Sperrvermerk

Die vorliegende Arbeit beinhaltet interne und vertrauliche Informationen der Firma dotSource GmbH. Die Weitergabe des Inhaltes der Arbeit und eventuell beiliegender Zeichnungen und Daten im Gesamten oder in Teilen ist grundsätzlich untersagt. Es dürfen keinerlei Kopien oder Abschriften - auch in digitaler Form - gefertigt werden. Ausnahmen bedürfen der schriftlichen Genehmigung der Firma dotSource GmbH.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CMS	Content Management System
CRM	Customer-Relationship-Management
CSRF	Cross-Site-Request-Forgery
CSV	Comma-separated-value
ERP	Enterprise-Ressource-Planning
FTP	File Transfer Protocol
http	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LGPL	Lesser General Public License
LTS	Long Term Support
MIT	Massachusetts Institute of Technology
PHP	Hypertext Preprocessor
PIM	Product Information Management
PSP	Payment Service Provider
UI	User Interface
VM	Virtual Machine
XML	Extensible Markup Language
XSS	Cross Site Scripting

1 Einleitung

„Content-Commerce, eine Zusammensetzung aus Content-Marketing und E-Commerce, bietet Händlern die Möglichkeit, ihre Fachkenntnis als Branchenexperten zu festigen und umfassende Beratungsleistungen in die Online-Welt zu verlagern. Die Kombination aus emotionaler Ansprache und Information schafft ein nahtloses Einkaufserlebnis, mit dem Händler den Preiskampf mit großen Plattformen umgehen können. Denn wenn das Erlebnis stimmt, sind Kunden häufig auch bereit, einen höheren Preis zu bezahlen.“[1] Content Commerce ist also die Kombination aus einem E-Commerce-System und einem Content-Management-System (CMS) (das sich um den Workflow und Usability kümmert) als leistungsfähiger Webshop.

In der dotSource GmbH wird die Kombination zweier Systeme (z. B. TYPO3 und Magento, TYPO3 und Shopware) verwendet, um einen Webshop zu implementieren. Diese Methode erfordert jedoch einen hohen Aufwand und wäre effizienter, wenn nur ein System verwendet werden könnte. Ein neuer Ansatz würde Kunden eine leichtgewichtige E-Commerce-Bibliothek wie das E-Commerce-Framework Aimeos bieten, die in andere Anwendungen wie TYPO3 oder Laravel eingebettet werden kann. Das Framework wird in die vorhandene Anwendung integriert und bietet eine Gesamtzahl an Tools für einen voll ausgestatteten Webshop. Um herauszufinden, inwieweit diese Methode sinnvoll ist, wird das E-Commerce-Framework Aimeos in TYPO3 integriert und die Funktionalität entsprechend bewertet. Es wird evaluiert, ob Aimeos eine Verbindung zweier Systeme tatsächlich ersetzen kann. Dabei werden vor allem Aufwand und Komplexität berücksichtigt.

Mit dieser Arbeit sollen die Funktionalitäten von Aimeos bezüglich ausgewählter Use Cases überprüft werden. Dazu gehören nicht nur kundenbezogene Shop-Aspekte, sondern auch die Handhabung und Konfiguration des Frameworks und der Support.

Zu diesem Zweck werden zunächst Aimeos und in diesem Zusammenhang E-Commerce-Frameworks vorgestellt. Anschließend werden die Tools vorgestellt, die zur Verwendung des Frameworks erforderlich sind. Dann werden die Funktionen von Aimeos bezüglich interner Use Cases betrachtet.

[1] [t3n18b]

2 E-Commerce-Frameworks

2.1 Prinzip von E-Commerce-Frameworks

E-Commerce hat seinen Beginn im Internet um das Jahr 1995 und kann jetzt in drei Generationen unterteilt werden. Die erste verwendet Customer Relationship Management (CRM) und CMS-Tools mit eingeschränkter Funktionalität in der Umgebung von SAP Enterprise Resource Planning (ERP). ERP ist die zeitnahe und bedarfsgerechte Planung und Steuerung von z.B. Personal-, Material- und Informations- und Kommunikationstechnologie im unternehmerischen Sinne. Hybris kann beispielhaft hierfür erwähnt werden. Die zweite Generation umfasst CRM- und CMS-Komponenten und andere Funktionalitäten, aber nur mit eingeschränkter Funktionsweise. Repräsentativ dafür ist das Magento-Shopsystem. E-Commerce-Frameworks können jedoch in die dritte Generation eingestuft werden. [2] Die Kerndomäne umfasst die Präsentation und den Verkauf von Produkten über eine eigenständige Einheit. Für andere E-Commerce-bezogene Daten und Aufgaben können die Frameworks mit spezialisierten Systemen wie Product Information Management (PIM), Ordermanagement und ERP sowie CRM verbunden werden und Daten austauschen. [3]

E-Commerce-Frameworks bieten im Vergleich zu den klassischen Shop-Systemen viel Flexibilität hinsichtlich des Integrierens verschiedenster IT-Systeme.[4] Durch den Einsatz sollte es in kurzer Zeit möglich sein, komplette E-Commerce-Applikationen zu erstellen. Diese Frameworks sind oft flexibel genug, um sie an kundenspezifische Anforderungen anzupassen. Anbieter solcher Frameworks müssen jedoch darauf achten, Anpassungen am Framework-Code selbst nicht zuzulassen. E-Commerce-Frameworks eignen sich beispielsweise für den Aufbau nahezu aller Arten von Online-Shops und E-Commerce-relevanten (Web-) Anwendungen. Die Verbreitung von solchen E-Commerce-Systemen ist jedoch gering.

Ein E-Commerce-Systeme sollte einige wichtige Funktionen aufweisen, wie Formularerstellung, benutzerdefinierte Favoriten und E-Mail-Kampagnen. Obwohl sich die Rahmenbedingungen in der detaillierten Struktur und den Zielgruppen unterscheiden, haben sie das gemeinsame Merkmal, dass sie bis zu bestimmten Punkten von Händlern und Herstellern

[2] Vgl. [aima]
[3] Vgl. [Dan16]
[4] Vgl. [eCo]

oft willkürlich und kundenorientiert erweiterbar[5] sind. E-Commerce-Frameworks definieren in der Regel einen allgemeinen Programmablauf und bestehen aus wiederverwendbaren Komponenten.

Darüber hinaus können die Frameworks nicht nur durch z.B. Cloud-Infrastrukturen betrieben werden, sondern auch vollständig auf eigenen Servern.[6] Das Framework bietet grundsätzlich eine vollständige Infrastruktur für verschiedene Webshops.[7] Darüber hinaus implementieren E-Commerce-Frameworks normalerweise den allgemeinen Programmfluss, d.h. wie Prozesse durch den Programmtext hindurch abgearbeitet werden. Dies ist oft anpassbar.

2.2 Vorstellung von Aimeos

Aimeos ist eine Open Source Hypertext Preprocessor (PHP)-Bibliothek und ein E-Commerce Framework. Dieses kann z.B. als Plugin in TYPO3 integriert werden. Aimeos bietet ein komplettes Shop-System, das mit Magento Core vergleichbar ist. Es verwendet eine umfangreiche Codebasis und einen Adapter für die native Integration. [8]

In Aimeos werden alle Daten wie Produkte oder Kategorien in einer „Domäne“ gespeichert [9], dies wird auch als Domain Driven Design bezeichnet. Eine Domäne enthält die Daten und den Code für einen allgemeinen Kontext. Ein Beispiel hierfür wäre die Katalogdomäne, die aus Kategorien und zugehörigen Daten besteht, einschließlich der Manager-Klassen, die diese Daten speichern, verarbeiten und abfragen. Die vertikale Trennung von Code in Geschäftsdomänen ermöglicht die Skalierung von Anwendungen, indem die Anwendung als Micro Services angeboten wird. Alle Daten oder Teile davon können in einer relationalen oder dokumentorientierten Datenbank gespeichert oder über SQL oder eine andere Abfragesprache abgerufen werden. Dies ist möglich, weil Manager und Elemente verwendet werden, die ein gemeinsames Interface für die gespeicherten Daten bereitstellen und den Zugriff auf die Daten ermöglichen, unabhängig davon, wo und wie sie gespeichert werden.

Die Implementierung für Aimeos Frontend ist nach Kategorien, Client-Files und Plugins unterteilt: Basket, Catalog, Account, Checkout und E-Mail. Plugins sind Klassen, die sich für

[5] Vgl. [sho]
[6] Vgl. [t3n18a]
[7] Vgl. Ebenda
[8] Vgl. [aimc]
[9] [Aim18c]

bestimmte Ereignisse registrieren [10]. Diese können benachrichtigt werden und reagieren dann auf das Ereignis. Zum Beispiel können die Plugins die Versandkosten ändern. Plugins sind eine sehr flexible Möglichkeit, die in einen Algorithmus implementiert werden kann. Die Client-Files bestehen aus HTML-embedded-PHP-Code.

Aimeos verwendet Software-Engineering-Techniken, Interfaces und Design Pattern, wie beispielsweise Dependency Injection, bei der eine Dependency ein Objekt und eine Injection die Weitergabe einer Dependency an ein anderes Objekt (einen Client) ist. Darüber hinaus verwendet Aimeos Factories von denen z. B. alle Komponenten instanziiert werden und Decorators. Ein Decorator kann durch Konfiguration zusätzliche Funktionen hinzufügen. Wenn zum Beispiel eine Zahlung nur per Rechnung verfügbar sein sollte, kann dies durch einen zusätzlichen Decorator erfolgen, welcher als eine Art Plugin einfach vor die zu dekorierende Klasse geschalten wird. Sie sind wie Regelsätze, die so angeordnet werden können, dass nur wenige Regeln zum Erstellen verschiedener Regelsätze erforderlich sind. Außerdem verwendet Aimeos das Nachrichtenmuster Publish Subscribe, bei dem sogenannte Publisher keine Nachrichten direkt an bestimmte Empfänger (Subscribers) senden, sondern in Klassen ohne Kenntnis der Subscriber[11] kategorisiert werden.

In Aimeos Struktur werden Bausteine zu Komponenten zusammengefügt (siehe Abbildung 1).

[10] Vgl. [Aim18a]
[11] [aimb]

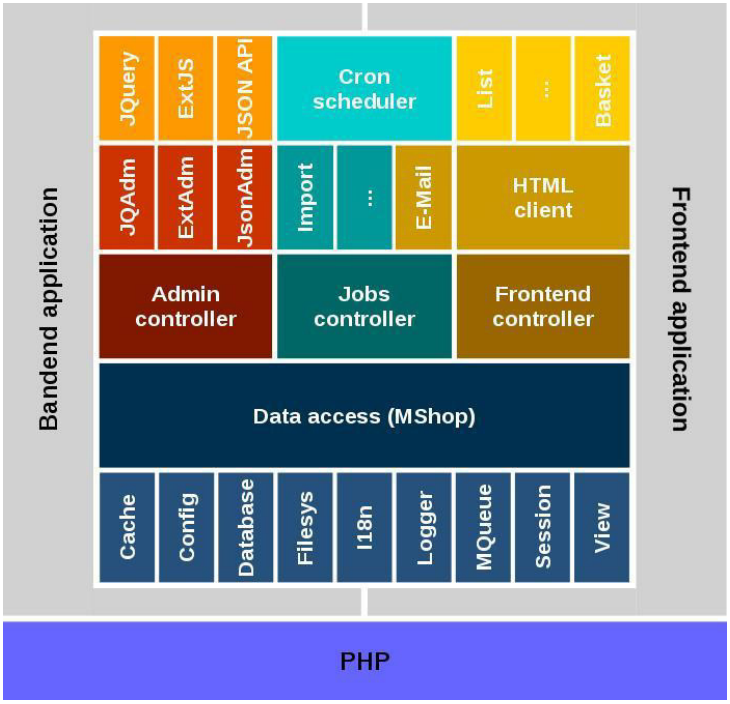


Abbildung 1: Aimeos-Stack[12]

Die PHP-basierten Interfaces für z.B. Cache und View bilden die unterste Ebene. Darüber befindet sich der Persistence Layer (MShop), der die Verwaltung aller Daten wie Produkte, Texte, Attribute usw. in Aimeos übernimmt. Es folgen die Controller (Admin, Jobs und Frontend), die Methoden enthalten, die sich um den Output der generierten HTML Seiten kümmern. Dadurch wird das Modell auch in drei entsprechende Bereiche unterteilt. Der Admin-Controller kümmert sich um die Admin-Interfaces, die entsprechend das Backend implementieren, wie das JQAdm-Administrations-Interface, das die Editoren beim Anmelden am Backend sehen. Die Interfaces können Erweiterungen wie JQuery (JavaScript Library) und die JavaScript Object Notation (JSON) Application Programming Interface (API) verwenden. Jobs Controller enthält die Controller, die Aimeos-Cronjobs über den Cron-Scheduler ausführen, z. B. den Import- und den E-Mail-Controller. Der Front-End-Controller ist der HTML-Client, der mehrere Komponenten enthält, z.B. die Listen- und die Warenkorbkomponenten.

Aimeos ist ein sehr schnell skalierbares Framework. Die E-Commerce Lösung ist anpassbar und erweiterbar, sodass jede Klasse dynamisch gestaltet werden kann. Darüber hinaus ist das System in hohem Maße konfigurierbar und es gibt ungefähr 2500 Konfigurationsoptionen[13] für

[12] [aim17]
[13] [Spe]

Shop-Inhaber und Entwickler, die je nach verwendeter Anwendung in TypoScript definiert werden können. Dies sind z.B. Frontend-Konfiguration, Controller-Einstellungen und Cronjobs-Gestaltung sowie Konfiguration der Administrationsschnittstelle. Es gibt auch über 70 Zahlungsanbieter, die über Omnipay, einer PHP-Zahlungsverarbeitungsbibliothek, verwendet werden können. In Bezug auf die Leistung ist Aimeos Multi-Site-fähig und ermöglicht das Speichern mehrerer Shops in einer Datenbank. Darüber hinaus verwendet Aimeos eine Lesser General Public License (LGPL) und eine Massachusetts Institute of Technology (MIT) -Lizenz.

Außerdem verfügt Aimeos über eine große Codebasis und wird ausführlich getestet (Unit Tests). Die Entwicklung von Aimeos findet auf GitHub statt und ist Open Source. Außerdem ist der Support sehr umfassend, so wird mindestens 4 Mal pro Jahr eine Long Term Support (LTS)-Version veröffentlicht und es gibt eine umfangreiche Dokumentation mit einer zunehmenden Anzahl von Übersetzungen.

3 Verwendete Tools

3.1 TYPO3

TYPO3 ist ein professionelles Open Source CMS-System. Mit TYPO3 können Websites, Intranets und Web-Apps erstellt werden. [14] Das System kann über verschiedene Plugins, Module, Klassen und Konfigurationscodes erweitert werden.

Im Back-End von TYPO3 gibt es eine Unterteilung in einen Kopfbereich, in dem Administrationskonfigurationen vorgenommen werden können, und in einen Hauptbereich. Im Hauptbereich von TYPO3 können zunächst einige Module ausgewählt werden, mit denen gearbeitet werden kann, z. B. Seiteninhalt oder Template-Tools. Wenn ein solches Modul gewählt wird, wird ein Seitenbaum geöffnet, in dem ausgewählt werden kann, welche TYPO3-Frontend-Seite bearbeitet werden soll. Im Arbeitsbereich können dann die entsprechenden TYPO3-Objekte bearbeitet werden.

Aimeos wird als Erweiterung verwendet und macht TYPO3 zur Plattform für Content Commerce und die entsprechenden E-Commerce-Anforderungen. Die Aimeos TYPO3-Erweiterung integriert die Arcavias PHP-Webshop-Bibliothek in jede TYPO3-Installation. Dies ermöglicht die direkte Integration von Aimeos in eine existierende TYPO3-Website. Die Erweiterung bietet mehrere Content-Plug-Ins für bestimmte Funktionalitäten (Produktfilter, Detailansicht, Checkout usw.). Zusätzlich ist die Aimeos Administrationsoberfläche, über die direkt Daten, wie Produkte, Attribute und Services verwaltet werden können, über das TYPO3-Backend erreichbar.

3.2 Wichtige TYPO3-Extensions

Um die Erweiterung Aimeos für TYPO3 zu nutzen, wurden einige zusätzliche Erweiterungen verwendet.

Dazu gehört die Scheduler-Erweiterung, die auch in TYPO3 installiert werden muss und von Aimeos zum Erstellen und Ausführen von Cron-Jobs verwendet wird. Wichtig ist auch die Erweiterung „static_info_tables“. Dies sind Tabellen für eine Datenbank, die Daten zu Ländern,

[14] Vgl. [typ]

Sprachen, Währungen und dergleichen enthalten. Diese werden auch standardmäßig von Aimeos verwendet.

Weiterhin kann z.B. die RealURL-Erweiterung, die statische Adressen anstelle der Websites mit dynamischen Adressen erstellt, die standardmäßig in CMS erstellt werden, verwendet werden. [15] Die Verwendung der statischen Adressen macht die TYPO3-URLs suchmaschinenfreundlicher.

Es ist auch möglich Fluid in TYPO3 zu verwenden. Fluid ermöglicht das dynamische Erstellen und Konfigurieren von Templates. Aimeos verwendet hierfür standardmäßig HTML und PHP. Diese können jedoch später bei der Bearbeitung leicht durch die Fluidsyntax ersetzt werden.

3.3 Xampp

„XAMPP ist eine vollständig kostenlose, [einfach] zu installierende Apache-Distribution, die MariaDB, PHP und Perl enthält.“[16] Es kann hauptsächlich zur Verwaltung von PHP-Entwicklungsumgebungen verwendet werden. Die Distribution ist Open Source und wird hierbei auch für die einfache Benutzung von Apache-Servern in Verbindung mit Datenbanken wie MySQL angewendet.

Im Rahmen dieser Arbeit wird Xampp zum Ausführen eines Webservers und eines Datenbankservers gebraucht, auf dem TYPO3 ausgeführt werden kann. Dazu wird TYPO3 heruntergeladen und lokal in den Ordner von Xampp, „htdocs“, entpackt. Im Ordner htdocs speichert der Apache-Server im Allgemeinen Dateien und Erweiterungen, auf die dann über den localhost oder andere verbundene Adressen zugegriffen werden kann. Wenn dann mit dem Apache-Server der localhost aufgerufen wird, wird der TYPO3-Installationsassistent gestartet, dessen Anweisungen nur befolgt werden müssen.

[15] Vgl. [Typ]
[16] Vgl. [apa19]

4 Zielstellung und Ist-Zustand

Gegenwärtig verwendet die dotSource eine Kombination von zwei Systemen als E-Commerce-System, z.B. eine Kombination aus TYPO3 und Magento. Diese Art von Lösung umfasst CRM, CMS sowie PIM-Komponenten und ähnliche Funktionen. Dies erfordert jedoch viel Entwicklungsarbeit, da eine Verbindung der beiden Systeme hergestellt werden muss, um den Funktionsumfang voll ausnutzen zu können.

Um die Verwendung von Aimeos erfolgreich zu bewerten, wurden insgesamt 21 Use Cases eingerichtet und unterschiedlich priorisiert. Die Buchstaben A-C wurden angegeben, wobei A der höchste und C der niedrigste war. Darüber hinaus sollte der Use Case nach dem Schulnotensystem anhand des Erfüllungsgrades bewertet werden. Dies sollte immer nach Überprüfung des Use Cases erfolgen und entsprechend begründet werden.

Die Priorität der Kategorie C hat der Import von Kundendaten.

Ein Use Case der Priorität B ist die Live-Abfrage des Bestands, wobei es auch darum geht, wie und ob ein ERP-System zum Artikelbestand angefordert werden kann. Ein weiterer Use Case wäre der Import des Bestellstatus, wobei angegeben werden muss, inwieweit hier ein Webservice eines Fullfilment Partners oder Logistiksystems genutzt werden kann. Weitere Use Cases wären der Export von Kundendaten und die Einbeziehung von Promotions, die Angebote im Sinne von „3 für 2 Produkte“ repräsentieren können. Es ist auch wichtig, wie der Zahlungsstatus in Aimeos aktualisiert wird und wie ein Payment Service Provider (PSP) verwendet und integriert wird. Außerdem werden die Use Cases für das Senden einer E-Mail-Bestätigung an Kunden (z.B. Dank-E-Mail oder „andere Kunden haben gekauft“) und das Ändern der Aimeos-internen Plugins, die zur Darstellung des Webshops verwendet werden, berücksichtigt (z. B. Checkout-Layout oder Produktdetail).

Am wichtigsten sind jedoch die Use Cases der Priorität A, von denen Aimeos mindestens 8 mit Note 3 erfüllen sollte. Dazu gehören Import- und Exportfunktionen, z. B. Importieren von Produkten und Lagerbestand oder Exportieren von Bestellungen. Wichtig ist auch die Erstellung von Promotions, wobei vor allem die Erstellung von Promotionssektionen, das Kreieren von prozentualen Rabatten, wobei auch nach Kopfrabatten und Positionsrabatten zu differenzieren ist, sowie die Erstellung von Gutscheinen untersucht und evaluiert werden sollten. Dazu gehört auch die Darstellung einer Produktkachel auf der Contentseite, auch im

Zusammenhang mit der Produktverknüpfung. Wichtig ist auch, wie PSPs (z.B. PayPal) in Aimeos eingerichtet und konfiguriert werden und welche Payment Provider über Aimeos bereitgestellt werden. Abschließend sollte geprüft werden, inwieweit Templates im Aimeos angepasst werden können. Beispielsweise werden Use Cases zur Anpassung von Formularen getestet, vor allem die Anpassung von Validierungs- und Pflichtfeldern sowie die Konfiguration einzelner Komponenten, wie z.B. Einrichten eines Teasers im Warenkorb oder ein Abschnitt „Andere Kunden haben auch gekauft“ im Warenkorb.

Am Ende der Prüfung dieser Use Cases wird Aimeos entsprechend bewertet und weitere Schritte zur Verwendung des Frameworks werden, sofern Aimeos positiv bewertet werden kann, unternommen.

5 Einrichtung von Aimeos

Aimeos wird im Rahmen dieser Arbeit in Verbindung mit TYPO3 eingesetzt und entsprechend als Extension integriert und installiert. TYPO3 läuft hier auf dem localhost der über einen Apache-Server aufgerufen wird, der mit Xampp gestartet wird. Die Erweiterung wird in Version 18.10.7 in die Version 8.7 TYPO3s integriert. Aimeos bietet eine vorkonfigurierte Distribution für Demozwecke an, die im Rahmen dieser Arbeit verwendet wird, da verschiedene Tests der oben definierten Use Cases mit vordefinierten Testdaten und Shopseiten durchgeführt werden sollen.

Nachdem alle Extensions in TYPO3 über die Registerkarte „Extensions“ hochgeladen und installiert wurden, wurde ein Skript zum Aktualisieren für die Aimeos-Shopextension ausgeführt und alle Tabellen in der verbundenen Datenbank wurden erstellt und mit Demodaten gefüllt. Zusätzlich wurde die Distribution der Aimeos Web Shop-Page installiert und aktiviert, um einen speziellen Seitenbaum zu erhalten. Danach musste der Shop-Seitenbaum konfiguriert werden.

Zu diesem Zweck wurde der ganze Templaterecord der Seite in der Benutzeroberfläche unter der Kategorie „Templates“ bearbeitet. Der vorhandene Wert *config.typolinkLinkAccessRestrictedPages* musste durch die ID der Seite „My Account“ ersetzt werden. Diese wurde standardmäßig von TYPO3 für jede Seite festgelegt. Letztlich musste im Ordner „Users“ für den Benutzer „test“ ein Passwort festgelegt werden. Dann konnten die Use Cases für Aimeos getestet werden.

6 Funktionstest interner Use Cases

6.1 Datentransfers

6.1.1 Importfunktionen

In der Regel verfügen Unternehmen bereits über ein ERP-System, das alle ihre Produkte verwaltet und die meisten Informationen, die benötigt werden, um den Shop zu füllen, werden dort gespeichert. In diesem Fall kann in Aimeos ein neuer Produktmanager erstellt werden, um auf die erforderlichen Informationen direkt aus dem ERP-System zuzugreifen. In diesem Teil der Arbeit sind jedoch vor allem der Import aus dem ERP-System in den Shop sowie die regelmäßigen Aktualisierungen der importierten Daten zu berücksichtigen. Der Produktimport-Jobcontroller wird für diesen Use Case verwendet.

Ein Import wird durch einen Cronjob/ Scheduler ausgelöst, der den entsprechenden Jobcontroller ausführt, z.B. der Jobcontroller "product/import/csv"[17]. In Anhang 1: Definition eines Cron-Jobs, wird dargestellt wie ein Cron-Job in TYPO3 definiert wird. Es gibt auch einen Import-Button im Admin-Interface, aber nur Texte für bereits vorhandene Produkte können hier importiert werden, nicht jedoch Produkte.

Um den Import von Produkten in Aimeos zu testen, wurden zunächst Produktdaten aus dem ERP-System exportiert, beispielsweise wurden hier Daten aus einem Demo-Shop von Magento verwendet, die Produkte eines Yoga-Shops darstellen. Die Produkte wurden als Comma-Separated-Value-Datei (CSV) und in einem Ordner gespeichert, der im Scheduler mit einem Location Setting im TypoScript angegeben wurde (Abbildung 2: TypoScript Konfiguration).

TypoScript configuration

```
controller.jobs.product.import.csv.location = C:\xampp\htdocs\fileadmin\product_imports
controller.jobs.product.import.csv.skip-lines = 1
```

Abbildung 2: TypoScript Konfiguration

[17] [Aim18b]

Der Containertyp und das Inhaltsformat konnten mit den Einstellungen *Controller.jobs.product.import.csv.container.type* und *Controller.jobs.product.import.csv.container.content* konfiguriert werden. Je nach Container / Inhaltstyp konnten zusätzliche Optionen verwendet werden, die ausführlich in der Aimeos-Dokumentation beschrieben wurden. Da die Datei in einem Directory gespeichert wurde, konnte der Standardcontainertyp „Directory“ verwendet werden. Weitere Containertypen, die genutzt werden konnten, sind ZIP-Container und PHPExcel-Container. Außerdem musste beachtet werden, dass sich die CSV-Datei nur einzeln im angegebenen Directory befinden durfte. Der Inhaltstyp musste immer ein CSV-ähnliches Format sein. Derzeit wurden zwei Formattypen unterstützt: CSV und PHPExcel.

Für CSV-Dateien gibt es im Allgemeinen viele Möglichkeiten, sie zu formatieren, da CSV nicht standardisiert ist, mit der Ausnahme, dass Felder normalerweise durch ein Komma getrennt werden, was z.B. im Englischen auch nicht gültig ist. Für Aimeos ist ein bestimmtes, festgeschriebenes Format erforderlich, das mit einem VisualBasic-Makro in Excel erstellt wurde. Die Regeln des Aimeos-Formats besagen hauptsächlich, dass Felder in doppelte Anführungszeichen („) eingeschlossen und durch Kommas getrennt werden müssen, doppelte Anführungszeichen müssen in doppelte Anführungszeichen gesetzt werden und das Ende einer Zeile muss ein New Line Char sein (oder ein „\r\n“ in Windows).[18] Die CSV-Datei kann auch mit einem Header beginnen, in dem zur besseren Lesbarkeit die Spaltennamen stehen können. (siehe Abbildung 3: Beispiel eines CSV-Formats)

```
"sku","attribute_set_code","product_type","categories","catalog
"24-MB01","Bag","default","Gear","Gear/Bags","","","base","Joust
<ul>
<li>Dual top handles.</li>
<li>Adjustable shoulder strap.</li>
<li>Full-length zipper.</li>
<li>L 29"" x W 13"" x H 11"".</li>
</ul>","","","1","Taxable Goods","Catalog, Search","350000","","'
\n
```

Abbildung 3: Beispiel eines CSV-Formats

In diesem Fall musste der Import im TypoScript so konfiguriert werden, dass die erste Zeile übersprungen wurde, indem die Konfiguration *controller.jobs.product.import.csv.skip-lines = 1* hinzugefügt wurde. Um den CSV-Import weiter zu optimieren, konnte die Einstellung

[18] Vgl. Ebenda

controller.common.product.import.csv.max-size verwendet werden, um den Import zu beschleunigen oder den Speicherverbrauch des Imports zu reduzieren. Dies war jedoch in diesem Fall nicht erforderlich, da der Import trotz einer CSV-Datei mit 2048 Zeilen sehr schnell lief (ca. 53,83s). Da TYPO3 in Verbindung mit Aimeos verwendet wurde, musste die gesamte Konfiguration in das Feld TSConfig der CSV-Scheduler-Task für den Produktimport eingegeben werden (siehe Abbildung 2: TypoScript Konfiguration). Wenn beim Importieren großer Dateien Probleme auftraten oder der Status des Fortschritts nicht angezeigt wurde, wurden Logs in der TYPO3-Logdatei oder auf der Registerkarte „Overview“ der Admin-UI von TYPO3 angegeben.

Das Standard-Mapping konnte für die genutzte CSV-Datei (Ausschnitt siehe Abbildung 4: CSV-Testdatei (Detail)) nicht verwendet werden und musste demnach entsprechend angepasst werden. Während des Imports werden die Felder einer Zeile in einem Array gespeichert und dann über TypoScript entsprechend ihrer Position ausgewählt, sodass sie einem Element des Produkts zugeordnet werden können. In TypoScript werden zusammengehörende Felder in Gruppen zusammengefasst, wobei die Keys dieser Gruppen den Namen der Datenprozessoren entsprechen, die den Import der Daten übernehmen. Wenn der Prozessor bei der Verarbeitung des Mappings denselben Domänenelementschlüssel findet (z.B. „Text.content“), beginnt er mit einem neuen Dataset. Außerdem muss der *product.code*-Elementschlüssel in der CSV immer an erster Stelle stehen. Es war auch zu beachten, dass die erste Position sowohl bei CSV als auch bei TypoScript bei 0 beginnt.

```
"MH01-L-Black","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-L-Gray","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default C
"MH01-L-Orange","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-M-Black","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-M-Gray","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default C
"MH01-M-Orange","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-S-Black","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-S-Gray","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default C
"MH01-S-Orange","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-XL-Black","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-XL-Gray","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-XL-Orange","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Defaul
"MH01-XS-Black","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-XS-Gray","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Default
"MH01-XS-Orange","Top","default","Men/Tops/Hoodies & Sweatshirts","Collections/Eco Friendly","Defaul
```

Abbildung 4: CSV-Testdatei (Detail)

Um das Standard-Mapping für die Datei mit den Produktdaten aus dem ERP anzupassen, wurde die TypoScript-Konfiguration *controller.common.product.import.csv.mapping* verwendet (siehe Anhang 2: TypoScript Mapping Konfiguration). Die Gruppennamen konnten frei umgeordnet werden. Dadurch wurde die Reihenfolge geändert, in der die Daten importiert wurden. Zuvor mussten jedoch die Kategorien und Attributtypen, die die Produkte im Magento verwendeten, manuell importiert oder erstellt werden. Anschließend konnte die CSV mit dem Mapping in TypoScript sowie das Format der CSV, insbesondere die Quotes, mit VisualBasic angepasst werden.

Beim Importieren der Kategorien war zu beachten, dass nur 32 Zeichen für Kategoriencodes (entsprechen IDs) verfügbar waren. Außerdem wurden Kategorien in Aimeos als Baumstruktur angeordnet, sodass es immer nur eine Stammkategorie gab. Die Kategorien mussten dann in der Oberfläche von Aimeos hinzugefügt werden.

Beim Produktimport wurde der Pfad der Datei zuerst in TypoScript angepasst. Einige weitere Merkmale der Produkte mussten separat angepasst werden, da sie bei dem Import nicht berücksichtigt wurden. Daher mussten Suchmaschineneinstellungen wie Meta-Keywords oder Sonderpreisdaten und Basketlimits in der UI manuell festgelegt werden. Bilder wurden in Aimeos extra per File Transfer Protocol (FTP) importiert, der Pfad konnte hier auch im TypoScript eingestellt werden. Mehrere Bilder konnten dabei mit \n (new Line) getrennt werden. Artikel mit Varianten mussten separat ausgelagert werden. Hierbei handelte es sich um Artikel mit Farbvarianten, verschiedenen Größen usw. Damit dies funktionierte, mussten zunächst die Unterartikel mit ihren Varianteneigenschaften und anschließend das Selektorprodukt erstellt werden, in das die Varianten integriert wurden. Der Selektor musste dann weiterhin aktiviert werden, da er von Aimeos standardmäßig deaktiviert wurde. Auch musste für jedes neue Variantenattribut eine neue „variant“-Spalte in der CSV erstellt werden, da Aimeos sonst nur ein Variantenattribut importierte. Bei den Preisen war zu beachten, dass eine Currency angegeben werden musste (default = Dollar oder EUR) und sie durch einen Punkt getrennt werden mussten. Bestimmte Spalten wie Texttypen (long oder short) mussten ebenfalls noch manuell hinzugefügt werden.

Abschließend lässt sich sagen, dass der Import durch die vielen Bearbeitungen und Anpassungen für eine Migration von Produktdaten ins Aimeos sehr aufwendig ist, für neue

Daten ist der Import eher bequem (auch für Bulks). Die Ladezeit ist dabei auch sehr kurz. Des Weiteren könnten auch Setup-Tasks verwendet werden, um Daten zu integrieren. Mit diesen kann jedoch nur die Datenbank bearbeitet werden, für diesen Fall ist dies jedoch eher ungeeignet, höchstens die Länge des Codes (entspricht der ID in Aimeos) für die Kategorien kann erhöht werden und Vorteile bringen. Eine andere Möglichkeit, den Import zu vereinfachen, wäre die Implementierung eines neuen Jobcontrollers mithilfe des Produktcontrollers.

6.1.2 Exportfunktionen

Als wichtiger Use Case für Exportfunktionen wurde der Export von Bestellungen definiert und der Export von Kundendaten als Use Case mit Priorität B. Zunächst wurde der Export von Bestellungen betrachtet.

Bestellungen können als CSV und als Extensible Markup Language (XML) exportiert werden, wobei der Lieferdienstanbieter „Standard“ Bestellungen im XML-Format exportiert und an ein Remote-Gateway über das Hypertext Transfer Protocol (http) sendet. Dieses muss als Zustellungsoption konfiguriert werden (der Name der Option ist selbst konfigurierbar) und die *process order delivery services* Scheduler Task kann ausgeführt werden. Der Task kann jedoch auch so konfiguriert werden, dass er eine Datei exportiert und ggf. das XML entsprechend angepasst wird.

Zuerst wurde der CSV-Export genauer betrachtet. Zu diesem Zweck wurden im Frontend des Webshops einige Bestellungen mit Beispielkunden angelegt. Beim Export von Standardbestellungen von Aimeos wurde der Scheduler zum Exportieren von Bestellungen verwendet, wenn auf die entsprechende Schaltfläche in der Admin-User Interface (UI) geklickt wurde. Es musste also manuell interagiert werden, um dies zu tun. Die entsprechende Scheduler-Task wurde also erstellt und entsprechend ausgeführt. Die Export Files landeten dann in der Admin-UI und konnten über eine Download-Schaltfläche heruntergeladen werden. Dies war eine einfache und schnelle Methode, um die Bestellungen als CSV-Datei abzurufen, erfüllte jedoch nur teilweise den Use Case, da jedes Mal eine manuelle Interaktion nötig war, indem auf die Schaltfläche „Download“ geklickt wurde.

Darüber hinaus gibt es den „Process order delivery services“ Job, der z.B. an ein ERP-System sendet. Zu diesem Zweck kann für jeden Delivery Service in Aimeos eingestellt werden, ob die

Order Handling manuell oder standardmäßig ausgeführt werden soll (siehe Abbildung 5: Optionen des Delivery Service Providers). Mit der Standardeinstellung können Bestellungen im XML-Format an einen Web Service gesendet werden, der diesen Content versteht (die URL des Web Services muss in Aimeos angegeben werden). Dafür kann ein Template in die eigene Erweiterung kopiert und geändert werden. In diesem können View-Helper verwendet werden wie „*$this->url()*“ für Links. Das Ganze ist so gemeint, dass es in bestimmten Abständen über den entsprechenden Cron-Job gestartet wird. Der entsprechende Scheduler oder der Provider der Task können auch programmgesteuert erweitert werden, sodass eine Datei exportiert wird.

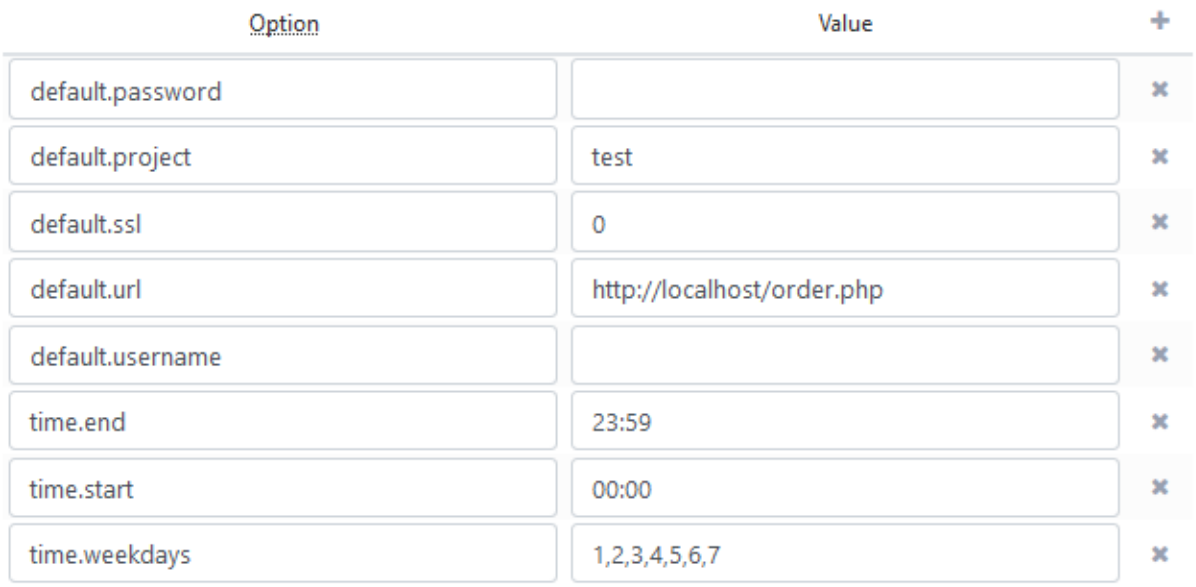


Abbildung 5: Optionen des Delivery Service Providers

Ein „Notify on Stock“ kann auch als Cron-Job für Artikel erstellt werden. Dieser benachrichtigt Kunden, wenn ein Artikel wieder auf Lager ist.

Ein weiterer Use Case, der Export von Kundendaten, ist in Aimeos nicht direkt möglich, ohne einen eigenen Job Controller oder eine Option für die UI zu erstellen, was programmatisch aufwendig ist, da Test und Implementierung hiervon viel Zeit in Anspruch nehmen können. Kundendaten werden in der Tabelle *fe_users* gespeichert.

Das Erstellen eines neuen Job-Controllers erfolgt hauptsächlich in einer Standard-Klasse. Der Name des Job Controllers entspricht seiner Position im Dateisystem. So ist *Controller/Jobs/Customer/Export/Csv/Standard.php* dem Namespace *Aimeos\Controller\Jobs\Customer\Export\Csv\Standard* zugeordnet. In neuen

Implementierungen wie in der Beispielimplementierung ist „Standard“ der Name der Klasse, da es sich um die Standardimplementierung handelt. Zusätzlich muss die abstrakte Basisklasse „*Aimeos\Controller\Jobs\Base*“ erweitert werden. Durch die Implementierung der *Aimeos\Controller\Jobs\Iface*-Schnittstelle wird die Beispielklasse schließlich zu einem Job-Controller. Code kann nicht im Konstruktor der Klasse implementiert werden, z.B. abrufen von Datensätzen aus der Datenbank oder Protokolle. Hierfür wird stattdessen die *run()*-Methode genutzt oder es werden neue Funktionen geschrieben.

Die Methode *getName()* wurde verwendet, um einen Namen in der Shop Besitzer Sprache anstelle des Schlüssels (z. B. *Customer/Export/Csv*) für den Beispiel-Job-Controller anzuzeigen. Diese gab eine Zeichenfolge zurück, die übersetzt werden konnte. Dies geschah durch die *dt()*-Methode des Internationalisierungs-/Übersetzungsobjekts.

Aimeos unterstützt mehrere Sprachen und Währungen in einer Shop-Site. Die Kernbibliothek (aimeos-core) enthält neben Composer-Paketen und Source Code auch mehrere Übersetzungsdomänen. Für Job-Controller muss aber immer die Übersetzungsdomäne „*Controller/Jobs*“ wie in Abbildung 7 gezeigt, verwendet werden. Der zweite Parameter der *dt()*-Methode ist der Name, d.h. der Name des Jobcontrollers auf Englisch (siehe Abbildung 6: *getName()*-Methode).

```
return $this->getContext()->getI18n()->dt
       ('controller/jobs', 'Customer export csv');
```

Abbildung 6: *getName()*-Methode

Eine aussagekräftigere Meldung über die Funktionalität des Beispieljobcontrollers musste von der *getDescription()*-Methode zurückgegeben werden. Die Übersetzungsdomäne wurde auch als „*Controller/Jobs*“ bezeichnet. Die Beschreibung durfte nicht länger als 250 Zeichen sein (siehe Abbildung 7: *getDescription()*-Methode).

```
return $this->getContext()->getI18n()->dt('controller/jobs',
       'Exports all available customers to Csv');
```

Abbildung 7: *getDescription()*-Methode

Die eigentliche Arbeit wird von der *run()*-Methode des Beispiel-Job-Controllers ausgeführt. Diese Methode führt die definierten Aufgaben aus. Hierbei werden die im Verzeichnis

lib/mshoplib/src verfügbaren Manager verwendet, z. B. um Daten zu speichern oder zu löschen (siehe Abbildung 8: *run()*).

```
$manager = \Aimeos\MShop\Factory::createManager
           ( $this->getContext(), 'customer');


$search= $manager->createSearch();
$search->setConditions($search->compare('==', 'pid', 48));


$manager->saveItem(false);
```

Abbildung 8: *run()*

Je nach Aufgabe kann die aktuelle Sprache und Währung des im Kontext gespeicherten Site-Elements geändert oder sogar auf null gesetzt werden. Dies ist für das Beispiel allerdings nicht weiter notwendig. Der Code selbst erstellt einen neuen Job-Eintrag für die exportierte Datei.

Das Testen der Funktion, der selbst implementierten Job-Controller, konnte über PHPUnit-Tests oder direkt am localhost durchgeführt werden. Die Implementierung der Unit-Testfälle unterschied sich nicht von anderen Unit-Tests. Die Testklasse nutzte dabei auch einen entsprechenden Namespace. Die Methoden aus Aimeos's Factory-Klasse *setCache()* und *clear()* wuren hier verwendet. (siehe Beispiel in Abbildung 9: Unit Tests)

```
\Aimeos\MShop\Factory::setCache(true);


$this->context = \TestHelperJobs::getContext();
$this->aimeos = \TestHelperJobs::getAimeos();
```

Abbildung 9: Unit Tests

Diese Methoden ermöglichen das Zwischenspeichern von Managerobjekten nur für die Testfälle dieser Komponententestklasse und löschen dann den Objektcache. In der *testRun()*-Beispielmethode wird die Unit-Test-Assertion indirekt überprüft, indem nach der generierten Datei gesucht wird.

6.2 Allgemeine Content-Funktionen

Für diesen Bereich der Arbeit wurden alle Use Cases berücksichtigt, die ohne großen Aufwand direkt über die UI des Aimeos bearbeitet werden konnten. Dazu gehörten vor allem die Erstellung von Promotions und Einstellungen für PSP. Im Zusammenhang mit Promotions wurde zunächst überlegt, wie sie für Produkte erstellt werden können. Somit konnten diese sowohl auf einzelne Artikel als auch auf Auswahlprodukte mit Varianten angewendet werden. Diese werden in Aimeos „Selection"-Produkte genannt und haben abhängige Produktvarianten. Produkte selbst haben konfigurierbare Optionen für alle Produkttypen, für die unterschiedliche Preise gelten, z.B. stellen sie benutzerspezifische Rabatte sicher.

In Aimeos können generell Preise für integrierte Kalkulationen verwendet werden. Dies berücksichtigt sowohl zusätzliche Kosten für konfigurierbare Optionen als auch Extrakosten pro Artikel. Außerdem können individuelle Steuersätze pro Artikel hinzugefügt werden. Für den speziellen Use Case „Promotions" ist es jedoch auch wichtig, dass Sonderangebote auch direkt zu reduzierten Preisen erstellt werden können.

Verfügbare Rabatte umfassen den Festpreis-Rabatt, Prozentsatz-Rabatt und kostenlosen Versand. Diese konnten anhand von Preisen erstellt werden. Dazu wurden die Preise der in Aimeos importierten Produkte genutzt. Bestimmten Preisen wurde der Typ Rabatt zugeordnet. Die Darstellung konnte dann anhand von Datum oder Nutzergruppe erfolgen. Somit wird der Preis entsprechend als Rabatt gehändelt. Kopfrabatte und Positionsrabatte lassen sich jedoch nur mit Abänderungen der Plugins in Aimeos darstellen.

Als Form einer Promotion können auch Abonnements direkt in der Produktverwaltung erstellt werden. Außerdem können virtuelle Produkte als Gutscheinprodukte erstellt werden, die dann einen Gutscheincode an den Kunden senden. Der generierte Gutscheincode kann vom Kunden selbst oder als Geschenk für eine andere Person verwendet werden. Zusätzlich muss der Gutschein-Provider eingerichtet werden. Aimeos kann verschiedene Rabatte anbieten, wenn die Gutscheinerweiterung installiert ist. Ein Gutscheinprodukt kann entweder ein echtes Produkt oder ein spezielles Rabattprodukt sein. Gutscheincodes können dann in einem Eingabefeld von Kunden entsprechend eingelöst werden, die Anzahl der Gutscheincodes pro Bestellung ist konfigurierbar.

In diesem Zusammenhang kann auch erwähnt werden, dass der Teaser im Warenkorb (auch ein Use Case) direkt hier angelegt und konfiguriert werden kann. Dargestellt werden können hier Produktabbild und -name sowie Attribute der Produktvarianten. Darüber hinaus werden in separaten Zeilen die Preise pro Einheit und insgesamt mit Mehrwertsteuer pro Steuersatz einschließlich Skonti und Versand- und / oder Zahlungskosten angezeigt. Alle diese Optionen werden direkt über die Plugin-Einstellungen vorgenommen. Aimeos kann dabei in TYPO3 in seiner Präsentation konfiguriert werden. Eines dieser Plugins sorgt für konfigurierbare Grenzwerte, z. B. wenn etwas versandkostenfrei ist. Ein anderes Plugin sorgt für Basketchecks und/ oder -aktualisierungen, z.B. werden Rabatte bei Korbänderungen unabdingbar geändert. Diese Plugins können auch entsprechend angepasst werden.

In Aimeos kann direkt der Use Case für die Bestandabfrage in kleinen Teilen erfüllt werden. In allen Listen- und Detailansichten gibt es dynamische Bestandsinformationen. Hier können auch Optionen für die Vorbestellung nicht vorrätiger Produkte und Angaben zu „Back in stock“ festgelegt werden. Standardmäßig zeigt das Frontend auch Hinweise für Kunden zu geringen Lagerbeständen an, die in den entsprechenden HTML-Files angepasst werden können. Es kann auch definiert werden, ob nicht vorrätige Produkte angezeigt werden oder nicht.

Auch der Bezahlstatus ist in Aimeos über Job-Controller abfragbar. Über Abfragen ist es auch möglich Bestellungen bei Nichtbezahlung zu stornieren. Ansonsten können Zahlungen im Backend des PSPs erstattet werden. Der PSP benachrichtigt in der Regel dann den Shop. Im Shop kann man dann daraufhin, auch über einen Job Controller, eine E-Mail-Nachricht an den Kunden auslösen. Dies erfüllt auch gleichzeitig den E-Mail Use Case. Über den Scheduler können hierbei verschiedene Zeiträume für den Versand von E-Mails eingestellt werden. Sobald der Cronjob ausgeführt wurde, erscheint in der entsprechenden Tabelle in der Datenbank der Datensatz „E-Mail versandt“.

6.3 Änderungen an Aimeos

6.3.1 Erstellen einer „myaimeos“-Erweiterung

Aimeos ist auf Erweiterbarkeit ausgelegt. Durch das Erstellen einer benutzerdefinierten Erweiterung können ganz einfach neue Funktionen für das Aimeos Shop-Beispielprojekt hinzugefügt werden. Daher kann noch auf spätere Versionen aktualisiert werden. Die neue Erweiterung für das Projekt wurde mit dem Aimeos-Extension-Generator erstellt. Dieser

generiert ein ZIP-Paket zum Herunterladen. Zusätzlich können installierbare Pakete für CMS mit vorkonfigurierten Aimeos-Erweiterungen erstellt werden.

Für die CMS-Anwendung TYPO3 musste die eigene Aimeos-Erweiterung Teil einer TYPO3-Erweiterung sein. So wurde dies mit dem Builder für Aimeos-Erweiterungen erstellt. Die aus der generierten ZIP-Datei entpackten Erweiterungsdateien und -verzeichnisse mussten in einem Unterverzeichnis des TYPO3-Erweiterungsverzeichnisses („*./ext/myaimeos/*“) gespeichert werden. Für das Projekt musste nur eine Erweiterung erstellt und der gesamte Code hinzugefügt werden. Dadurch wurden die Antwortzeiten niedrig gehalten, da weniger nach Konfigurationseinstellungen und Übersetzungen gesucht werden musste. Die Erweiterung konnte sofort verwendet werden.

Eine Aimeos-Erweiterung spiegelt die Verzeichnisstruktur des Kerns wider und enthält die folgenden Hauptverzeichnisse: *admin/jqadm*, *admin/jsonadm*, *client/html*, *client/jsonapi*, *controller/common*, *controller/frontend*, *controller/jobs*, *lib/custom*. *Client/html* enthält das HTML-Frontend und die Klassen sowie Templates, die den HTML-Output generieren. Es sind auch die Standarddesigns enthalten. In diesem Verzeichnis werden Anpassungen vorgenommen, um die Templates für den Shop zu testen. Jede Klasse in „controller/jobs“ führt einen Task aus. Weiterer Code befindet sich im „common“ Verzeichnis. Dies umfasst hauptsächlich den Quellcode für die Medien- und Auftragsbearbeitung sowie CSV-Importprozessoren und -caches. Die untere Ebene des Cores, einschließlich der Adapter- und Datenzugriffsmanagerebene (lib/mwlib und lib/mshoplib im aimeos core), verwendet das Verzeichnis *lib/custom*.

6.3.2 Anpassen von Templates

Die Template Ansichten selbst sind in Aimeos recht einfach: Sie bestehen aus HTML mit alternativer PHP-Syntax. Auf alle Daten kann zugegriffen werden. Bei derselben Directory Struktur können die eigenen Templates die Templates von Aimeos überschreiben.

Die HTML-Client-Struktur organisiert diese Templates im Aimeos z. B. so:

```
|Client/Html/Catalog/Detail/Standard
| => template: catalog/detail/body-default.php
| => assigns: detailProductItem, detailProductMediaItems, etc.
|- Client/Html/Catalog/Detail/Basket/Standard
| => template: catalog/detail/basket-body-default.php
| => assigns: basketStockUrl, etc.
|-- Client/Html/Catalog/Detail/Basket/Selection/Standard
| => template: catalog/detail/basket-selection-body-default.php
| => assigns: selectionProducts, selectionAttributeItems, etc.
```

Client/Html/Catalog/Detail/Standard HTML Client-Template hat z.B. Zugriff auf detailProductItem. Um herauszufinden, welche Daten zugeordnet werden können, ist es möglich, die Templates und die HTML-Clients zu durchsuchen. Darüber hinaus sind einige View Helper verfügbar. Sie befinden sich in Aimeos-Core.

View Helper sind in Aimeos Funktionen, die in bestimmten View Helper-Klassen definiert werden und in z.B. den Template-Ansichten verwendet werden können (Beispiel siehe Abbildung 10: Verwendung des *url*()-View Helpers

). Die Parameter für jeden View Helper sind unterschiedlich.

```
<?php echo $this->url( $reviewTarget, $reviewController,
$reviewAction, $reviewParams, $reviewTrailing, $reviewConfig ); ?>
```

Abbildung 10: Verwendung des *url*()-View Helpers

Die View Helper umfassen z. B.:

- *url* (generiert eine URL für das definierte Ziel)
- *block* (Erweiterung der übergeordneten Templates wie in Fluid)
- *config* (Aufruf der Konfigurationseinstellungen)
- *content* (erstellt eine URL aus einem in der Datenbank gespeicherten Pfad)

- *csrf* (sichert Formpost mit einem Cross-Site-Request-Forgery-Token (CSRF-Token))
- *date* (übersetzt ein Datum in lokalisiertes Format)
- *encoder* (Daten werden vor dem Schreiben in Anführungszeichen gesetzt, um Sicherheitsverletzungen wie Cross Site Scripting (XSS) zu verhindern)
- *formparam* (generiert den Wert für das Attribut „name“ der Eingabeelemente)
- *mail* (bietet den Zugriff auf eine E-Mail-Nachricht an)
- *number* (gibt den lokalisierten Integer oder den Dezimalwert für die angegebene Zahl zurück)
- oder *param* (gibt den GET / POST / URL-Parameter oder eine Liste zurück).

Hinsichtlich der Use Cases konnten über das Anpassen der Templates die HTML-Ansichten der Plugins und der Shopseiten von Aimeos bearbeitet werden. Dazu zählten z.B. das Checkout-Layout oder das Produktdetail.

7 Auswertung

Die Use Cases wurden schließlich unter Beachtung ihrer Prioritäten mit den Schulnoten von eins bis sechs bewertet. Ein grober Überblick wie viele Use Cases mit Aimeos erfüllt werden konnten, ist in Tabelle 1: Überblick zur Erfüllung der Use Cases zu sehen.

Priorität der Use Cases	**Anzahl der Use Cases**	**Use Cases die in oder mit Aimeos erfüllt werden können (Schulnoten 1-3)**
C	1	1
B	7	6
A	13	10

Tabelle 1: Überblick zur Erfüllung der Use Cases

Der Use Case „Import von Kundendaten“ der Kategorie C wurde mit 3 bewertet, da dieser nicht direkt verfügbar war. Es bestand jedoch die Möglichkeit, eigene Klassen zu implementieren, so dass ein Import über einen neuen Job Controller oder eine Option fürs UI möglich war. Dafür war allerdings Entwickleraufwand notwendig.

Für Use Cases mit Priorität B beträgt die durchschnittliche Bewertung 2.43, was sozusagen einer drei entspricht. Die schlechteste Bewertung mit einer sechs ist der Use Case für einen Rabatt wie „3 für 2“. Das liegt daran, dass dies in Aimeos standardmäßig nicht verfügbar ist. Auch der Use Case für den Export der Kundendaten wird mit einer 3 eher mittelmäßig bewertet, was jedoch ebenso wie beim Import darauf zurückzuführen ist, dass dies nur durch die Implementierung eines neuen Job Controllers möglich ist. Die verbleibenden Use Cases der Kategorie B wurden von eins bis drei besser bewertet.

Damit Aimeos jedoch als Framework genutzt werden kann, sollten mindestens acht der Use-Cases mit Priorität A mindestens eine Note drei erreicht haben, was bereits in Kapitel vier: Zielstellung und Ist-Zustand definiert wurde. Tatsächlich konnten bereits neun der Use Cases dieses Ziel erreichen, darunter die Erstellung von Gutscheinen, Produktimport, Bestandsimport, Promotion-Erstellung, Teaser-Erstellung, Erstellung von PSPs und Formularanpassungen,

Pflichtfelder und Validierung. Obwohl Use Cases wie Bestellexport, Prozentsatzrabatt und Produktverlinkung mit vier bewertet wurden, muss bei diesen Use Cases jedoch berücksichtigt werden, dass sie auch mit einem geringen Entwicklungsaufwand erfüllt werden können. So könnte z. B. ein Basket-Plugin geschrieben werden, dass den gesamten Warenkorb bearbeitet und einen Prozentsatzrabatt bzw. Kopf- und Positionsrabatt entsprechend hinzufügt. Ein besonderes Problem mit dem Use Case „Produktkachel auf Contentseite“ besteht darin, dass Einstellungen für Aimeos für Laravel (PHP Framework) hier zwar möglich sind, aber nicht für TYPO3, da es hier standardmäßig keine Kacheln als Content-Elemente gibt. Allerdings ist es möglich eigene Content-Elemente zu implementieren. Allerdings ist dies sehr aufwendig, da hierfür im Bereich der Admin-UI-Files implementiert werden muss. Außerdem ist es nicht sehr effizient für jedes Content-Element mehrere Klassen zu implementieren, da es für diese oft großen bzw. speziellen Bedarf gibt.

Abschließend kann gesagt werden, dass Aimeos definitiv als E-Commerce-Framework in der dotSource verwendet werden kann. Das wahre Potenzial dieses Frameworks wird sich aber erst im produktiven Einsatz mit einer Kombination aller Schnittstellen wie PIM, ERP, PSPs usw. zeigen.

8 Zusammenfassung

In dieser Projektarbeit wurden verschiedene Use Cases aufgestellt und im Laufe der Arbeit entsprechend bewertet. Insbesondere wurden die Priorität des Use Cases und der Umfang, in dem er bereits in Aimeos integriert wurde, und/ oder wie die Implementierungsoptionen aussahen, betrachtet, um diese Bewertung zu erstellen. Die Use Cases wurden dann nach Schulnoten bewertet. Das Ganze wurde in eine Excel-Tabelle geschrieben und die Bewertung wurde mit einer kurzen Beschreibung begründet (siehe Anhang 3: Use Cases Aimeos). Darüber hinaus wurden mögliche Erweiterungen der Use Cases in Aimeos in Betracht gezogen, was einige Ratings positiv beeinflussen konnte. Dann konnte eine Schlussfolgerung gezogen werden. Abschließend konnte nun entschieden werden, ob sich Aimeos als Ersatz für ein Shop-System aus zwei Systemen als E-Commerce-Framework etablieren konnte.

Ein besonderer Fokus wurde auf die Use Cases für Import- und Exportfunktionen sowie die Templatebearbeitung gelegt. Sie konnten also anhand von Beispieldaten, die das Aimeos selbst zur Verfügung gestellt hatte, sowie mit Daten aus dem Magento als Beispielsystem getestet werden. Die Anwendungsbeispiele für Import- und Exportfunktionen funktionierten alle mit einem Cronjob-Scheduler und waren über TypoScript konfigurierbar (z.B. Directory, Mapping, Backups). Der gesamte Importvorgang erfolgte über eine CSV-Datei. Exportvorgänge waren auch mit CSV-Dateien möglich. Der Standardexport erforderte jedoch eine manuelle Interaktion mit dem Scheduler. Zur Erweiterung der Funktionalitäten konnten die Job Controller entsprechend angepasst werden.

Wichtige Bereiche, wie das Erstellen von Promotions konnten im Aimeos selbst im Katalog- oder Produktpanel vorgenommen werden. Hier wurde Aimeos eigenes Rabattkonzept verwendet. Mit Templates und Clients konnte deren Darstellung vollständig den eigenen Bedürfnissen angepasst werden. Somit konnten Erweiterungen von mehreren, z.B. Promotionssektionen erfolgen. Das Rabattkonzept von Aimeos konnte auch durch Bearbeiten des Basket-Plugins geändert werden, das dem Warenkorb ein Rabattprodukt mit einem negativen Preis hinzufügte. Es musste also nicht das gesamte System angepasst werden. Ansonsten konnten Rabatte auch auf Preisbasis angelegt werden. Hier mussten die Preise nur in Aimeos angelegt werden. Danach wurde ihnen nur noch der Typ „Rabatt“ zugewiesen. Weitere Use Cases konnten auch über Template Anpassungen erfüllt werden.

9 Fazit

Mit der angewandten Methodik konnten die Use Cases erfolgreich nacheinander abgearbeitet und bewertet sowie umschrieben werden. Jedoch hätte man, statt das System lokal mit Xampp aufzusetzen, auch eine Virtual-Machine (VM) nutzen können. Dadurch könnten die Beispielimplementierungen, welche zusätzlich auch auf GitLab hochgeladen werden könnten, dort in PhpStorm implementiert werden und diese sowie die ganzen Systemeinstellungen für Aimeos und TYPO3 für andere Mitarbeiter des Unternehmens zugänglich gemacht werden. Auch Xampp hätte ersetzt werden können, da das System über dieses langsam lief. Stattdessen hätte man so einen Apache-Server auf Ubuntu laufen lassen und entsprechend eine MySQL Datenbank einrichten können.

Insgesamt hätte das Testen von Aimeos auch durch das Aufsetzen eines kompletten Beispielprojekts erfolgen können. So hätte ein klareres Gesamtergebnis mit einem beispielhaft aufgebauten Webshop mit entsprechenden Funktionalitäten dargestellt werden können. Jedoch konnten die einzelnen Use Cases mit der in dieser Arbeit angewandten Methodik spezieller und umfassender betrachtet werden. Außerdem konnten so mehr Use Cases betrachtet werden, als es mit dem Aufsetzen eines beispielhaften Web Shops möglich gewesen wäre.

Mit der aus dieser Arbeit resultierenden Bewertung kann Aimeos nun für ein geeignetes Kundenprojekt verwendet werden.

Literaturverzeichnis

[aima] aimeos: „Was ist ein E-Commerce Framework?“. https://aimeos.org/tips/ecommerce-framework-de/ Abruf: 2019.04.09

[aimb] aimeos: „Was ist ein E-Commerce Framework?“. https://aimeos.org/tips/ecommerce-framework-de/ Abruf: 2019.04.03

[aimc] aimeos: „What is Aimeos?“. https://aimeos.org/project/what-is-aimeos/ Abruf: 2019.04.17

[aim17] aimeos: „Aimeos-stack.png“, 2017. https://aimeos.org/tips/wp-content/uploads/2017/06/Aimeos-stack.png Abruf: 2019.04.08

[Aim18a] Aimeos documentation: „User Manual/Administration Interface/Plugin list“, 2018. https://aimeos.org/docs/User_Manual/Administration_Interface/Plugin_list Abruf: 2019.04.17

[Aim18b] Aimeos documentation: „Developers/Controller/Import products from CSV“, 2018. https://aimeos.org/docs/Developers/Controller/Import_products_from_CSV Abruf: 2019.04.04

[Aim18c] Aimeos documentation: „Developers/Library/Managing items“, 2018. https://aimeos.org/docs/Developers/Library/Managing_items Abruf: 2019.04.05

[apa19] apachefriends: „XAMPP Installers and Downloads for Apache Friends“, 2019. https://www.apachefriends.org/de/index.html Abruf: 2019.04.03

[Dan16] Daniel Becker: „E-Commerce Systeme - Was war, was ist, was kommt?“, 2016. https://www.netz98.de/blog/ecommerce-trends/entwicklung-der-e-commerce-systeme-the-age-of-frameworks/ Abruf: 2019.04.09

[eCo] eCommerce Leitfaden: „E-Commerce-Frameworks“. https://www.ecommerce-leitfaden.de/studien/item/e-commerce-frameworks Abruf: 2019.04.17

[sho] shopanbieter.de:: *E-Commerce Frameworks und E-Commerce As A Service (EAAS)*

[Spe] SpeakerDeck: „High performance e-commerce in Laravel“. https://speakerdeck.com/aimeos/high-performance-e-commerce-in-laravel?slide=13 Abruf: 2019.04.10

[t3n18a] t3n – digital pioneers: „Shoptech 2.0: E-Commerce-Systeme der nächsten Generation“, 2018. https://t3n.de/magazin/shoptech-20-e-commerce-systeme-naechsten-generation-246453/ Abruf: 2019.04.10

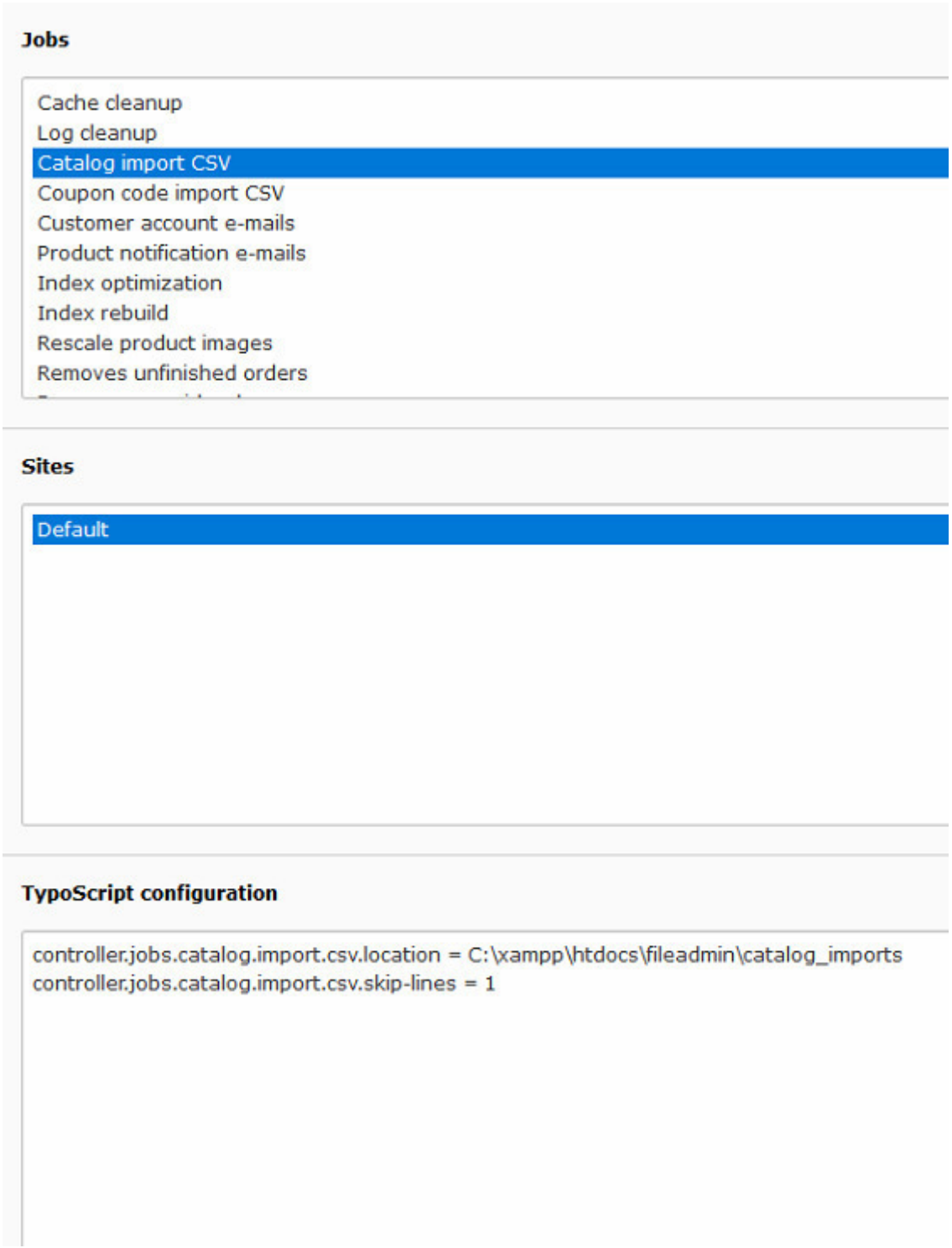
[t3n18b] t3nMagazin: „Content Commerce: 13 inspirierende Beispiele für Onlineshops“, 2018. https://t3n.de/news/content-commerce-beispiele-1105062/ Abruf: 2019.04.12

[typ] typo3: „The TYPO3 Project & Community“. https://typo3.org/ Abruf: 2019.04.03

[Typ] Typo3-websites: „TYPO3-Extension realurl für sprechende Adressen“. https://www.typo3-websites.eu/typo3-erweiterungen/optimierung/realurl/ Abruf: 2019.04.03

Anhangsverzeichnis

Anhänge



Anhang 1: Definition eines Cron-Jobs

```
controller.common.product.import.csv.mapping {
      item {
            0 = product.code
            8 = product.label
            2 = product.type
            12 = product.status
      }
      text {
            73 = text.type
            9 = text.content
            74 = text.type
            10 = text.content
      }
      media {
            20 = media.label
            19 = media.url
            69 = media.label
            68 = media.url
      }
      price {
            71 = price.currencyid
            15 = price.value
            71 = price.currencyid
            16 = price.value
      }
      attribute{
            26 = attribute.code
            25 = attribute.type
            72 = product.lists.type
            28 = attribute.code
            27 = attribute.type
```

```
            72 = product.lists.type
            30 = attribute.code
            29 = attribute.type
            72 = product.lists.type
            32 = attribute.code
            31 = attribute.type
            72 = product.lists.type
            34 = attribute.code
            33 = attribute.type
            72 = product.lists.type
            36 = attribute.code
            35 = attribute.type
            72 = product.lists.type
            38 = attribute.code
            37 = attribute.type
            72 = product.lists.type
            40 = attribute.code
            39 = attribute.type
            72 = product.lists.type
            42 = attribute.code
            41 = attribute.type
            72 = product.lists.type
            44 = attribute.code
            43 = attribute.type
            72 = product.lists.type
      }
      product{
            70 = product.code
      }
      property{
            11 = product.property.value
```

```
            76 = product.property.type
      }
      catalog{
            3 = catalog.code
            77 = catalog.lists.type
            4 = catalog.code
            77 = catalog.lists.type
            5 = catalog.code
            77 = catalog.lists.type
            6 = catalog.code
            77 = catalog.lists.type
      }
      stock{
            45 = stock.stocklevel
      }
```

Anhang 2: TypoScript Mapping Konfiguration

Use Case	Priorität	Use Case überprüft	Use Case Erfüllungsgrad (Schulnoten 1-6)	Kurzbeschreibung
Kundendaten importieren	C	ja	3	nicht direkt möglich - neuen Job Controller oder Option für UI erstellen -- Entwicklungsaufwand nötig
3 für 2	B	ja	6	ist nicht vorhanden
Kundendaten exportieren	B	ja	3	nicht direkt möglich - neuen Job Controller oder Option für UI erstellen -- Entwicklungsaufwand nötig
Live-Abfrage Bestand	B	ja	2	über Export und aimeos direkt - Bestand wird in aimeos aktualisiert - abfrage kann über export erfolgen - ERP-System zu Artikelbestand anfragen->über Erstellen von Service Providern möglich
Bestellstatus importieren	B	ja	2	nicht direkt über Aimeos - Bestellstatus im UI anpassbar - Anpassung soll durch Logistiksystem oder Fullfilment Partner erfolgen - eigener Job Controller für Import implementierbar - Webservice
Bezahlstatus	B	ja	2	Job-Controller - über Abfragen, Bestellungen werden bei Nichtbezahlung

				storniert - Zahlungen können im Backend des PSPs erstattet werden -> PSP benachrichtigt in der Regel Shop -> kann dann E-Mail an Kunden auslösen
E-Mail Bestätigung an Kunden (z.B. Danke-Email, andere Kunden kauften auch)	B	ja	1	Cronjob - kann über Scheduler für bestimmt Zeiträume eingestellt werden - in Tabelle Datensatz "E-Mail Versand"
Veränderung von Aimeos-Plugins (z.B. Checkout-layout oder Produktdetail)	B	ja	1	Templateanpassung - bei selber Directory Struktur überschreiben eigene Templates aimeoss Templates -- gut machbar -- muss aber auf richtige Struktur und Benennung achten
Bestellung exportieren	A	ja	3	CSV-Export - Default Export mit Scheduler -> manuell interagieren -- einfach - Job Controller anpassen XML-Export und senden an Remote-HTTP-Gateway - Job der Bestellungen an ERP-System senden kann - hier auch Erweiterungen von Scheduler und Provider möglich
Prozent Rabatt	A	ja	4	Basket-Plugin - Entwickler-Aufwand Rabatte anhand von preisen erstellen - Preise in Aimeos erstellen -> Darstellung kann anhand von Datum oder Nutzergruppe

				erfolgen -> somit wird Preis als Rabatt gehändelt - Preise können als Typ "Rabatt" haben Kopfrabatte und Positionsrabatte?
Gutschein	A	ja	3	E-Mail Job an Kunden in aimeos unter Coupon List - fügt Warenkorb bei Codeeingabe ein Rabattprodukt mit verbleibenden Gutscheinbetrag hinzu -- gibt einiges an Einstellungen zu beachten, was auch nicht im Nutzerhandbuch so genau steht -- in der Theorie einfacher als in der Praxis
Produkt importieren	A	ja	2	Produktimport von CSV - mit Beispieldatei und Magentodaten getestet -- für neue Produkte einfach -- für Migration Produkte aus Magento viele Anpassungen nötig (sowohl Typoscript als auch CSV) - via cronjob scheduler, über Typoscript konfigurierbar (directory, mapping, backups)
Bestand importieren	A	ja	2	CSV-Import -- durch Anpassung des Mappings einfach

Promotions anlegen	A	ja	2	im Katalog- oder Produktpanel - Fixrabatte an sich gibt es allerdings nicht s.u. - mit Templates usw. Darstellung bearbeitbar -- einfach - Erweiterungen um mehrere z.B. Promotions-Sektionen möglich?
Produktkachel auf Contentseite	A	nein	6	Seitenanpassung - Komponenten sollten einfach angepasst und hinzugefügt werden können -- das ganze funktioniert auf jeden Fall für Laravel, für TYPO3 hab ich das allerdings erstmal nicht gefunden
Produktverlinkung	A	nein	5	Produktdetails aimeos - Target Links für Produkte - Parameternamen können in TYPO3 bearbeitet werden
Teaser im Warenkorb/ Andere Kunden kauften auch	A	ja	1	Anpassung HTML + Typoscript - Basket Templates bearbeitet werden und Konfigurationen/ Parameter in Typoscript angepasst -- gut machbar
PSP (z.B. PayPal)	A	ja	2,5	über aimeos direkt - Payment Provider werden über aimeos bereitgestellt
Formulare	A	ja	1	Templateanpassung - z.B. CSS, HTML, PHP, Jquery -- gut machbar

Pflichtfelder	A	ja	2	Templateanpassung - z.B. Formular in Checkout in Checkout-Template bearbeitbar - z.B. auch configurations um zu verhindern das Nutzer neue Liefer und Zahladressen eingeben können -- gut machbar
Validierung anpassen	A	ja	2	Templateanpassung - z.B. Formular in Checkout in Checkout-Template bearbeitbar -- gut machbar

Anhang 3: Use Cases Aimeos