Angesichts der schnellen Entwicklung im E-Commerce-Bereich ist es wichtig, gegen Schnittstellen anstelle von Implementationen zu programmieren. Dieser Ansatz ermöglicht eine grössere Flexibilität und erleichtert den Austausch oder die Aktualisierung von Implementierungen in der Zukunft, ohne die Gesamtstruktur des Systems zu beeinträchtigen.

ID	NFR05
Kategorie	Qualität
Unterkategorie	Wartbarkeit
Szenario	Der Webshop soll erweitert werden.
Auslöser	Implementierung neuer Zahlungsmethoden.
Reaktion	Dank modularem Code ist eine einfache Erweiterbarkeit gewährleistet.
Messkriterium	Manuelle Kontrolle des Repositories.
Priorität	Hoch

Tabelle 3.10: Beschreibung NFR05: Qualität (Wartbarkeit)

3.3.3 Externes Shopsystem

Bei der Integration eines bestehenden E-Commerce-Systems ergeben sich zusätzliche nicht-funktionale Anforderungen (NFR). Diese spezifischen Anforderungen berücksichtigen die Eigenheiten eines bereits etablierten Softwaresystems und passen daher nicht in das bisher genutzte Schema zur Definition von NFRs. Aus diesem Grund wurde eine separate tabellarische Darstellung für diese besonderen Anforderungen verwendet, die in den Tabellen 3.11 bis 3.15 dargestellt sind. Sie sind so formuliert, dass sie eindeutig mit 'Ja' oder 'Nein' beantwortet werden können, wodurch eine weitere Messbarkeit entfällt.

ID	NFR06
Titel	Lizenzierung und Verfügbarkeit
Anforderung	Das Shopsystem sollte als Open-Source-Projekt verfügbar sein und die Möglichkeit für eine kommerzielle Nutzung bieten.
Nutzen	Der Source-Code ist öffentlich zugänglich und ermöglicht somit eine Anpassung aller Funktionalitäten vom System.
Priorität	Mittel

Tabelle 3.11: Beschreibung NFR06: Lizenzierung und Verfügbarkeit

ID	NFR07
Titel	Pluginsystem
Anforderung	Das System sollte ein Pluginsystem bereitstellen, um Erweiterungen und Logikanpassungen ohne Änderungen am Kerncode des Systems zu ermöglichen.
Nutzen	Durch die Trennung von Kernsystem und Erweiterungen können Updates des externen Systems effizienter und ohne Beeinträchtigung der individuellen Anpassungen durchgeführt werden.
Priorität	Hoch

Tabelle 3.12: Beschreibung NFR07: Pluginsystem

ID	NFR08
Titel	Event/Webhook
Anforderung	Das System sollte ein Event/Webhook-Mechanismus bereitstellen, um auf bestimmte Ereignisse reagieren zu können.
Nutzen	Durch dieses System können bei bestimmten Ereignissen, wie z.B. einer neuen Bestellung, individuelle Aktionen wie die Übermittlung an Abacus automatisch ausgelöst werden.
Priorität	Hoch

Tabelle 3.13: Beschreibung NFR08: Event/Webhook

ID	NFR09
Titel	Architektur
Anforderung	Die Architektur des Systems sollte den Zugriff auf Entitäten über klar definierte Services ermöglichen.
Nutzen	Ein solcher Ansatz gewährleistet eine strukturierte und konsistente Interaktion mit den Entitäten und fördert die Wartbarkeit und Skalierbarkeit des Systems.
Priorität	Hoch

Tabelle 3.14: Beschreibung NFR09: Architektur

ID	NFR10
Titel	Headless
Anforderung	Das System sollte in der Lage sein, headless betrieben zu werden, sodass es unabhängig von einer spezifischen Benutzeroberfläche funktioniert.
Nutzen	Durch die headless Nutzung kann das System flexibel mit verschiedenen Frontend-Lösungen integriert werden. Dies fördert die Anpassungsfähigkeit und ermöglicht eine breitere Kompatibilität mit unterschiedlichen Technologien und Plattformen.
Priorität	Mittel

Tabelle 3.15: Beschreibung NFR10: Headless Nutzung

Kapitel 4

Analyse

In diesem Kapitel wird die Vorarbeit für die Erarbeitung vom Lösungskonzept des Proof of Concept dokumentiert und begründet. In der Analyse der Schnittstelle von Abacus wird geprüft, ob mit dieser ein integrierter Webshop gemäss der in Kapitel 3 definierten Anforderungen umsetzbar ist. Nach der Evaluation verschiedener E-Commerce-Systeme wird entschieden, ob und mit welchem gearbeitet wird. Die Entscheidungen in diesem Kapitel hatten erheblichen Einfluss auf die Architektur vom System under Development (SuD), welche im Abschnitt 5.4 dokumentiert sind.

4.1 RESTful HTTP-Schnittstelle Abacus

Die Analyse wurde mit Hilfe vom Abacus API Hub gemacht [1]. Die Schnittstelle implementiert den Open Data Protocol (OData) Standard. OData ist ein offenes Protokoll, das einen Standard für die Implementation von Schnittstellen definiert [31]. Es ermöglicht die Erstellung und Nutzung von RESTful HTTP-Schnittstellen in einer einfachen und standardisierten Weise und unterstützt komplexe Abfragen, Filterung und Pagination. Ein Schlüsselelement von OData ist der '$metadata' Endpunkt, der das Datenmodell des Services in einem standardisierten XML-Format beschreibt. Dieses Modell liefert detaillierte Informationen über die Struktur der Daten, einschliesslich Typisierung der Antworten, Definitionen von Entitäten, deren Eigenschaften und Beziehungen.

4.1.1 Authentifizierung

Abacus ermöglicht eine tokenbasierte Authentifizierung gemäss dem standardisierten OAuth 2.0-Protokoll, wie in RFC6749 definiert [18]. Sowohl der Authorization Code Grant als auch der Client Credentials Grant werden unterstützt. Für die Integration in ein Webshopsystem ist jedoch nur der Client Credentials Grant relevant, da die API-Anfragen nicht an einen spezifischen Benutzer in Abacus ERP gebunden sein müssen. Listing 4.1 zeigt, wie ein Token abgefragt werden kann.

```
curl -X POST {base_url}/oauth/oauth2/v1/token \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "grant_type=client_credentials&client_id={}&client_secret={}"
```

Listing 4.1: cURL Befehl, um ein OAuth 2.0-Token unter Verwendung von Client Credentials zu erhalten.

Die Antwort auf diese POST-Anfrage ist ein Token, der zur Authentifizierung allen weiteren Endpoints als Bearer-Token übermittelt werden muss. Dieser Token hat eine Gültigkeitsdauer von 600 Sekunden. Nach Ablauf dieser Zeit muss ein neuer Token angefordert werden.

4.1.2 URL-Struktur

Der Uniform Resource Locator (URL) eines Endpunktes folgt dabei stets dem Aufbau gemäss Listing 4.2.

```
{base_url}/api/entity/{version}/mandanten/{mandantennummer}/{rest_ressource}
```

Listing 4.2: Aufbau URL Abacus Schnittstelle

- **base_url:** DNS-Name, unter dem Abacus via HTTP (Port 80) oder HTTPS (Port 443) erreichbar ist.
- **version:** Definiert die genutzte Schnittstellen-Version. Zum Zeitpunkt der Durchführung dieser Arbeit (12/2023) ist ausschliesslich v1 verfügbar.
- **mandatennummer:** Eine Abacus-Installation kann mehrere eigenständige Mandanten haben. Durch diesen Parameter kann der gewünschte Mandant angesteuert werden.
- **rest_ressource:** Die entsprechende Ressource, die abgefragt oder beschrieben wird.

In diesem Kapitel wird die darauf verzichtet, die gesamte URL auszuschreiben. Es wird jeweils die entsprechende '`rest_ressource`' als Endpoint aufgeführt.

4.1.3 Produkt

Stammdaten

Titel	Details
HTTP Methode	GET
Endpoint	`/ShopProducts`
Parameter	(ProductId=) um ein spezifisches Produkt abzufragen, z.B. `/ShopProducts(ProductId=3,VariantId=0)`
Beschreibung	Dieser Endpoint liefert alle für den Webshop freigegebenen Artikel.

Tabelle 4.1: Abacus API Produkt

Alle für einen Webshop relevanten Informationen werden über diesen Endpoint geliefert. Es fehlen die individuellen Attribute/Userfields, die jedoch durch einen zusätzlichen Aufruf verfügbar sind (Tabelle 4.3).

Produkt-Bilder

Titel	Details
HTTP Methode	GET
Endpoint	`/ShopProductAttachments`
Parameter	(ProductId=,VariantId=,Id=) um ein spezifisches Produktbild abzufragen, wobei die Id eine eindeutige Identifikation eines Bildes pro Produkt darstellt.
Beschreibung	Dieser Endpoint liefert die Metadaten der Produktbilder.

Tabelle 4.2: Abacus API Produktbilder

Das Bild selbst ist in der Antwort nicht enthalten und muss mit der OData-Action `/FileStream` separat abgefragt werden [30]. Dabei ist zu beachten, dass die Action immer nur auf einer spezifisch aufgerufen Ressource ausgeführt werden kann, weshalb der Paramter `(ProductId=,VariantId=,Id=)` zwingend ist. Listing 4.3 zeigt, wie ein Bild abgerufen werden kann.

```
curl -X GET /ShopProductAttachments(ProductId={},VariantId={},Id={})/FileStream
```

Listing 4.3: Abacus API Action: Produktbild als Stream

Userfields

Titel	Details
HTTP Methode	GET

Endpoints	`/ShopProductAdditionalField`
	`/ShopProductAdditionalFreeField`
	`/ShopProductTextAdditionalFields`

Parameter	(ProductId=) um Userfields von einem spezifischen Produkt abzufragen, z.B. `/ShopProductAdditionalField(ProductId=3)`
Beschreibung	Diese Endpoints liefern die individuell definierten Attribute der Produkte.

Tabelle 4.3: Abacus API Userfield

Abacus bietet drei Möglichkeiten Userfields auf Produkten zu erfassen. Diese können entweder in der Abacus-Datenbanktabelle ADO (AdditionalField), IBF (AdditionalFreeField) oder AFO (TextAdditionalFields) erfasst werden. Zwischen ADO und IBF gibt es keinen Unterschied, wobei die meisten Kunden der Customize AG die ADO verwenden In der Abacus-Tabelle AFO sind übersetzte Attribute gespeichert, weshalb dort in der Response anstelle eines Objekts ein Array von Objekten zurückgegeben wird. Im JSON-Auschnitt 4.4 ist ein Beispiel so einer Antwort aufgeführt.

```
"ShopProductTextAdditionalFields": [
    {
        "@odata.etag": "",
        "ProductId": 3,
        "VariantId": 0,
        "Language": "de",
        "UserFields": {
            "UserField1": "2 Jahre Garantie inkludiert",
            "UserField2": "Dies ist ein Demotext in Deutsch"
        }
    }
]
```

Listing 4.4: Abacus API Response: übersetzte Attribute

Die Herausforderung bei dieser Antwort liegt darin, dass die Schlüssel in der Form 'Userfieldx' enthalten sind, wobei 'x' aufsteigend nummeriert wird. In Abacus erhalten solche Attribute jedoch eigene Namen. Daher sollte der Eintrag nicht als `"UserField1": "2 Jahre Garantie inkludiert"`, sondern als `"Garantie": "2 Jahre Garantie inkludiert"` formuliert werden.

Über den OData Endpoint '`/$metadata`' kann ein XML bezogen werden, welches den Aufbau dieser Userfields beschreibt.

```
<Property Name="UserField1" Type="String" MaxLength="50">
    <Annotation Term="Description">
        <String>Garantie</String>
        <Annotation Term="IsLanguageDependent">
            <Bool>false</Bool>
        </Annotation>
    </Annotation>
</Property>
```

Listing 4.5: Abacus API Response: Metadaten der Attribute

Durch diese Umgehungslösung ist es möglich, den korrekten Namen des Userfields zu erhalten. Allerdings erhält man dadurch nur die Attributbezeichnung in Deutsch. In Abacus werden die Bezeichnungen vom Userfields jedoch mehrsprachig geführt.
Zurzeit gibt es keine adäquate Lösung, um mit Radio Userfields umzugehen. Diese sind Userfields mit Auswahllisten bestehend aus Key-Value-Paaren, wobei in der Datenbank der Entität jeweils der Key des ausgewählten Elements gespeichert wird. Die Schnittstelle liefert in diesem Fall den in der Datenbank gespeicherten Key, wobei für die Darstellung im Webshop eigentlich der Value benötigt wird.

Radio Userfields

Bei den Artikeln der Bürobedarfs AG soll ein Attribut für die Garantiebestimmungen des Produkts geführt werden. Diese Bestimmungen liegen als vordefinierte Liste vor und werden daher als Radio-Auswahlliste im Userfield bereitgestellt. Abbildung 4.1 zeigt, wie eine Radio Auswahlliste in Abacus erfasst ist. Der Vorteil gegenüber einem Text-Userfield ist, dass hier fix vordefiniert werden kann, welche Werte möglich sind. Dadurch können Erfassungsfehler von Mitarbeitern der Bürobedarfs AG minimiert werden.

WERT	DEUTSCH	...	ENGLISCH
1	keine Garantie		no guarantee
2	2 Jahre Garantie		2 years guarantee
3	4 Jahre Garantie		4 years guarantee

Abbildung 4.1: Beispiel Radio Auswahlliste

Atkuell retourniert die Schnittstelle in diesem Fall die Spalte Wert anstelle der Spalten Deutsch/Englisch.

Diese beiden Herausforderungen wurden der Abacus Research AG gemeldet und befinden sich zum Zeitpunkt dieser Studienarbeit (12/2023) in der Umsetzungsphase.

4.1.4 Klassierung

Definition

Titel	Details
HTTP Methode	GET
Endpoint	`/ShopProductClassificationDefinitions`
Parameter	-
Beschreibung	Dieser Endpoint liefert alle für den Webshop erstellten Klassierungsdefinitionen zurück.

Tabelle 4.4: Abacus API Klassierungsdefinition

Die Antwort enthält ein Array von Klassierungsdefinitionen, da man in Abacus mehrere verschiedene Klassierungen führen kann. Relevant ist hier nur die gelieferte Id, da diese als Einstiegspunkt benötigt wird, um die Baumstruktur der jeweiligen Klassierung abzufragen.

Klassierungselemente

Titel	Details
HTTP Methode	GET
Endpoint	`/ShopProductClassificationElements`
Parameter	(Id=) um ein spezifisches Element der Klassierung abzufragen
Beschreibung	Dieser Endpoint liefert alle Elemente der Klassierungsdefinition zurück.

Tabelle 4.5: Abacus API Klassierungselemente

In dieser Antwort wird die gesamte Baumstruktur der Produktklassierung aufgebaut. Das Listing 4.6 zeigt einen gekürzten JSON-Ausschnitt, wie ein Klassierungselement retourniert wird.

```
"Id": "14c37b0d-cd28-8fb6-9c8c-f0fc46b8466b",
"DefinitionId": "332328b8-758e-27b7-08a7-0da67bda45c6",
"ParentId": "b8dd4dbe-461d-a527-4493-76743cf6f6eb"
```

Listing 4.6: Abacus API Klassierung Baumstruktur

Mithilfe der '`ParentId`' referenziert ein Element sein Elternelement. Die '`DefinitionId`' verweist auf die Definition, die im ersten Schritt geladen wurde (Tabelle 4.4).

Produktzuordnungen

Titel	Details
HTTP Methode	GET
Endpoint	`/ShopProductClassificationDetails`
Parameter	-
Beschreibung	Dieser Endpoint liefert alle Produktzuordnungen zu den Klassierungselementen zurück.

Tabelle 4.6: Abacus API Produktzuordnungen

Die Antwort enthält ein Array von Zuordnungen zwischen Produkt und Klassierungselement. Im Listing 4.7 befindet sich eine gekürzte Antwort einer einzelnen Produktzuordnung.

```
"Id": "3e3bf72f-a809-48c2-f14a-bef7153bf52c",
"DefinitionId": "332328b8-758e-27b7-08a7-0da67bda45c6",
"ParentId": "6640c9d0-6ff1-847f-dac7-bc4cfad5dffd",
"Number": 3
```

Listing 4.7: Abacus API Klassierung Produktzuordnung

Relevant ist hier die '`Number`', die für die Produktnummer steht und mithilfe der '`ParentId`' auf das Klassierungselement verweist. Es ist möglich, dass ein Produkt mehreren Klassierungselementen zugeordnet ist. In diesem Fall gibt es mehrere dieser Objekte mit dem gleichen Wert unter `Number`.

Klassierungs-Bilder

Titel	Details
HTTP Methode	GET
Endpoint	`/ProductClassificationAttachments`
Parameter	(ClassificationId=,Language=,SortOrder=) um ein spezifisches Klassierungsbild abzufragen, wobei die `SortOrder` zusammen mit `Language` eine eindeutige Identifikation eines Bildes pro Klassierungselement darstellt.
Beschreibung	Dieser Endpoint liefert die Metadaten aller Klassierungsbilder.

Tabelle 4.7: Abacus API Klassierungsbilder

Der Mechanismus ist identisch wie bei den Produktbildern (Tabelle 4.2). Die Entität '`ProductClassificationAttachments`' stellt ebenfalls eine OData-Action '`FileStream`' zur Verfügung.

4.1.5 Subscriptions

Die Subscriptions Application Programming Interface (API) ermöglicht es Konsumenten, mit Benachrichtigungen über Änderungen an Stammdaten informiert zu werden. Diese API arbeitet nach dem Muster des Publish-Subscribe-Prinzips, welches in der Fachliteratur oft als 'Publish-Subscribe-Pattern' bezeichnet wird [17]. Abacus agiert dabei als Publisher und veröffentlicht Nachrichten über Änderungen an Stammdaten in einem spezifischen Topic. Konsumenten, die an diesen Updates interessiert sind, können sich als Subscriber für diese Topics registrieren. Nach der Registrierung können sie ihre Nachrichten abholen, wenn es Änderungen an den Stammdaten gibt, die für das abonnierte Topic relevant sind. In Abbildung 4.2 wird die Funktionsweise von diesem Pattern visualisert.

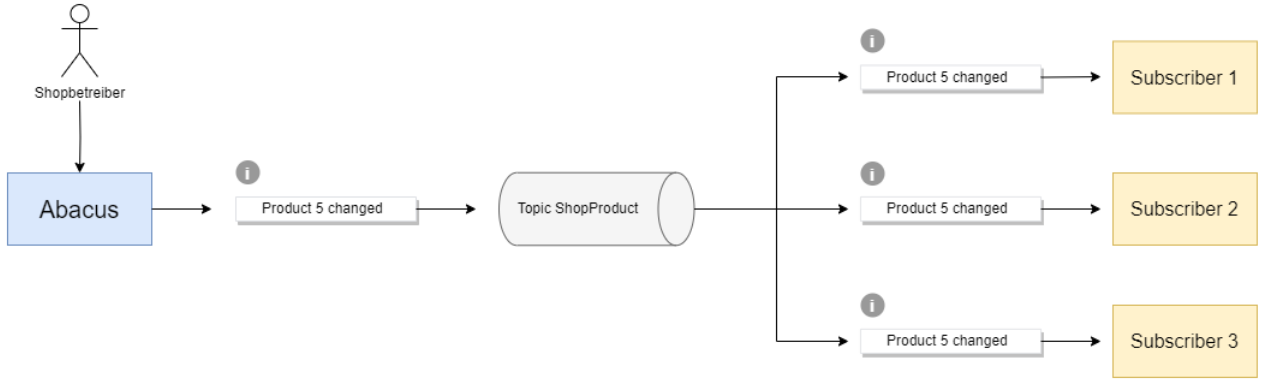

Abbildung 4.2: Publish-Subscribe Pattern

Es ist zu beachten, dass der Konsument die Nachrichten nicht automatisch über einen Webhook zugestellt bekommt [4]. Stattdessen werden die Nachrichten durch einen expliziten Aufruf der API in einem vom Konsumenten selbst definierten Intervall abgerufen. Diese Vorgehensweise wird in den nachfolgenden Abschnitten detaillierter erläutert.

Aktivierung

Titel	Details
HTTP Methode	POST
Endpoint	`/subscribe/beispiel_name/topic`
Parameter	Kein Body im Post-Request erforderlich.
Beschreibung	Mit diesem Endpoint kann man sich für eine Subscription anmelden. Der Parameter '`beispiel_name`' kann durch einen beliebigen Namen ersetzt werden. Dieser Name wird später benötigt, um Änderungen abzurufen und entspricht der ID dieser Subscription. Als Topic können verschiedene Bereiche ausgewählt werden, in denen dann über Änderungen informiert werden.

Tabelle 4.8: Abacus API Subscription Anmeldung

Konsumation

Titel	Details
HTTP Methode	GET
Endpoint	`/consume/beispiel_name`
Parameter	Als '`beispiel_name`' wird der Name verwendet, der bei `subscribe` (Tabelle 4.8) festgelegt wurde.
Beschreibung	Mit diesem Endpoint können die aktuellen Änderungen einer Subscription abgerufen werden.

Tabelle 4.9: Abacus API Subscription Konsum

Mit diesem Aufruf kann der Subscriber alle offenen Nachrichten auf einem Topic abholen. Die Antwort auf eine Konsumation ist nachfolgend im Listing 4.8 aufgeführt.

```
"messages": [
    {
        "topic": "Product",
        "event": "Update",
        "timestamp": "2023-10-06T10:49:54.355Z",
        "payload": {
            "ProductId": 3
        }
    }
],
"acknowledgeKey": "ae99d6c8-6435-11ee-913d-c49deda99d6c"
```

Listing 4.8: Abacus API Subscription Consume

Wie die Beispielantwort zeigt, sieht ein Konsument in diesem Fall lediglich, dass ein Artikel mutiert wurde, ohne jedoch die spezifischen Änderungen zu erfahren.
Als Event können die Werte Create, Update oder Delete zurückgegeben werden. Pro Anfrage an den Endpoint erhält man maximal 1000 Nachrichten. Dies bedeutet, dass der Konsument diesen Aufruf mehrfach machen muss.

Bestätigung

Titel	Details
HTTP Methode	POST
Endpoint	`/acknowledge/beispiel_name/key`
Parameter	Als '`beispiel_name`' wird der Name verwendet, der bei `subscribe` (4.8) festgelegt wurde. Der Schlüssel '`key`' wird aus der Antwort des Konsums (4.9) im Feld '`acknowledgeKey`' entnommen.
Beschreibung	Mit diesem Endpoint können die konsumierten Änderungen bestätigt werden.

Tabelle 4.10: Abacus API Subscription Bestätigung

Nach einem erfolgreichen Konsum sollte dieser Endpoint aufgerufen werden, da erst in diesem Moment die Nachrichten auf dem Topic für den aufrufenden Subscriber entfernt werden. Andernfalls werden bei der nächsten Konsumanfrage dieselben Änderungen erneut angezeigt. Dies stellt sicher, dass der Konsument bei einem Fehler die Nachrichten erneut empfangen kann.
Abbildung 4.3 zeigt den gesamtem Prozess der Subscription API als UML Sequenzdiagramm aus Sicht vom Konsumenten.

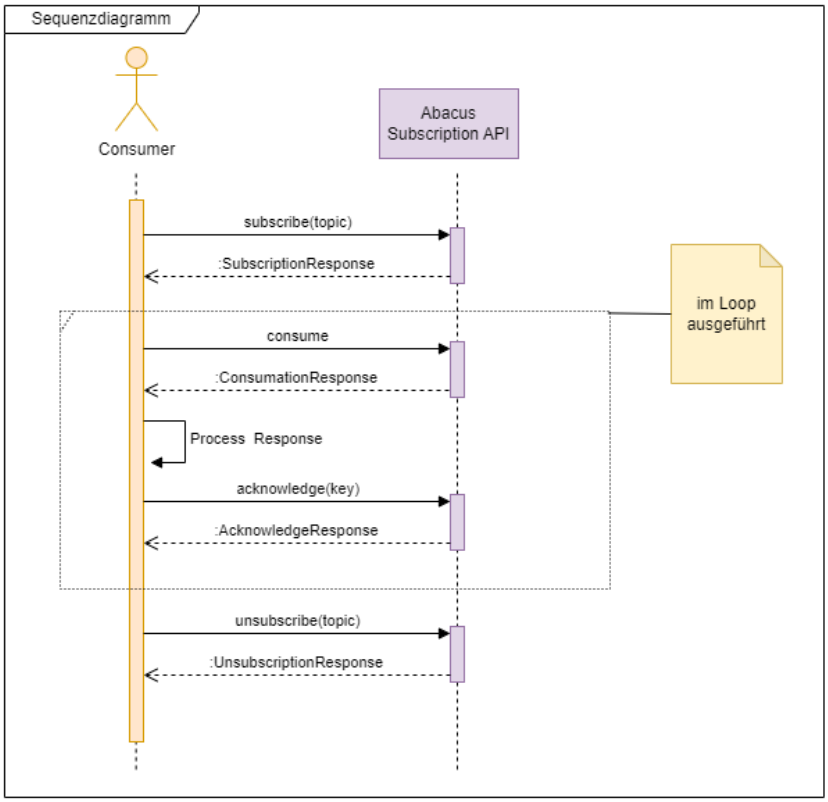

Abbildung 4.3: Prozess Subscription API (UML Sequenzdiagramm)

4.1.6 Verkaufsauftrag

Erstellung

Titel	Details
HTTP Methode	POST
Endpoint	`/CreateSalesOrderInbox`
Beschreibung	Dieser Endpoint ermöglicht das Erstellen eines Eintrages in der Inbox.

Tabelle 4.11: Abacus API Verkaufsauftrag Erstellung

Gemäss Ausführungen in Abschnitt 2.4.5 werden Bestellungen nicht direkt als Verkaufsaufträge in Abacus eingetragen. Stattdessen wird eine Nachricht in der E-Business Inbox erzeugt, die anschliessend in Verkaufsaufträge umgewandelt wird. Die Inbox-Schnittstelle ist sehr umfangreich, weshalb im Rahmen dieser Studienarbeit ein Verkaufsauftrag mit den minimal notwendigen Informationen erstellt wurde. Ein Beispiel für einen solchen minimalen Body ist im Anhang unter B.1 zu finden.

Warenkorb

Titel	Details
HTTP Methode	POST
Endpoint	`/FindProductsPriceShoppingCart`
Beschreibung	Dieser Endpoint ermöglicht es, einen Warenkorb an Abacus zu senden, um eine Preisfindung durchzuführen.

Tabelle 4.12: Abacus API Warenkorb Preisfindung

Da Abacus über einen komplexen Preisfindungsmechanismus verfügt, welcher in einem Webshop nicht vollständig nachgebildet werden kann, sollte diese Schnittstelle verwendet werden. Sie ermöglicht es, einen Warenkorb an Abacus zu senden, um eine finale Preis- und Rabattfindung durchzuführen. Ein Beispiel für einen Body und die entsprechende Antwort ist im Anhang unter B.2 zu finden.

4.1.7 Gesamtübersicht der Schnittstellen

Ein Ziel dieser Arbeit war es, durch die ausgewählten Use Cases einen Proof of Concept zu erstellen, der die Abacus API möglichst breit abdeckt.
Die folgende Tabelle zeigt, welche Endpunkte der Schnittstelle für die Erfüllung der ausgewählten Use Cases benötigt wurden:

Use Case	**Endpoints**
UC01 - Konfiguration	`/token` Bei der Konfiguration wird ein Token abgerufen, um einen Healthcheck auszuführen und die korrekte Konfiguration zu bestätigen.
UC02 - Produktsuche	`/ShopProducts`, `/ShopProductAttachments`, `/ShopProductAdditionalField`, `/ShopProductAdditionalFreeField`, `/ShopProductTextAdditionalFields` , `/$metadata`, `/ShopProductClassificationDefinitions`, `/ShopProductClassificationElements`, `/ShopProductClassificationDetails`, `/ShopProductAttachments`, `/subscribe`, `/consume`, `/acknowledge` Neben allen Schnittstellen für Produkt- und Klassifizierungsdaten wurde die Subscriptions-API benutzt.
UC03 - Warenkorb	`/FindProductsPriceShoppingCart`
UC04 - Bestellung	`/CreateSalesOrderInbox`

Tabelle 4.13: Gesamtübersicht der Abacus API

Um eine fundierte Aussage darüber treffen zu können, ob die neue RESTful HTTP-Schnittstelle die in Kapitel 3 gestellten Anforderungen erfüllen kann, wurde auf theoretischer Basis untersucht, ob auch die Use Cases, die im Proof of Concept nicht umgesetzt wurden, implementiert werden können. In Tabelle 4.14 wird dargestellt, welche zusätzlichen Endpunkte für einen MVP-Webshop notwendig sind. Es ist nicht auszuschliessen, dass bei der Umsetzung der Use Cases noch weitere Endpunkte dazukommen.

Use Case	**Endpoints**
UC05 - Erweiterte Produktsuche	Die Userfields werden bereits durch UC02 - Produktsuche angeliefert. Da es in Abacus keine Möglichkeit gibt, ein Userfield als filterbar zu markieren, muss die Implementation im E-Commerce-System passieren.

UC06 - Registrierung	`/ShopperAccounts` Über diesen Endpunkt können neue registrierte Shopper in die E-Business Inbox geschrieben werden, wo sie dann einer Adresse in Abacus zugeordnet werden.
UC07 - Login	Wird im Webshop ohne Abacus abgehandelt.
UC08 - Cross Selling	`/ShopRelatedProducts` Deser Endpunkt gibt die verwandten Artikel von einem Produkt aus.

UC09 - Konditionen und Rabatte	`/VariantDimensions`
	`/VariantDimensionsItems`
	`/ExchangeRates`
	Um einem Shopper die korrekten Konditionen und Rabatte anzuzeigen, kann die Schnittstelle aus UC03 - Warenkorb genutzt werden. Dabei wird für den geöffneten Artikel, jeweils mit der Menge 1, ein Warenkorb erstellt und an Abacus geschickt. In einem Webshop, der mehrere Währungen unterstützt, ist es notwendig, über den Endpunkt `ExchangeRates` die aktuellen Wechselkurse abzufragen, um eine korrekte Preisdarstellung zu gewährleisten. Für die Anzeige der Konditionen bei konfigurierbaren Artikeln ist es zudem erforderlich, die Informationen zu den jeweiligen Varianten zu beziehen (`VariantDimensions` und `VariantDimensionsItems`).

UC10 - Einkaufsverlauf anzeigen	`/OrderProcessingShopStatus`
	`/OrderProcessingStatusDocuments`
	Damit der Status einer Bestellung abgerufen werden kann, gibt es den Endpunkt `OrderProcessingShopStatus`. Der zusätzliche Endpunkt `OrderProcessingStatusDocuments` erlaubt es ausserdem, zu einem Status ein gewünschtes Dokument anzuzeigen (beispielsweise ein Lieferschein).

Tabelle 4.14: Abacus API für das MVP

Die RESTful-API von Abacus ist um ein Vielfaches umfangreicher und deckt Bereiche ab, die für einen Webshop keinen Mehrwert bieten. In dieser Arbeit wurde die Schnittstelle nur aus der Sicht eines E-Commerce-Systems beleuchtet. Es ist nicht auszuschliessen, dass bei der Umsetzung der Use Cases noch weitere Endpunkte dazukommen.

4.1.8 Resultat

Ein entscheidender Erfolgsfaktor dieser Studienarbeit ist die Bewertung der Eignung der Schnittstelle als potenzielle Nachfolgelösung für den AbaShop. Nach einer umfassenden Analyse in diesem Kapitel lässt sich bestätigen, dass alle für einen Minimum Viable Product (MVP) Webshop erforderlichen Use Cases realisierbar sind. Da die benötigten Schnittstellen für UC05 bis UC10 in dieser Arbeit nicht implementiert wurden, kann nicht ausgeschlossen werden, dass bei deren Umsetzung Herausforderungen entstehen könnten. Da das MVP alle Features umfasst, die der AbaShop bisher bot, ist es möglich, dass Customize seinen Kunden mit den neuen RESTful HTTP-Schnittstellen von Abacus eine adäquate Nachfolgelösung anbietet.
Einzig im Bereich der Userfields, wie im Abschnitt 4.1.3 dokumentiert, besteht noch Optimierungsbedarf. Da der Softwarehersteller über diese Anforderung informiert ist und entsprechende Anpassungen zugesagt hat, wird dies im Rahmen dieser Arbeit als erfüllt angesehen.

4.2 Evaluation externe Shopsysteme

Gemäss der definierten Aufgabenstellung (siehe Anhang D) ist eine Überprüfung bestehender E-Commerce-Systeme erforderlich. Das primäre Ziel dieser Überprüfung ist es, zu evaluieren, ob die Entwicklung eines Connectors für ein bereits existierendes System ausreichend ist oder ob ein vollständiger Neuaufbau des E-Commerce-Grundgerüsts notwendig wird.

4.2.1 Vorgehen

Die Grundlage wurde durch eine umfassende Recherche zum Thema 'E-Commerce-Systeme' gelegt. Basierend auf den in Abschnitt 3.2 definierten Use Cases und den nicht funktionalen Anforderungen aus Abschnitt 3.3 sowie weiteren Gesprächen mit dem Industriepartner wurde ein Kriterienkatalog erstellt. Mit Hilfe dieses Katalogs und einer anschliessenden Nutzwertanalyse konnte das am besten geeignete System ermittelt werden. Das ausgewählte System wurde daraufhin einer detaillierten Analyse unterzogen, um sicherzustellen, dass sowohl die funktionalen als auch die nicht-funktionalen Anforderungen erfüllt werden können. Auf Grundlage dieser Analyse wurde schliesslich entschieden, ob eine Eigenentwicklung vorgenommen oder ein Connector eingesetzt wird (siehe Abschnitt 4.2.11).

4.2.2 Kriterien

Die folgende Tabelle 4.15 zeigt die Evaluationskriterien, die in der Analyse der Shopsysteme und der anschliessenden Nutzwertanalyse in Abschnitt 4.2.8 verwendet wurden.

Kriterium	**Beschreibung**
Kosten / Lizenzierung	Die Kosten für die Softwarenutzung. Bei Open Source-Projekten sollte geprüft werden, inwiefern das System kommerziell genutzt werden kann.
Erweiterbarkeit	Es ist zu klären, wie die Erweiterung der Backendlogik in das Shopsystem integriert werden kann, sei es durch ein Plugin oder durch die Erweiterung der Codebasis bei Open-Source-Projekten. Dabei ist eine Abschätzung des Aufwands für eine solche Erweiterung von grosser Bedeutung. Ein Beispiel hierfür könnte die Implementierung eines Bewertungssystems gemäss UC11 - Produktbewertungen sein. Zudem sollte das System eine headless Nutzung ermöglichen, sodass ein eigenes Frontend implementiert werden kann.
Community	Grösse und Aktivität der Community. Existiert beispielsweise ein Forum, in dem Unterstützung angeboten wird? Falls das Projekt Open Source ist, sollte die Aktivität des Repositories überprüft werden.
Umfang der Features	Abdeckungsgrad der benötigten Features: • **Mehrsprachigkeit:** Produktdaten, Adminbereich • **Währungen:** Mehrere Währungen nutzbar • **Mehrwertsteuer:** Gemäss Anforderungen der Schweiz [8] • **Versandkosten:** Individuelle Berechnung • **Kreditkartenzahlung:** Datatrans [7] oder Saferpay [43] • **Produktfotos** • **Individuelle Attribute:** Auf den Produkten • **Produktbeschreibung in HTML** • **Variantenprodukte** Konfigrierbare Artikel mit auswählbaren Variantenattributen (siehe Abschnitt 2.4.3) • **Preise:** Listenpreis, Aktionspreis, Staffelpreis • **Rabatte:** Standardrabatt, Aktionsrabatt, Staffelrabatt
Dokumentation	Qualität der Dokumentation.
Gesamteindruck	Dieses Kriterium berücksichtigt den Gesamteindruck der Plattform.

Tabelle 4.15: Evaluationskriterien für E-Commerce-Systeme

4.2.3 Shopify

Shopify ist eine der aufstrebenden Plattformen auf dem Markt und überzeugt vor allem durch seine Benutzerfreundlichkeit und umfassende Dokumentation. Die Software wird vom gleichnamigen kanadischen Unternehmen angeboten [37].

Kriterium	**Beschreibung**
Kosten	Nur als SaaS Lösung verfügbar. Basic: \$39 pro Monat plus 2.95% und \$0.30 pro Verkauf Shopify: \$105 pro Monat plus 2.75% und \$0.30 pro Verkauf Advanced: \$399 pro Monat plus 2.55% und \$0.30 pro Verkauf [36]
Erweiterbarkeit	Der Source Code von Shopify ist nicht öffentlich zugänglich. Erweiterungen werden neuerdings mit dem auf React basierenden Fullstack Framework Remix entwickelt [33]. Die Dokumentation zur Integration eines Plugins in den Adminbereich ist sehr knapp gehalten, was die Testintegration sehr aufwändig machte. Eine headless Nutzung ist dank der vorhandenen GraphQL Schnittstelle möglich [16].
Community	Ein Forum ist vorhanden und wird aktiv genutzt. Es ist jedoch stark anwenderorientiert und bietet weniger Inhalte für Entwickler.
Umfang der Features	• **Mehrsprachigkeit:** Vorhanden • **Währung:** Nur wenn Shopify Payments genutzt wird [38] • **Mehrwertsteuer:** Vorhanden • **Versandkosten:** Nach Gewicht vorhanden, andere Logiken benötigen ein Plugin • **Kreditkartenzahlung:** Durch ein Plugin des Marketplace möglich • **Produktfotos:** Vorhanden • **Individuelle Attribute:** Nur über Metafields möglich, diese sind jedoch nicht übersetzbar • **Produktbeschreibung in HTML:** Vorhanden • **Variantenprodukte:** Vorhanden • **Preise:** Listenpreis • **Rabatte:** Standardrabatt
Dokumentation	Die Dokumentation der API ist sehr gut, jedoch ist zum Zeitpunkt dieser Arbeit (12/2023) die Dokumentation der Pluginentwicklung mit Remix noch nicht abgeschlossen.

Gesamteindruck	Die Plattform punktet mit vielen Features und einer hohen Benutzerfreundlichkeit. Die Integration eigener Apps gestaltete sich jedoch als umständlich. Zudem sind die Kosten relativ hoch, und da Shopify ausschliesslich Closed Source als SaaS angeboten wird, sind Änderungen am Source Code nicht möglich.

Tabelle 4.16: Analyse von Shopify

4.2.4 WooCommerce

WooCommerce [42] ist mit über 15 Millionen Download die am weitesten verbreitete Plattform auf dem Markt [40]. Das als WordPress-Plugin verfügbare E-Commerce-System ist in PHP geschrieben und als Open Source Lösung erhältlich.

Kriterium	**Beschreibung**
Kosten	Open Source mit einer GPLv2 Lizenz.
Erweiterbarkeit	Da es sich um eine Open Source-Lösung handelt, kann der Source Code nach Bedarf angepasst werden. Das vorhandene Pluginsystem erlaubt es mittels Action Hooks das Handling von einem Request innerhalb von WordPress zu modifizieren. Stellt WordPress/WooCommerce für eine Action keinen Hook zur Verfügung, kann ein Plugin das Verhalten nicht anpassen. Eine Headless Nutzung ist mit einer vorhandenen RESTful HTTP-Schnittstelle möglich.
Community	Es existieren lediglich inoffiziell organisierte Community-Foren. Aufgrund der grossen Verbreitung sind jedoch viele Ressourcen verfügbar.
Umfang der Features	• **Mehrsprachigkeit:** Nur mit einem externen Plugin möglich • **Währung:** Vorhanden • **Mehrwertsteuer:** Vorhanden • **Versandkosten:** Nach Gewicht vorhanden • **Kreditkartenzahlung:** Nur mit einem externen Plugin möglich • **Produktfotos:** Vorhanden • **Individuelle Attribute:** Nur mit einem externen Plugin möglich • **Produktbeschreibung in HTML:** Vorhanden • **Variantenprodukte:** Vorhanden • **Preise:** Listenpreis • **Rabatte:** Standardrabatt

Dokumentation	Ein Dokumentation ist vorhanden, wobei für Entwickler lediglich Code Snippets zur Verfügung stehen.
Gesamteindruck	WooCommerce ist nur in Kombination mit WordPress nutzbar, wodurch der Kunde stark an ein umfangreiches Ökosystem gebunden ist. Die Plattform setzt zudem stark auf Plugins, sodass viele benötigte Features über externe Plugins hinzugefügt werden müssen. Ein Problem dabei ist, dass die Wartbarkeit dieser Plugins nicht immer garantiert ist.

Tabelle 4.17: Analyse von WooCommerce

4.2.5 Magento

Magento wurde 2008 als PHP Open Source Software veröffentlicht und 2018 von Adobe übernommen. Seitdem wird das System als Adobe Commerce bezeichnet. Die Software wird in zwei Editionen angeboten: Open Source (Community) und Enterprise [2]. Diese Analyse bezieht sich ausschliesslich auf die Community Edition.

Kriterium	**Beschreibung**
Kosten	Community: Open Source mit einer Open Software License (OSL) 3.0 Lizenz, Enterprise: Nicht öffentlich
Erweiterbarkeit	Da es sich um eine Open Source-Lösung handelt, kann der Source Code nach Bedarf angepasst werden. Das vorhandene Pluginsystem erlaubt das Verhalten von allen public Methoden anzupassen. Eine Headless Nutzung ist mit einer vorhandenen RESTful HTTP-Schnittstelle möglich.
Community	Es existieren lediglich inoffiziell organisierte Community-Foren. Aufgrund der grossen Verbreitung sind jedoch viele Ressourcen verfügbar.

Umfang der Features	• **Mehrsprachigkeit:** Vorhanden • **Währung:** Vorhanden • **Mehrwertsteuer:** Vorhanden • **Versandkosten:** Fehlt • **Kreditkartenzahlung:** Über ein Plugin vom Marketplace möglich • **Produktfotos:** Vorhanden • **Individuelle Attribute:** Fehlt • **Produktbeschreibung in HTML:** Vorhanden • **Variantenprodukte:** Vorhanden • **Preise:** Listenpreis • **Rabatte:** Standardrabatt
Dokumentation	Als Entwickler ist es schwierig, in der Dokumentation die gewünschten Informationen zu finden.
Gesamteindruck	Seit der Übernahme durch Adobe wird Magento stärker kommerzialisiert. Viele neue Features werden nur in der Enterprise-Version angeboten.

Tabelle 4.18: Analyse von Magento

4.2.6 nopCommerce

NopCommerce ist eine mit Microsoft-Technologien entwickelte Open Source Lösung [29]. Sie basiert auf ASP.NET Core und stellt ein Frontend mit Razor Pages zur Verfügung [26].

Kriterium	**Beschreibung**
Kosten	Open Source mit eigener Lizenz ('powered by nopCommerce' muss auf jeder Seite vorhanden sein)
Erweiterbarkeit	Da es sich um eine Open Source-Lösung handelt, kann der Source Code nach Bedarf angepasst werden. Ein vorhandenes Plugin-System ermöglicht Erweiterungen, ohne den Kerncode zu verändern. Eine headless Nutzung ist allerdings nicht direkt möglich, da keine API bereitgestellt wird. Es wäre jedoch denkbar, selbst eine Web-API zu implementieren.

Community	Das GitHub-Repository wird aktiv von einer breiten Community gepflegt. Das offizielle Forum ist stark auf Entwickler ausgerichtet und wird dementsprechend aktiv genutzt.
Umfang der Features	• **Mehrsprachigkeit:** Vorhanden • **Währung:** Vorhanden • **Mehrwertsteuer:** Vorhanden • **Versandkosten:** Nach Gewicht vorhanden, andere Logiken benötigen ein Plugin • **Kreditkartenzahlung:** Fehlt (kann eigenständig entwickelt werden) • **Produktfotos:** Vorhanden • **Individuelle Attribute:** Vorhanden • **Produktbeschreibung in HTML:** Vorhanden • **Variantenprodukte:** Vorhanden • **Preise:** Listenpreis • **Rabatte:** Standardrabatt
Dokumentation	Eine Dokumentation ist vorhanden, allerdings nicht sehr detailliert.
Gesamteindruck	NopCommerce bietet nahezu alle benötigten Features. Besonders überzeugend sind der verwendete Technologiestack, da der industriepartner ebenfalls mit Microsoft-Technologien arbeitet. Überzeugend ist auch die ausgezeichnete Architektur, die eigene Erweiterungen optimal unterstützt [28]. Diese Architektur folgt den Prinzipien der Clean Architecture, beschrieben in [23]. Ein Nachteil von NopCommerce ist jedoch, dass eine headless Nutzung ohne die Eigenentwicklung einer Web-API nicht möglich ist.

Tabelle 4.19: Analyse von nopCommerce

4.2.7 Medusa

Medusa ist eine aufstrebende E-Commerce Plattform, die das JavaScript-Ökosystem bedient. Das 2021 gegründete System zeichnet sich durch seine grosse Zahl an mitwirkenden Open Source Entwicklern, dem modularen Aufbau und ihren Headless-First Ansatz aus [24].

Kriterium	**Beschreibung**
Kosten	Open Source mit einer MIT Lizenz.
Erweiterbarkeit	Da es sich um eine Open Source-Lösung handelt, kann der Source Code nach Bedarf angepasst werden. Ein vorhandenes Plugin-System ermöglicht Erweiterungen, ohne den Kerncode zu verändern. Eine Headless Nutzung ist mit einer vorhandenen Restful HTTP und GraphQL Schnittstelle möglich.
Community	Das Projekt profitiert von einer enorm grossen Zahl an Open Source Entwicklern und hat dementsprechend eine grosse Community.
Umfang der Features	• **Mehrsprachigkeit:** Fehlt • **Währung:** Vorhanden • **Mehrwertsteuer:** Vorhanden • **Versandkosten:** Nach Gewicht vorhanden, andere Logiken benötigen ein Plugin • **Kreditkartenzahlung:** Fehlt (kann eigenständig entwickelt werden) • **Produktfotos:** Vorhanden • **Individuelle Attribute:** Nur durch Speichern eines JSON in der Datenbank möglich • **Produktbeschreibung in HTML:** Vorhanden • **Variantenprodukte:** Vorhanden • **Preise:** Listenpreis • **Rabatte:** Standardrabatt
Dokumentation	Die Dokumentation ist umfassend und sehr gut erklärt.
Gesamteindruck	Medusa überzeugt durch seinen modularen Aufbau und den Headless-First Ansatz. Die Plattform wirkt durchdacht und bietet viele Features. Die fehlende Möglichkeit, Entitäten zu übersetzen, stellt jedoch eine grosse Hürde dar. Eine bereits laufende Diskussion lässt jedoch hoffen, dass dieses Feature in naher Zukunft implementiert wird [13].

Tabelle 4.20: Analyse von Medusa

4.2.8 Nutzwertanalyse

In der nachfolgenden Abbildung 4.4 ist das Ergebnis der Nutzwertanalyse der untersuchten E-Commerce-Systeme dargestellt. Die Bewertung basiert auf den zuvor durchgeführten Analysen der Systeme. Die Gewichtung der einzelnen Kriterien wurde zusammen mit dem Industriepartner erarbeitet. Besonders wichtig ist die Erweiterbarkeit, damit Spezialanforderungen von bestimmten Kunden zielführend implementiert werden können. Ebenfalls wird der Gesamteindruck stark gewichtet, da hier auch Punkte berücksichtigt werden können, die nicht zwingend als Kriterium erfasst sind.

		Shopify		WooCommerce		Magento		nopCommerce		Medusa	
Kriterien	**Gewichtung**	**Bewertung**	**Nutzwert**	**Bewertung**	**Nutzwert**	**Bewertung**	**Nutzwert**	**Bewertung**	**Nutzwert**	**Bewertung**	**Nutzwert**
Kosten	10	1	10	10	100	5	50	10	100	10	100
Erweiterbarkeit	30	2	60	7	210	7	210	10	300	10	300
Community	5	5	25	8	40	8	40	6	30	10	50
Umfang Features	20	10	200	8	160	6	120	9	180	6	120
Benutzerfreundlichkeit	5	10	50	5	25	3	15	5	25	9	45
Gesamteindruck	30	5	150	6	180	3	90	9	270	8	240
Total	**100**		**495**		**715**		**525**		**905**		**855**
Rang			5		3		4		1		2

Abbildung 4.4: Ergebnis der Nutzwertanalyse

Die Nutzwertanalyse hat nopCommerce als das zu verwendende Shopsystem ausgewählt. In den nachfolgenden Abschnitten wird aufgezeigt, wie die gestellten Anforderungen mit diesem System abgeglichen und gelöst werden können.

4.2.9 Domänenmodell nopCommerce

In der folgenden Abbildung 4.5 wird das Domänenmodell von nopCommerce aufgezeigt. Damit mit diesem System eine Lösung für Abacus entwickelt werden kann, muss die Abacus Domäne aus Abschnitt 2.4 mit jener von nopCommerce verknüpft werden.

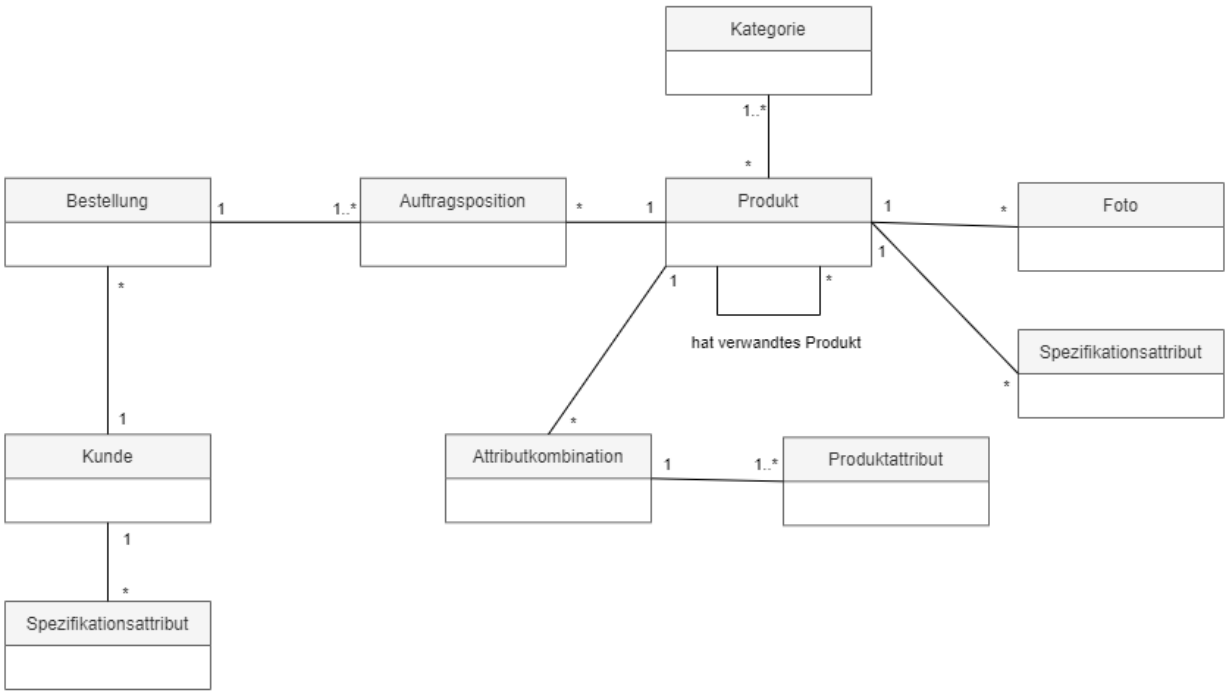

Abbildung 4.5: Domänenmodell nopCommerce

Im nachfolenden Abschnitt wird aufgezeigt, wie nopCommerce die gestellten Anforderungen aus Kapitel 3 abdecken kann. Dabei wird bei den funktionalen Anforderungen darauf eingegangen, wie die Abacus Domäne auf obiges Diagramm abgebildet wird.

4.2.10 Abgleich Anforderungen mit nopCommerce

Um die Erfüllbarkeit der funktionalen und nicht-funktionalen Anforderungen des ausgewählten Systems nopCommerce zu verifizieren, wurden diese abgeglichen.

Nicht-funktionale Anforderungen

Im Abschnitt 3.3.3 wurden explizite nicht-funktionale Anforderungen an ein externes E-Commerce System gestellt. Diese Anforderungen werden in nachfolgender Tabelle 4.21 mit nopCommerce verzifiziert.

NFR	**Resultat**
NFR06 - Lizenzierung	NopCommerce ist als Open Source auf Github verfügbar [10]. Die Lizenz erlaubt eine kommerzielle Nutzung unter der Voraussetzung, dass jede Seite im Fussbereich den Hinweis 'powered by nopCommerce' trägt. Eine einmalige Zahlung von $250 ermöglicht die Entfernung dieses Textes [27]. ✔ Das NFR ist erfüllt.
NFR07 - Pluginsystem	Die Infrastruktur für Plugins ist vorhanden. Durch die Implementierung des Interfaces IPlugin lädt nopCommerce ein kompiliertes Projekt zur Laufzeit aus dem Verzeichnis /Presentation/Nop.Web/Plugins. ✔ Das NFR ist erfüllt.
NFR08 - Event	Die Architektur ist eventbasiert, und man kann sich für Create/Update/Delete-Events von gewünschten Entitäten registrieren. ✔ Das NFR ist erfüllt.
NFR09 - Architektur	NopCommerce implementiert eine geschichtete Architektur nach den Prinzipien von Clean Architecture [23]. Die Geschäftslogik befindet sich dabei in einem eigenen Layer `Nop.Services`. Dies ermöglicht es, direkte Datenbankzugriffe zu vermeiden und Zugriffe auf nopCommerce über diesen Servicelayer zu machen. ✔ Das NFR ist erfüllt.
NFR10 - Headless	Das Frontend ist mit Razor-Pages erstellt, weshalb für die Entwicklung vom Proof of Concept in dieser Studienarbeit keine Web-API benötigt wurde. Eine Erfüllung dieser Anforderung wäre über ein Plugin selbst implementierbar. ✗ Das NFR ist im Standard nicht erfüllt.

Tabelle 4.21: Abgleich nicht-funktionale Anforderungen mit nopCommerce

Funktionale Anforderungen

Da nopCommerce als Open Source Software verfügbar ist, können alle Use Cases durch eine Kombination aus Eigenentwicklung des Frontends und entsprechenden Erweiterungen von nopCommerce realisiert werden. Gemäss NFR07 - Pluginsystem (Tabelle 3.12) ist es jedoch wichtig, dass die Entwicklung der Use Cases möglich ist, ohne Änderungen am Code-Code von nopCommerce vorzunehmen. In den nachfolgenden Abschnitten wird dokumentiert, wie die funktionalen Anforderungen mit nopCommerce umgesetzt werden können. Es wird aufgezeigt, wie sich die Domänenkonzepte von Abacus auf die von nopCommerce anwenden lassen. Dabei wird theoretisch erläutert, wie die Anforderungen realisiert werden können, um eine Grundlage für die Entscheidung zwischen der Verwendung von nopCommerce und einer Eigenentwicklung zu schaffen.

UC01 - Konfiguration

Es gibt ein Interface `IAdminMenuPlugin`, welches durch die Methode `ManageSiteMapAsync(SiteMapNode rootNode)` die Integration einer eigenen Seite in den Adminbereich von nopCommerce ermöglicht.
Ein weiteres Interface, `ISettings`, erleichtert das Erstellen von Einträgen in der Settings-Datenbanktabelle und bietet somit eine einfache Möglichkeit zur Speicherung der Konfiguration des Plugins.

✔ Der UC kann umgesetzt werden.

UC02 - Produktsuche

Die Baumstruktur der Abacus-Produktklassierung kann mit Kategorien abgebildet werden, wobei eine Mehrfachzuordnung von Artikeln zu verschiedenen Kategorien ebenfalls möglich ist. Userfields lassen sich mittels Spezifikationsattributen den Produkten zuordnen. Sowohl Kategorien als auch Spezifikationsattribute können mehrsprachig gepflegt werden. Der Zugriff auf diese Entitäten ist über den Servicelayer von nopCommerce aus dem Plugin heraus möglich.

✔ Der UC kann umgesetzt werden.

UC03 - Warenkorb

Durch die Erfüllung von NFR10 - Event (Tabelle 3.13) kann ein Plugin auf Änderungen im Warenkorb reagieren und die finale Preisfindung durch Abacus initiieren. Der Preis von Artikeln im Warenkorb kann dabei über den Servicelayer angepasst werden.

✔ Der UC kann umgesetzt werden.

UC04 - Verkauf

Der Event `OrderPlaced` ermöglicht es einem Plugin, auf einen Verkauf zu reagieren und diesen an Abacus zu übermitteln. Der Zugriff auf die benötigten Daten erfolgt über den Servicelayer. Zudem bietet nopCommerce die Möglichkeit, eigene Kreditkartenanbieter zu einem späteren Zeitpunkt zu integrieren. Das Interface `IPaymentMethod` erlaubt es, durch Implementierung der Methode `ProcessPaymentAsync()`, eine eigene Bezahlmethode zu implementieren.

✔ Der UC kann umgesetzt werden.

UC05 - Erweiterte Produktsuche

Dies ist ein reines Frontend-Feature, dessen benötigte Daten durch UC02 bereitgestellt werden können.

✔ Der UC kann umgesetzt werden.

UC06 - Registrierung

Mit dem Event `CustomerRegistered` kann ein Plugin auf die Registrierung eines Benutzers reagieren und den neuen Kunden an Abacus übermitteln. In nopCommerce entspricht ein Kunde einem Shopper in Abacus. Dabei kann die von nopCommerce generierte Globally Unique Identifier (GUID) an Abacus mitgegeben werden um eine Verknüpfung zwischen Abacus Shopper und nopCommerce Kunde herzustellen. Auf dem Kundenprofil können mittels Spezifikationsattributen Informationen zu der zugehörigen Abacus Adresse und dem entsprechenden Kunden gespeichert werden. Der Zugriff auf diese Daten erfolgt über den Servicelayer.

✔ Der UC kann umgesetzt werden.

UC07 - Login

Dies muss lediglich im Frontend implementiert werden, da die vorhandene Backend-Funktionalität ausreichend ist.

✔ Der UC kann umgesetzt werden.

UC08 - Cross Selling

NopCommerce unterstützt die Einbindung verwandter Produkte, wodurch Ersatzprodukte aus Abacus dargestellt werden können. Ein Artikel kann mehrere verwandte Produkte besitzen, was diesen Use Case erfüllt. Der Zugriff auf die erforderlichen Entitäten erfolgt über den Servicelayer.

✔ Der UC kann umgesetzt werden.

UC09 - Konditionen und Rabatte

Das Interface `IDiscountRequirementRule` ermöglicht es, eigene Konditionen durch Implementierung der Methode `CheckRequirementAsync(DiscountRequirementValidationRequest request)` zu erstellen. Solche Konditionen können sich auf Attributkombinationen beziehen und die Variantenartikel aus Abacus abbilden. Eine Kombination besteht dabei aus mehreren Produktattributen, die den Variantenausprägungen in Abacus entsprechen.

✔ Der UC kann umgesetzt werden.

UC10 - Einkaufsverlauf anzeigen

Dies ist ebenfalls ein reines Frontend-Feature. Der Zugriff auf die benötigten Daten erfolgt über den Servicelayer.

✔ Der UC kann umgesetzt werden.

UC11 - UC15

Da diese Use Cases nicht zu den Anforderungen eines MVP gehören, wurden sie nicht im Detail analysiert. Es ist wichtig zu erwähnen, dass nopCommerce eine Infrastruktur bietet, um bestehende Entitäten zu erweitern oder neue Entitäten ins System zu integrieren. Diese Flexibilität wird durch die Möglichkeit in einem Plugin eigene Migrations hinzuzufügen, ermöglicht und deckt diese sowie zukünftige Anforderungen ab. Der nachfogelnde Codeausschnitt 4.9 zeigt auf, wie so eine Migration erstellt werden kann.

```
[NopMigration("2023/01/01", "Add Property to Category")]
public class AddSomeNewProperty: AutoReversingMigration
{
    public override void Up()
    {
        Create.Column(nameof(Category.SomeNewProperty))
        .OnTable(nameof(Category))
        .AsString(255)
        .Nullable();
    }
}
```

Listing 4.9: Migration in einem nopCommerce Plugin erstellen

4.2.11 Entscheidung

Die durchgeführte Nutzwertanalyse (4.2.8) hat ergeben, dass nopCommerce im aktuellen Stadium am besten geeignet ist. Obwohl das System Medusa (4.2.7) noch jung ist, könnte es nach der Implementierung der Mehrsprachigkeit ebenfalls ein interessanter Kandidat werden.

Nach der Analyse stand zur Debatte, ob nopCommerce verwendet oder eine komplette Eigenentwicklung durchgeführt werden sollte. Sowohl die funktionalen als auch die nicht funktionalen Anforderungen (Kapitel 3) können mit beiden Ansätzen erfüllt werden. Die Verwendung von nopCommerce ermöglicht eine schnelle Entwicklung und bietet den Vorteil, von jahrelangem Wissen über den Aufbau eines sicheren, performanten und erweiterbaren E-Commerce-Backends zu profitieren. Bei einer Eigenentwicklung hingegen kann die gesamte Logik und Architektur an die Bedürfnisse des Industriepartners angepasst werden.

Nach Rücksprache mit dem Industriepartner wurde entschieden, nopCommerce zu verwenden und einen Conncetor, genannt Abacus Connector, für den Datenaustausch zwischen Abacus und nopCommerce als Plugin zu entwickeln.

Kapitel 5

Design und Implementation

Dieses Kapitel dokumentiert die Implementierung des Proof of Concept, der in dieser Studienarbeit entwickelt wurde. Im Rahmen dieser Arbeit entstanden zwei eigenständige C#-Projekte:

- **Abacus Client**: Die Implementierung eines Clients für die Interaktion mit der Schnittstelle von Abacus.
- **Abacus Connector**: Ein Plugin in nopCommerce, das die Integration von Abacus in das E-Commerce-System ermöglicht.

Die beiden Projekte werden in den Abschnitten 5.1 und 5.2 jeweils detailliert beschrieben. Die Dokumentation zur Architektur, die aufzeigt, wie die beiden Projekte in das Gesamtsystem integriert sind, findet sich im Abschnitt 5.3.

5.1 Abacus Client

Die detaillierte Analyse der RESTful HTTP-Schnittstelle von Abacus, die in Kapitel 4.1 präsentiert wird, führte zur Entwicklung einer C# Class-Library. Diese Bibliothek vereinfacht die Authentifizierung und Interaktion mit der Abacus-Schnittstelle durch eine Reihe von Servicemethoden. Die Überlegungen, die zu dieser Architekturentscheidung führten, sind in Architekturentscheidung 2 (5.4.2) ausführlich erläutert.
Der Abacus Client bietet derzeit fünf verschiedene Services an, die jeweils spezifische Funktionen erfüllen:

- **ProductService**: Umfasst den Umgang mit Produktstammdaten, Userfields und Fotos.
- **ClassificationService**: Ermöglicht die Abfrage von Klassifizierungen und die Zuordnung von Produkten zu Klassifizierungselementen.
- **OrderService**: Beinhaltet Funktionen für das Schreiben in die E-Business Inbox und ermöglicht das Abrufen spezifischer Konditionen für Artikel und Kunden.
- **ShopperService**: Ermöglicht die Zuordnung von Abacus Shoppern zu Adressen und Kunden.
- **SubscriptionService**: Bezieht sich auf die Handhabung der Subscriptions API, wie in Abschnitt 4.1.5 dokumentiert.

Die spezifischen Funktionalitäten jedes Services werden über sein Interface dokumentiert, das als Membervariable in einer Client-Instanz verfügbar ist. Diese Strukturierung ermöglicht eine klare und modulare Nutzung der verschiedenen Services.

Der folgende Codeausschnitt 5.1 zeigt die Anwendung des Clients aus Sicht des Aufrufers.

```
1 var client = new AbacusClient("https://abacus.customize.ch/", 7777,
2                               "clientId", "clientSecret");
3 var products = await client.ProductService.GetAllProductsAsync();
```

Listing 5.1: Anwendung Abacus Client

Die Nummer 7777 repräsentiert dabei die Mandantennummer in Abacus.

5.1.1 Dependencies

Der Abacus Client verwendet die Bibliothek Simple.OData.Client (nachfolgend OData-Bibliothek bezeichnet) [12]. Die Begründung für diese Entscheidung ist im Abschnitt 5.4.3 dargelegt. Diese Bibliothek dient als HTTP-Client für die Schnittstelle von Abacus, wodurch anstelle von JSON ein typisiertes Dictionary zurückgegeben wird. Die OData-Bilbiothek holt mithilfe des OData-Endpoints '`$metadata`' die dokumentierten Typen eines Properties ab und gibt diese entsprechend typisiert zurück. Die Anwendung der OData-Bibliothek wurde in einen Wrapper ausgelagert, um die Unit-Testbarkeit der Services zu gewährleisten. Details zur Implementierung sind im Testkonzept unter Abschnitt 5.5.1 ausführlicher beschrieben.

5.1.2 Modelle

Die primäre Funktion des Abacus Clients besteht darin, Antworten der Schnittstelle in strukturierten Modellen an den Aufrufer zurückzuliefern. Innerhalb der Servicemethoden werden oftmals mehrere API-Aufrufe kombiniert. Zum Beispiel werden bei der Methode `GetAllProductsAsync()` Daten wie Produktstammdaten, Userfields und Produktbilder abgerufen und in einem aggregierten Produktmodell zurückgegeben.
Gemäss Architekturentscheidung 1 (5.4.1) ist eine zusätzliche Persistierung der Produktstammdaten im E-Commerce-System erforderlich. Obwohl durch die Aggregierung der Schnittstellenaufrufe eine Performanceeinbusse entsteht, wird diese akzeptiert. Die Ausführung dieser Servicemethoden erfolgt im Hintergrund, was keinen direkten Einfluss auf die Performance aus Sicht der Shopkunden hat.

Um Dictionaries, die aus den Aufrufen der OData-Bibliothek stammen (5.1.1), in definierte Modelle zu konvertieren, wurde eine Erweiterungsmethode für `Dictionary<string, object>` implementiert. Diese Methode besitzt die Signatur `T GetValueOrDefault<T>(string dictionaryKey)`. Die Modelle erwarten im Konstruktor ein Dictionary. Mit der Erweiterungsmethode werden alle Eigenschaften des Modells aus dem Dictionary ausgelesen und mit dem generischen Typen `T` zurückgegeben. Diese Methode funktioniert auch rekursiv, wodurch verschachtelte Modelle verarbeitet werden können.

```
1 public ShopProduct(IDictionary<string, object> dictionary)
2 {
3     if(dictionary != null)
4     {
5         Type = dictionary.GetValueOrDefault<ProductRelatedType>("Type");
6         ProductId = dictionary.GetValueOrDefault<int>("ProductId");
7         ValidFrom = dictionary.GetValueOrDefault<DateTime>("ValidFrom");
8     }
9 }
```

Listing 5.2: Parsen von Dictionary in ein Modell

5.1.3 Versionierung

Die RESTful HTTP-Schnittstelle von Abacus ist über die URL versioniert (4.1.2). In dieser Studienarbeit wurde noch keine Versionierung implementiert. Die Struktur dieses Projekts wurde jedoch so konzipiert, dass eine spätere Implementierung möglich ist. Eine Instanz von `AbacusClient` implementiert die Interfaces aller fünf Services, was bedeutet, dass im Hintergrund mehrere Implementierungen dieser Services existieren können.

5.2 Abacus Connector

Der Abacus Connector, die Kernkomponente dieser Studienarbeit, ist als eigenständiges Projekt in der Model-View-Controller-Architektur [9] implementiert und wird als Plugin für nopCommerce genutzt. Durch das Implementieren des `IPlugin`-Interfaces wird das Projekt von nopCommerce während der Laufzeit aus einem festgelegten Verzeichnis geladen und ausgeführt. Dieser Mechanismus wird in der Deplyoment-Stage der Gitlab-Pipeline gelöst.

5.2.1 Views

Das ausgewählte E-Commerce-System ermöglicht die Integration eigener Routen in das Backend. Der Abacus Connector nutzt Razor-Pages [26] für die Implementierung von Views, die auf diesen Routen dargestellt werden. Dies ermöglicht die Konfiguration des Abacus Connectors direkt im Backend von nopCommerce. Dabei können notwendige Metadaten zur Anbindung an die Abacus-Schnittstelle gepflegt und der Synchronisationsdienst gesteuert werden (5.2.2). Bei jeder Änderung der Konfiguration wird ein Healthcheck der Abacus Schnittstelle durchgeführt, um sicherzustellen, dass die aktualisierte Konfiguration korrekt ist.

5.2.2 Synchronisationsdienst

Der Abacus Connector hat als Hauptaufgabe, Stammdaten aus dem Abacus ERP nahtlos und kontinuierlich in das E-Commerce-System zu integrieren. Diese Integration wird durch einen Synchronisationsdienst realisiert, der sich in zwei Hauptmethoden aufgliedert:

- **Vollsynchronisation**: Diese Methode importiert alle Produkte, Klassierungen und Kundeninformationen aus Abacus und sorgt für eine umfassende Aktualisierung der Datenbasis.
- **Teilsynchronisation**: Unter Nutzung der Subscription-API (4.1.5) werden laufend aktuelle Datenänderungen abgeholt.

Mit der Installation des Plugins wird automatisch ein `ScheduleTask` generiert, eine Funktion, die in regelmässigen, anpassbaren Zeitintervallen von der nopCommerce-Infrastruktur ausgeführt wird. Die Konfiguration dieser Intervalle erfolgt über die View. Der Scheduled Task entscheidet mit einem Flag in der Settingsdatenbank vom Conncetor eigenständig, ob eine Voll- oder Teilsynchronisation durchgeführt wird. Dieses Flag ist initial nach der Installation vom Plugin auf Vollpublikation gesetzt. Der Scheduled Task setzt diesen nach einer erfolgreich durchgeführten Vollsynchronisation auf Teilsynchronisation um. Diese Einstellung kann in der View zurückgesetzt werden, um eine erneute Vollsynchronisation zu initiieren.

5.2.3 Mapping von Primärschlüsseln

NopCommerce bietet keine Möglichkeit, individuelle Primärschlüssel für Entitäten zu definieren. Im Einklang mit der Einhaltung von NFR07 - Pluginsystem (Tabelle 3.12), welches Modifikationen am Quellcode des E-Commerce-Systems verhindern soll, wurde von einer Implementierung dieser Funktionalität abgesehen. Um sicherzustellen, dass bestehende Entitäten bei Aktualisierungen durch die Teilsynchronisation korrekt identifiziert werden, wurde eine spezielle Mappingtabelle eingerichtet. Diese Tabelle wird durch eine Migrationsklasse während der Plugin-Installation in ein separates Schema innerhalb der Datenbank eingefügt. Neben der Abbildung zwischen den Schlüsseln von Abacus und nopCommerce enthält diese Tabelle auch ein Enum, das den Typ der gemappten Entität spezifiziert. Dies ermöglicht die zentrale Speicherung aller Mappings in dieser Tabelle. Der Zugriff auf die Datenbank wird über eine zentrale Repository-Klasse geregelt.

5.2.4 Übermittlung von Bestellungen

Bei Eingang einer neuen Bestellung im Webshop wird diese umgehend an Abacus übermittelt. Dies erfolgt durch einen Handler, der auf das Domain-Event 'OrderPlaced' reagiert, um den Übermittlungsprozess zu initiieren. Der nachfolgende Codeausschnitt im Listing 5.3 illustriert die Registrierung eines solchen Handlers:

```
1  public class OrderPlacedEventHandler : IConsumer<OrderPlacedEvent>
2  {
3      private readonly IAbacusOrderService _abacusOrderService;
4
5      public OrderPlacedEventHandler(IAbacusOrderService abacusOrderService)
6      {
7          _abacusOrderService = abacusOrderService;
8      }
9
10     public async Task HandleEventAsync(OrderPlacedEvent eventMessage)
11     {
12         await _abacusOrderService.SubmitOrderToAbacus(eventMessage.Order);
13     }
14 }
```

Listing 5.3: Handler für das Domainevent 'neue Bestellung'

Der Eventhandler wird asynchron ausgeführt. Dies impliziert, dass eine Bestellung im Webshop für den Kunden als erfolgreich abgeschlossen gilt, auch wenn die Übermittlung an Abacus möglicherweise nicht erfolgt. Die Gründe hierfür sind in der Architekturentscheidung 5 (5.4.3) dargelegt. Zudem markiert der Eventhandler eine Bestellung, falls bei der Übermittlung Fehler auftreten. Ein Benachrichtigungsmechanismus für solche Fälle ist momentan noch nicht implementiert. In Zukunft soll zudem eine Option geschaffen werden, um diese Bestellungen manuell erneut übermitteln zu können.

5.2.5 Kundenhandling

Im Domänenmodell von Abacus (2.4) wird ersichtlich, dass ein Kunde mehrere Adressen besitzen und eine Adresse zugleich verschiedenen Shoppern zugeordnet sein kann. Im Gegensatz dazu kennt das Domänenmodell von nopCommerce (4.2.9) nur einzelne Kunden. Wie in Abschnitt 4.2.10 erläutert, entspricht ein Kunde aus nopCommerce einem Shopper in Abacus. Der Abacus Connector ordnet Kunden über Spezifikationsattribute zu, die Aufschluss darüber geben, zu welcher Adress- und Kundennummer ein nopCommerce-Kunde gehört. Das Kundenhandling gestaltet sich jedoch komplexer, was nachfolgend anhand eines fiktiven Beispiels mit der Firma Bürobedarfs AG illustriert wird.

Umgang mit Kunden

Ein neuer Kunde der Bürobedarfs AG, Roger Tester, registriert sich im Webshop und tätigt eine Bestellung. Da Roger Tester in Abacus noch nicht erfasst ist, existieren keine Adress- und Kundennummern von ihm. In diesem Fall vergibt der Abacus Connector temporäre, stets negative Nummern, um Konflikte mit existierenden Adress- und Kundennummern von Abacus zu vermeiden. Die Bestellung wird in die Abacus-Inbox übertragen. Erst nach der Bearbeitung der Inbox-Nachricht erhält Roger Tester eine feste Adress- und Kundennummer in Abacus. Sollte Roger Tester vor dieser Bearbeitung eine weitere Bestellung aufgeben, kann Abacus dank der temporären Kundennummer diese Bestellung dem korrekten Abacus-Kunden zuordnen. Nachdem ein Mitarbeiter der Bürobedarfs AG die erste Inbox-Nachricht in einen Auftrag umgewandelt hat, wird Roger Tester eine offizielle Kunden- und Adressnummer zugewiesen. Der Abacus Connector erhält durch die Teilsynchronisation der Subscription-API eine Benachrichtigung über die Mutation des Shoppers Roger Tester. Nun kann der Connector über die Abacus-Schnittstelle die zugewiesenen Nummern abfragen und diese in einem Spezifikationsattribut des Kunden speichern. Zukünftige Bestellungen von Roger Tester können dann mit den korrekten Nummern an die Abacus-Inbox übermittelt werden, was den Mitarbeitern der Bürobedarfs AG eine automatisierte Auftragsverarbeitung ohne manuellen Eingriff ermöglicht.

5.2.6 Preisfindung

Um sicherzustellen, dass Kunden bei der Bestellung ihre individuellen Preise und Rabatte erhalten, wird bei jeder Änderung in ihrem Warenkorb eine Anfrage zur Preisfindung an Abacus gesendet. Die Implementierung dieses Prozesses folgt dem Muster der Bestellungsübermittlung (5.2.4), wobei hier ein Domainevent-Handler synchron implementiert wird. Diese Synchronität ist entscheidend, um zu verhindern, dass der Warenkorb kurzzeitig in einem inkonsistenten Zustand verbleibt.

Sollte bei der Preisanfrage ein Fehler auftreten, wird dem Kunden der Listenpreis angezeigt. Dieser Preis wird in nopCommerce durch die Teilsynchronisation aktualisiert und gespeichert.

5.3 Architektur

Die Systemarchitektur wird unter Verwendung des standardisierten C4-Modells dargestellt [5]. Die optionale vierte Ebene (Code) wird in dieser Dokumentation nicht aufgeführt. Bei Bedarf kann diese Darstellung mithilfe des Toolings einer integrierten Entwicklungsumgebung (IDE) generiert werden.

5.3.1 Level 1 - Kontext

Abbildung 5.1 zeigt die Kontextebene des C4-Modells. Hier wird das Zusammenspiel zwischen den Akteuren (Consultant, Shopbetreiber und Shopper), den Fremdsystemen (Abacus ERP, E-Commerce-System (nopCommerce), Mailsystem) und dem System under Development (SuD) dargestellt.
Der Consultant ist zuständig für die Konfiguration und das Monitoring des SuD. Der Shopbetreiber verwaltet das E-Commerce System und ist für die Pflege der Stammdaten in Abacus ERP verantwortlich. Der Shopper nutzt das E-Commerce System, um Produkte zu erwerben.

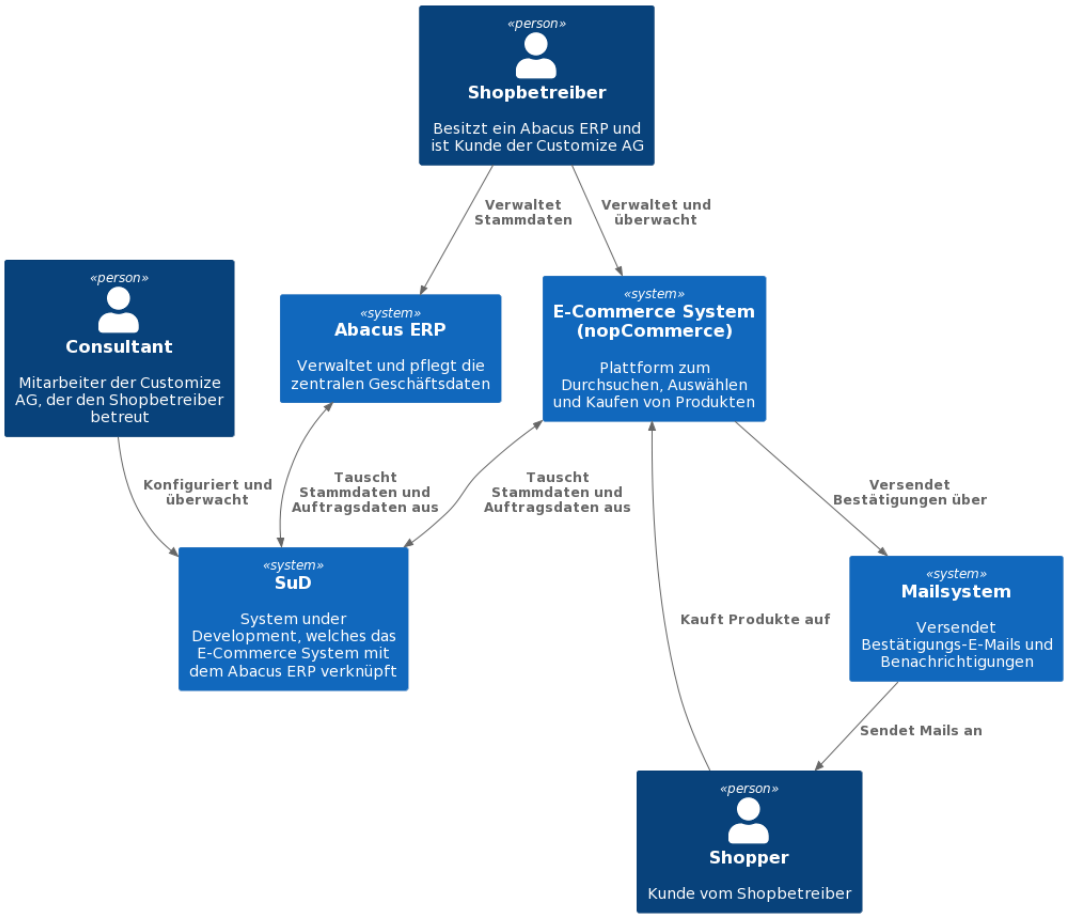

Abbildung 5.1: Architektur: Kontextdiagramm (C4)

5.3.2 Level 2 - Container

Die Containerebene, dargestellt in Abbildung 5.2, unterteilt das Gesamtsystem in seine verschiedenen Container. Im Kern dieses Modells steht nopCommerce, ein E-Commerce-System, das als C# ASP.NET Core-Applikation [25] implementiert wurde. Der Zugang zu nopCommerce wird durch mit Razor Pages [26] erstellte Views ermöglicht. Kunden interagieren über das Storefront mit dem Webshop, um Einkäufe zu tätigen. Parallel dazu dient das Admin Frontend als Konfigurationsschnittstelle zur Verwaltung des Webshops, in der auch der Abacus Connector integriert ist. In diesem Diagramm werden die beiden Container, der Abacus Client und der Abacus Connector, als separate Einheiten dargestellt, da sie als eigenständige C#-Projekte entwickelt wurden. Im realen Betrieb werden sie innerhalb des nopCommerce Containers ausgeführt.

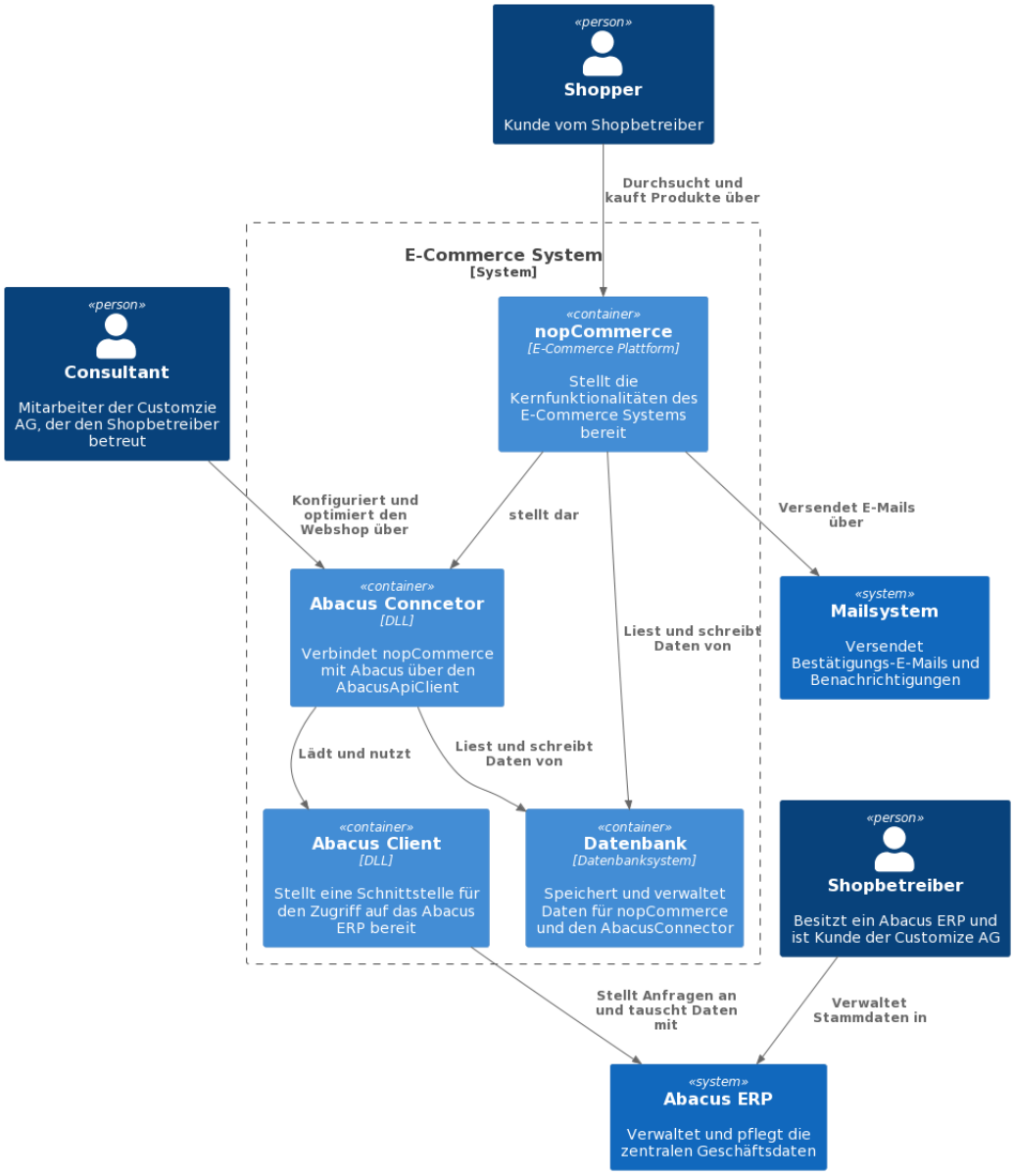

Abbildung 5.2: Architektur: Containerdiagramm (C4)

Im Diagramm wird deutlich, dass sowohl nopCommerce als auch der Abacus Connector Zugriff auf die Datenbank haben. Dies ist notwendig, da der Connector eigene Datenbanktabellen in einem separaten Schema für seine Funktionalitäten anlegt. Für den Zugriff auf die von nopCommerce erstellten Tabellen werden ausschliesslich die Service-Methoden von nopCommerce verwendet. Dies verhindert direkte Datenbankzugriffe durch den Connector und gewährleistet die Integrität des Systems.

5.3.3 Level 3 - Komponenten

Im unter Abbildung 5.3 dargestellten Komponentendiagramm werden die im Rahmen der Studienarbeit entwickelten Container in ihre einzelnen Komponenten zerlegt. Die Akteure sind nicht im Diagramm dargestellt, da diese aus dem Kontext- und Containerdiagramm bekannt sind.

Abacus Connector
Der Abacus Connector ist eine MVC-Applikation [9], welche die Views mit Razor Pages implementiert [26]. Die Controller sind schlank gehalten, indem die gesamte Logik in Service-Komponenten ausgelagert wird. Diese Services definieren Interfaces, um eine lose Kopplung und eine einfache Bereitstellung durch Dependency Injection zu ermöglichen. Die Entitäten des Plugins werden in der Domain-Komponente verwaltet und in der Datenbank gespeichert. Die Infrastrukturkomponente ist für das Hinzufügen der Services zum Dependency Injection Container und das Einbetten der Views in das Admin-Frontend von nopCommerce zuständig.

Abacus Client
Der Abacus Client bietet verschiedene Services, die die Kommunikation mit der Abacus API abstrahieren. Die von der API empfangenen Daten werden in Modelle gemappt und an den Konsumenten weitergegeben. Die Authentifizierung mit der Abacus API erfolgt über die Authenticator-Komponente, die einen einfachen Token-Cache implementiert. Diese Komponente implementiert die Authentifizierung nach dem OAuth 2.0 Standard [18]. Da die Tokens eine Gültigkeit von 10 Minuten haben, wird durch einen Tokencache sichergestellt, dass nur in diesem Intervall neue Tokens beantragt werden.

5.3.4 Level 3 - Komponenten

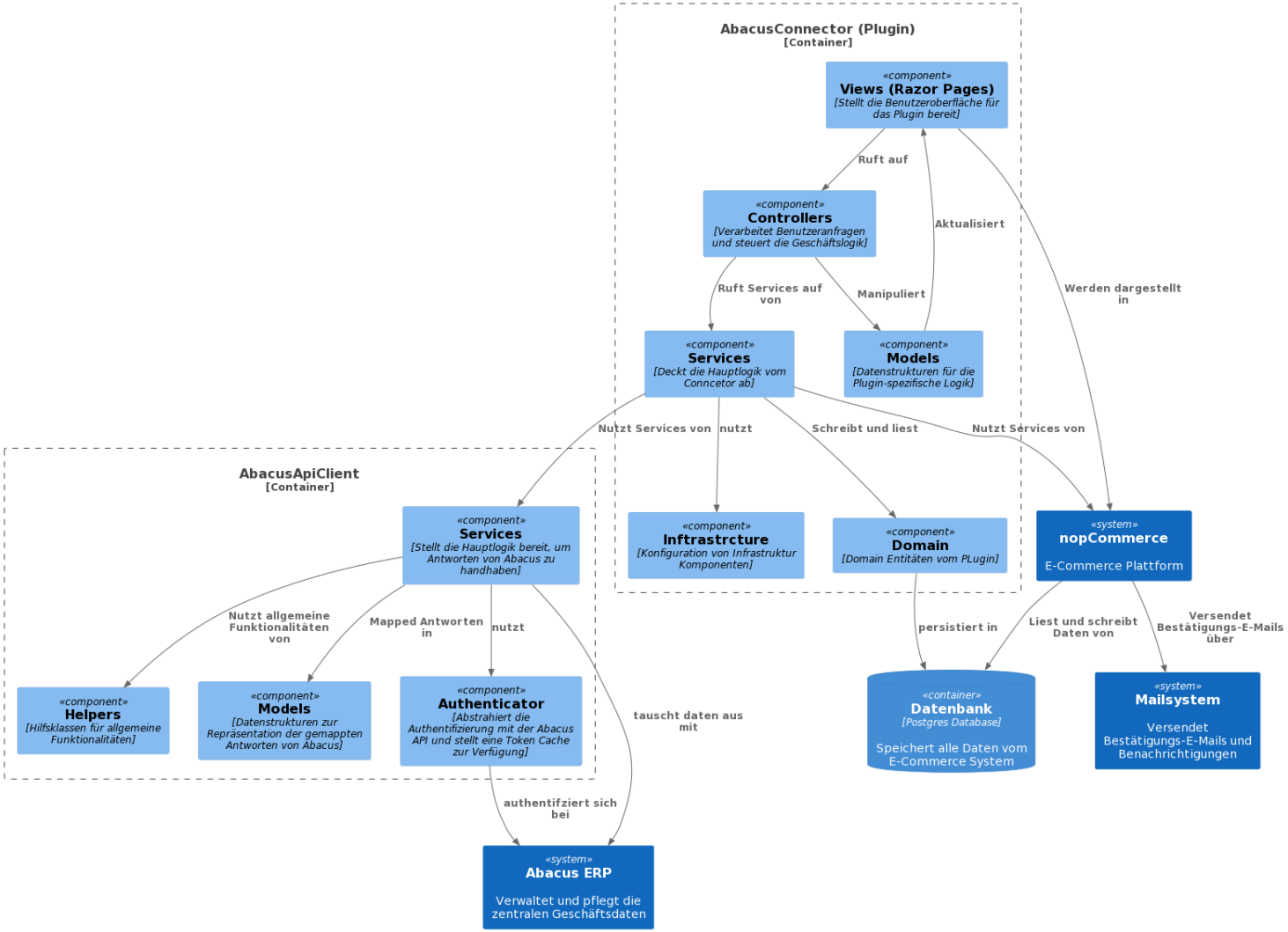

Abbildung 5.3: Architektur: Komponentendiagramm (C4)

5.3.5 Sequenzdiagramm Architektur anhand Teilsynchronisation

Um das Zusammenspiel aller Komponenten genauer zu veranschaulichen, wird auf nachfolgender Seite in Abbildung 5.4 ein Sequenzdiagramm aufgezeigt. Dieses Diagramm veranschaulicht die Teilsynchronisation (5.2.2), die erforderlich ist, um die Anforderungen von UC02 - Produktsuche zu erfüllen.

Die Sequenz wird in regelmässigen Abständen ausgeführt, um sicherzustellen, dass alle relevanten Stammdaten aus dem Abacus ERP-System in den Webshop aktualisiert werden. In diesem konkreten Beispiel geht es um die Aktualisierung der geänderten Produktstammdaten.

Der ausführende Akteur kann variieren, da dieser Prozess entweder automatisch durch den geplanten Task des Abacus Connectors oder manuell durch einen Berater oder den Shopbetreiber durchgeführt werden kann.

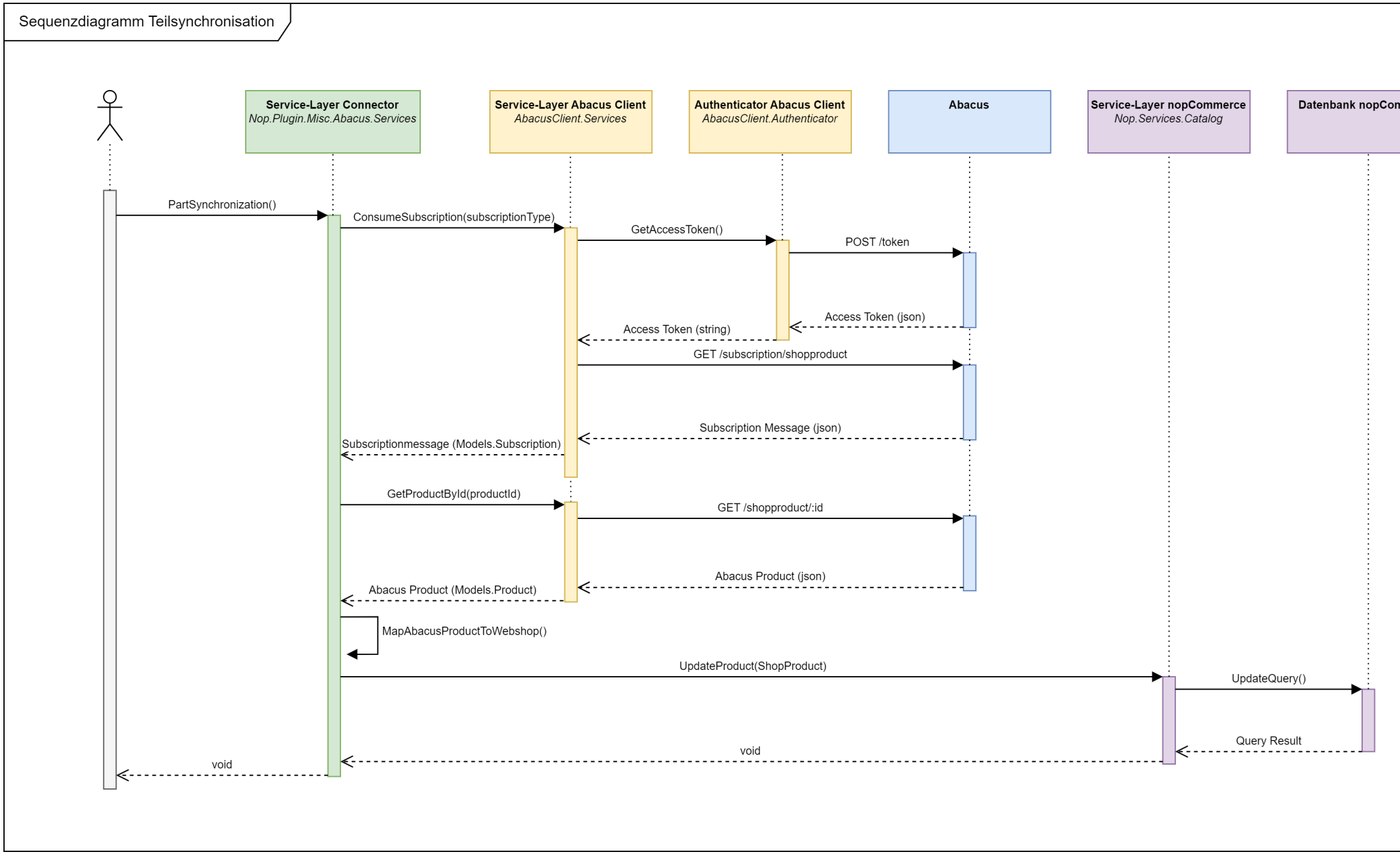

Abbildung 5.4: Sequenzdiagramm Teilsynchronisation Produktdaten (UML)

5.4 Architekturentscheidungen

Alle getroffenen Architekturentscheidungen wurden als strukturierte Architectural Decision Records (ADR) nach Vorlage von Michael Nygard erfasst [3]. Diese ADR-Einträge befinden sich im Anhang C.

5.4.1 Persistierung von Produktdaten (ADR01)

Zu Beginn dieser Studienarbeit war unklar, ob die Produktstammdaten direkt aus dem Abacus-System im Frontend bezogen werden können, wodurch eine redundante Persistierung der Stammdaten im E-Commerce-System überflüssig wäre. Es war von entscheidender Bedeutung, dass die nichtfunktionale Anforderung NFR01 - Performance eingehalten wird. Da der Abruf aller Produkte basierend auf den Abacus-Mustermandanten bis zu 7 Sekunden dauerte, war eine Erfüllung dieser Anforderung nicht möglich. Zusätzlich bestand keine Möglichkeit, spezielle Produkte, wie jene für die Startseite, in einem einzelnen Aufruf der Restful HTTP-Schnittstelle von Abacus zu erhalten. Das Attribut '`Frontpage`' konnte nicht als Filterparameter an die Schnittstelle übergeben werden. Daraus folgte, dass alle Produkte abgerufen und anschliessend im Frontend gefiltert werden mussten, um jene mit '`Frontpage: true`' zu identifizieren. Dieser Prozess hätte einen erheblichen negativen Einfluss auf die Performance. Daher wurde entschieden, dass unabhängig vom Entwicklungsweg - ob Eigenentwicklung oder Anbindung an ein E-Commerce-System - eine zusätzliche Persistierung der Produktstammdaten notwendig ist.

Eine wesentliche Konsequenz dieser Architekturentscheidung ist, dass zur kontinuierlichen und korrekten Persistierung der Stammdaten die Nutzung der Subscriptions API erforderlich ist (4.1.5). Ziel ist es, zu vermeiden, dass der gesamte Produktstamm in regelmässigen Zeitabständen abgerufen und mit dem Datenbestand des Webshops abgeglichen werden muss. Durch die Subscriptions API wird es möglich, lediglich die Produktnummern der seit dem letzten Abruf geänderten Produkte zu ermitteln und ausschliesslich diese zu synchronisieren.

5.4.2 Entwicklung API Client (ADR02)

Bei der Analyse der RESTful HTTP-Schnittstelle wurde beschlossen, eine eigenständige Client-Library für Abacus zu entwickeln. Diese kann unabhängig vom gewählten Entwicklungsweg eingesetzt und bei Bedarf auch in anderen Projekten des Industriepartners integriert werden. Wie in Abschnitt 4.1.2 erläutert, ist die Schnittstelle mittels der URL versioniert. Dies impliziert, dass es zukünftig weitere Versionen mit veränderten Endpunkten oder vollständig anderen Antwortmodellen geben könnte. Da das System under Development (SuD) möglicherweise von vielen Kunden mit verschiedenen Abacus-Versionen verwendet wird und somit gleichzeitig mehrere Schnittstellenversionen zum Einsatz kommen können, ist eine zentrale Implementierung der Schnittstelle vorteilhaft.
Zudem ermöglicht dies, dem Client saubere Servicemethoden bereitzustellen. So werden beispielsweise beim Aufruf von `AbacusClient.ProductService.getProductById(id)` im Hintergrund mehrere Anfragen durchgeführt, um neben den Stammdaten auch die Bilder und Userfields des Produkts in einem vollständigen Modell zurückzugeben. Die Implementation dieses Clients ist in Abschnitt 5.1 dokumentiert.

5.4.3 Verwendung Simple.OData.Client (ADR03)

Die Abacus-Schnittstelle ist nach dem OData-Standard implementiert [31]. In diesem Standard ist ein '`$metadata`' Endpunkt vorgesehen, der die Typeninformationen aller Entitäten als XML bereitstellt. Durch die Verwendung der Bibliothek Simple.OData.Client [12] als HTTP-Client können die Vorteile dieses Standards genutzt werden.
Es gibt dabei zwei Methoden um die Bilbiothek zu verwenden.

1. Definieren von Entitätsklassen, sodass die Bibliothek automatisch ein Mapping auf diese Klassen durchführt.
2. Die Antworten als `Dictionary<string, object>` erhalten, wobei der Schlüssel (ein String) den Identifikator des Entitätsmembers darstellt und der Wert (geboxt als Objekt) den typisierten Wert des Members enthält.

Die Entscheidung fiel auf die zweite Methode, da die erste Methode bei der Handhabung von Userfields auf Schwierigkeiten stösst. Die Abacus-Schnittstelle gibt Userfields in einem Format zurück, wie es im Listing 5.4 dargestellt ist.

```
{
    "ProductId": 3,
    "UserFields": {
        "UserField1": "Demotext",
        "UserField2": "9.00",
        "UserField3": "",
        "UserField4": 3
    }
}
```

Listing 5.4: Struktur Userfields

Die Herausforderung dabei ist, dass nicht vorhersehbar ist, wie viele Userfields ein Kunde besitzen wird. In Abacus können fortlaufend zusätzliche Userfields ergänzt werden, was bedeutet, dass sich die Struktur dieser Antwort ständig verändern kann. Eine Struktur, wie sie im Listing 5.5 vorgeschlagen wird, würde der eingesetzten Bibliothek den Umgang mit Userfields erleichtern.

```
{
    "ProductId": 3,
    "UserFields": [
        {
            "Name": "UserField1",
            "Value": "Demotext"
        },
        {
            "Name": "UserField2",
            "Value": "9.00"
        }
    ]
}
```

Listing 5.5: Gewünschte Struktur Userfields

In Abschnitt 5.1.2 wird dokumentiert, wie mit Hilfe einer implementierten Erweiterungsmethode aus den Dictionaries die Instanzen der Antwortmodelle erstellt werden.

5.4.4 Einsatz von nopCommerce als E-Commerce-Framework (ADR04)

Im Rahmen dieser Studienarbeit erfolgte die Evaluation verschiedener E-Commerce-Systeme, um zu bestimmen, ob eines dieser Systeme angebunden wird oder ob eine komplette Eigenentwicklung erforderlich ist. Basierend auf einer Nutzwertanalyse und in Abstimmung mit dem Industriepartner fiel die Entscheidung, den Proof of Concept mit nopCommerce zu realisieren. Die detaillierte Analyse, die zur Auswahl von nopCommerce führte, ist in Abschnitt 4.2 dokumentiert.

5.4.5 Asynchrone Übermittlung der Bestellungen (ADR05)

Wenn im Webshop eine neue Bestellung aufgegeben wird, muss diese an Abacus übermittelt werden. Diese Aufgabe übernimmt der Abacus Connector, der einen Handler für das Domainevent `OrderPlaced` registriert. Diese Handler können entweder asynchron oder synchron ausgeführt werden. In diesem Kontext bedeutet eine synchrone Ausführung, dass die Bestellung aus der Perspektive des Shoppers fehlschlägt, falls im Handler ein Fehler auftritt.

Die Übermittlung wurde jedoch asynchron implementiert, um sicherzustellen, dass eine Bestellung auch dann erfolgreich abgeschlossen wird, wenn die Übermittlung an Abacus fehlschlägt. Dies ist besonders wichtig, da der Webshop auch bei einem Update des Abacus ERP weiterhin funktionsfähig bleiben muss. Bestellungen, bei denen die Übermittlung fehlschlägt, werden vom Eventhandler entsprechend gekennzeichnet.
Zukünftig wird eine Funktion benötigt, durch die der Shopbetreiber über fehlgeschlagene Übermittlungen informiert wird und die Möglichkeit hat, die Übermittlung erneut zu initiieren.

5.5 Testkonzept

Moderne Softwareentwicklung betrachtet das Testen als entscheidender Faktor zur Sicherstellung der Qualität und Zuverlässigkeit entwickelter Anwendungen. Dieser Abschnitt stellt das Testkonzept für die im Rahmen dieser Studienarbeit entwickelten Projekte Abacus Client und Abacus Connector vor.

Im Fokus stehen zwei zentrale Testmethoden: Unit Tests und Integrationstests. Unit Tests überprüfen die kleinsten testbaren Einheiten einer Anwendung, in diesem Fall spezifische Funktionen und Methoden des Abacus Client und des Abacus Connector, um die korrekte Funktionsweise jeder Komponente isoliert zu gewährleisten. In beiden Projekten wurde das weit verbreitete C# Testframework NUnit eingesetzt.

Da der Abacus Connector als Plugin in nopCommerce integriert ist und der Abacus Client mit der Schnittstelle von Abacus ERP interagiert, sind viele externe Abhängigkeiten vorhanden. Daher liegt ein besonderer Schwerpunkt auf Integrationstests, um die Interaktion mit diesen Umsystemen zu überprüfen. Dies hilft, potenzielle Fehler durch Updates von nopCommerce oder Abacus ERP frühzeitig zu erkennen. Eine grosse Herausforderung dabei ist die Unmöglichkeit, das Abacus ERP wiederholt automatisiert für Tests hochzufahren.

5.5.1 Unit Tests

Wie bereits erwähnt, stellt die Vielzahl an externen Abhängigkeiten eine Herausforderung dar. Um dennoch Unit Tests zu ermöglichen, wurde ein Mocking-Framework verwendet, um Antworten von externen Methodenaufrufen zu simulieren. Dabei wird der gesamte Aufrufstack nicht durchlaufen, sondern es wird stattdessen eine vom Entwickler vorgegebene Antwort zurückgegeben. Hierfür wurde die Bibliothek Moq eingesetzt.

Abacus Client

Um das Abacus Client Projekt unit-testbar zu machen, mussten besondere Massnahmen ergriffen werden. Eine Schwierigkeit bestand darin, dass die genutzte Bibliothek OData.Client.Simple [12] kein Interface für den HTTP-Client bereitstellt. Dadurch war es unmöglich, die Antworten von Abacus durch einen Mock zu ersetzen..

Deshalb wurde ein eigener Wrapper erstellt, der den Client der Bibliothek beinhaltet und die benötigten Funktionalitäten über ein Interface bereitstellt. Nachfolgender Codeausschnitt zeigt, wie die Bibliothek in einen Wrapper ausgelagert wurde.

```
public class ODataClientWrapper : IODataClientWrapper
{
  private readonly ODataClient _client;

  public ODataClientWrapper(ODataClient client)
  {
      _client = client;
  }
  public async Task<IEnumerable<IDictionary<string, object>>> GetAllProducts()
  {
      var products =  await _client
          .For("ShopProducts")
          .Expand("ShopProductAdditionalField")
          .Expand("ShopProductAdditionalFreeField")
          .Expand("ShopProductTextAdditionalFields")
          .Expand("ShopProductVariantAdditionalField")
          .Expand("ShopProductAttachments")
          .FindEntriesAsync();
      return ProcessDictionaries(products);
  }
  // Weitere Methoden
}
```

Listing 5.6: OData Wrapper

Wie in der Architekturentscheidung 4 (Abschnitt 5.4.4) beschrieben, liefert der zu mockende ODataClient Dictionaries zurück. Diese Dictionaries sind teilweise sehr umfangreich und komplex verschachtelt. Ein manuelles Nachbauen wäre zu aufwendig gewesen. Deshalb wurden im Testprojekt unter '/Data' JSON-Dateien von echten Antworten der Abacus Schnittstelle gespeichert. Diese JSON-Dateien werden beim Setup des Testprojekts in Dictionaries serialisiert, um den oben aufgeführten `ODataClientWrapper` korrekt mocken zu können. Nachfolgender Code zeigt, wie das Setup eines solchen Tests durchgeführt wird.

```
[TestFixture]
public class ProductServiceTests
{
  private Mock<IODataClientWrapper> _mockClient;
  private ProductService _productService;

  [SetUp]
  public async Task SetupAsync()
  {
    _mockClient = new Mock<IODataClientWrapper>();
    _productService = new ProductService(_mockClient.Object);
    var json = await File.ReadAllTextAsync(@"/Data/GetAllProductsAsync.json");
    var dic = JsonConvert.DeserializeObject<IDictionary<string, object>>(
        json, new JsonConverter[] { new MyConverter() });
    _mockClient.Setup(m => m.GetAllProductsAsync()).ReturnsAsync(dic);
  }

  [Test]
  public async Task GetAllProductsAsync_ReturnsCorrectNumberOfProducts()
  {
      var products = await _productService.GetAllProductsAsync();
      int expectedProductCount = 55;

      Assert.That(products.Count(), Is.EqualTo(expectedProductCount));
  }
}
```

Listing 5.7: Aufbau Testing Abacus Client

Mit diesen Massnahmen kann der Servicelayer effektiv getestet werden. Neben den Services gibt es diverse Hilfsklassen, die ohne Abhängigkeiten funktionieren und daher klassischen Unit Tests unterzogen werden können.

Abacus Connector

Das Connector-Projekt interagiert intensiv mit den Servicemethoden von nopCommerce. In dieser Arbeit wird das ausgewählte E-Commerce-System als Framework betrachtet, und es wird bewusst darauf verzichtet, alle Servicemethoden von nopCommerce zu mocken. Dies erhöht die Aussagekraft der Tests erheblich, erfordert jedoch die Einbindung einer Datenbank in die Unit Tests, da die Servicemethoden von nopCommerce mit dieser interagieren. Für die automatisierte und reproduzierbare Durchführung der Tests wird eine In-Memory SQLite-Datenbank verwendet. Zudem besteht eine Abhängigkeit dieses Projekts zum Abacus Client, der komplett gemockt wird.

Jeder Unit Test erbt von der Klasse `BaseAbacusPluginTest`. In dieser Klasse wird eine temporäre Datenbank mittels eines statischen Konstruktors erstellt, die während der Laufzeit des Testprojekts genutzt wird. Der statische Konstruktor gewährleistet durch den C#-Compiler, dass er nur einmal pro Laufzeit ausgeführt wird. In diesem Konstruktor wird auch die Migration auf der temporären Datenbank durchgeführt. Um alle nopCommerce- und Connector-Services im Dependency Injection Mechanismus [35] verfügbar zu machen, werden sie in den Servicecontainer des Testrunners eingefügt.

5.5.2 Integrationstests

Integrationstests gewährleisten, dass alle beteiligten Komponenten des implementierten Webshopsystems korrekt zusammenarbeiten. Wie zu Beginn dieses Kapitels erwähnt, besteht die Herausforderung darin, dass es nicht möglich ist, eine reproduzierbare Instanz von Abacus ERP automatisiert hochzufahren. Trotzdem ist es wesentlich, das Abacus ERP in die Integrationstests einzubeziehen. Da eine produktive Abacus-Installation regelmässig aktualisiert wird, muss sichergestellt werden, dass der Webshop weiterhin wie vorgesehen funktioniert.

Zu diesem Zweck wurde auf einem von Customize bereitgestellten Server ein Abacus Testmandant eingerichtet, der für das Integrationstesting genutzt werden kann. Dieser basiert auf dem Mustermandanten von Abacus, wurde jedoch so angepasst, dass alle implementierten Funktionalitäten getestet werden können.

Die Integrationstests wurden in den Abacus Connector integriert und nutzen die unter Kapitel 5.5.1 beschriebene Testinfrastruktur des Abacus Connectors. Im Gegensatz zu Unit Tests wird der Abacus-Client jedoch nicht durch einen Mock ersetzt, sondern die Integrationstests laufen direkt auf der Testinstanz des Abacus ERP.

5.5.3 Resultate Testing

Mit insgesamt 109 Unit Tests über beide Projekte konnte das oben beschriebene Testkonzept eine gute Abdeckung aller zentralen Funktionalitäten erreichen. Wie in den Abbildungen 5.5 und 5.6 dargestellt, wurde eine Testabdeckung von rund 80% erzielt.

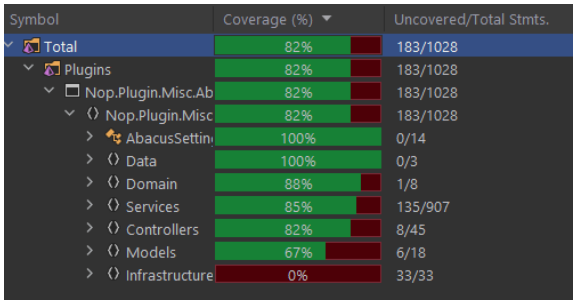

Abbildung 5.5: Test Coverage Abacus Connector

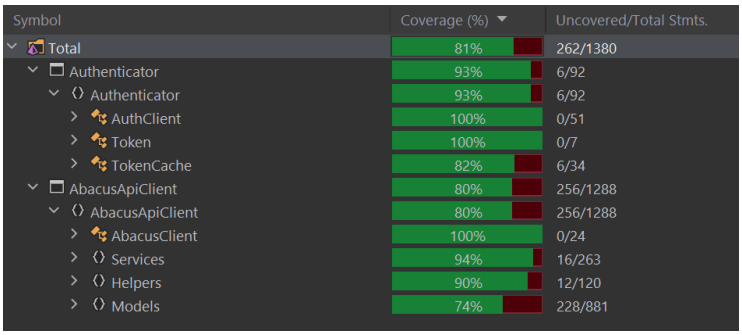

Abbildung 5.6: Test Coverage Abacus Client

Durch die Durchführung von 16 Integrationstests wird die gewünschte Funktionalität unter Einbezug aller Systeme überprüft. Schwieriger gestaltete sich das Integrationstesting für schreibende Zugriffe in das Abacus-System, da es an Möglichkeiten mangelt, das Ergebnis dieser Zugriffe zu verifizieren. Daher wird empfohlen, diese Aspekte zukünftig durch manuelles End-To-End Testing abzudecken.

Kapitel 6

Ergebnisdiskussion

Dieses Kapitel widmet sich der Diskussion der in dieser Studienarbeit erzielten Ergebnisse. Es erfolgt ein Vergleich dieser Ergebnisse mit der ursprünglichen Aufgabenstellung und den in Kapitel 3 definierten Anforderungen. Ziel ist es, Übereinstimmungen sowie eventuelle Abweichungen transparent darzustellen und zu bewerten.

6.1 Resultate der Studienarbeit

Dieser Abschnitt befasst sich mit der von Customize definierten Aufgabenstellung (Anhang D) und den dort festgelegten vier Liefergegenständen:

1. **Anforderungserhebung und -Analyse**: Eine detaillierte Analyse der funktionalen und nicht-funktionalen Anforderungen wurde im Kapitel 3 durchgeführt. Die funktionalen Anforderungen decken alle Funktionen des bisherigen Produkts AbaShop ab und berücksichtigen zudem die wichtigsten Anforderungen der Kunden von Customize, die vom AbaShop bisher nicht abgedeckt wurden.
2. **Produktevaluation**: Die Analyse von fünf externen Shopsystemen lieferte eine solide Grundlage für das Lösungskonzept. Ein Kriterienkatalog ermöglichte eine fundierte Entscheidung basierend auf einer Nutzwertanalyse.
3. **Prototypentwicklung**: Als Ergebnis dieser Arbeit entstand ein Prototyp, der anhand von vier ausgewählten Use Cases zeigt, wie ein integrierter Webshop für Abacus entwickelt werden kann.
4. **Migrationsfähigkeit**: Die Migrationsfähigkeit wird durch die Implementierung des Synchronisationsdienstes gewährleistet. Es wurde jedoch noch nicht dargelegt, wie die bestehenden Logindaten aus dem AbaShop migriert werden könnten.

Ein kritischer Erfolgsfaktor neben den Liefergegenständen ist eine Aussage über die Eignung der Abacus-Schnittstelle. Diese Einschätzung findet sich am Ende der Schnittstellenanalyse im Abschnitt 4.1.8. Es wird aufgezeigt, dass die vorhandenen Möglichkeiten mit der neuen RESTful HTTP-Schnittstelle von Abacus sehr gute Voraussetzungen bieten, eine integrierte Webshoplösung zu implementieren.

6.2 Verifikation der Anforderungen

Für einen aussagekräftigen Proof of Concept wurden vier Use Cases ausgewählt, die die Abacus-Schnittstelle möglichst umfassend abdecken und die zentralen Anforderungen der Kunden von Customize erfüllen. In diesem Abschnitt wird die Umsetzung dieser Use Cases verifiziert.

UC01 - Konfiguration

Die Spezifikation dieses Use Cases ist in Tabelle A.1 zu finden.

Das implementierte Plugin kann von nopCommerce installiert und deinstalliert werden. Abbildung 6.1 zeigt die View, über die das Plugin installiert und deinstalliert werden kann.

Abbildung 6.1: Installationsmenü des Abacus Plugins

Im nopCommerce Backend kann unter der Kategorie 'Abacus' in der Navigationsleiste die Konfiguration aufgerufen werden. Dort kann der Consultant alle nötigen Einstellungen vornehmen, um die Verbindung zu Abacus herzustellen und den Synchronisationsdienst zu steuern. Bei der Einrichtung des Plugins wird ein Healthcheck der Abacus-Schnittstelle durchgeführt, um den Consultant über den Status zu informieren. Abbildung 6.2 zeigt die Konfigurationsseite.

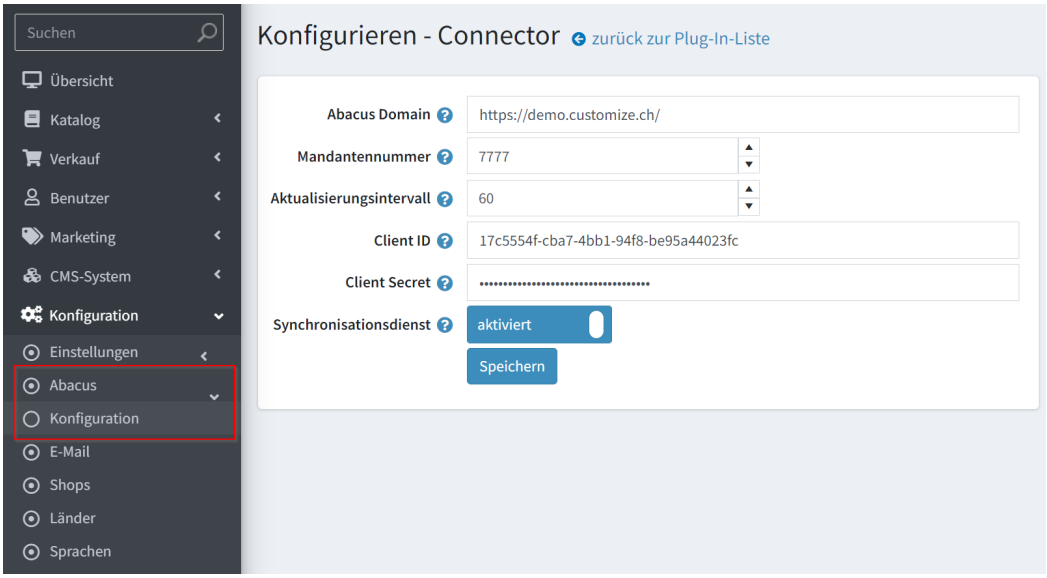

Abbildung 6.2: Konfigurationsseite des Abacus Plugins

✔ Der Use Case ist erfüllt.

UC02 - Produktsuche
Die Spezifikation dieses Use Cases ist in Tabelle A.2 zu finden.

Ein Synchronisationsdienst, der auf einem Scheduled Task basiert, überträgt in einem konfigurierbaren Intervall Stammdaten vom Abacus-System in das Webshopsystem. Die folgenden Entitäten werden synchronisiert:

- **Produkt**: Stammdaten, Texte mit Übersetzungen, Preise, Userfields mit Übersetzungen, Bilder, Lagerbestand
- **Klassierung**: Klassierungstexte mit Übersetzungen, Produktzuordnungen, Bild

Durch die Implementierung einer Teilsynchronisation wird vermieden, dass in jedem Intervall alle Stammdaten geladen werden müssen. Der Dienst holt lediglich die Änderungen seit der letzten Abholung ab.

✔ Der Use Case ist erfüllt.

UC03 - Warenkorb
Die Spezifikation dieses Use Cases ist in Tabelle A.3 zu finden.

Ein Shopper kann im Standardfrontend von nopCommerce Artikel in den Warenkorb legen. Bei jeder Warenkorbmutation sendet das Plugin die Daten an Abacus zur Preisfindung. Falls kundenspezifische Preise oder Rabatte vorhanden sind, werden diese angezeigt.

✔ Der Use Case ist erfüllt.

UC04 - Bestellung
Die Spezifikation dieses Use Cases ist in Tabelle A.4 zu finden.

Nach Abschluss einer Bestellung wird diese an Abacus übermittelt. Kunden aus nopCommerce, die in Abacus eine Kundennummer erhalten, werden durch die Teilsynchronisation im E-Commerce-System mit den Abacus relevanten Daten aktualisiert. Zukünftige Bestellungen können dann mit dieser Kundennummer übermittelt werden, sodass sie in der Abacus-Inbox ohne manuellen Eingriff verarbeitet werden können. Eine detaillierte Beschreibung dieses Prozesses befindet sich im Abschnitt 5.2.5.

Dieser Use Case hat noch Optimierungspotential. Übermittlungen an Abacus erfolgen asynchron, wodurch eine Bestellung im Webshop erfolgreich sein könnte, aber nicht an Abacus gesendet wird. Die Überlegungen dazu befinden sich in Architekturentscheidung 5 (5.4.5). Bestellungen mit Übermittlungsfehlern werden in nopCommerce markiert. Zukünftig wird ein Benachrichtigungsmechanismus benötigt, um den Shopbetreiber über solche Fälle zu informieren, sowie eine Möglichkeit, die Übermittlung manuell erneut anzustossen.

? Der Use Case ist erfüllt, jedoch mit Optimierungspotential.

Nicht-funktionale Anforderungen

In Tabelle 6.1 wird die Evaluation der nicht-funktionalen Anforderungen dargestellt.
Hierbei ist wichtig zu beachten, dass sich NFR01, NFR02 und NFR03 mit den Standardfrontend von nopCommerce evaluiert wurden. Bei einer potentiellen Eigenimplementation lassen sich diese Anforderungen selbst steuern.

NFR	**Soll**	**Ist**	**Erfüllt**
NFR01 - Performance	80 Punkte im Lighthouse-Report / maximale Ladezeit einer Seite von 3 Sekunden	Es wurden 78 Punkte im Lighthouse-Report (Abbildung 6.3) erreicht, die maximale Ladezeit beträgt jedoch nur 2.1 Sekunden.	✗
NFR02 - Responsivität	Webshop ist durchgängig auf allen Bildschirmgeräten nutzbar	Eine Evaluation mittels Usertests wurde ausgelassen, da in dieser Arbeit mit dem Standardfrontend von nopCommerce gearbeitet wurde. Dies soll in einem nächsten Schritt durch eine Eigenimplementation abgelöst werden.	-
NFR03 - SEO	80 Punkte im Lighthouse-Report / sprechende URLs	Es wurden 100 Punkte im Lighthouse-Report erreicht (Abbildung 6.3). Der Abacus Connector vergibt sprechende SEO-Namen für Produkte und Klassierungen.	✔
NFR04 - Testing	70% Test-Coverage	Die Test-Coverage liegt bei rund 80% (siehe 5.5.1).	✔
NFR05 - Zukunftssicherheit	Programmierung gegen Interfaces	In beiden C#-Projekten, dem Abacus Connector und dem Abacus Client, wird ausschliesslich gegen Interfaces programmiert und die Services werden mittels Dependency Injection bereitgestellt.	✔

Tabelle 6.1: Evaluation der nicht-funktionalen Anforderungen

Abbildung 6.3 zeigt den Google Lighhouse Report mit einer Webshopinstanz basierend auf dem Mustermandanten von Abacus.

Abbildung 6.3: Lighthouse Report

Die Anforderungen NFR08 bis NFR12, die an das evaluierte E-Commerce-System gestellt wurden, wurden im Abschnitt 4.2.10 verifiziert.

6.3 Ausblick

Diese Semesterarbeit demonstriert mit dem entwickelten Proof of Concept, dass ein in das ERP integrierter Webshop unter Nutzung der RESTful HTTP-Schnittstelle von Abacus realisierbar ist. Durch die umfassende Anforderungsanalyse wurden sowohl die aktuellen Funktionalitäten des AbaShops als auch die zusätzlichen Wünsche der Kunden des Industriepartners dargelegt. Die Architektur des Prototyps bietet eine solide Grundlage für die Implementierung weiterer Use Cases.

Da das evaluierte Shopsystem nopCommerce jedoch die Anforderung NFR10 - Headless (siehe Tabelle 3.15) nicht erfüllt, muss nun untersucht werden, wie diese Anforderung in die bestehende Architektur integriert werden kann. Angesichts der Tatsache, dass das Standardfrontend von nopCommerce nicht den Ansprüchen des Industriepartners genügt, ist die Entwicklung einer Headless-Lösung für das E-Commerce-System erforderlich. Diese Basis ermöglicht es, die Implementierung eines eigenen Frontends in Angriff zu nehmen.

Eine Herausforderung hierbei ist die breite Kundenbasis, da jeder Kunde individuelle visuelle Anforderungen an seinen Webshop stellt. Um eine wartbare Lösung zu gewährleisten, muss der Kern des Frontends, einschliesslich aller wesentlichen Logiken, bei allen Kunden identisch bleiben. Dadurch wird eine kontinuerliche Updatefähigkeit garantiert. Es bedarf der Entwicklung eines Theming-Konzepts, das in einer zusätzlichen Schicht die gesamte Darstellung des Frontends parametrisieren kann. Die Herausforderung dieses Themingkonzeptes ist, dass nicht nur die Farbgebung beeinflusst werden muss, sondern die beinhalteten Daten und Formen wie auch Platzierung einer Komponente kann bei jedem Kunden unterschiedlich sein. Beispielsweise variiert bei jedem Kunden, welche Daten aus dem Produktstamm auf der Detailansicht des Produktes dargestellt werden. Auch die Platzierung und Gestaltung einzelner Komponenten, wie etwa die Navigation durch die Produktklassifizierung, sind oft individuell gestaltet.

Anhang

Anhang A

Fully Dressed Beschreibung funktionale Anforderungen

UC01 - Konfiguration

Ein Consultant verbindet den Shop mit Abacus zur Synchronisierung von Stammdaten.

Primärer Akteur	Consultant
Stakeholder und Interessen	• **Consultant:** Möchte in einfach Schritten ein Abacus ERP mit dem Webshop verbinden. • **Shopbetreiber:** Möchte den Shop mit Abacus verbinden, um Stammdaten automatisch zu synchronisieren und den Betrieb effizient zu führen. • **Shopper:** Möchte im Shop aktuelle Produktdaten sehen.
Preconditions	• Das Admin-Panel des Shops ist betriebsbereit. • Der Akteur ist im Admin-Panel angemeldet und verfügt über die erforderlichen Berechtigungen. • Die notwendigen Schnittstellen und Konfigurationen für die Verbindung zu Abacus sind eingerichtet und verfügbar.
Postconditions	• Der Shop ist erfolgreich mit Abacus verbunden, und die automatische Synchronisierung von Stammdaten wurde konfiguriert • Produkte aus Abacus können im Adminbereich eingesehen werden

Haupterfolgsszenario	1. Der Consultant
	  • richtet den API Zugriff in Abacus ein.
	  • greift auf das Admin-Panel zu.
	  • navigiert zur Konfigurationsseite im Admin-Bereich.
	  • wählt die Option zur Verbindung mit Abacus.
	  • gibt die notwendigen Informationen ein.
	2. Das System überprüft die Verbindung und die Eingaben.
	3. Wenn die Überprüfung erfolgreich ist, zeigt das System eine Bestätigung an, dass die Verbindung hergestellt wurde und die Stammdaten-Synchronisierung konfiguriert wurde.

Alternatives Szenario	Wenn die Verbindung fehlschlägt oder die Überprüfung der Eingaben nicht erfolgreich ist, zeigt das System eine Fehlermeldung an.
Häufigkeit	Einmalig

Tabelle A.1: Fully Dressed Use Case 01: Konfiguration

UC02 - Produktsuche

Ein Shopper durchsucht das Produktangebot über die Produktklassierung.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte Produkte basierend über die Produktklassierung in seiner gewählten Sprache suchen und finden. Kann Detailinformationen zu einem gewünschten Produkt einsehen. • **Shopbetreiber:** Hat Interesse an der Wirksamkeit der Produktklassifikationen und der Kundeninteraktion damit.
Preconditions	• Der Webshop ist Betriebsbereit und die Produktdaten aus Abacus stehen im Webshop zur Verfügung(A.1). • Der Webshop unterstützt die gewünschte Benutzersprache, wobei diese von Abacus als Mastersystem vorgegeben werden.
Postconditions	• Das System zeigt eine Liste relevanter Produkte basierend auf der ausgewählten Produktklassifikation in der gewählten Benutzersprache an. • Der Shopper hat die Möglichkeit, individuelle Detailinformationen für jedes Produkt einzusehen.
Haupterfolgsszenario	1. Der Shopper wählt die Option zur Navigation über Produktklassifikationen. 2. Das System zeigt eine Liste von verfügbaren Produktklassifikationen in der gewählten Benutzersprache an. 3. Der Shopper wählt eine Produktklassifikation aus der Liste. 4. Das System zeigt eine Liste von Produkten an, die dieser Produktklassifikation zugeordnet sind. 5. Der Shopper kann auf ein Produkt klicken, um Details in seiner gewählten Sprache anzuzeigen.

Alternatives Szenario	Wenn keine Produkte in der ausgewählten Produktklassifikation vorhanden sind, zeigt das System eine Nachricht in der gewählten Benutzersprache an, die darauf hinweist, dass keine Ergebnisse gefunden wurden.
Häufigkeit	Sehr hoch

Tabelle A.2: Fully Dressed Use Case 02: Produktsuche

UC03 - Warenkorb

Ein Shopper kann Produkte zum Warenkorb hinzufügen und bearbeiten.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte Produkte zum Warenkorb hinzufügen, entfernen und bearbeiten, um eine Einkaufsliste zu erstellen. • **Shopbetreiber:** Ziel ist es, die Interaktion des Shoppers mit dem Warenkorb effizient zu gestalten um eine möglichst tiefe Barriere für einen Einkauf zu setzen. • **Abacus:** Möchte seine eigene Preisfindung auf dem Warenkorb durchführen.
Preconditions	• Die Produktdaten sind verfügbar (A.1)
Postconditions	• Der Warenkorb des Shoppers wurde aktualisiert und enthält alle Artikel mit den aktuellen Konditionen.
Haupterfolgsszenario	1. Der Shopper durchsucht den Webshop nach Produkten (A.2). 2. Der Shopper wählt Produkte aus und fügt sie zum Warenkorb hinzu. 3. Das System sendet den Warenkorb zu Abacus um eine finale Preisfindung durchzuführen. 4. Der Shopper kann die Anzahl der Produkte im Warenkorb bearbeiten oder Produkte entfernen.
Alternatives Szenario	Wenn der Warenkorb leer ist, erhält der Shopper eine Nachricht, die ihn zur Auswahl von Produkten auffordert.
Häufigkeit	Hoch

Tabelle A.3: Fully Dressed Use Case 03: Warenkorb

UC04 - Bestellung

Ein Shopper kann eine Bestellung tätigen.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte einen Verkauf abschliessen. • **Shopbetreiber:** Möchte möglichst hohe Verkaufszahlen generieren. • **Abacus:** Hat Interesse an der korrekten Erfassung der Verkaufsdaten und der Verarbeitung der Bestellung.
Preconditions	• Der Shopper hat Produkte im Warenkorb (A.3).
Postconditions	• Der Verkauf des Shoppers wurde erfolgreich erfasst und ist an Abacus übermittelt.
Haupterfolgsszenario	1. Der Shopper geht zum Warenkorb und wählt die Option zur Bestellung. 2. Das System zeigt ein Formular an, in dem der Shopper zusätzliche Informationen eingeben kann (z. B., Lieferadresse, Versandart, Zahlungsinformationen). 3. Der Shopper gibt die erforderlichen Informationen ein. 4. Das System überprüft die Informationen und die Verfügbarkeit der Produkte. 5. Der Shopper bestätigt die Bestellung. 6. Das System erfasst die Bestellung des Shoppers und generiert einen Auftrag. 7. Die Verkaufsdaten werden an Abacus übertragen. 8. Das System verschickt eine Auftragsbestätigung via Mailsystem.
Alternatives Szenario	Wenn die Übertragung der Verkaufsdaten an Abacus fehlschlägt, generiert der Mailprovider eine E-Mail und verschickt diese an den Shopper.
Häufigkeit	Hoch

Tabelle A.4: Fully Dressed Use Case 04: Bestellung

Hinweis: Dieser Use Case ist als 'Must Have' markiert und soll entsprechend im finalen Prototypen umgesetzt werden. Dabei wird aber ausschliesslich eine Bestellung auf Rechnung umgesetzt. Andere

Zahlungarten wie zum Beispiel Kreditkarte oder Twint sind ausserhalb vom Projektscope, da dort externe Systeme angebunden werden müssen.

UC05 - Erweiterte Produktsuche

Ein Shopper kann mittels Filter oder Volltextsuche nach Artikeln suchen.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopbetreiber:** Möchte seine Artikel für die Shopper möglichst einfach auffindbar machen.
Preconditions	• Die Produktdaten sind synchronisiert (A.1). • Auf den Produkten sind filterbare Userfields vorhanden.
Postconditions	Der Shopper hat alle Artikel gefunden, die seinen Suchkriterien entsprechen.
Haupterfolgsszenario	**Der Shopbetreiber** 1. navigiert im Adminbereich zur Attributverwaltung. 2. legt fest, welche Attribute von den Artikeln gefiltert werden können. **Der Shopper** 1. gibt in der Suchfunktionen ein Stichwort ein. 2. Das System zeigt Suchergebnisse basierend auf dem eingegebenen Suchbegriff an. 3. Der Shopper schränkt die Ergebnisse mittels Filter auf den Attributen der gefunden Artikel weiter ein. 4. Der Shopper kann Artikel aus den gefilterten Ergebnissen auswählen und Detailinformationen dazu anzeigen.
Alternatives Szenario	Filterung ohne vorherige Suche. **Der Shopper** 1. navigiert über die Produktklassierung (A.2) 2. schränkt die Erbenisse mittels Filter auf den Attributen der gefunden Artikel weiter ein. 3. kann Artikel aus den gefilterten Ergebnissen auswählen und Detailinformationen dazu anzeigen.
Häufigkeit	Mittel

Tabelle A.5: Fully Dressed Use Case 05: Erweiterte Produktsuche

UC06 - Registrierung

Ein Shopper erstellt im Webshop einen Account.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte einen neuen Account im Webshop erstellen, um künftige Einkäufe zu erleichtern. • **Abacus:** Durch die Registrierung können zukünftige Bestellungen automatisiert dem korrekten Kunden zugeordnet werden.
Preconditions	• Der Shopper hat noch keinen Account im Webshop.
Postconditions	• Der Shopper hat erfolgreich einen neuen Account erstellt, der sowohl im Webshop als auch in Abacus erfasst wurde.
Haupterfolgsszenario	1. Der Shopper navigiert zur Registrierungsseite im Webshop. 2. Das System zeigt ein Registrierungsformular an, in dem der Shopper persönliche Informationen eingeben kann (z. B., Name, E-Mail-Adresse, Passwort). 3. Der Shopper gibt die erforderlichen Informationen ein. 4. Das System überprüft die Eingaben auf Richtigkeit und Verfügbarkeit der E-Mail-Adresse. 5. Der Shopper bestätigt die Registrierung. 6. Das System erfasst die Registrierungsdaten des Shoppers und generiert einen neuen Account. 7. Die Shopper-Daten werden an Abacus übertragen. 8. Das Mailsystem generiert eine Bestätigung.

Alternatives Szenario	1. Der Shopper navigiert zur Registrierungsseite im Webshop. 2. Das System zeigt ein Registrierungsformular an, in dem der Shopper persönliche Informationen eingeben kann (z. B., Name, E-Mail-Adresse, Passwort). 3. Der Shopper gibt die erforderlichen Informationen ein. 4. Das System überprüft die Eingaben auf Richtigkeit und Verfügbarkeit der E-Mail-Adresse. 5. Das System zeigt an, dass die E-Mail Adresse bereits einen Account besitzt. 6. Der Shopper wählt, dass er das Passwort vergessen hat (Alternative: Shopper navigiert zur Loginseite und loggt sich ein). 7. Das Mailsystem generiert eine E-Mail um das Passwort neu zu vergeben.
Häufigkeit	Mittel

Tabelle A.6: Fully Dressed Use Case 06: Registrierung

UC07 - Login	
Ein Shopper meldet sich im Webshop an.	
Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte sich in seinen bestehenden Account im Webshop anmelden, um Einkäufe zu tätigen. • **Abacus:** Durch den Login kann der Shopper identifiziert und Bestellungen seiner Kundennummer zugeordnet werden.
Preconditions	• Der Shopper hat bereits einen Account im Webshop erstellt (A.7).
Postconditions	• Der Shopper hat sich erfolgreich in seinen Account im Webshop angemeldet.
Haupterfolgsszenario	1. Der Shopper navigiert zur Login-Seite im Webshop. 2. Das System zeigt ein Login-Formular an, in das der Shopper seine E-Mail-Adresse und das Passwort eingeben kann. 3. Der Shopper gibt seine Anmeldeinformationen ein. 4. Das System überprüft die Eingaben auf Richtigkeit und Existenz eines entsprechenden Accounts. 5. Der Shopper wird erfolgreich in seinen Account eingeloggt und kann Einkäufe tätigen.
Alternatives Szenario	Wenn die Anmeldeinformationen nicht korrekt sind oder der Account nicht existiert: 1. Das System zeigt eine Fehlermeldung an und informiert den Shopper über das Problem. 2. Der Shopper kann seine Anmeldeinformationen erneut eingeben oder das Passwort zurücksetzen.
Häufigkeit	Mittel

Tabelle A.7: Fully Dressed Use Case 07: Login

UC08 - Cross Selling

Ein Shopper erhält auf der Detailseite eines Artikels Vorschläge für ähnliche Produkte.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte auf der Detailseite eines Artikels Vorschläge für ähnliche Produkte erhalten, um seine Einkaufsoptionen zu erweitern und relevante Produkte zu entdecken. • **Shopbetreiber:** Möchte mittels Cross Selling seinen Umsatz maximieren.
Preconditions	• Der Shopper befindet sich auf der Detailseite eines Artikels. • Der Artikel hat in Abacus ein oder mehrere ähnliche Artikel hinterlegt.
Postconditions	• Dem Shopper wurden erfolgreich ähnliche Artikel vorgeschlagen.
Haupterfolgsszenario	1. Der Shopper öffnet die Detailseite eines Artikels im Webshop. 2. Das System zeigt dem Shopper eine Liste von ähnlichen Produkten an. 3. Der Shopper kann die vorgeschlagenen Produkte anzeigen und weitere Informationen erhalten.
Alternatives Szenario	-
Häufigkeit	Mittel

Tabelle A.8: Fully Dressed Use Case 08: Cross Selling

U09 - Konditionen und Rabatte

Ein Shopper kann mit seinen individuellen Konditionen einkaufen.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte seine individuellen Konditionen und Rabatte in Anspruch nehmen. • **Shopbetreiber:** Hat Interesse daran, dass die in Abacus erfassten Konditionen im Webshop korrekt angezeigt und angewendet werden.
Preconditions	• Der Shopper ist im Webshop eingeloggt (A.8). • Die spezifischen Konditionen und Rabatte des Shoppers sind in Abacus erfasst und im Webshop verfügbar.
Postconditions	Der Shopper hat seine spezifischen Konditionen und Rabatte in der Produktübersicht und im Warenkorb erfolgreich angezeigt bekommen.
Haupterfolgsszenario	1. Der Shopper meldet sich im Webshop an. 2. Das System zeigt dem Shopper seine persönlichen Konditionen und Rabatte an, die in Abacus erfasst sind. 3. Der Shopper durchsucht den Webshop und wählt Produkte aus. 4. Beim Hinzufügen von Produkten zum Warenkorb werden automatisch die spezifischen Konditionen und Rabatte des Shoppers angewendet.
Alternatives Szenario	-
Häufigkeit	Mittel

Tabelle A.9: Fully Dressed Use Case 09: Konditionen und Rabatte

UC10 - Einkaufsverlauf anzeigen

Ein Shopper kann seinen Einkaufsverlauf und den Verarbeitungsstatus vergangener Bestellungen einsehen.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte seinen Einkaufsverlauf im System einsehen, um Informationen zu vergangenen Bestellungen und deren Verarbeitungsstatus zu erhalten. • **Shopbetreiber:** Kann Anfragen beim Kundenservice reduzieren, da der Shopper den Status aktueller Bestellungen selbst einsehen kann.
Preconditions	• Der Shopper ist im Webshop angemeldet (A.8). • Es liegen vergangene Bestellungen vor (A.4).
Postconditions	• Der Shopper hat erfolgreich seinen Einkaufsverlauf und den Verarbeitungsstatus vergangener Bestellungen eingesehen.
Haupterfolgsszenario	1. Der eingeloggte Shopper navigiert zu seinem Benutzerkonto im Webshop. 2. Das System zeigt eine Übersicht des Einkaufsverlaufs mit vergangenen Bestellungen an. 3. Der Shopper wählt eine vergangene Bestellung aus, um weitere Details anzuzeigen. 4. Das System System fragt bei Abacus den Bestellstatus ab und zeigt ihn an.
Alternatives Szenario	Wenn keine vergangenen Bestellungen für den Shopper vorhanden sind oder der Verarbeitungsstatus nicht abgerufen werden kann, wird eine entsprechende Nachricht angezeigt.
Häufigkeit	Mittel

Tabelle A.10: Fully Dressed Use Case 10: Einkaufsverlauf anzeigen

UC11 - Produktbewertungen

Ein Shopper kann Produktbewertungen für gekaufte Produkte abgeben und einsehen.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte Produktbewertungen und -rezensionen für gekaufte Produkte abgeben, um anderen Shoppern bei ihrer Kaufentscheidung zu helfen und Feedback an den Webshop zu geben. • **Shopbetreiber:** Möchte den Communitygedanken seiner Kunden stärken.
Preconditions	• Der Shopper ist im Webshop angemeldet (A.8). • Es liegen vergangene Bestellungen vor (A.4).
Postconditions	• Die Produktbewertung und -rezension des Shopper wurde erfolgreich im Webshop gespeichert und ist für andere Shopper sichtbar.
Haupterfolgsszenario	1. Der Shopper navigiert zu seiner Einkaufsübersicht im Webshop. 2. Das System zeigt eine Liste der gekauften Produkte an. 3. Der Shopper wählt ein Produkt aus, für das er eine Bewertung oder Rezension abgeben möchte. 4. Das System zeigt eine Bewertungs- und Rezensionsseite für das ausgewählte Produkt an. 5. Der Shopper gibt seine Bewertung und Rezension ein. 6. Das System speichert die Bewertung des Shopper für das Produkt. 7. Die Bewertung und Rezension werden auf der Produktseite für andere Shopper sichtbar.
Alternatives Szenario	-
Häufigkeit	Selten

Tabelle A.11: Fully Dressed Use Case 11: Produktbewertungen

UC12 - Wunschliste

Ein Shopper erstellt Wunschliste eine von Artikeln.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte eine bequeme Möglichkeit haben, Produkte, die er später kaufen möchte, zu speichern und zu organisieren.
Preconditions	• Der Shopper ist im Webshop angemeldet (A.8). • Der Shopper hat mindestens ein Produkt im Webshop angesehen.
Postconditions	• Der Shopper hat erfolgreich eine Wunschliste erstellt und Produkte hinzugefügt. Die Wunschliste steht bei einem späteren Login weiterhin zur Verfügung.
Haupterfolgsszenario	1. Der eingeloggte Shopper navigiert zu einem Produkt, das er zu seiner Wunschliste hinzufügen möchte. 2. Das System zeigt eine Schaltfläche oder Option 'Zur Wunschliste hinzufügen' auf der Produktseite an. 3. Der Shopper klickt auf diese Schaltfläche, um das Produkt zur Wunschliste hinzuzufügen. 4. Das System bestätigt die Hinzufügung zur Wunschliste und gibt dem Shopper die Möglichkeit, die Wunschliste anzuzeigen oder weitere Produkte hinzuzufügen. 5. Der Shopper kann seine Wunschliste jederzeit aufrufen, um die gespeicherten Produkte anzuzeigen und zu verwalten.
Alternatives Szenario	-
Häufigkeit	Selten

Tabelle A.12: Fully Dressed Use Case 12: Wunschliste

UC13 - Rücksendungen und Umtausch

Ein Shopper löst eine Rücksendung oder Umtausch aus.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte die Möglichkeit haben, Produkte zurückzusenden oder umzutauschen, die Mängel aufweisen oder nicht den Erwartungen entsprechen. • **Shopbetreiber:** Kann Anfragen beim Kundenservice reduzieren, da der Shopper selbstständig einen Umtausch/Rücksendung auslösen kann.
Preconditions	• Der Shopper ist im Webshop angemeldet (A.8). • Der Shopper hat mindestens ein Produkt im Webshop gekauft (A.4).
Postconditions	• Die Rücksendung oder der Umtausch wurde erfolgreich initiiert und verarbeitet.
Haupterfolgsszenario	1. Der eingeloggte Shopper navigiert zu seinem Benutzerkonto. 2. Das System zeigt eine Liste der bisherigen Einkäufe des Shoppers an (A.10). 3. Der Shopper wählt den Kauf aus, für den er eine Rücksendung oder einen Umtausch initiieren möchte. 4. Das System zeigt dem Shopper eine Option zur Rücksendung oder zum Umtausch der ausgewählten Produkte an. 5. Der Shopper wählt diese Option aus und gibt den Grund für die Rücksendung oder den gewünschten Umtausch an. 6. Das System führt den Shopper durch den Prozess, um eine Rücksendungsanfrage zu erstellen. 7. Die Anfrage wird an Abacus übermittelt. 8. Der Shopper erhält vom Mailsystem eine Bestätigung, dass die Rücksendungsanfrage erfolgreich erstellt wurde, und erhält Informationen zur Vorgehensweise.

Alternatives Szenario	Falls ein Umtausch ausgeschlossen ist: 1. Das System zeigt dem Benutzer an, wieso ein Umtausch oder Rücksendung nicht möglich ist.
Häufigkeit	Selten

Tabelle A.13: Fully Dressed Use Case 13: Rücksendungen und Umtausch

UC14 - Gutscheinverwaltung

Ein Shopper möchte einen Gutschein kaufen oder eine Bestellung damit bezahlen.

Primärer Akteur	Shopper
Stakeholder und Interessen	• **Shopper:** Möchte Gutscheine erwerben und diese für Rabatte bei Einkäufen nutzen. • **Shopbetreiber:** Möchte Gutscheine erstellen, verwalten und verkaufen, um Kunden zu binden.
Preconditions	• Der Shopper ist im Webshop angemeldet (A.8).
Postconditions	• Der Shopbetreiber hat erfolgreich einen Gutschein erstellt und im System verfügbar gemacht. • Der Shopper hat einen Gutschein erworben und kann ihn für zukünftige Einkäufe verwenden.
Haupterfolgsszenario	**Sicht Shopbetreiber:** 1. Der Shopbetreiber meldet sich im Adminbereich des Systems an. 2. Das System bietet eine Option zur Gutscheinverwaltung an. 3. Der Shopbetreiber erstellt einen neuen Gutschein, indem er die Details wie Rabattbetrag, Gültigkeitsdauer und Verwendungsbedingungen festlegt. 4. Das System generiert einen eindeutigen Gutscheincode und macht den Gutschein im Webshop verfügbar. **Sicht Shopper:** 1. Der Shopper wählt im Webshop die Option zum Kauf eines Gutscheins aus. 2. Das System zeigt eine Liste der verfügbaren Gutscheine an. 3. Der Shopper wählt einen Gutschein aus und gibt die gewünschte Menge an. 4. Das System berechnet den Gesamtbetrag und ermöglicht die Bezahlung. 5. Nach erfolgreicher Bezahlung erhält der Shopper den Gutscheincode per E-Mail oder in seinem Konto. 6. Der Shopper kann den Gutscheincode bei zukünftigen Einkäufen im Webshop verwenden, um den Rabatt anzuwenden.

Alternatives Szenario	-
Häufigkeit	Mittel

Tabelle A.14: Fully Dressed Use Case 14: Gutscheinverwaltung

UC15 - Content Management

Der Shopbetreiber kann selbstständig Content-Seiten des Systems erfassen und editieren. 

Primärer Akteur	Shopbetreiber
Stakeholder und Interessen	• **Shopbetreiber:** Möchte ohne Programmierkentnisse Content-Seiten einfach erstellen, bearbeiten und veröffentlichen, um Kunden aktuelle Informationen bereitzustellen. • **Shopper:** Profitieren von aktuellen und informativen Inhalten auf der Website.
Preconditions	• Der Shopbetreiber ist im Adminbereich des Systems angemeldet.
Postconditions	Die Content-Seite wurde erfolgreich erstellt oder aktualisiert und ist für die Shopper sichtbar.
Haupterfolgsszenario	1. Der Shopbetreiber meldet sich im Adminbereich des Systems an. 2. Das System bietet eine Option zur Content-Verwaltung an. 3. Der Shopbetreiber wählt die zu bearbeitende Content-Seite aus oder erstellt eine neue. 4. Das System zeigt einen WYSIWYG-Editor (What You See Is What You Get) zur Bearbeitung des Inhalts an. 5. Shopbetreiber kann den Text formatieren, Bilder und Multimedia-Inhalte hinzufügen und den Inhalt speichern. 6. Nach dem Speichern wird die aktualisierte Content-Seite im Webshop angezeigt.
Alternatives Szenario	-
Häufigkeit	Mittel

Tabelle A.15: Fully Dressed Use Case 15: Content Management

Anhang B

Abacus Schnittstelle Beispiele

Der nachfolgende Codeausschnitt zeigt, wie ein minimaler Body bei einem POST eines Verkaufsauftrages aussehen kann.

B.1 Abacus API: Verkaufsauftrag erstellen

```
{
  "InboxOrderRequest": {
    "TaxType": "Included",
    "RecalculationMethod": "NoRecalculation",
    "Shopper": {
      "Id": "a095b567-13b8-4e29-89e1-6d2a1f0e7e83",
      "Account": "demo@customize.ch"
    },
    "Summary": {
      "CurrencyId": "CHF",
      "TotalAmountIncludingTax": 352,
      "TotalAmountExcludingTax": 326.83
    },
    "PositionList": [
      {
        "Quantity": 1,
        "PriceInfo": {
          "PriceFromApi": true,
          "PerQuantityExcludingTax": 326.83,
          "PositionTotalExcludingTax": 326.83,
          "PerQuantityExcludingTaxRelevant": 326.83,
          "TaxInfo": {
            "Code": 311,
            "Rate": 7.7,
            "Amount": 25.17
          }
        },
        "Product": {
          "Name": "0955"
        }
      }
    ],
    "CustomerSubject": {
      "Id": 234,
      "FirstName": "Ramon",
      "Name": "Ebneter",
      "Address": {
        "Street": "Kaustrasse",
        "HouseNumber": "46",
        "PostCode": "9050",
        "City": "Appenzell"
```

```
        }
      }
    }
  }
```

Listing B.1: Abacus API Verkaufsauftrag Body

B.2 Abacus API: Preisfindung Warenkorb

Die beiden nachfolgenden Codeauschnitte zeigen, wie der Body bei einer Abfrage des Preises von einem Warenkorb aussieht und was die Struktur der Antwort von Abacus ist.

```
{
  "ProductsPricingRequest": {
    "Currency": "CHF",
    "CustomerNumber": 59,
    "CalculationDate": "2023-10-06",
    "Positions": [
      {
        "ProductId": 3,
        "VariantId": 0,
        "Quantity": 3
      }
    ]
  }
}
```

Listing B.2: Abacus API Warenkorb Preisfindung Body

```
{
  "Positions": [
    {
      "RequestKey": "123",
      "PriceType": "LIST",
      "PerUnitValue": {
      "PriceInclTax": 284.32800000,
      "PriceExclTax": 264.00000000,
      "PriceInclTaxBeforDiscount": 379.10400000,
      "PriceExclTaxBeforDiscount": 352.00000000
      },
      "QuantityDetail": {
        "Ordered": 3.00
      },
      "TaxDetail": {
        "Code": "311",
        "Rate": 7.70
```

```
        },
         "DiscountDetails": [
          {
            "Type": "STANDARD",
            "Percent": 25.00,
            "UseSubTotal": true
          }
        ]
      }
    ]
  }
```

Listing B.3: Abacus API Warenkorb Preisfindung Antwort

Anhang C

Architectural Decision Records

ADR Nr.	1
Titel	Persistierung von Produktdaten
Datum	20.09.2023
Status	Akzeptiert
Kontext	Das Abrufen aller Produktdaten über die Abacus API dauert bis zu 7 Sekunden.
Entscheidung	Abacus kann nicht direkt als Backend verwendet werden; es wird ein System evaluiert, das die Stammdaten redundant persistieren kann.
Begründung	Eine Antwortzeit von bis zu 7 Sekunden für das Laden einer Seite ist aus der Perspektive der Nutzer gemäss NFR01 (Tabelle 3.6) inakzeptabel. Zudem fehlen der API derzeit wichtige Abfragemöglichkeiten, wie beispielsweise für alle Produkte, die auf der Startseite angezeigt werden sollen.
Konsequenzen	Zu den positiven Konsequenzen zählen verkürzte Antwortzeiten des Webshops. Durch die Entkopplung von der Abacus API können flexible Abfragemöglichkeiten implementiert werden. Negative Konsequenzen sind die Herausforderungen im Umgang mit dem Cache; die redundanten Stammdaten müssen stets aktuell und korrekt gehalten werden.

Tabelle C.1: Beschreibung ADR 01: Persistierung von Produktdaten

ADR Nr.	2
Titel	Entwicklung des Abacus API Client als eigenständiges Projekt
Datum	05.10.2023
Status	Akzeptiert
Kontext	Das Abacus ERP stellt eine RESTful HTTP-Schnittstelle zur Verfügung. In dieser Arbeit wird die Machbarkeit einer integrierten Webshoplösung unter Verwendung dieser neuen Schnittstelle untersucht (4.1). Dabei hat sich gezeigt, dass bestimmte API-Aufrufe aus Sicht der Verbraucher zusammengeführt werden können und zukünftig eine Lösung benötigt wird, um verschiedene Versionen der Schnittstelle zu unterstützen.
Entscheidung	Es hat sich herausgestellt, dass es unabhängig vom gewählten E-Commerce-System von Vorteil ist, ein separates Class-Library-Projekt zur Abstraktion der API-Logik zu entwickeln. Dieses Projekt kann vom Verbraucher als DLL-Artefakt verwendet werden und könnte später möglicherweise als NuGet-Paket zur Verfügung gestellt werden.
Konsequenzen	Eine positive Auswirkung ist die Wiederverwendbarkeit des Clients für andere Projekte der Customize AG. Darüber hinaus ermöglicht die Abstraktion der API-Logik eine effizientere Nutzung, da beispielsweise beim Aufruf von ‘getProduct()‘ im Hintergrund mehrere Anfragen an Abacus gestellt werden können. Eine negative Konsequenz ist der anfängliche höhere Aufwand und die erforderliche Infrastruktur, um diesen Client dem Connector bereitzustellen (dies wird in der CI/CD-Pipeline gelöst, welche die DLL vom Client in das Repository des Connectors überträgt). Mit der Veröffentlichung neuer Versionen der Abacus API kann der Client versioniert werden, um Verbrauchern die Nutzung verschiedener Schnittstellenversionen zu ermöglichen.

Tabelle C.2: Beschreibung ADR 02: Trennung von AbacusApiClient und AbacusConnector

ADR Nr.	3
Titel	Verwendung von Simple.OData.Client für die API-Kommunikation
Datum	06.10.2023
Status	Akzeptiert
Kontext	Das Abacus ERP stellt eine RESTful HTTP-Schnittstelle im OData-Standard zur Verfügung. Diese Schnittstelle bietet den Vorteil eines $metadata-Endpoints, über den alle Typen der Attribute von den Entitäten der Schnittstelle abgerufen werden können. Dadurch ist es möglich, Typinformationen von der API zu erhalten, bei denen beispielsweise Datumswerte bereits als Datumsfelder interpretiert werden.
Entscheidung	Wir haben uns dafür entschieden, den Simple.OData.Client für die Kommunikation mit der OData-API des Abacus ERP zu verwenden. Diese Library bietet die Möglichkeit, die Antworten direkt in C#-Modelle zu mappen. Es wurde entschieden dies nicht zu nutzen und anstelle typisierte Dictionaries für die Entgegennahme der Antworten zu verwenden. Dies geschah, da einige Strukturen der Antworten nicht wie gewünscht (zum Beispiel die Userfields) waren. Dies ermöglicht eine flexiblere Verarbeitung der Daten und eine einfachere Anpassung an mögliche Änderungen in der API-Struktur.
Konsequenzen	Die Verwendung von typisierten Dictionaries erfordert zusätzlichen Code zur Umwandlung der API-Antworten in saubere C#-Modelle. Dieser Code wurde im API-Client implementiert, um die Verarbeitung der Daten zu erleichtern. Obwohl diese Entscheidung zu mehr Komplexität im Code geführt hat, hat sie uns die Möglichkeit gegeben, besser mit den unerwarteten Strukturen in den API-Antworten umzugehen und die Flexibilität für zukünftige Änderungen zu erhöhen. Allfällige Bugs durch Abacus Updates können damit gegebenenfalls abgefangen werden.

Tabelle C.3: Beschreibung ADR 03: Verwendung von Simple.OData.Client für die API-Kommunikation

ADR Nr.	4
Titel	Einsatz von nopCommerce als E-Commerce-Framework
Datum	20.10.2023
Status	Akzeptiert
Kontext	Gemäss der Aufgabenstellung wird evaluiert, ob eine komplette Eigenentwicklung oder die Anbindung an ein bestehendes E-Commerce-System vorzuziehen ist.
Entscheidung	Die umfangreiche Analyse zur Evaluation von nopCommerce ist unter 4.2 dokumentiert.
Konsequenzen	Das Lösungskonzept sieht die Entwicklung eines Connectors für nopCommerce und Abacus vor, der als Plugin bereitgestellt wird.

Tabelle C.4: Beschreibung ADR 04: Einsatz von nopCommerce als E-Commerce-Framework

ADR Nr.	5
Titel	Asynchrone Übermittlung der Bestellungen
Datum	01.11.2023
Status	Akzeptiert
Kontext	Sobald eine Bestellung im nopCommerce getätigt wurde, muss diese mit dem Abacus Conncetor an das Abacus ERP übermittelt werden.
Entscheidung	Mittels einem Handler auf dem Domainevent 'OrderPlaced' kann der Abacus Conncetor auf einer neue bestellung reagieren und diese entsprechend übermitteln. Es gibt dabei die Möglichkeit, einen Handler Synchron oder Asynchron auszuführen. Bei einer Synchronen Ausführung wartet nopCommerce auf eine Antwort vom Handler und die Bestellung könnte bei einem Fehler von Kunden nicht abgesetzt werden. Es wurde jedoch Entschieden, die Übermittlung an Abacus Asynchron auszuführen, da die Schnittstelle der Inbox (siehe Abschnitt 2.4.5) nicht validiert. Die Validierung erfolgt innerhalb vom Abacus, sobald ein Eintrag in der Inbox zu einem Auftrag umgewandelt wird.
Konsequenzen	Es ist möglich, dass eine Bestellung erfolgreich abgesetzt wird und diese dann nicht in Abacus landet. Da die Inbox nicht validiert, dürfte dies in der Theorie nur geschehen, wenn die Schnittstelle von Abacus nicht erreichbar ist. Dies wird auch so vorfallen, zum Beispiel bei einem Update vom ERP-System. Es ist aber gewollt, dass in diesem Moment weiterhin Bestellung im Shop abgesetzt werden können.

Tabelle C.5: Beschreibung ADR 05: Asynchrone Übermittlung der Bestellungen

Anhang D

Aufgabenstellung

D.1 Ausgangslage

Abacus ist der führende Anbieter von Unternehmensressourcenplanung (ERP) Software in der Schweiz. Seit langem bietet Abacus eine integrierte Shop-Lösung namens 'Abashop' an, um die Anforderungen seiner Kunden im Bereich E-Commerce zu erfüllen. Diese Lösung wurde durch die Verwendung von statischem HTML und JSP realisiert und ermöglichte es Unternehmen, nahtlos in ihre ERP-Systeme zu integrieren. Allerdings hat Abacus kürzlich angekündigt, dass sie den Support für Abashop bis Ende 2025 einstellen wird.

Diese Ankündigung stellt viele Unternehmen vor die Herausforderung, alternative E-Commerce-Lösungen zu finden, die ihren Anforderungen entsprechen. Insbesondere hat der Auftraggeber dieser Arbeit, die Firma Customize AG, ein Vertriebspartner von Abacus, zahlreiche Kunden, die derzeit Abashop nutzen. Daher ist es für Customize von entscheidender Bedeutung, potenzielle Nachfolgelösungen zu evaluieren und sicherzustellen, dass sie den Bedürfnissen ihrer Kunden gerecht werden.

In diesem Zusammenhang hat Abacus eine neue RESTful HTTP-Schnittstelle eingeführt, die es den Kunden ermöglicht, eigene Shop-Lösungen auf der Grundlage ihres Abacus-Systems zu entwickeln. Dies eröffnet sowohl für Customize als auch für andere Unternehmen spannende Möglichkeiten, aber es wirft auch Fragen auf, wie diese Schnittstelle optimal genutzt werden kann und ob alle bestehenden Funktionalitäten des AbaShop abgedeckt werden können.

D.2 Ziele der Arbeit und Liefergegenstände

In dieser Arbeit liegt der Hauptfokus auf der Evaluation der neuen RESTful-Schnittstellen von Abacus und ihrer Eignung zur Abdeckung der E-Commerce-Bedürfnisse der Kunden von Customize. Hierbei wird geprüft, ob es sinnvoll ist, ein Plugin für ein bestehendes E-Commerce-System zu entwickeln oder ob eine Eigenentwicklung die bessere Lösung darstellt.

Um dieses Hauptziel zu erreichen, werden folgende Liefergegenstände verfolgt:

1. **Anforderungserhebung und -analyse**: Es sollen detaillierte Anforderungen der Kunden von Customize an ihre E-Commerce-Lösungen ermittelt werden. Diese Anforderungen werden repräsentativ und aussagekräftig evaluiert, um eine solide Grundlage für die weitere Arbeit zu schaffen.

2. **Produktevaluation**: Die verbreitesten E-Commerce-Plattformen und -Lösungen (Minimum: 5, Maximum: 10) werden analysiert und bewertet. Dabei wird untersucht, wie gut sie die zuvor identifizierten Anforderungen erfüllen können. Dieser Schritt ist signifikant für die Entscheidung, ob ein Plugin oder eine Eigenentwicklung die bessere Wahl ist.

3. **Prototypentwicklung**: Im Rahmen des Projekts wird ein Prototyp entwickelt, der die wichtigsten Use Cases für die Kunden von Customize abdeckt. Dieser Prototyp dient dazu, die Machbarkeit der Umsetzung der Anforderungen zu demonstrieren und erste praktische Erfahrungen mit den Schnittstellen zu sammeln. Hierbei wird insbesondere Wert auf die Evaluation der Eignung der

Abacus API zur Erstellung des besagten Prototypen gelegt.

4. **Migrationsfähigkeit**: Es wird untersucht, wie eine mögliche Migration der bestehenden Abashop-Lösung auf die neue Shoplösung durchgeführt werden kann.

Folgende Punkte werden als kritische Erfolgsfaktoren erachtet:

1. Kundenanforderungen werden repräsentativ und ausreichend erfasst. Es werden 3-4 repräsentative Usecases definiert und implementiert. Repräsentativ bedeutet in diesem Kontext, dass die Usecases die Bedürfnisse der Nutzer und Kunden möglichst breit abdecken. Ein Usecase, der nur wenige Bedürfnisse erfasst, ist somit aufgrund der Begrenztheit der Studienarbeit zu vermeiden.

2. Evaluation des Prototyp auf Benutzbarkeit, Erweiterbarkeit sowie Wartbarkeit. Es gelten die üblichen Ansprüche an Software: Die Software soll insbesondere hohe Kohäsion sowie geringe Kopplung umsetzen. Zur Erstellung dieser Software werden industrieübliche Prozesse eingesetzt. Dies inkludiert insbesondere Software Engineering Hygienefaktoren wie automatisierte Builds, Tests, angemessene Versionierung mit git.

3. Ein abschliessendes Urteil über die Eignung der Abacus Schnittstelle wird gefällt sowie begründet.

Literaturverzeichnis

[1] Abacus. Api hub. https://apihub.abacus.ch/, 2023. [Online; Abgerufen am 25.09.2023].

[2] Adobe. Magento. https://business.adobe.com/ch_de/products/magento/magento-commerce.html, 2023. [Online; Abgerufen am 25.09.2023].

[3] Arc42. Adr template by nygard. https://docs.arc42.org/examples/decision-use-adrs/, 2023. [Online; Abgerufen am 22.10.2023].

[4] Matthias Biehl. *Webhooks: Events for RESTful APIs*. API-University Series. CreateSpace, 2017.

[5] C4Model. C4 architektur modell. https://c4model.com/, 2023. [Online; Abgerufen am 01.10.2023].

[6] Mike Cohn. *Agile Estimating and Planning*. Prentice Hall, 2005.

[7] Datatrans AG. Datatrans: Ihr spezialist für online-zahlungen. https://www.datatrans.ch/de/, 2023. [Online; Abgerufen am 20.10.2023].

[8] Eidgenössische Steuerverwaltung ESTV. Mehrwertsteuer: Steuersätze. https://www.estv.admin.ch/estv/de/home/mehrwertsteuer/mwst-steuersaetze.html, 2023. [Online; Abgerufen am 20.10.2023].

[9] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2002.

[10] GitHub. Github nopcommerce. https://github.com/nopSolutions/nopCommerce, 2022. [Online; Abgerufen am 01.10.2022].

[11] GitHub. Editor config .net runtime repository. https://github.com/dotnet/runtime/blob/main/.editorconfig, 2023. [Online; Abgerufen am 23.11.2023].

[12] Github. Github repository simple.odata.client bibliothek. https://github.com/simple-odata-client/Simple.OData.Client, 2023. [Online; Abgerufen am 20.10.2023].

[13] GitHub. Medusa github discussion about translation of entities. https://github.com/medusajs/medusa/discussions/2003, 2023. [Online; Abgerufen am 01.10.2023].

[14] Google. Google lighthouse. https://chrome.google.com/webstore/detail/lighthouse, 2022. [Online; Abgerufen am 01.10.2022].

[15] Google. Google mobile-friendly test. https://search.google.com/test/mobile-friendly, 2022. [Online; Abgerufen am 01.10.2022].

[16] GraphQL. Graphql. https://graphql.org/, 2023. [Online; Abgerufen am 20.09.2023].

[17] Bobby Woolf Gregor Hohpe. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, 1 edition, 2003.

[18] IETF. Oauth 2.0. https://datatracker.ietf.org/doc/html/rfc6749, 2023. [Online; Abgerufen am01.10.2023].

[19] Inztitut. Iso25010 inztitut. https://inztitut.de/blog/glossar/iso-25010/, 2023. [Online; Abgerufen am 02.10.2023].

[20] JetBrains. dotcover: .net unit test runner and code coverage tool. https://www.jetbrains.com/dotcover/, 2023. [Online; Abgerufen am 25.09.2023].

[21] Philippe Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley Professional, 2000.

[22] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, 2004.

[23] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.

[24] Medusa. Medusa. https://medusajs.com/, 2023. [Online; Abgerufen am 01.10.2023].

[25] Microsoft. Asp.net core. https://learn.microsoft.com/en-us/aspnet/core, 2023. [Online; Abgerufen am 23.10.2023].

[26] Microsoft. Razor. https://learn.microsoft.com/en-us/aspnet/core/razor-pages, 2023. [Online; Abgerufen am 25.09.2023].

[27] nopCommerce. nopcommerce entfernung urheberrecht. https://www.nopcommerce.com/de/nopcommerce-copyright-removal-key, 2022. [Online; Abgerufen am 01.10.2022].

[28] nopCommerce. Architektur nopcommerce. https://docs.nopcommerce.com/en/developer/tutorials/architecture-of-nopCommerce.html, 2023. [Online; Abgerufen am 20.10.2023].

[29] nopCommerce. nopcommerce. https://www.nopcommerce.com/, 2023. [Online; Abgerufen am 25.09.2023].

[30] OData. Actions in odata. https://www.odata.org/blog/actions-in-odata/, 2023. [Online; Abgerufen am 01.10.2023].

[31] OData. Open data protocol. https://www.odata.org/, 2023. [Online; Abgerufen am 01.10.2023].

[32] OuterBox. Mobile ecommerce statistics: Percentage of shoppers on mobile devices. 2023. [Online; Abgerufen am 23.11.2023].

[33] Remix. Remix. https://remix.run/, 2023. [Online; Abgerufen am 20.09.2023].

[34] IT Reseller. Abacus ist das am meisten genutzte erp-system in der schweiz. https://www.itreseller.ch/Artikel/98885/Abacus_ist_das_am_meisten_genutzte_ERP-System_in_der_Schweiz.html, 2023. [Online; Abgerufen am 10.10.2023].

[35] Mark Seemann and Steven van Deursen. *Dependency Injection: Principles, Practices, and Patterns*. Manning Publications, 2019.

[36] Shopify. Preisstruktur shopify. https://www.shopify.com/pricing, 2023. [Online; Abgerufen am 20.10.2023].

[37] Shopify. Shopify. https://www.shopify.com/de, 2023. [Online; Abgerufen am 20.09.2023].

[38] Shopify. Shopify payments. https://help.shopify.com/en/manual/payments/shopify-payments, 2023. [Online; Abgerufen am 01.12.2023].

[39] ITBE Staff. How poor website performance impacts revenue. *IT Business Edge*, 2014. [Online; Abgerufen am 23.11.2023].

[40] Statista. Market share of leading e-commerce software platforms. https://www.statista.com/statistics/710207/worldwide-ecommerce-platforms-market-share/, 2023. [Online; Abgerufen am 20.09.2023].

[41] Vue Storefront. Headless commerce. https://vuestorefront.io/blog/headless-commerce, 2022. [Online; Abgerufen am 10.10.2023].

[42] WooCommerce. Woocommerce. https://woocommerce.com/, 2023. [Online; Abgerufen am 20.09.2023].

[43] Worldline. Worldline: Leading payment services. https://worldline.com/de-ch/home.html, 2023. [Online; Abgerufen am 20.10.2023].

Glossar

ABEA ABEA (Auftragsbearbeitung) ist ein Modul von Abacus, welches den Warenprozess eines Unternehmens abbildet (Produktstammdaten, Lager, Verkauf, Einkauf). 3

API Application Programming Interface, kurz API, ist eine Schnittstelle in der Softwareentwicklung, die es verschiedenen Anwendungen ermöglicht, miteinander zu interagieren und Funktionen oder Daten auszutauschen. Sie legt fest, wie Programme miteinander kommunizieren, und wird oft als "Schnittstelle" zwischen verschiedenen Softwarekomponenten bezeichnet. APIs sind zentral für die Integration und das Zusammenspiel moderner Software-Systeme.. 32, 36

B2B Business-to-Business, ein Geschäftsmodell, das den Handel zwischen Unternehmen beschreibt. Dies kann den Austausch von Produkten, Dienstleistungen oder Informationen umfassen.. 11

B2C Business-to-Consumer, ein Geschäftsmodell, bei dem Unternehmen direkt an Endverbraucher verkaufen. Dies ist typisch für den Einzelhandel und Online-Shops.. 11

CRM Unter Customer Relationship Managemeng (CRM) versteht man eine Software, die auf die Verwaltung, Analyse und Pflege der Interaktionen und Beziehungen eines Unternehmens zu seinen Kunden ausgerichtet ist. Ziel ist es, die Kundenbindung zu stärken, die Kundenzufriedenheit zu erhöhen und letztendlich den Vertriebserfolg zu steigern, indem die Kundenbedürfnisse effektiv erfasst und adressiert werden.. 4, 8

ERP Unter Unternehmensressourcenplanung (ERP) versteht man eine Software, welche die Geschäftsprozesse eines Unternehmens durchgängig und integriert in einem zentralen System digital abbildet.. iii, 3–5, 14, 26, 76, 79, 118

GUID Globally Unique Identifier, eine Zeichenfolge, die zur Identifizierung von Informationen in Computersystemen verwendet wird. GUIDs sind in ihrer Struktur standardisiert und werden so entworfen, dass sie einzigartig sind. Eine GUID besteht typischerweise aus 32 Hexadezimalzahlen, die durch Bindestriche in fünf Gruppen unterteilt sind (z.B., 21EC2020-3AEA-4069-A2DD-08002B30309D). Sie werden häufig in Softwareentwicklung und Datenbanken verwendet, um eindeutige Schlüssel zu erzeugen, die Kollisionen vermeiden.. 50

IDE Auch bekannt als IDE (Integrated Development Environment), bezeichnet eine Softwareanwendung, die Softwareentwicklern eine umfassende Umgebung für die Softwareentwicklung bietet. Sie integriert typischerweise einen Quellcode-Editor, Debugging-Werkzeuge und häufig auch Tools zur Versionskontrolle. Integrierte Entwicklungsumgebungen erleichtern die Softwareentwicklung, indem sie alle notwendigen Werkzeuge in einer einheitlichen Anwendung zusammenführen.. 58

MVP Ein Minimum Viable Product (MVP) ist die einfachste Version eines Produkts, mit der es möglich ist, Kunden zu gewinnen und maximales Lernen über Kundenpräferenzen mit minimalem Aufwand zu erzielen. Es enthält nur die grundlegenden Funktionen, die notwendig sind, um es verkaufsfähig zu machen und Feedback von den Nutzern zu erhalten.. 2, 10, 36, 38, 51

NFR Nicht-funktionale Anforderungen (NFR) beziehen sich auf die Qualitätsattribute eines Softwaresystems, wie z.B. Leistung, Sicherheit, Benutzerfreundlichkeit und Skalierbarkeit. Im Gegensatz zu funktionalen Anforderungen, die beschreiben, was ein System tun soll, definieren NFRs, wie gut das System seine Funktionen ausführen soll.. 19, 21–23, 48

OData Open Data Protocol (OData) ist ein offenes Webprotokoll zur Abfrage und Aktualisierung von Daten. Es ermöglicht die Erstellung und Nutzung von RESTful APIs in einer standardisierten Weise, unterstützt komplexe Abfragen, Filterung, Pagination und Typisierung der Antworten. OData erleichtert die Integration und Interaktion zwischen verschiedenen Plattformen und Anwendungen durch eine einheitliche Schnittstelle und bietet durch den $metadata Endpunkt eine detaillierte Beschreibung des Datenmodells in einem standardisierten XML-Format.. 25, 53

Open Source Open Source bezieht sich auf Software oder Technologien, deren Quellcode öffentlich zugänglich ist und von einer Gemeinschaft von Entwicklern bearbeitet und weiterentwickelt werden kann.. 39, 41–43, 45, 48, 49

PoC Ein Proof of Concept (PoC) ist eine Demonstration, um das Potenzial einer bestimmten Idee oder Methode zu verifizieren oder zu bestätigen. Es dient dazu, die Machbarkeit und Funktionalität eines Projekts oder einer Technologie zu zeigen, ohne ein vollständiges Produkt oder System zu entwickeln.. iii, 2, 10, 12, 14–16

SaaS SaaS (Software as a Service) bezeichnet ein Geschäftsmodell, bei dem Softwareanwendungen über das Internet bereitgestellt und als Abonnementdienst genutzt werden, anstatt sie auf eigenen Servern oder lokal auf den Endgeräten der Nutzer zu installieren.. 40, 41

SEO Search Engine Optimization (SEO) bezeichnet den Prozess, die Sichtbarkeit einer Website in den Suchergebnissen von Suchmaschinen zu verbessern. Ziel ist es, durch relevante Keywords und Optimierungen mehr Besucher auf die Website zu ziehen.. 21

SuD SuD (System und discussion) bezeichnet in der Use Case Modellierung das System, für welches die Requirements erhoben werden. 11, 25, 58, 64

UML Die Unified Modeling Language (UML) ist eine standardisierte Modellierungssprache in der Softwaretechnik. Sie dient der Spezifikation, Konstruktion und Dokumentation von Software-Teilen und anderen Systemen. UML bietet verschiedene Diagrammtypen, die verschiedene Aspekte von Systemmodellen darstellen können.. 18

URL Uniform Resource Locator, ein standardisiertes Format für Adressen im World Wide Web. Eine URL ermöglicht den Zugriff auf Ressourcen wie Webseiten, Bilder oder Videos und besteht aus mehreren Teilen, einschliesslich Protokoll (wie http oder https), Domänennamen und gegebenenfalls einem Pfad zu einer spezifischen Ressource. URLs sind ein grundlegendes Element des Internets und erleichtern die Navigation und den Zugriff auf Inhalte im Web.. 21, 26, 64

Abbildungsverzeichnis

Tabellenverzeichnis