HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr
Jan Künzel

Analyse und Optimierung des Produktverwaltungsprozesses und der Bestellverarbeitung eines Magento-Onlineshops sowie dessen verbesserte Integration in die bestehende IT-Landschaft eines mittelständischen Unternehmens.

2017

Fakultät **Angewandte Computer- und Biowissenschaften**

BACHELORARBEIT

Analyse und Optimierung des Produktverwaltungsprozesses und der Bestellverarbeitung eines Magento-Onlineshops sowie dessen verbesserte Integration in die bestehende IT-Landschaft eines mittelständischen Unternehmens.

Autor:
Jan Künzel

Studiengang:
Angewandte Informatik, Wirtschaftsinformatik

Seminargruppe:
IF13wW-B

Erstprüfer:
Prof. Dr. Dirk Pawlaszczyk

Zweitprüfer:
Dipl.-Medieninformatiker René Iwan

Mittweida, 2017

Bibliografische Angaben

Künzel, Jan: Analyse und Optimierung des Produktverwaltungsprozesses und der Bestellverarbeitung eines Magento-Onlineshops sowie dessen verbesserte Integration in die bestehende IT-Landschaft eines mittelständischen Unternehmens, 85 Seiten, 31 Abbildungen, 4 Tabellen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Bachelorarbeit, 2017

Referat

Die vorliegende Arbeit befasst sich mit einer Verbesserung des Produktverwaltungsprozesses eines Magento Onlineshops sowie einer verbesserten Bestellverarbeitung und dessen Integration zu Unternehmenssystemen. Mögliche Optimierungsansätze werden anhand der Systemlandschaft des Unternehmens XECURIS GmbH & Co. KG aufgezeigt. Unternehmen sind darum bemüht, Workflows möglichst effektiv und mit geringem Zeitaufwand zu gestalten. Insbesondere Unternehmen im IT-Umfeld sind darauf angewiesen, dass komplexe Geschäftsabläufe automatisiert und fehlerfrei abgearbeitet werden. Ziel der Arbeit ist es, Schwächen an der bestehenden Infrastruktur des Unternehmens zu analysieren, Optimierungsansätze zu entwickeln sowie Ansätze zu deren Implementation aufzuzeigen.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AJAX ..... Asynchronous JavaScript and XML, Seite 17
BPMN ..... Business Process Model and Notation, Seite 5
CRUD-Funktion ... Create-Read-Update-Delete-Funktion, Seite 14
CSS ..... Cascading Style Sheets, Seite 51
DOM ..... Document Object Model, Seite 16
EBC ..... EGIS Business Connector, Seite 43
EK ..... Einkaufspreis, Seite 24
HTML ..... Hypertext Markup Language, Seite 11
JSON ..... JavaScript Object Notation, Seite 12
LINQ ..... Language Integrated Query, Seite 14
PHP ..... Hypertext Preprocessor, Seite 19
SCP ..... Secure Copy, Seite 52
SEO ..... Search Engine Optimization, Seite 20
SQL ..... Structured Query Language, Seite 14
URL ..... Uniform Resource Locator, Seite 43
VK ..... Verkaufspreis, Seite 24
XML ..... Extensible Markup Language, Seite 11

1 Einleitung

Die vorliegende Bachelorarbeit untersucht in Kooperation mit der XECURIS GmbH & Co. KG[1] mögliche Ansätze zur Optimierung des Workflows in dem unternehmenseigenen Magento-Onlineshop. Die Arbeit knüpft an ein vorangegangenes 13-wöchiges Praktikum im fünften Fachsemester an, welches im Bereich der Softwareentwicklung absolviert wurde. In diesem Zeitraum ist ein eigenständiges webbasiertes Softwaresystem entstanden, das bereits Grundfunktionalitäten für eine verbesserte Verwaltung der angebotenen Produkte im Onlineshop realisiert. Die Arbeitsschritte im Unternehmen gelten als zeitintensiv, gering automatisiert, redundant sowie fehleranfällig und stellen sich in ihrer Durchführung als komplex heraus.

1.1 Motivation

Im Zeitalter des 21. Jahrhunderts sehen sich besonders Unternehmen im IT-Geschäft zunehmend einer dynamischen und schnelllebigen Marktentwicklung ausgesetzt. Infolge globaler Vernetzung wird es notwendig, einen stetigen Überblick über das aktuelle Marktangebot zu behalten. XECURIS ist der Betreiber eines Onlineshops für PC-Systeme im Finanzbereich. Die darin angebotenen Produkte zeichnen sich durch eine hohe individuelle Anpassbarkeit aus. In der detaillierten Produktkonfiguration werden Einzelkomponenten ausgewählt, welche letztendlich im Gesamtprodukt verbaut werden. Die Einkaufspreise und Bestände dieser Artikel bei den Distributoren können innerhalb von Minuten stark variieren. Ein stetiger Konkurrenzdruck erfordert es, einen Verkaufspreis anzubieten, welchen der Kunde bereit ist zu zahlen. Gleichzeitig ist es erforderlich, eine Mindestmarge für jeden Artikel einzuhalten, sodass das Unternehmen bei jeder Bestellung einen vertretbaren Gewinn erzielt. Ein System, welches in diesem Bereich genutzt wird, muss demnach nicht nur mit stetig aktualisierten Einkaufspreisen arbeiten, sondern ebenfalls die festgelegten Produktmargen dynamisch und hoch automatisiert auf die Produkte abbilden. Gleichzeitig muss das Ergebnis dieser Kalkulation innerhalb kurzer Zeit in den Onlineshop übertragbar sein.

Auf dem Markt existiert eine Vielzahl an Standardsoftware, welche die beschriebenen Funktionalitäten implementiert. Sowohl geringere Kosten zur Administration, die problemlose und überschneidungsfreie Einbindung bestehender Geschäftsprozesse in den Unternehmensablauf als auch ein hoher Funktionsumfang sind Gründe, dass Individualentwicklungen zunehmend abgelöst werden.[2] Die nutzungsgerechte Anpassung betrieblicher Anwendungssoftware ist dennoch nicht in jedem Fall durch Konfigurations-

[1] Das Unternehmen XECURIS GmbH & Co. KG wird nachfolgend verkürzt unter dem Namen XECURIS genannt.

[2] vgl. [EHI Retail Institute 2013]

einstellungen oder Customizing möglich. Aus Sicht von XECURIS konnte keine Standardsoftware identifiziert werden, welche die individuellen Anforderungen des Unternehmens zureichend implementiert. Nach dem derzeitigen Stand kommt ein komplexer Konfigurator zur Verwaltung der angebotenen Produkte zum Einsatz. Die schlechte Bedienbarkeit sowie eine mangelnde Integration und Interoperabilität zwischen den im Unternehmen eingesetzten Anwendungen sind ausschlaggebend für die Entwicklung eines eigenen Softwaresystems. Das entstehende Softwaresystem soll als zentrale Schnittstelle für eine vereinfachte, automatisierte und weniger fehleranfällige Nutzung des Onlineshops dienen. Gleichzeitig wird angestrebt, durch die Synchronisation zu den eingesetzten Teilsystemen der Unternehmensinfrastruktur eine bessere Administration des unternehmenseigenen Produktangebotes zu erreichen. Die Integration dieser Systeme soll dazu beitragen, systemübergreifende Arbeitsabläufe zu schaffen und die Datenhaltungen der eingesetzten Teilsysteme zu steuern.

1.2 Zielstellung und Nutzen

Sowohl fachliche als auch technische Grundlagen stehen zunächst im Fokus der Arbeit. Wichtige Begriffsklärungen legen ein einheitliches Begriffsverständnis für die nachfolgenden Kapitel. Anschließend wird anhand der Ausgangslage ein Einblick in die Problematik im Umgang mit dem Onlineshop gegeben, dessen Optimierung den Gegenstand der Untersuchung darstellt. Hierbei liegt ein besonderes Augenmerk auf der Infrastruktur des Unternehmens. In Verbindung mit bestehenden Vor- sowie Nachteilen werden nun Konzepte zur Verbesserung des Produktverwaltungs- und Bestellverarbeitungsprozesses gefunden. Daraufhin fokussiert das Kapitel *Methoden* die gefundenen Optimierungsansätze und erläutert Teile der Implementation bzw. Umsetzung der Teilabschnitte. Das Ziel der Arbeit ist es, ein Verständnis für die Problematik im Einsatz von nicht integrierter betrieblicher Standardsoftware näher zu bringen. In diesem Zusammenhang werden Lösungsansätze aufgezeigt, welche die Optimierung sowie Integration eines Softwaresystems als Individualsoftware hin zu einem Gesamtsystem ermöglichen. Hierbei soll ein einsatzfähiges Softwaresystem entstehen, in welchem die vollständige Produktverwaltung für den eingesetzten Onlineshop ablaufen kann. Neben der Schaffung systemübergreifender Arbeitsabläufe sowie der Bereitstellung der erforderlichen Funktionalität zur Administration der angebotenen Produkte steht weiterhin die effiziente Bedienbarkeit des Systems im Vordergrund.

1.3 Unternehmensumfeld

Bei XECURIS handelt es sich um ein mittelständiges Unternehmen, welches in Tronitz bei Döbeln ansässig ist. Die Firma wurde im Jahr 2000 unter dem Namen L&H Consulting GbR gegründet und war zu diesem Zeitpunkt noch als reines Beratungsunternehmen für Wertpapiere tätig. Das mit den Geschäftsführern Sebastian Lohr und Thomas Hoyer inhabergeführte Unternehmen wuchs in den darauffolgenden Jahren um weitere Bereiche. Als richtungsweisend stellte sich hierbei der Aufbau eines Bereiches zur Herstellung von Multimonitorworkstations und Monitorträgersystemen heraus. Ebenso hat sich eine eigenständige Abteilung mit dem Fokus in der Softwareentwicklung etabliert. 2013 ging das Unternehmen im Zuge einer Umfirmierung vollständig in die Firma XECURIS über.[3]

Mit insgesamt 13 Mitarbeitern im Jahr 2016 vertreibt XECURIS ihre Produkte unter drei verschiedenen Marken: Trading-PC, vAppX und sedergo. Über die Marke Trading-PC werden die beworbenen Rechner- und Monitorsysteme im unternehmenseigenen Onlineshop angeboten (siehe Abbildung 1.1). Dieser ist nach einem Jahr Entwicklung im März 2016 veröffentlicht worden. In dem Shop erhält der Kunde die Möglichkeit, ein Produkt individuell nach seinen Wünschen zusammenzustellen.[4]

Abbildung 1.1: Der XECURIS Onlineshop

Über vAppX hingegen werden mit den angebotenen Rechenzentren-Kapazitäten verschiedene Hosting- und Cloud-Computing-Lösungen bereitgestellt. Durch dieses Angebot sollen Arbeitsplätze mobil gestaltbar und nach beliebiger Größe skalierbar sein.

[3] vgl. [XECURIS GmbH & Co. KG 2016a,b,c,d]
[4] vgl. [XECURIS GmbH & Co. KG 2016c]

Sedergo umfasst als dritte Marke die Herstellung sowohl individueller als auch hochwertiger Büromöbel. Das Angebot von XECURIS ist gezielt auf Geschäftszweige wie Banken, Versicherungen, Fondsgesellschaften, Energie-Handelsunternehmen sowie Vermögensverwaltungen abgestimmt.

2 Grundlagen

Dieses Kapitel bringt notwendige Begriffe näher, welche für das grundlegende Verständnis der Arbeit wichtig sind. Hierbei stehen sowohl fachliche als auch technische Grundlagen im Mittelpunkt der Untersuchung.

2.1 Fachliche Grundlagen

Dieser Teil ist den fachlichen Grundlagen der Arbeit gewidmet. Begriffsdefinitionen und Kategorisierungen sind Inhalt des Abschnitts, um grundlegende Begriffe näher zu bringen.

2.1.1 Workflow

Ein Workflow bezeichnet einen rechnergestützten Geschäftsprozess, d.h. Workflows umfassen alle elektronischen Abläufe, welche die Erarbeitung des Geschäftsresultates unterstützen und von wiederkehrender Natur sind. Workflows können Gegenstand einer Integration oder Optimierungsvorgängen sein. In diesem Fall sollen Geschäftsabläufe möglichst geringe Latenzen aufweisen. Um dieses Ziel zu erreichen, müssen redundante Arbeitsabläufe vermieden werden.

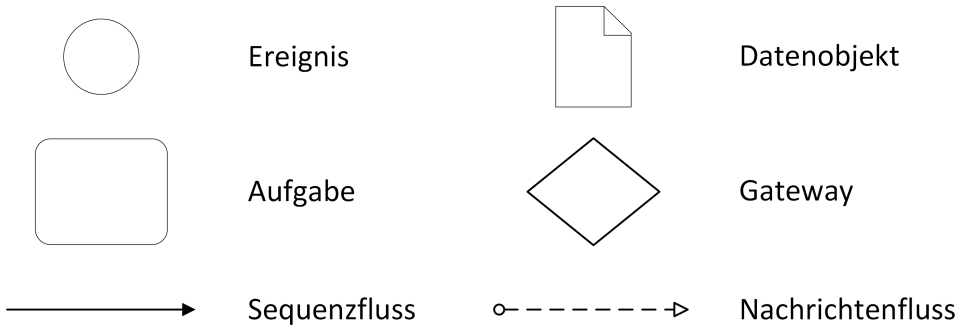

Abbildung 2.1: BPMN Notationselemente

Die Logik hinter Prozessen kann komplex sein. Um einen Prozess mithilfe von strukturierten Elementen zu beschreiben, haben sich Notationen entwickelt. Diese legen einheitlich Gestaltungselemente sowie deren Bedeutung bzw. Nutzung fest. Eine weit verbreitete Spezifikationssprache ist die sogenannte *Business Process Model and Notation* (BPMN). Diese wurde 2001 durch die *Business Process Management Initiative* erarbeitet und stellt verschiedene Notationselemente zur Verfügung (siehe Abbildung 2.1).

Jeder modellierte Prozess beginnt jeweils mit einem Startereignis und endet mit einem Endereignis. Teilabläufe lassen sich mithilfe von sogenannten Aufgaben modellieren. Sowohl Parallelisierungen in Form von logischen AND-Verknüpfungen als auch Fallunterscheidungen wie logische OR- und XOR-Verknüpfungen können durch Gateways definiert werden. Die Reihenfolge der Abarbeitung wird durch sogenannte Sequenzflüsse festgelegt. Optional lassen sich Datenobjekte in das Modellierungsschema einbinden. Die Zuordnung erfolgt über Kommunikationsbeziehungen. Derartige Strukturen werden eingesetzt, um die Verständlichkeit für den Prozessablauf zu erhöhen. Die Verwendung der Modellierungselemente wird in Abbildung 2.2 anhand eines Beispieles gezeigt. Informationen zu weiteren Gestaltungselementen sowie Konventionen zur Nutzung können der nachfolgenden Quelle entnommen werden.[5]

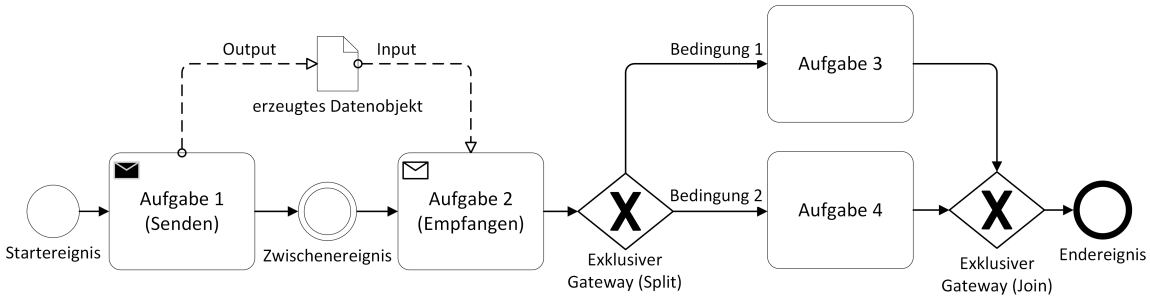

Abbildung 2.2: BPMN Beispiel

BPMNs finden bei der Modellierung von Geschäftsprozessen bis hin zu IT-Anforderungsplanungen Anwendung. Die Sprache bietet Elemente zur Gestaltung und Unterstützung des Verständnisses von Prozessabläufen. Sie wird innerhalb dieser Arbeit sowohl zur Beschreibung von Geschäftsprozessen als auch zur Erläuterung der Funktionsweise von Programmcode eingesetzt.[6]

Die Qualität eines Prozesses lässt sich anhand von zwei Faktoren beurteilen: Effektivität und Effizienz. Ersteres stellt Kosten und Nutzen eines Prozesses aus der internen Sicht des Unternehmens gegenüber. Grundlage einer hohen Effektivität ist eine geringe Latenz zwischen den Vorgänger- und Nachfolgeprozessen, ein geringer Ressourceneinsatz sowie eine verringerte Redundanz der Arbeitsschritte. Um eine Optimierung bezüglich der Effektivität durchzuführen, bedarf es häufig einer Neustrukturierung des Prozessablaufes. Effizienz hingegen betrachtet die Qualität des Prozesses aus Kundensicht. Ein Prozess gilt dann als effizient, wenn das gewünschte Geschäftsresultat in einer gewünschten Zeitspanne eingetreten ist.[7]

[5] vgl. [Object Management Group Inc. 2016]
[6] vgl. [MID GmbH, S. 1 ff.]
[7] vgl. [Schneider et al. 2014, S. 67]

2.1.2 Softwaresystem

Dieser Abschnitt gibt einen Überblick über den Begriff Softwaresystem. Neben Zusammenhängen zu verwandten Begriffen wie Software und Informationssystemen werden auch mögliche Klassifizierungen des Softwarebegriffes in den Vordergrund gestellt (siehe Abbildung 2.3).

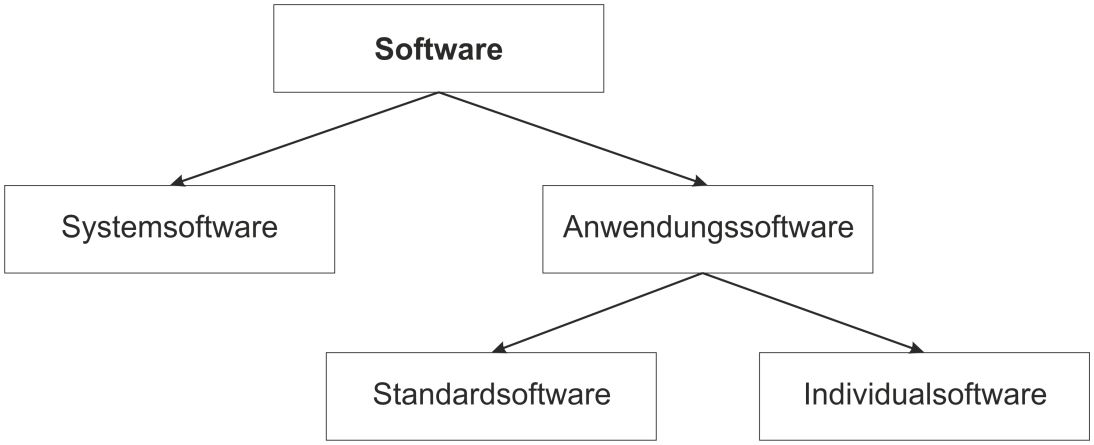

Abbildung 2.3: Klassifizierung von Software[8]

Begriffsklärung

Der Begriff Software beinhaltet diejenigen Komponenten eines Rechnersystems, welche immaterieller Natur sind. Hierzu zählen neben dem Programmcode zur zielgerichteten Steuerung und Verarbeitung der Eingabedaten auch organisatorische Richtlinien, Verfahrensregeln sowie eine Dokumentation des vorliegenden Programms. Auf Grundlage dieser Einordnung wird eine Software, die sich aus mehreren modularen Bestandteilen zusammensetzt, als Softwaresystem bezeichnet. Ein Informationssystem erweitert den Begriff des Softwaresystems um zusätzliche Kommunikationsbeziehungen. Der Schwerpunkt liegt auf dem Austausch von Informationen zwischen Systemanwender und der Maschine, auch soziotechnisches System genannt.[9,10]

Integration

Der Begriff *Integration* entstammt dem lateinischen Wort *integrare* und bedeutet wörtlich übersetzt *wiederherstellen* bzw. *zusammensetzen* und bezeichnet den Zusammenschluss logisch zusammengehöriger Teile. Eine Integration wird je nach Anwendungsgebiet auf verschiedene Sachverhalte angewandt und ist somit vom Kontext abhängig.[11]

[8] vgl. [Mertens et al. 2012, S. 18]
[9] vgl. [Brich u. Hasenbalg 2013, S. 174]
[10] vgl. [Kohnke 2015, S. 35 f.]
[11] vgl. [Mertens et al. 2012, S. 9]

Im Bereich Informationssysteme wird der Integrationsbegriff zusätzlich in drei Ansätze gegliedert, welche nach dem Integrationsgegenstand unterschieden werden: Die Funktionsintegration, Datenintegration und Prozessintegration. Das Prinzip einer *Funktionsintegration* besteht darin, mehrere Teile eines Systems auf identische Funktionen zugreifen zu lassen, sodass zum einen kein redundanter Programmcode entsteht und zum anderen kein Mehraufwand zur Implementierung oder Erweiterung bereits vorhandener Funktionalität auftritt. *Datenintegration* hingegen zielt darauf ab, dass alle Teilsysteme auf eine gemeinsame, konsistente Datenbasis zugreifen, beispielsweise durch die Nutzung einer relationalen Datenbank. *Prozessintegration* bezeichnet schließlich die Schaffung einer zentralen Schnittstelle, von welcher aus alle Funktionalitäten anderer Systeme genutzt und nahtlos miteinander verbunden werden. Diese Schnittstelle wird auch als Integrationsplattform bezeichnet.[12]

Klassifizierung

Software lässt sich durch ihre Nähe zur Hardware in zwei Teilbereiche klassifizieren: Die System- und Anwendungssoftware (siehe Abbildung 2.3). Beide Begriffe unterscheiden sich dahingehend voneinander, dass Systemsoftware eine nähere Kommunikation mit den zugrunde liegenden Hardwarekomponenten erlaubt. Unter dieser Bezeichnung werden alle Programme zusammengefasst, welche zum Betrieb des Rechnersystems notwendig sind und demnach die Hardware nutzbar machen. Neben dem Betriebssystem werden auch Treiber und Systemprotokolle diesem Punkt zugeordnet.[13,14]

Anwendungssoftware hingegen besitzt im Vergleich zu Systemsoftware eine geringere Nähe zur Hardware. Der Vorteil besteht darin, dass diese Software unabhängiger vom zugrundeliegenden Hardwarekomponenten nutzbar sind und somit ohne große Anpassungen vielseitiger auf Systemen eingesetzt werden kann.[15]

Bei Anwendungssoftware kann wiederum eine Unterscheidung zwischen Standardsoftware und Individualsoftware getroffen werden. Standardsoftware wird für einen breiten anonymen Markt entwickelt und setzt standardisierte Workflows um. Beispielsweise kann ein Textverarbeitungsprogramm als Standardsoftware betrachtet werden, da dieses unabhängig von der eingesetzten Branche identische Funktionalitäten beinhaltet. Individualsoftware umfasst unternehmensinterne Eigenentwicklungen. Diese sind dann notwendig, wenn ein Unternehmen Anforderungen an eine Anwendung stellt, diese aber von keiner Standardsoftware zufriedenstellend umgesetzt werden können, zu viele Anpassungen durch Konfiguration bzw. Customizing nötig sind oder die Software für den Betrieb finanziell nicht erschwinglich ist.[16]

[12] vgl. [Mertens et al. 2012, S. 9]
[13] vgl. [Mertens et al. 2012, S. 17 ff.]
[14] vgl. [Kohnke 2015, S. 35 ff.]
[15] Nachfolgend wird unter dem Begriff Software bzw. Softwaresystem der Bereich Anwendungssoftware verstanden.
[16] vgl. [Mertens et al. 2017, S. 16 ff.]

Aufgrund des umfangreichen Angebotes an Standardsoftware auf dem Markt gerät Individualsoftware zunehmend in den Hintergrund. Wie eine Statistik aus dem Jahr 2013 zeigt, nutzen 70 Prozent aller Unternehmen im deutschsprachigen Handel Warenwirtschaftssysteme, welche nicht auf einer Eigenentwicklung basieren. Weitere 7 Prozent der Unternehmen mit Individualsoftware gaben an, zukünftig eine Standardsoftware zu verwenden (siehe Abbildung 2.4).[17]

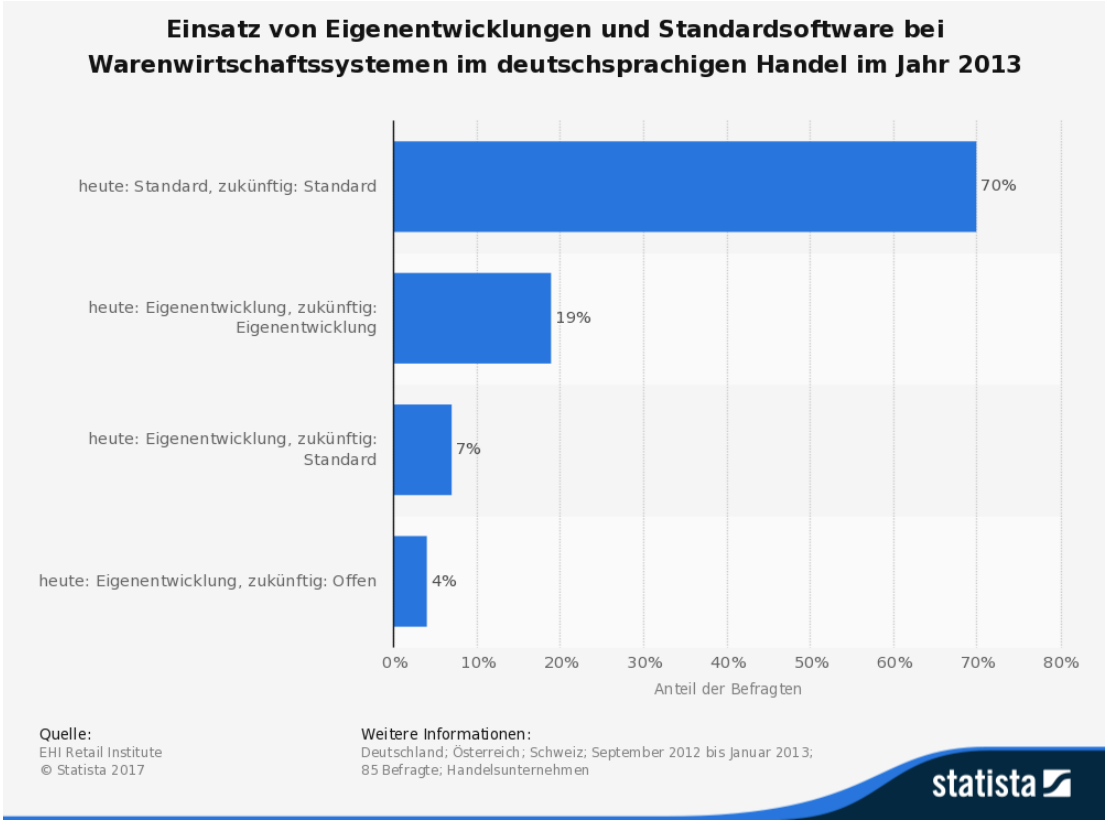

Abbildung 2.4: Einsatz von Eigenentwicklungen im Jahr 2013[18]

Als Sonderform der Individualsoftware wird die sogenannte Middleware gesehen. Diese bildet eine zentrale Anwendung zwischen verschiedenen eingesetzten Softwaresystemen und verbindet diese über Schnittstellen in Form einer Integrationsplattform. Hierbei können sowohl Eigenentwicklungen als auch Standardsoftware mit Middleware verbunden werden. Die Umsetzung einer derartigen Lösung bringt verschiedene Vorteile mit sich. Zum einen muss nicht auf bereitgestellte Funktionalitäten der einzelnen Teilsysteme verzichtet werden, da die Selbständigkeit der eingebundenen Teilanwendungen erhalten bleibt. Weiterhin ist es auf diese Weise möglich, eine Benutzerschnittstelle zu schaffen, um das Gesamtsystem über ein zentrales Portal zu bedienen. Da Individualsoftware unternehmensintern entwickelt wird, sind sowohl Benutzeroberfläche als auch Workflows individuell gestaltbar.[19]

[17] vgl. [EHI Retail Institute 2013]
[18] [EHI Retail Institute 2013]
[19] vgl. [Tresch 1996, S. 253 ff.]

2.1.3 Onlineshop

Der Begriff Onlineshop wird in den Bereich des E-Commerce eingeordnet. E-Commerce, auch Elektronischer Handel bzw. Online-Handel genannt, bezeichnet alle Ein- und Verkaufsvorgänge, welche mithilfe des Internets geschehen. Durch Datenübertragung wird eine unmittelbare Geschäftsbeziehung zwischen dem Anbieter und dem Konsumenten erreicht. Aufbauend auf dieser Einordnung wird ein Onlineshop als Teilbereich des Elektronischen Handels verstanden. Der Anbieter stellt über das Internet, in Form einer Webseite, gewerbliche Waren oder Dienstleistungen zum Kauf oder zur Miete bereit. Anbieter im Elektronischen Handel besitzen aufgrund des freien und internetweiten Zugriffs eine höhere Reichweite als der Einzelhandel, sehen sich aber aufgrund steigender Onlineshops einer zunehmenden Konkurrenz ausgesetzt. Ein Onlineshop dient nicht nur dem Verkauf von Waren, sondern auch der Produktpräsentation.[20]

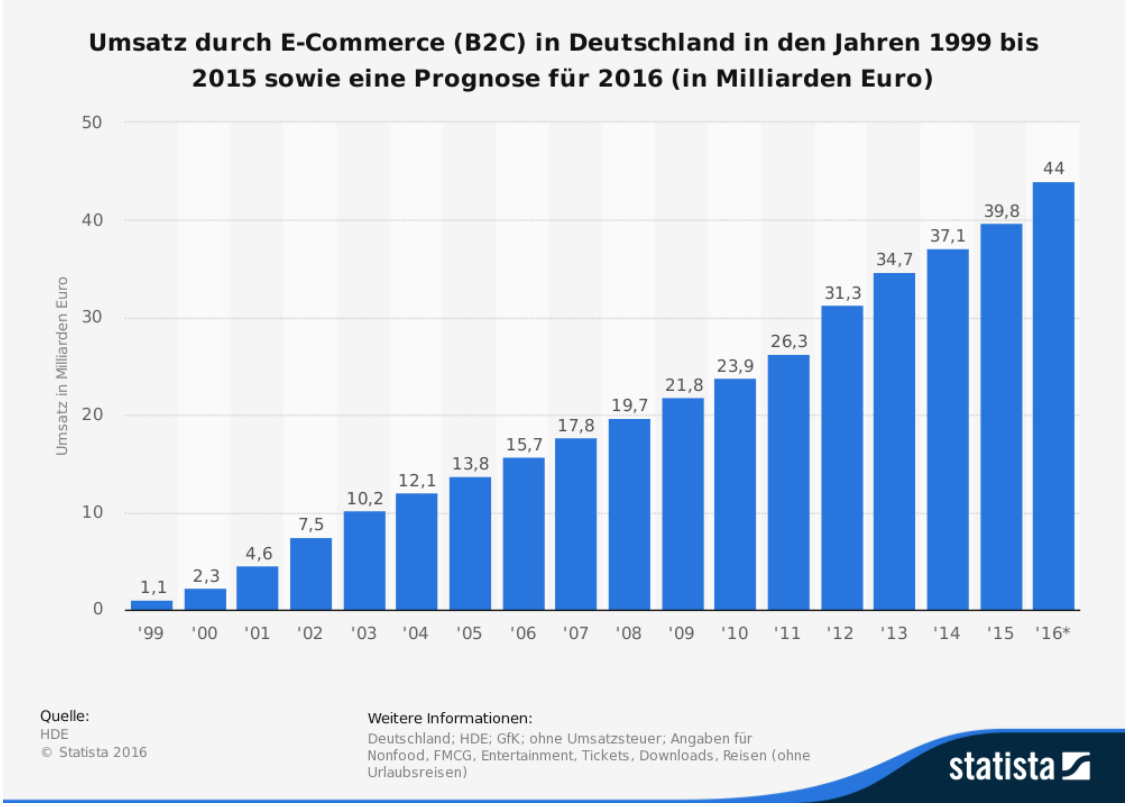

Abbildung 2.5: Umsatz durch E-Commerce zwischen 1999 und 2016 in Deutschland[21]

Wie aus einer Umfrage des Handelsverbands Deutschland hervorgeht (siehe Abbildung 2.5), wurden 1999 1,1 Milliarden Euro im Bereich des E-Commerce erwirtschaftet. Im Jahr 2015 waren es bereits 39,8 Milliarden Euro.[22] Der Umsatz ist in diesem Zeitraum um mehr als das 36-fache gestiegen. Prognosen für Entwicklungen im Jahr 2016 schät-

[20] vgl. [Sjurts 2011, S. 139]
[21] [Handelsverband Deutschland 2016]
[22] vgl. [Handelsverband Deutschland 2016]

zen einen weiteren Anstieg der Umsätze. Der Online-Handel ist somit ein Sektor, welcher weiterhin einen steigenden Trend verzeichnet und zunehmend an Bedeutung gewinnt. Wie eine Umfrage der RegioPlan Consulting GmbH aus dem Jahr 2012 zeigt, wird im Bereich der Elektronikbranche 78 Prozent des Gesamtumsatzes über den Onlinehandel erreicht.[23]

Im Bereich des E-Commerce hat sich eine Reihe von Softwaresystemen, sogenannten Shopsystemen, entwickelt, welche es Händlern erleichtert, einen Onlineshop aufzubauen, diesen nach eigenen Bedürfnissen zu individualisieren und gleichzeitig eine hohe Standardfunktionalität anzubieten. Der Administrator des Shopsystems kann häufig aus einer Vielzahl von Funktionen wählen, beispielsweise verschiedene Farbdesigns, Layouts und Sprachen sowie die Konfiguration eigener Produkte. Aufgrund steigender Konkurrenz müssen Händler bzw. Hersteller eine hohe Aufmerksamkeit in die Gestaltung sowie die Nutzerfreundlichkeit ihrer Webseite investieren und gewonnenes Nutzerfeedback sinnvoll einsetzen, um die Absprungrate potenzieller Kunden gering zu halten. Insbesondere für Bestandskunden als wiederkehrende Käufer muss das Onlineangebot des Händlers langfristig ansprechend erscheinen.[24]

2.2 Technische Grundlagen

In den technischen Grundlagen werden anwendungsbezogene Aspekte der Arbeit näher beleuchtet. In diesem Abschnitt stehen Technologien im Fokus, die für den Datenaustausch an Schnittstellen benötigt werden. Auch Technologien, welche im aufgebauten Softwaresystem sowohl für serverseitige als auch clientseitige Strukturen Anwendung finden, bilden den Inhalt des Abschnitts.

2.2.1 Datenaustausch an Schnittstellen

Um einen Austausch zwischen Systemen ohne Datenintegration durchzuführen, müssen erforderliche Daten in eine passende, interpretierbare Struktur überführt werden. Nachfolgend werden XML und JSON als häufig eingesetzte Technologien in diesem Bereich näher erläutert und anhand eines Beispiels verdeutlicht.

Extensible Markup Language (XML)

Extensible Markup Language, kurz XML, ist eine deskriptive Auszeichnungssprache. XML basiert auf der allgemeineren Sprache *Standard Generalized Markup Language* (SGML) und weist demnach eine ähnliche Syntax wie die Websprache *Hypertext Markup Language* (HTML) auf.[25]

[23] vgl. [a3 Wirtschaftsverlag Gesellschaft m.b.H. 2012]
[24] vgl. [Heinemann 2010]
[25] vgl. [Nurseitov et al. 2009]

Mit ihr ist es möglich, hierarchisch strukturierte Datenobjekte in Form einer Baumstruktur abzubilden und als Datei abzuspeichern. XML-Strukturen werden insbesondere an flexiblen Schnittstellen verwendet. Datenstrukturen lassen sich nicht nur auf die Richtigkeit ihrer Syntax (Wohlgeformtheit) überprüfen, sondern auch die inhaltliche Richtigkeit der Struktur kann durch ein XML-Schema als XSD-Datei festgestellt werden. Beispielsweise können so inkonsistente oder unvollständige Daten identifiziert werden.[26]

Listing 2.1 zeigt den Aufbau einer XML-Struktur anhand eines Beispiels:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Kunden>
  <Kunde>
    <vorname>Lea</vorname>
    <nachname>Lack</nachname>
    <geburt>22.01.1990</geburt>
  </Kunde>
  <Kunde>
    <vorname>Max</vorname>
    <nachname>Meier</nachname>
    <geburt>10.03.1973</geburt>
  </Kunde>
</Kunden>
```

Listing 2.1: beispielhafte XML-Struktur

JavaScript Object Notation (JSON)

Bei *JavaScript Object Notation*, kurz JSON, handelt es sich um ein Datenformat mit einer speziellen Syntax. Im Gegensatz zu XML ist JSON keine Auszeichnungssprache. Das Format wird unter anderem dazu verwendet, um den Austausch von Datenobjekten auch über Systemgrenzen hinweg zu realisieren. Auch hierarchisch strukturierte und komplexe Objekte lassen sich mithilfe von JSON darstellen. Im Vergleich zu XML können JSON-Datenformate nicht gegen ein Schema validiert werden. Erzeugte Strukturen sind weniger komplex als XML, auf eine schnelle Datenverarbeitung ausgelegt und sind nicht typisiert.[27]

Listing 2.2 zeigt den exemplarischen Aufbau einer JSON-Struktur:

```
{"Kunden":[
  {"vorname":"Lea", "nachname": "Lack", "geburt":
    "22.01.1990"},
  {"vorname":"Max", "nachname": "Meier", "geburt":
    "10.03.1973"}
]}
```

Listing 2.2: beispielhafte JSON-Struktur

[26] vgl. [Nurseitov et al. 2009]
[27] vgl. [W3Schools 2017]

JSON wird vordergründig an Schnittstellen verwendet, die Antworten in einer rigiden Form geben sollen. Sowohl XML- als auch JSON-Strukturen können mithilfe von Parsern eingelesen und in Softwaresystemen mittels Programmiersprachen verarbeitet werden.[28]

2.2.2 .NET-Technologien

Bei .NET handelt es sich um eine von Microsoft herausgegebene Standardumgebung für die Anwendungsentwicklung auf Windows-Systemen. .NET wurde im Jahr 2002 erstmalig veröffentlicht und stellt Funktionalitäten als Frameworks bereit, um die Entwicklung und Veröffentlichung von Anwendungssoftware zu unterstützen. Aufgrund ihrer umfassenden Struktur wird .NET auch als zentrale Softwareentwicklungsinfrastruktur bezeichnet.[29]

Entity Framework

Das Entity Framework schafft mithilfe von O/R-Mapping eine Abstraktionsschicht, welche den Zugriff auf eine zugrundeliegende Datenbank vereinfacht. Über objektrelationale Abbildungen werden in C# definierte Objekte in eine objektorientierte Datenbankstruktur überführt.[30]

```
1  DatabaseContext db = new DatabaseContext();
2
3  //Create-Funktion
4  Template createdTemplate = new Template();
5  template.Title = "Neue Produktvorlage";
6  db.Templates.Add(createdTemplate);
7  db.Entry(createdTemplate).State = EntityState.Added;
8  await db.SaveChangesAsync();
9
10 //Read-Funktion
11 Template template = db.Templates
12   .FirstOrDefault(template => createdTemplate.Id == id);
13
14 //Update-Funktion
15 template.Title += " (bearbeitet)";
16 db.Entry(template).State = EntityState.Modified;
17 await db.SaveChangesAsync();
18
19 //Delete-Funktion
20 db.Templates.Remove(template);
21 db.Entry(template).State = EntityState.Deleted;
22 await db.SaveChangesAsync();
```

Listing 2.3: Nutzung des Entity Frameworks

[28] vgl. [Afsari et al. 2017, S. 26]
[29] vgl. [Schwichtenberg 2016]
[30] vgl. [Microsoft Corporation 2016]

Das Framework wird dazu eingesetzt, um den benötigten Code für Datenzugriffe, beispielsweise auf eine Datenbank, zu reduzieren. Create-Read-Update-Delete-Funktionen (CRUD-Funktionen) lassen sich ohne SQL definieren (siehe Listing 2.3). Alle Datenabfragen geschehen innerhalb der Softwareumgebung über ein instanziiertes Database-Context-Objekt. Dieses ermöglicht den einfachen Zugriff auf Datenbanktabellen.[31]

Language Integrated Query (LINQ)

Innerhalb von C# in Verbindung mit .NET existiert eine Vielzahl an vordefinierten Datentypen. Um einen einheitlichen bekannten Satz von Methoden für den Zugriff auf Datenstrukturen zur Verfügung zu stellen, existieren sogenannte *Language Integrated Queries*, kurz LINQ. Die Zugriffe erfolgen hierbei über ein Provider-Model. LINQ ist auf beliebige Datenstrukturen anwendbar, für welche ein derartiges Model definiert ist. Über LINQ kann nicht nur der Zugriff auf Objekte wie Arrays und Listen erfolgen, sondern auch auf Datenbanken. Somit müssen für Zugriffe auf die zugrundeliegende Datenbank manuell keine SQL-Anweisungen mehr geschrieben werden. Benötigte Joins über mehrere Tabellen werden automatisiert durchgeführt. Datenbankabfragen bzw. Filter lassen sich äquivalent zu dem WHERE-Befehl in SQL-Answeisungen auf beliebige Datenstrukturen anwenden. Auch die Extraktion einzelner Attribute aus einer Liste ist über LINQ-Befehle möglich. Bei Datenstrukturen, für welche keine LINQ-Funktionalität implementiert wurde, existiert ebenfalls die Möglichkeit, eigene Provider-Models zu definieren und so auch hier die bereitgestellte Funktionalität zu nutzen.[32]

Listing 2.4 zeigt beispielhaft die Anwendung von LINQ-Methoden auf Grundlage von Arrays bzw. Listen:

```
1  //ungefilterte Vorlagenelemente
2  List<Template> tempList = GetTemplateList();
3  Template[] tempArray = GetTemplateArray();
4
5  //Filter auf Liste/Array durch dieselbe Methode
6  List<Template> tempFilterList = tempList
7    .Where(x => x.Id == 5).ToList();
8  Template[] tempFilterArray = tempArray
9    .Where(x => x.Id == 5).ToArray();
10
11 //dynamische Konvertierung zwischen Datentypen
12 tempArray = tempFilterList.ToArray();
13 tempList = tempFilterArray.ToList();
14
15 //Extraktion einzelner Properties
16 List<long> listIds = tempList.Select(x => x.Id).ToList();
17 List<long> arrIds = tempArray.Select(x => x.Id).ToList();
```

Listing 2.4: Beispiel zur Nutzung der LINQ-Funktionalität

[31] vgl. [Microsoft Corporation 2016]
[32] vgl. [Microsoft Corporation 2008]

Code First Migrations

Mithilfe von Code First Migrations lässt sich automatisiert eine objektrelationale Datenstruktur aufbauen bzw. eine bestehende Datenbankstruktur aktualisieren. Grundlage für die Erstellung der Datenstruktur bilden die Model-Klassen des Quellcodes. Bei einer Strukturänderung wird eine Migration der Datenbank durchgeführt, wobei bereits existierende Daten bestehen bleiben. Bei der stetigen Weiterentwicklung einer Anwendung sind Änderungen an der zugrundeliegenden Datenstruktur notwendig. Verschiedene Entwickler, die an einem System arbeiten, sowie der damit verbundene Einsatz mehrerer lokaler sowie entfernter Teilsysteme erfordern es, die Datenhaltung durch Ausführung identischer Datenbankoperationen zu synchronisieren. Das manuelle Einspielen der Änderungen zieht einen großen Zeitaufwand mit sich und stellt sich als fehleranfällig heraus. Um die fehlerfreie Ausführung der Migration zu gewährleisten, wird eine selbständige Klasse erstellt, welche sowohl eine Methode für die Durchführung der Änderungen als auch für dessen Umkehr enthält. Bei Ausführung der benötigten Methoden wird die Datenstruktur automatisch auf den gewünschten Stand gebracht. Hierbei wird kein SQL für die Data Definition Language (DDL) benötigt.[33]

ASP.NET MVC

ASP.NET MVC wird für die Erstellung von dynamischen webbasierten Softwaresystemen eingesetzt. Wie aus dem Namen bereits hervorgeht, wird hierbei das modulare Model-View-Controller (MVC) Entwurfsmuster umgesetzt (siehe Abbildung 2.6). Der Webserver kann mithilfe der bereitgestellten Funktionalität dynamische Weboberflächen generieren.

Abbildung 2.6: Model-View-Controller-Enwurfsmuster

Während der Entwicklung des Softwaresystems wird in den Models die grundlegende Datenstruktur anhand von Klassen definiert. Webansichten lassen sich in den Views erstellen. Mithilfe von ASP.NET MVC werden auf dem Server vordefinierte Webansichten in Form von CSHTML-Dokumenten geschaffen, welche eine spezielle Razor-Syntax enthalten. Diese beginnen jeweils mit einem @-Tag und werden direkt in die HTML-Struktur eingebunden. Bevor eine Webseite an den Client gesendet wird, werden alle Razor-Elemente gerendert. Der enthaltene Razor-Code wird verarbeitet und gegen HTML-Code ausgetauscht, der bereits die erforderlichen Server-Daten beinhaltet. Der Client erhält so lediglich die generierte HTML-Seite. Methoden im Controller geben schließlich durch HTTP-Requests die passenden Views mit den durch die Models bereitgestellten Daten an den Client zurück.[34]

[33] vgl. [EntityFrameworkTutorial 2016]
[34] vgl. [Anderson u. Mullen 2016]

Abbildung 2.5 zeigt exemplarisch die Implementierung eines Controllers, welcher eine Methode zum Laden einer Webansicht beinhaltet. Dem Aufruf wird ein View-Model mit den erforderlichen Daten übergeben, die beim Rendern verarbeitet werden:

```
1  public class TemplateController : Controller
2  {
3    private DatabaseContext db = new DatabaseContext();
4
5    //HTTP-Aufruf: /Template/Details/id
6    public ActionResult Details(long id)
7    {
8      var template = db.Templates.FirstOrDefault(x =>
9        x.Id == id);
10     return View(new TemplateViewModel(template));
11   }
12 }
```

Listing 2.5: beispielhafte Implementierung eines Controllers

2.2.3 JavaScript-Technologien

JavaScript ist eine weit verbreitete webbasierte objektorientierte Skriptsprache und wird insbesondere für den dynamischen Aufbau von Webseiten verwendet. JavaScript findet Anwendung für die clientseitige Verarbeitung von Daten und ist deshalb Inhalt dieser Arbeit. Die Technologie stellt Funktionalitäten bereit, um aus einer gegebenen statischen Webseite eine dynamische *Document Object Model* (DOM) Struktur aufzubauen. Hierbei handelt es sich um eine hierarchische Baumstruktur der Webseite, die den Aufbau einer HTML-Seite als einzelne Objekte wiedergibt. Das Editieren dieser Datenobjekte bewirkt eine Veränderung der eigentlich statischen Webseite. Seiteninhalte lassen sich dynamisch verändern oder auslesen. Auf JavaScript basiert eine Vielzahl an Technologien, welche die Kommunikation zwischen Client und Server vereinfachen.[35,36]

jQuery

Bei jQuery handelt es sich um eine leichtgewichtige, schnelle und funktionsreiche JavaScript-Bibliothek. Die Bibliothek vereinfacht die Nutzung von JavaScript und besteht aus drei Komponenten: Selektion, Manipulation und Animation. Auf Grundlage der Selektion in Verbindung mit der durch JavaScript bereitgestellten DOM-Struktur kann durch eine Webseite traversiert und nach passenden Selektoren wie Elementtypen, Attribute, Klassen und vergebenen IDs gefiltert werden. Diese selektierten Elemente können beliebig manipuliert werden. Beispielsweise besteht die Möglichkeit, die Attribute eines Elements zu editieren, zu löschen oder die zugrunde liegende Struktur zu erweitern.

[35] vgl. [Steyer 2014, S. 189 f.]
[36] vgl. [Mozilla Developer Network 2016]

Ebenso können den selektierten Elementen mehrere Events zugeordnet werden, welche abhängig von der Art ausgelöst werden. Eine detaillierte Dokumentation ist auf der offiziellen Seite der jQuery Foundation verfügbar.[37,38]

Asynchronous JavaScript and XML (AJAX)

Statische Webseiten, die lediglich eine Serverkommunikation über HTTP-Requests ermöglichen, besitzen eingeschränkte Möglichkeiten, um dynamische Seiteninhalte bereitzustellen. Der Client sendet synchron einen HTTP-Request an den Server. Daten, welche über einen HTML-Response an den Client gesendet werden, lassen sich lediglich über ein komplettes Neuladen der Webseite einbinden. Dateneingaben werden über Formulare an den Server gesendet und erfordern ebenfalls einen erneuten Seitenaufbau.

Javascript stellt mit *Asynchronous JavaScript and XML*, kurz AJAX, eine Technologie bereit, welche es ermöglicht, Daten asynchron zwischen Server und Client auszutauschen. Mit deren Hilfe kann eine Webseite aktualisiert werden, ohne diese erneut vollständig laden zu müssen. Des Weiteren können auf diese Weise Datenabfragen an den Server gestellt werden, nachdem eine Webseite bereits geladen wurde. Es müssen demnach nicht alle Daten beim erstmaligen Laden an den Client gesendet werden. Vom Server erhaltene Daten können wieder dynamisch in die Webseitenstruktur eingebunden werden. Auch Speicheroperationen sind ohne Zuhilfenahme von Formularen und erneutes Laden der Webseite möglich. Listing 2.6 zeigt einen beispielhaften AJAX-Aufruf mithilfe von jQuery.[39]

```
1  var data = { //zu sendendes Datenobjekt
2    TemplateId: getTemplateId(),
3    Id: $("#Id").val(),
4    Title: $("#Title").val(),
5    Value: $("#Value").val()
6  };
7
8  $.ajax({
9    data: JSON.stringify(data), //JSON-Struktur generieren
10   type: "POST",
11   contentType: "application/json; charset=utf-8",
12   url: "/OptionValue/Edit",
13   success: function (data) {
14     alert("Das Speichern war erfolgreich");
15   }
16 });
```

Listing 2.6: beispielhafter AJAX-Aufruf mithilfe von jQuery

[37] vgl. [jQuery Foundation 2016a]
[38] vgl. [jQuery Foundation 2016b]
[39] vgl. [W3Schools 2016]

Kendo UI

Kendo UI ist ein Framework von Telerik, welches auf Basis von JavaScript optisch ansprechende grafische Benutzerelemente bereitstellt. Diese lassen sich individuell mit Daten befüllen und miteinander verbinden. Kendo UI bietet eine Integration zu ASP.NET MVC an und gibt so die Möglichkeit, die grafischen Elemente bereits auf dem Server ohne zusätzlichen Einsatz von JavaScript mit Daten zu befüllen. Die bereitgestellten Benutzerelemente werden über Razor-Code in die serverseitige HTML-Struktur eingefügt. Während des Render-Vorgangs werden die so eingebundenen Objekte in HTML-Strukturen überführt und an den Client gesendet.[40]

Listing 2.7 zeigt den exemplarischen Code für eine von Kendo definierte mehrspaltige Tabelle, ein sogenanntes Grid:

```
@(Html.Kendo().Grid<TemplateViewModel>()
    .Name("TemplateGrid")
    .Columns(columns =>
    {
        columns.Bound(x => x.Name).Title("Kategorie");
        columns.Bound(x => x.BasePrice).Format("{0:c2}");
        columns.Bound(x => x.SalePrice).Format("{0:c2}");
        columns.Bound(x => x.Profit).Format("{0:p2}");
    })
    .DataSource(ds => ds.Ajax().Read("Read", "Template"))
    .Resizable(resize => resize.Columns(true))
)
```

Listing 2.7: Ein Kendo-Element als Razor-Code

[40] vgl. [Telerik 2017a,b]

3 Problembeschreibung und Ausgangssituation

Die Infrastruktur von XECURIS besteht aus verschiedenen Teilsystemen. Nachfolgend wird ein Augenmerk auf diejenigen Systeme der Unternehmensinfrastruktur gelegt, welche für den Produktverwaltungs- und Bestellverarbeitungsprozess primär genutzt werden.

3.1 Magento

Bei Magento handelt es sich um eines der weltweit führenden Onlineshop-Systeme. Es basiert auf der Skriptsprache PHP und ist modular aufgebaut. Magento ist speziell für Shop-Lösungen im Bereich des E-Commerce ausgerichtet. In XECURIS wird die Magento Version 1.9 verwendet.[41] Da das Softwaresystem Open Source ist, können am Quellcode beliebig Anpassungen vorgenommen werden. Ebenso existiert eine Vielzahl an kostenlosen und kostenpflichtigen Plugins, welche mithilfe des zentralen Portal Magento Connect bezogen werden. Über eine zentrale Administrationsübersicht, das sogenannte Admin Panel, lassen sich mehrere Onlineshops gleichzeitig verwalten:

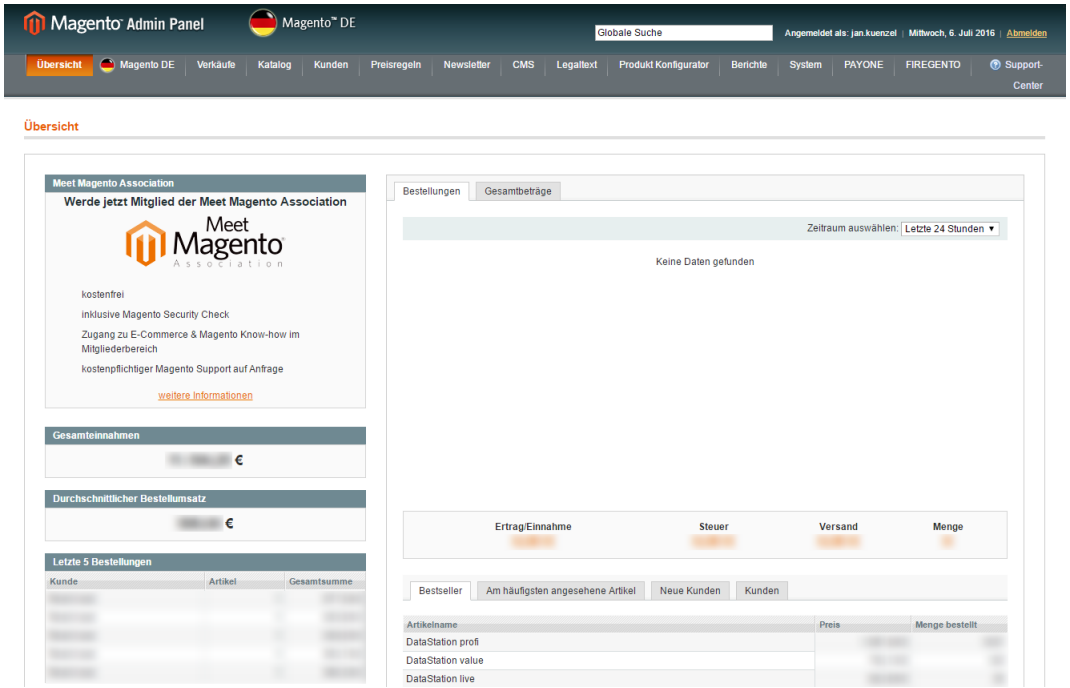

Abbildung 3.1: Das Magento Admin Panel

[41] Stand: März 2017

In Magento sind bereits umfangreiche Funktionalitäten zum Betreiben eines Onlineshops integriert, welche keine zusätzlichen Plugins oder Zusatzsoftware benötigen. Beispielsweise liefert Magento im Onlineshop ein Tool zur Suchmaschinenoptimierung (SEO) sowie den dynamischen Umgang mit verschiedenen Währungen und Steuersätzen. Für Kunden, die den Onlineshop über Smartphones oder Tablets aufrufen, erhält der Betreiber des Onlineshops die Möglichkeit, aus mehreren verschiedenen Responsive Webdesigns zu wählen oder ein individuelles Design anzulegen.[42]

Magento wird in zwei verschiedenen Versionen angeboten: Zum einen als kostenlose Community Edition und zum anderen als kostenpflichtige Enterprise Edition. Die kostenlose Version erhält in dem bereitgestellten Forum eine große Unterstützung durch eine im Forum aktive Mitglieder. Die kostenpflichtige Version enthält neben den Funktionalitäten der Community Version einen zusätzlichen Funktionsumfang sowie einen Support und eine Gewährleistung für das Magento-System.[43]

3.2 Produkt Konfigurator von justSelling

Im E-Commerce-Bereich haben sich Unternehmen entwickelt, welche Produkte zur Erweiterung der Funktionen von gängigen Onlineshop-Lösungen anbieten. Magento liefert in der Grundfunktionalität nicht die benötigte Komplexität für die Produkte, welche in XECURIS angeboten werden. Die justSelling GmbH bietet einen Produkt Konfigurator an, mit dem sich in Magento die vorhandenen Möglichkeiten zur Individualisierung von Artikeln erweitern lassen. Auf dieser Grundlage ist die Entscheidung gefallen, den Produkt Konfigurator im Unternehmen einzusetzen. Die Erweiterung wird derzeit für Magento-Onlineshops bis Version 2.1 unterstützt.[44] Der Konfigurator wird als Extension in die Struktur von Magento eingebunden. Die Konfiguration erfolgt über einen zentralen Reiter in der Magento-Administrationsübersicht (siehe Abbildung 3.1).[45]

3.2.1 Struktur

Die grundlegende Struktur des justSelling Produkt Konfigurators bilden vier hierarchisch gegliederte Komponenten. Die höchste Ebene bildet eine sogenannte *Vorlage*. Diese wird als das Produkt im Onlineshop verstanden, welches individuell konfigurierbar ist. Die Vorlage wird anschließend mit dem jeweiligen Artikelstamm im Magento-Produktkatalog verknüpft. An jede Vorlage des Produkt Konfigurators können Optionen geknüpft werden. Diese sind einer übergeordneten Kategorie zugeordnet, den *Optionsgruppen*. Eine *Option* umfasst immer Komponenten oder Dienstleistungen der gleichen

[42] vgl. [Magento Inc. 2014, S. 2, 5 ff.]
[43] vgl. [Magento Inc. 2016a,b]
[44] Stand: März 2017
[45] vgl. [justSelling GmbH 2016a,b]

Art. Diese werden unter *Optionswerten* zusammengefasst. Sie stellen jeweils im Onlineshop auswählbare Komponente dar, beispielsweise ein spezifisches Arbeitsspeicher-Modul (siehe Tabelle 3.1).

Komponente	**Erläuterung**	**Beispiel**
Vorlage	Produkt	Datastation Value
Optionsgruppe	übergeordnete Kategorie	Hardware
Option	beinhaltet Komponenten der gleichen Art	Arbeitsspeicher
Optionswert	Auswählbare Komponente	8GB DDR3-RAM (2 x 4GB DIMMs)

Tabelle 3.1: Erläuterung der Komponenten des Produkt Konfigurators

3.2.2 Funktionsumfang

Der Produkt Konfigurator lässt sich in einem separaten Backend administrieren (siehe Abbildung 3.2). Vorlagen, Optionsgruppen und Optionen sind jeweils über eigene Reiter erreichbar. Über das Konfigurationsmenü einer Option lässt sich einstellen, wie die Auswahl von zugeordneten Optionswerten geschehen soll. Entscheidend ist hierbei der gewählte Eingabetyp. Der administrierende Mitarbeiter erhält die Möglichkeit, aus einer Vielzahl von Elementen wie Radio Buttons, Dropdown-Listen oder die Wahl über ein anklickbares Bild zu wählen. Auch das Hochladen von Dateien oder die Eingabe von Texten ist über den Eingabetyp der Option realisierbar.

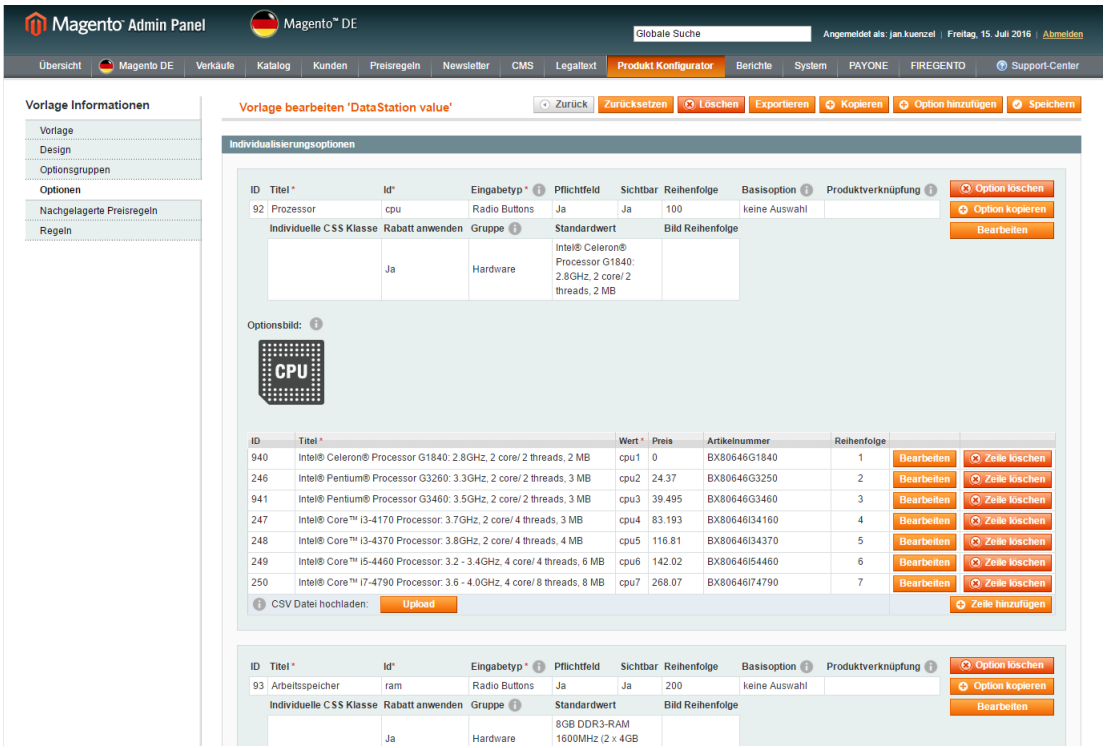

Abbildung 3.2: Magento Backend

Durch den justSelling Produkt Konfigurator besteht die Möglichkeit, bestimmte Optionswerte ausblenden zu lassen. Dieses Feature ist als Blacklisting bekannt. Optionswerte können aufgrund von verschiedenen Auswahlkombinationen ausgeblendet werden. Diese Funktionalität ist für den Onlineshop von XECURIS notwendig, da die Komponenten einer Konfiguration aufeinander abgestimmt werden müssen und folglich nicht alle Kombinationen zulässig sind. Hierarchische Abhängigkeiten bilden die Grundlage für das Ausblenden von Optionswerten. Beispielsweise ist es denkbar, dass für verschiedene Monitortypen bestimmte Farben nicht zur Verfügung stehen bzw. bei Auswahl des Monitortyps *Ohne* alle Farben ausgeblendet werden (siehe Abbildung 3.3). Die Optionen, welche voneinander abhängig sind, müssen miteinander über sogenannte *Basisoptionen* verbunden werden. Anschließend muss für jeden Optionswert eine separate Filterbedingung definiert werden. Durch die Definition von Filtern lassen sich nicht nur einfach, sondern auch komplex strukturierte Produkte im Produkt Konfigurator abbilden.

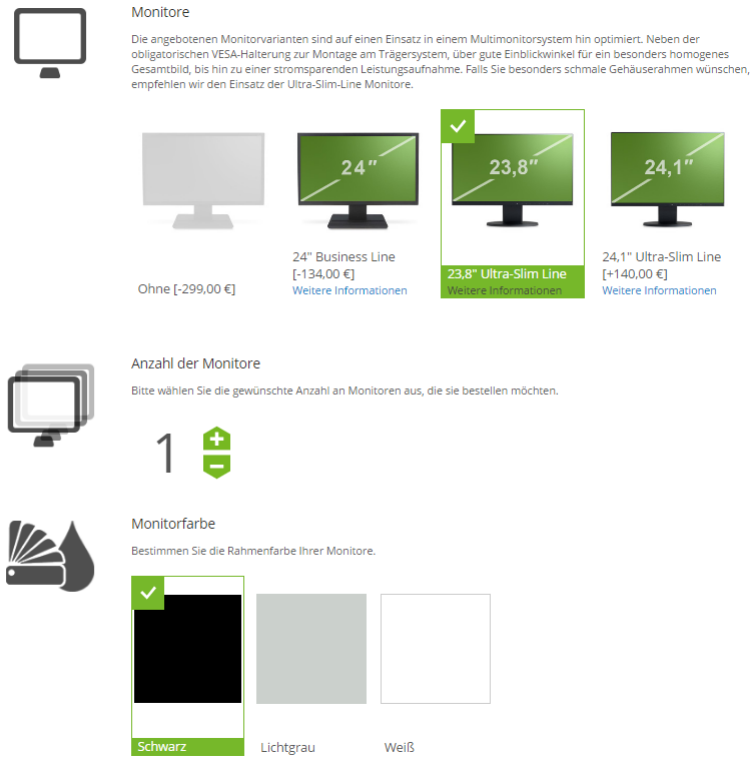

Abbildung 3.3: Die Farbauswahl im Onlineshop

Die Daten des Produkt Konfigurators lassen sich extern verwalten. Eine Import- und Exportfunktionalität ermöglicht es, Vorlagen extern im gegebenen Dateiformat zu manipulieren und anschließend über die Importfunktion in das Magento-System einzuspielen. Die Exportfunktionalität generiert ein ZIP-Archiv mit allen an die Vorlagen geknüpften Bildern sowie einer Datei, welche die Vorlagenstruktur im JSON-Format beinhaltet.

3.3 Angrenzende Systeme

In XECURIS werden verschiedene Softwaresysteme eingesetzt, die neben dem Onlineshop ebenso für die Bestellverarbeitung relevant sind. Diese werden nachfolgend erläutert.

3.3.1 Factura

Bei Factura handelt es sich um ein ERP-System. Die 2002 veröffentlichte Software ist ein Produkt der combit GmbH und setzt sich aus verschiedenen Modulen zusammen. Dazu zählen Bereiche wie eine Stammdaten-, Lager- und Belegverwaltung, ein Bestellwesen sowie ein Modul zur Aufgabenplanung. In XECURIS wird die Anwendung zur Auftragsabwicklung verwendet. Hierzu zählen sowohl Kundenaufträge als auch interne Aufträge, Produktions- bzw. Wartungsaufträge oder ein Vor-Ort-Aufbau bei einem Kunden.[46]

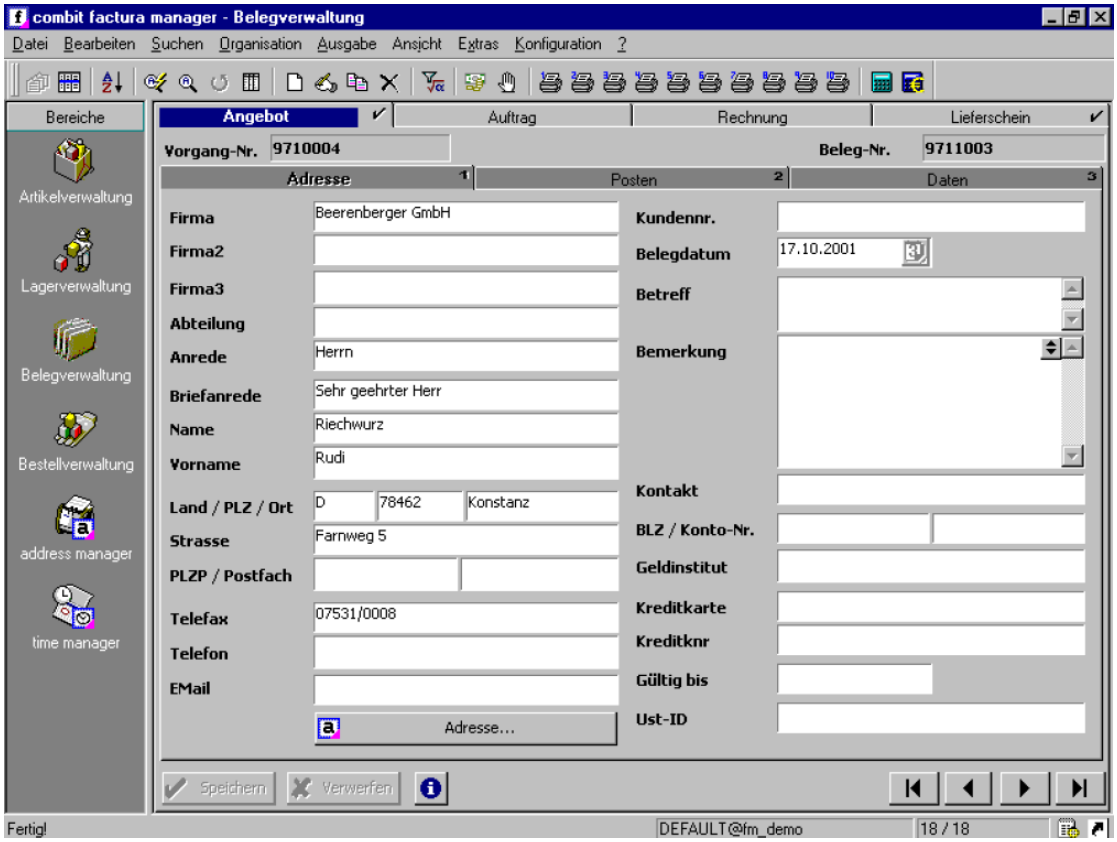

Abbildung 3.4: Die Factura-Benutzeroberfläche mit Beispieldatensatz[47]

[46] vgl. [combit GmbH 2002, S. 13 ff.]
[47] [combit GmbH 2002, S. 82]

Eine Integration zu diesem System ist nicht möglich, da die Software zum einen keine Schnittstellen für einen externen Zugriff auf die Daten enthält und zum anderen der Zugriff auf die Datenbasis nicht durchgängig gewährleistet werden kann, weil die zugrundeliegende Datenbank lediglich für einen Single-User-Betrieb ausgelegt ist. Auch Einkaufspreise (EKs) sowie verschiedene Verkaufspreise (VKs) können hierbei hinterlegt werden. Das System lässt jedoch maximal fünf Lieferanten für ein Produkt zu. Für Produkte existiert eine Fülle an Distributoren. Die vorgeschriebene maximale Anzahl von fünf Distributoren ist somit nicht einhaltbar, da aus allen verfügbaren Preisen und Beständen ein einheitlicher Einkaufspreis als Basis generiert werden muss. Ebenso besteht nicht die Möglichkeit, eine automatisierte Berechnung des Einkaufspreises durchzuführen. Folglich ist die Nutzung der Funktionalität für XECURIS nur mit einem hohen Aufwand verbunden und nicht automatisierbar.

3.3.2 X1

Bei X1 handelt es sich um ein hauseigenes ERP-System von XECURIS. Dieses befindet sich in der Entwicklung und basiert auf dem Framework ASP.NET MVC. Das Softwaresystem ist aus dem Ausgangspunkt entstanden, dass ERP-Systeme im Bereich der Standardsoftware nicht genügend an die Bedürfnisse des Unternehmens anpassbar sind. Hierbei wurde ein gesamtheitlicher Integrationsansatz gewählt. Über das zentrale System soll eine Prozess-, Daten- sowie Funktionsintegration erreicht werden. Alle Geschäfts- und Verwaltungsprozesse des Unternehmens finden in einem zentralen Portal statt. Langfristig zielt das System darauf ab, einen Großteil der im Unternehmen eingesetzten Standardsoftware abzulösen, relevante Workflows bedeutend zu optimieren und eine Integration zu notwendigen angrenzenden Systemen zu schaffen.[48]

3.3.3 EGIS

Bei EGIS handelt es sich um ein Online-Portal der in Deutschland ansässigen Synaxon AG. Das System besitzt eine Anbindung zu Distributoren für IT- und Elektronik-Produkte. Die Preise im IT-Umfeld fluktuieren stark und erlauben keine großen Produktmargen. Wie bereits in der Einleitung erwähnt, sind für eine Preiskalkulation aktuelle Einkaufspreise und ein stetiger Marktüberblick sehr wichtig. Nutzer des Portals erhalten Zugang zu stetig aktualisierten Produktpreisen von verschiedenen Distributoren sowie einer umfangreichen Artikeldatenbank. Preise werden ständig aktualisiert und den Nutzern zur Verfügung gestellt. Auch Bestellabwicklungen bei den Lieferanten sind direkt über das zentrale Portal möglich.[49]

[48] Stand: März 2017
[49] vgl. [Synaxon AG 2016b]

SYNAXON-Kunden können über eine Schnittstelle auf die EGIS-Daten zugreifen. Die Zugriffsmöglichkeiten erstrecken sich über mehrere Bereiche bzw. Komponenten mit jeweils eigenen Funktionen. So lassen sich beispielsweise nicht nur Artikel-Stammdaten abrufen, sondern auch Artikel zusammenstellen und abfragen. Diese erlaubt es, eigene Tools zu entwickeln, um automatisierte Preisabfragen zu starten. Produkte können über die angegebenen Artikelnummern eindeutig identifiziert und so mit eigenen Datenbeständen abgeglichen werden.[50,51]

3.4 Ablauf der Produktverwaltung und der Bestellverarbeitung

Wie aus Abschnitt 3.2 hervorgeht, ist die Konfiguration komplexer Produkte mit einem sehr hohen Aufwand verbunden. Diese Komplexität muss aber eingehalten werden, um die Produkte im Onlineshop so anpassbar wie möglich zu gestalten. In der Produktverwaltung existieren zwei elementare Arbeitsfelder. Zum einen werden stetig neue Produkte entwickelt und somit über den Produkt Konfigurator angelegt. Zum anderen müssen die bestehenden Produkte administriert werden, sodass sich diese stets auf einem aktuellen Stand befinden.

3.4.1 Produkterstellung

Um ein Produkt zu erstellen, muss zunächst eine Vorlage angelegt werden. Anschließend können über den passenden Reiter die Optionsgruppen und Optionen erstellt werden. Im Anschluss können über die Bearbeiten-Dialoge die hierarchischen Abhängigkeiten sowie die Filter gesetzt werden. Elemente lassen sich im Produkt Konfigurator nicht wiederverwenden, d.h. Komponenten mit derselben Struktur müssen in verschiedenen Vorlagen immer neu angelegt werden. Insbesondere an Optionswerten bestehen viele konfigurierbare Felder, welche auszufüllen sind. Da die Daten für jeden Optionswert manuell eingepflegt werden müssen, entstehen an dieser Stelle redundante, zeitaufwendige und fehleranfällige Arbeitsschritte.

Wie in Absatz 3.2.2 erläutert wurde, können im Magento Produkt Konfigurator Optionswerte durch hierarchische Abhängigkeiten ausgeblendet werden. Wenn ein Optionswert gefiltert werden soll, so muss die Hierarchie zu allen Optionen aufgebaut werden, welche für die Filter relevant sind. Die Definition dieser Abhängigkeiten geschieht über sogenannte Basisoptionen. Problematisch hierbei ist, dass für eine Option nicht mehrere Basisoptionen definiert werden können, d.h. für den hierarchischen Aufbau müssen Optionen verkettet werden.

[50] vgl. [Synaxon AG 2016a]
[51] vgl. [Synaxon AG 2015a]

Die Verkettung der Basisoptionen läuft hierbei der Zugriffskette entgegen:

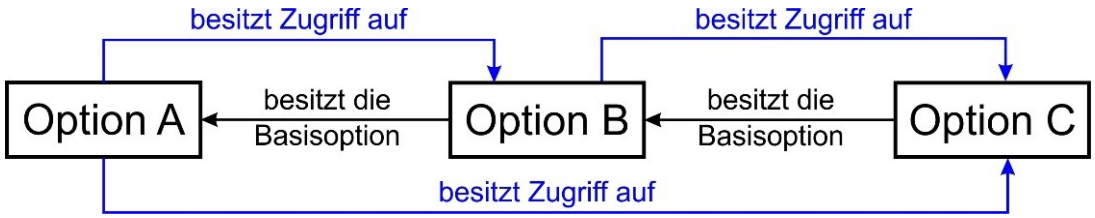

Abbildung 3.5: Verkettung der Optionen über Basisoptionen

Das Ausblenden von Optionswerten mit mehreren Abhängigkeiten ist nur mit sogenannten Expressions möglich. Dabei handelt es sich um eine JavaScript-Programmierschnittstelle des Produkt Konfigurators, mit dessen Hilfe eigene Rückgabewerte und zusätzliche Abhängigkeiten definiert werden können. Die Umsetzung vieler Abhängigkeiten ist jedoch mit einem verhältnismäßig großen Aufwand verbunden und erfordert Programmierkenntnisse von dem administrierenden Mitarbeiter des Onlineshops. In Abbildung 3.6 existieren exemplarisch zwei unabhängige Filter-Bedingungen zum Ausblenden eines Optionswertes, die bei Eintreten jeweils den Rückgabewert 1 liefern:

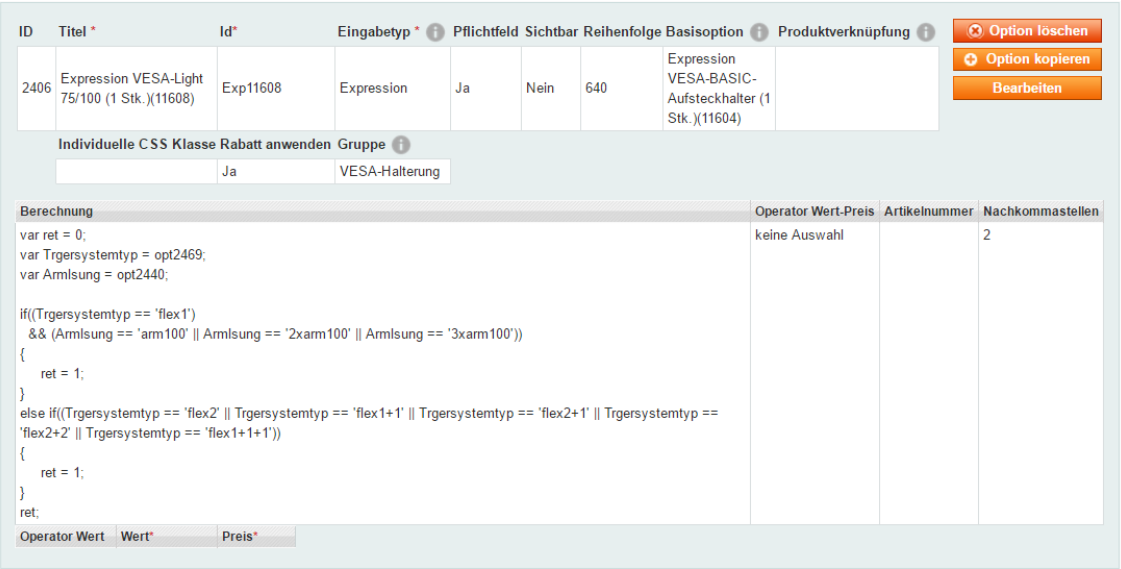

Abbildung 3.6: Beispiel einer Expression mit mehreren Abhängigkeiten

Des Weiteren müssen im Bearbeiten-Dialog der Expression unter dem Menüpunkt *Einstellungen abhängiger Optionen* diejenigen Optionen markiert werden, welche für die Filterung genutzt werden. Auf diese Weise wird bei jedem Klick auf eine betroffene Option eine neue Berechnung durchgeführt und das Ein- sowie Ausblenden der nötigen Optionswerte realisiert.

Wie aus Abbildung 3.7 hervorgeht, geht bei steigender Anzahl der verbundenen Optionen die Übersichtlichkeit verloren:

Abbildung 3.7: Aktualisierung von Abhängigkeiten für die Filterung von Optionswerten

3.4.2 Preiskalkulation

Die Berechnung der Verkaufspreise für den Magento Onlineshop geschieht nach einem komplexen Muster. Artikel, welche in einem Optionswert enthalten sind, lassen sich von verschiedenen Lieferanten über Portale wie EGIS beziehen. Ziel ist es, einen kalkulatorischen Einkaufspreis zu erhalten, der zum einen konkurrenzfähig ist und zum anderen mögliche kurzfristige Preisänderungen einkalkuliert. Jeder Lieferant besitzt eigene Einkaufspreise. Um einen einheitlichen Artikel-Einkaufspreis als Grundlage für die weitere Berechnung zu erhalten, werden die verschiedenen Einkaufspreise zuerst durch einen Mittelwert errechnet. Anschließend werden auf deren Grundlage absolute Einkaufspreise für Optionswerte kalkuliert. Diese bilden schließlich in Verbindung mit den definierten Artikel-Margen den Ausgangspunkt für die Verkaufspreise der Artikel.

Abbildung 3.8 gibt einen Überblick über die einzelnen Schritte zur Berechnung des Artikel-Verkaufspreises:

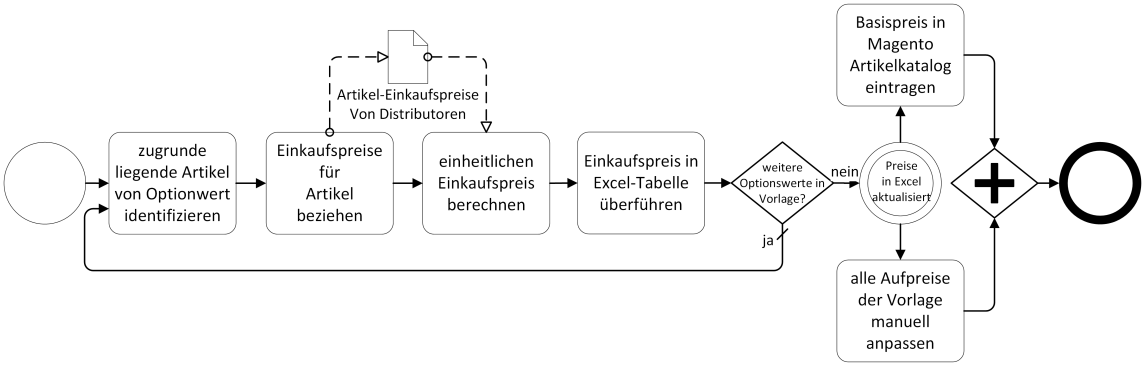

Abbildung 3.8: allgemeiner Ablauf der Preiskalkulation in XECURIS

Preise, die im System des justSelling Produkt Konfigurators festgeschrieben werden, sind speziell formatiert: Es existiert ein Basispreis für jedes Produkt. Dieser befindet sich nicht in der Vorlagenzuordnung des Produkt Konfigurators, sondern in der separaten Produktdatenbank von Magento. Dieser Basispreis enthält alle Verkaufspreise der Standardkonfiguration des Produktes. Alle gewählten Optionswerte, welche von der Standardkonfiguration abweichen, werden zum Basispreis als relativer Aufpreis aufaddiert und sind schließlich im Frontend am Optionswert sichtbar (siehe Abbildung 3.9).

Abbildung 3.9: Die Option Arbeitsspeicher im Frontend

Standardwerte besitzen immer einen Aufpreis von 0,00 Euro, da deren Preis bereits in der Standardkonfiguration enthalten ist. Um die Aufpreise zu erhalten, wird der absolute Verkaufspreis des Optionswertes mit dem absoluten Verkaufspreis des standardmäßigen Optionswertes subtrahiert. Um einen runden Brutto-Aufpreis zu erhalten, muss dieser Aufpreis schließlich mit der Mehrwertsteuer multipliziert und somit in einen Brutto-Betrag umgewandelt werden. Anschließend wird der Preis aufgerundet und wieder in einen Netto-Betrag zurückgerechnet (siehe Tabelle 3.2).

Der Prozess der Preiskalkulation wird in der folgenden Abbildung verdeutlicht:

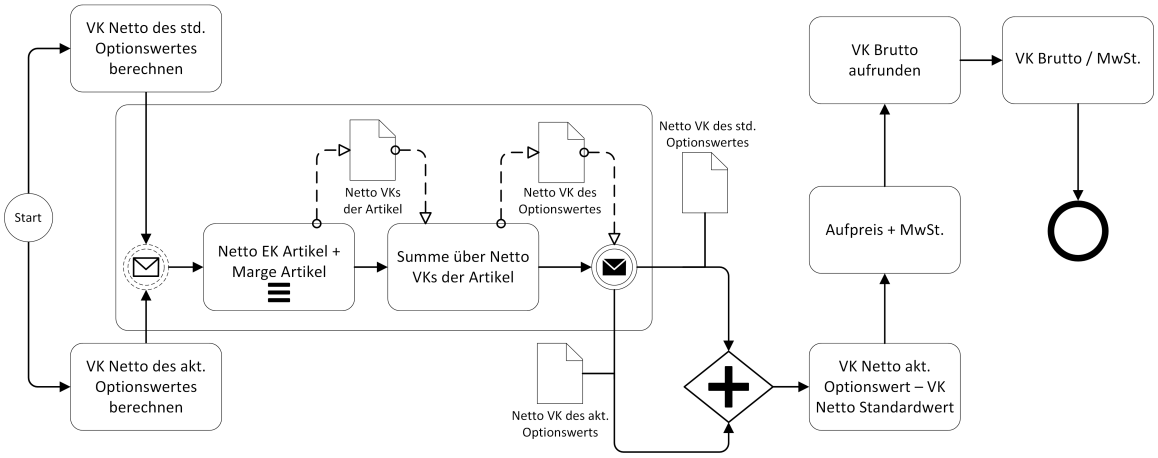

Abbildung 3.10: Die Aufpreiskalkulation für den Magento Produkt Konfigurator

Die kalkulierten Netto-Aufpreise werden schließlich an den Optionswerten festgeschrieben. Alle Preise sind hierbei als Netto-Beträge gespeichert. Der Produkt Konfigurator wandelt die Preise für die Anzeige im Frontend dynamisch in Brutto-Beträge um. Die direkte Berechnung der Aufpreise aus den realen Verkaufspreisen ist im Produkt Konfigurator nicht vorgesehen. Dies bedeutet, dass für die Erzeugung eines adäquaten Verkaufspreises eine externe Preisberechnung und -fortschreibung stattfinden muss. Einzutragende Preise müssen zunächst extern vorkalkuliert und aufgearbeitet werden. In XECURIS geschieht die Berechnung zum gegenwärtigen Zeitpunkt über eine Excel-Tabelle, welche komplex ist und bei jeder Preisaktualisierung händisch mit den gegenwärtigen Einkaufspreisen aktualisiert werden muss (siehe Abbildung A.3). Insbesondere Optionswerte, welchen mehrere Artikel zugrunde liegen, bedeuten einen großen Aufwand für die Kalkulation der Aufpreise, da für jeden Artikel eine individuelle Marge gesetzt werden soll. Nachfolgend ist eine beispielhafte Aufpreisberechnung für zwei Optionswerte dargestellt, wobei der Optionswert in der linken Spalte als Standardwert der Option angesehen werden kann:

Bezeichnung	8GB DDR4-RAM 2133MHz (2 x 4GB DIMMs)	16GB DDR4-RAM 2133MHz (2 x 8GB DIMMs)
Artikel	KINGSTON HyperX Fury Black 8GB Kit (2 x 4GB)	KINGSTON HyperX Fury Black 16GB Kit (2 x 8GB)
Netto EK	46.5000 €	90.5000 €
Marge	17.00%	17.00%
Netto VK	54.4050 €	105.8850 €
Netto Aufpreis	0.0000 €	51.4800 €
Netto Aufpreis (gerundet)	0.0000 €	52.1008 €

Tabelle 3.2: beispielhafte Preisberechnung für zwei Optionswerte aus der Option Arbeitsspeicher

Das Schema zur Berechnung der Aufpreise ist in den nachfolgenden Gleichungen zusammengefasst und wird anhand des Beispiels in Tabelle 3.2 verdeutlicht:

$$Netto\,VK = Netto\,EK * \left(\frac{Marge}{100}+1\right)$$

$$Netto\,Aufpreis = VK\,Netto - VK\,Netto\,(Std.)$$

$$Netto\,Aufpreis\,(gerundet) = \frac{aufgerundet\left(Netto\,Aufpreis * \left(\frac{MwSt.}{100}+1\right)\right)}{\frac{MwSt.}{100}+1}$$

$$Netto\,VK = 90.5000\,€ * 1.17 = 105.8850\,€$$

$$Netto\,Aufpreis = 105.8850\,€ - 54.4050\,€ = 51.4800\,€$$

$$Netto\,Aufpreis\,(gerundet) = \frac{aufgerundet\,(51.4800\,€ * 1.19)}{1.19} = 52.1008\,€$$

Wie die Aktualisierung von gleichen Elementen muss auch das Eintragen der Produktpreise manuell stattfinden. Identische Optionswerte, welche im Konfigurator enthalten sind, müssen einzeln bei einer Änderung durch einen Unternehmensmitarbeiter über das Magento Admin Panel bearbeitet werden. Im IT-Bereich ändern sich Einkaufspreise fortlaufend. Folglich ist es notwendig, für die Kalkulation des Verkaufspreises stets tagesaktuelle Einkaufspreise zu verwenden. Die stetige Änderung der Einkaufspreise macht es weiterhin notwendig, eine regelmäßige Aktualisierung der Produktvorlagen im Produkt Konfigurator durchzuführen. Es existiert keine Historisierung der Preisdaten. Dies hat zur Folge, dass durch den Magento Onlineshop nicht nachverfolgt werden kann, wann eine Preisänderung stattgefunden oder wie viel eine Konfiguration zu einem Zeitpunkt gekostet hat. Bestellungen können folglich nach einer Änderung an der Vorlage nicht mehr auf ihre Richtigkeit überprüft werden.

3.4.3 Bestellverarbeitung

Nachdem eine Bestellung im Onlineshop eingegangen ist, wird diese über das Magento Admin Panel sichtbar. In diesem existiert eine Ansicht *Positionsübernahme* (siehe Abbildung 3.11), welcher jede Position der Bestellung mit dem jeweiligen Netto-Aufpreis auflistet. Dieser Teil ist standardmäßig nicht in Magento enthalten, sondern wurde von einem Mitarbeiter von XECURIS implementiert.

Die Aufpreise der Bestellpositionen müssen zurückgerechnet werden, sodass wieder ein absoluter Verkaufspreis entsteht. Hierbei muss schließlich eine Auflistung aller den Optionen zugrunde liegenden Artikel sowie ihren jeweiligem Verkaufspreis entstehen. Die Summe der Artikelpreise muss schließlich dem angegebenen summierten Netto-Verkaufspreis entsprechen. Um die Berechnung durchzuführen, werden die Aufpreise der gewählten Konfiguration des Kunden wieder in das Excel-Dokument überführt. Hierbei findet eine Kalkulation statt, welche den Aufpreis wieder mit dem absoluten Verkaufspreis des Standardwertes in Relation setzt (siehe Abbildung A.3).

Kategorie	Komponente	Artikelnummer		Name	Preis (netto)
Hardware	Basis	MTS-TRA	Copy	Monitorträgersystem Traverse	125.2101
Anzahl Ebenen	Ebenen	xx	Copy	Zwei Ebenen	112.60504201681
Anordnung der Monitore	Anordnung der Monitore	10	Copy	3+1	315.12605042017
Traversenform	Traversenform	parabolisch	Copy	parabolisch	12.605042016807
Traversenlänge	Traversenlänge	50	Copy	parabolisch 3 + 1 1660mm bis 27"	20.168067226891
Befestigungsvarianten	Befestigungsvarianten	8	Copy	Einbauadapter mit Kabel durchl. (2 Stk.)	65.546218487395
Säulenlänge	Säulenlänge	21	Copy	Säule 30/800 (2 Stk.)	0
Armlösung	Armlösung	4x 5158x	Copy	VESA-Schlitten (4 Stk.)	0
VESA-Halterung	VESA-Halterung	4x 51100	Copy	5D-VESA 75/100 (4 Stk.)	122.68907563025
VESA-Adapter	VESA-Adapter		Copy	Ohne	0
Farbe	Farbwahl	8-RAL.9011	Copy	RAL 9011 Schwarz matt	0

Summe: 552.9412

Versandart	Versandkosten
Versand - Versand mit DHL	10.8824

Summe: 0

Gesamtsumme: 552.9412

Abbildung 3.11: Positionsübernahme

4 Analyse und Konzeption

Obwohl viele Optimierungen bereits Teil des Softwaresystems sind, bestehen dennoch Schwächen im Umgang mit dem System bzw. weitere Optimierungsansätze bezüglich der umgesetzten Workflows. Nachfolgend werden ausgehend von einer Auflistung von existierenden Schwächen des Softwaresystems Anforderungen an eine mögliche Optimierung der Arbeit im Umgang mit dem Magento Onlineshop thematisiert. Ziel ist es, einen halbautomatisierten Workflow für den Produktverwaltungsprozess zu schaffen. Wiederholende Prozessabläufe werden, wenn möglich, automatisiert, sodass lediglich überprüfende Schritte durch den administrierenden Mitarbeiter erforderlich werden.

4.1 Vorteile des bestehenden Softwaresystems

Die Infrastruktur von XECURIS setzt sich durch die unabhängigen Systemen aus verschiedenen Insellösungen zusammen, welche jeweils in sich selbst ein Teilsystem bilden. Die Prozesse der Teilsysteme sind wenig integriert, gering automatisiert und redundant. Der Umgang mit den komplexen Systemen ist folglich als ineffizient und redundant zu betrachten. Das geschaffene Softwaresystem bildet die Grundlage für eine effizientere Verarbeitung. Nachfolgend wird auf wichtige Teilaspekte eingegangen, welches die Vorteile des bestehenden Softwaresystems näher erläutert.

4.1.1 Funktionalität

Das Softwaresystem enthält bereits Grundfunktionalitäten, die Workflows des justSelling Produkt Konfigurators abbilden bzw. erweitern.

Artikelzuordnung

Im Softwaresystem müssen Optionswerte nicht mehr als die kleinste zuordenbare Einheit verstanden werden. Neben der standardmäßigen Produktstruktur von justSelling lässt sich im eigenen Konfigurator ein Artikelbestand pflegen. In diesem werden alle Stammdaten der Artikel gehalten. Optionswerte und Artikel stehen in einer n:m-Beziehung zueinander, sodass einerseits einem Optionswert mehrere Artikel zugeordnet werden können. Zum anderen besteht die Möglichkeit, Artikel für mehrere Optionswerte wiederzuverwenden. Auch eine Zuordnung von Optionswerten direkt an eine Produktvorlage ist bereits implementiert. Die so zugeordneten Artikel sind unabhängig von den ausgewählten Optionswerten immer in der Konfiguration vorhanden und bilden somit einen Teil des Basispreises der Vorlage. Diese können auch als Grundlage für die Anlage von Optionswerten verwendet werden.

Filtern von Optionswerten

Komplexe Zusammenhänge wie das Setzen von Filtern müssen nicht mehr über komplexe Abhängigkeiten zwischen Optionswerten durchgeführt werden, sondern lassen sich händisch über das Setzen von Checkboxen realisieren. Die Bedienung ist somit auch für ungeschultes Personal möglich und bedarf einer geringeren Einarbeitungszeit. Benötigte hierarchische Abhängigkeiten sowie die benötigten Expressions zum Ausblenden der Werte werden bei einem Export bereits automatisiert angelegt. Eine nebenläufige Beschreibung des aktuell gesetzten Filters unterstützt den administrierenden Mitarbeiter dabei, den gegenwärtig festgelegten Filter nachzuvollziehen.

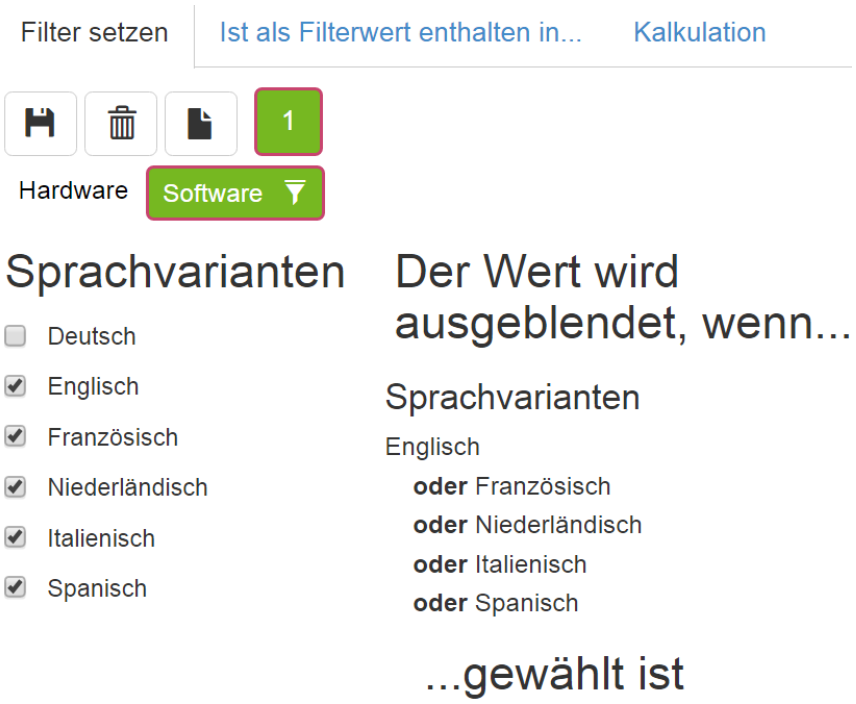

Abbildung 4.1: Filter für den Optionswert *Windows 10 Home 64 Bit, DE* in der Option *Betriebssystem*

Festlegen der Element-Reihenfolgen

Im justSelling Produkt Konfigurator müssen für Optionsgruppen, Optionen und Optionswerte Reihenfolgen in Form von numerischen Werten festgelegt werden. Elemente, welche im Vergleich einen niedrigeren Wert besitzen, werden im Frontend weiter oben angezeigt. Das Festlegen dieser Sortierreihenfolgen ist im Softwaresystem als Drag and Drop Funktionalität implementiert. Demnach müssen nicht mehr manuell numerische Werte festgelegt werden. Diese werden automatisiert festgelegt und in der Datenbank gespeichert.

Export der Daten

Das Softwaresystem besitzt bereits eine Export-Funktionalität. Diese ermöglicht es, eine ZIP-Datei zu generieren, welche über eine Import-Funktion des justSelling Produkt Konfigurators in das Live-System importierbar ist. Das Archiv beinhaltet eine JSON-Datei mit allen Informationen über das Produkt, welches verkauft werden soll, beispielsweise die Vorlage, die Optionsgruppen, Optionen, Optionswerte sowie alle nötigen Filter und Abhängigkeiten. In einem separaten Ordner befinden sich alle exportierten Bilder. Die Referenzen zu dem jeweiligen Bild befinden sich in der Zuordnung der Option bzw. des Optionswertes. Ein automatisierter Import ins Live-System ist noch nicht implementiert. Die Vorlage muss manuell heruntergeladen und importiert werden.

4.1.2 Wiederverwendbarkeit

Das Softwaresystem besitzt Funktionalitäten, um Komponenten wiederverwenden zu können. Optionsgruppen und Optionen lassen sich an der entsprechenden Komponente mit Daten einer anderen Komponente gleicher Art befüllen. Auch Vorlagen können kopiert werden. Neben dem Austausch der Daten einer Komponente sind auch dessen untergeordnete Elemente Teil der Duplikation. Für Optionsgruppen beispielsweise werden alle zugrunde liegenden Optionen, Optionswerte sowie deren Zuordnung zu den Artikeln exakt übernommen. Dieser Ablauf erleichtert es, Vorlagen mit ähnlicher Struktur zu administrieren. Gleichermaßen ist es auf diese Weise möglich, gleiche Komponenten, welche auch in anderen Vorlagen vorkommen, zu aktualisieren.

Viele Optionswerte bestehen aus einem Hauptartikel. In diesem Fall sind die zugeordneten Stammdaten des Optionswertes mit dem des Artikels weitestgehend identisch. Deshalb gibt es beim Anlegen des Optionswertes die Möglichkeit, direkt einen Artikel zu wählen und Felder aus diesem zu übernehmen. Gleiches gilt beispielsweise auch für ein zugeordnetes Produktbild.

4.1.3 Nutzung des Magento Onlineshops

Die Nutzung der Lösung als Middleware ermöglicht es, das Softwaresystem zu nutzen, gleichzeitig aber nicht auf die Funktionalität des Magento Onlineshops bzw. des justSelling Produkt Konfigurators zu verzichten. Insbesondere die Blacklisting-Funktionalität ist ein elementares Kriterium für die Auswahl des justSelling Produkt Konfigurators, auf dessen Funktionalität nicht verzichtet werden kann. Die eigene Implementierung der Filter-Funktionalität ist als komplex und aufwendig angesehen und soll deshalb vermieden werden. Ebenso muss hierbei die konsistente Integration zu Magento betrachtet werden. Insbesondere Administrationsprozesse, welche das Hauptgeschäftsfeld des Unternehmens tangieren, sollten so produktiv wie möglich gestaltet werden. Durch wenig optimierte Workflows und veraltete Software werden Prozesse ineffizient.

Im Vergleich zu dem Arbeitsablauf ohne das Softwaresystem ist bereits eine Steigerung der Effizienz im Bereich der Workflows zu verzeichnen. Die bestehende Softwarelösung trägt bereits dazu bei, dass eine Teilautomatisierung bei der Produktpflege stattfindet und ist auf die Anforderungen des Unternehmens angepasst.

4.1.4 Struktur

Die Struktur des Softwaresystems ist so angepasst, dass eine Integration zu anderen Projekten stattfinden kann. Der Vorteil hierbei liegt darin, dass die zugrunde liegende Datenbank des Softwaresystems unabhängig von Magento ist. Sie soll als Middleware dienen, aus welcher alle Produktverwaltungsprozesse und -aktualisierungen zu anderen Systemen ablaufen. Auf diese Weise können Inkonsistenzen vermieden werden. Des Weiteren besitzt die Software bereits Grundfunktionalitäten, welche den Produktverwaltungsprozess vereinfachen. Weiterhin ist es möglich, den Workflow im Konfigurator so anzupassen, dass für das Unternehmen ein maximaler Nutzen erzielt wird. Abbildung 4.2 zeigt die Datenbankstruktur des Softwaresystems:

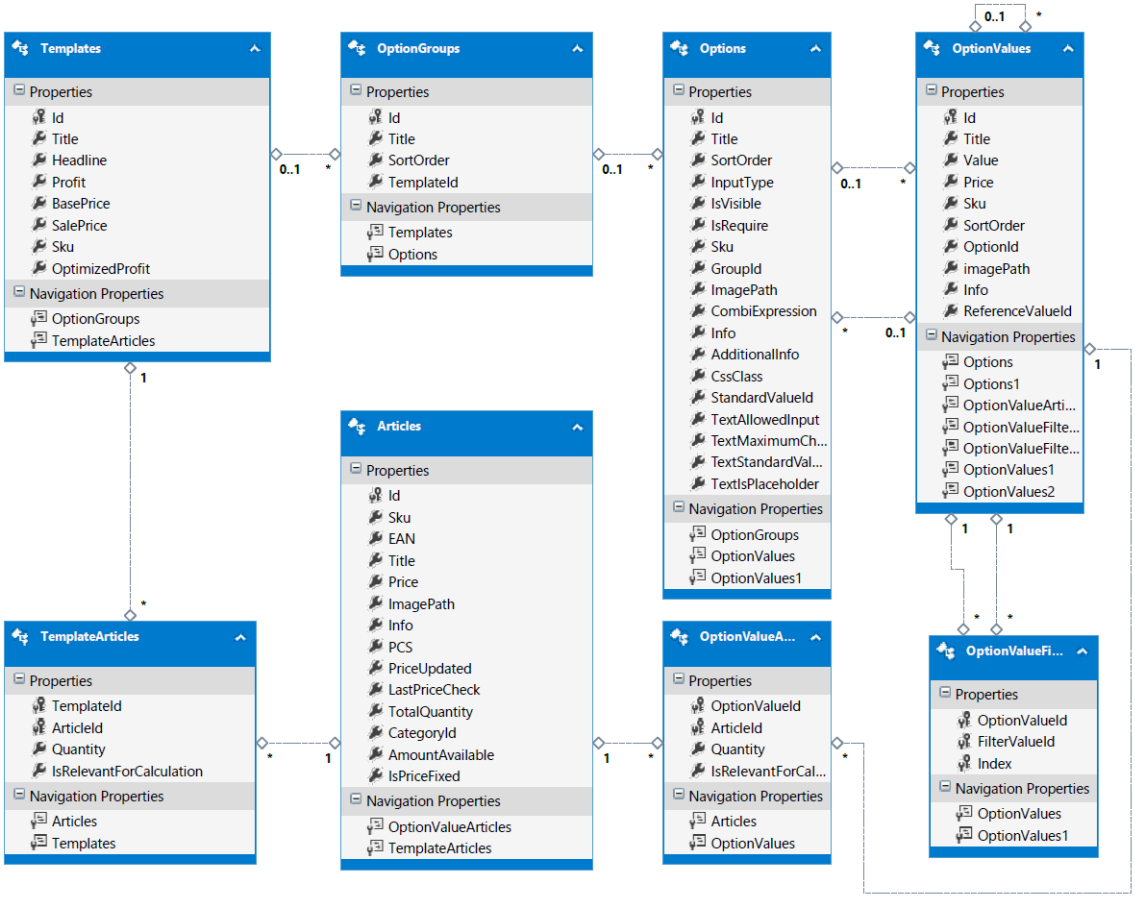

Abbildung 4.2: Datenbankstruktur des Softwaresystems

4.1.5 Erweiterbarkeit

Da das derzeitige System wenig automatisiert ist, sollten an dem System weitere Optimierungen durchgeführt werden. Ebenso zu betrachten ist die Weiterverarbeitung von Bestellungen im Magento Onlineshop. Durch den modularen Aufbau des Softwaresystems ist es problemlos möglich, Erweiterungen am System und somit an der Funktionalität des Produkt Konfigurators vorzunehmen. Auch eine Integration des Softwaresystems in andere auf C# basierenden Projekte wie dem ERP-System X1 ist vorstellbar.

4.2 Schwächen des bestehenden Softwaresystems

Dieser Abschnitt zeigt bestehende Defizite am Softwaresystem auf. Diese bilden die Grundlage für die Anforderungen im Implementierungsprozess, welcher in Kapitel 5 beschrieben wird.

4.2.1 Automatisierungsgrad

Aufgrund einer fehlenden Preiskalkulation für die Optionswerte findet die Kalkulation der Aufpreise nach wie vor in einem separaten Excel-Dokument statt. Da die Kalkulation für jede Vorlage einzeln durchgeführt werden muss, ist es nötig, alle Einkaufspreise sowie die Margen einzeln in das Dokument zu übertragen. Aufpreise bedürfen einer manuellen und sehr aufwendigen Berechnung, indem zunächst alle einem Optionswert zugrunde liegenden Artikel mit ihren jeweils eigenen Margen verrechnet und anschließend zu einem Gesamtergebnis addiert werden.

Der Umgang mit einer Vielzahl an Artikeln ist unübersichtlich und fehleranfällig. Gleichermaßen ist auch die Berechnung des Basispreises nur über eine manuelle Kalkulation möglich und muss nach jeder Preisaktualisierung neu berechnet werden. Auch die Umwandlung von Netto- und Brutto-Preisen muss stets in der Kalkulation berücksichtigt werden. Je nach Ansicht variiert die Darstellung sowie die Formatierung der Preise.

4.2.2 Integration

Das Softwaresystem besitzt bislang keine Integration zu angrenzenden Systemen. XECURIS bezieht beispielsweise aktuelle Distributorenpreise aus dem EGIS-Portal. Da das Unternehmen eine hohe Anzahl an Produkten verarbeitet, ist bereits die manuelle grundlegende Aktualisierung der Artikel-Einkaufspreise mit einem hohen Aufwand verbunden. Sucheingaben müssen manuell getätigt und aus den Lieferanten-Einkaufspreisen ein einheitlicher Einkaufspreis gewonnen werden. Dieser Preis muss schließlich über den Bearbeiten-Dialog eines Artikels abgespeichert werden. Da von EGIS eine

zentrale Schnittstelle angeboten wird und somit die Möglichkeit zu einer Automatisierung besteht, ist dieser Gesichtspunkt als großes Defizit des Softwaresystems anzusehen.

Auch eine direkte Anbindung an den justSelling Produkt Konfigurator ist im Softwaresystem nicht implementiert. Angelegte Strukturen müssen manuell über eine Exportfunktionalität als ZIP-Datei heruntergeladen und anschließend im Produkt Konfigurator importiert werden. Besonders in Testphasen müssen viele Exports erstellt und manuell importiert werden. Dieser Workflow ist zeitintensiv. Der Basispreis, welcher nicht Teil der Struktur des justSelling Produkt Konfigurators ist, wird nicht über die Import-Funktionalität übernommen und bedarf einer manuellen Eintragung.

Weiterhin können auch Bestellungen nicht wieder zurück in das Softwaresystem übernommen werden. Da im Magento Onlineshop lediglich die Aufpreise eines Optionswertes gespeichert sind, lässt sich nicht zurückverfolgen, welche Artikel der aktuellen Bestellung zugrunde liegen oder für die zu dem Zeitpunkt aktuelle Kalkulation genutzt wurden. Bei der Bestellung ist es möglich, dass bereits eine Änderung der Vorlage im Softwaresystem durchgeführt wurde oder Komponenten der Vorlage bereits nicht mehr existieren. Preise werden bei einer Änderung überschrieben, ohne dass diese für spätere Überprüfungen noch einsehbar sind. Dies gilt sowohl für eingetragene Verkaufspreise an den Optionswerten als auch für Einkaufspreise, welche sich an der Artikelzuordnung befinden.

4.2.3 Laden der Dialoge

Die Auswahl der Ansichten des Softwaresystems bleibt nicht erhalten. Wenn der Browser beispielsweise geschlossen wurde, kann nach Aufruf derselben Internetadresse nicht die alte Auswahl wiederhergestellt werden. Auch beim Kopieren der URL, beispielsweise durch Senden an einen anderen Nutzer, wird die bestehende Ansicht nicht gespeichert.

Die Ansichten basieren auf Formularen. Sobald ein Formular abgeschickt ist, wird nach Antwort des Servers ein Neuladen der aktuellen Ansicht vorausgesetzt. Das erneute Laden einer Webseite unterbricht den Nutzer in seinem Arbeitsablauf und führt gegebenenfalls zu Fehlern. Die wiederholte Auswahl derselben Komponente in der Ansicht ist mit einer hohen Redundanz ineffizienter Arbeitsschritte verbunden, da immer erneut die vorher selektierten Elemente ausgewählt werden müssen. Auch die Übersichtlichkeit geht im Bearbeitungsvorgang verloren.

4.3 Konzeption

Ziel der Umsetzung ist es, den Workflow des Softwaresystems so zu gestalten, dass Workflows für Unternehmensmitarbeiter bedeutend effizienter als im Produkt Konfigurator nutzbar sind. Funktionalität und Nutzerfreundlichkeit stehen im Vordergrund der Entwicklung. Da es sich um ein unternehmensinternes Tool handelt und die Anzahl an administrierenden Mitarbeitern, welche die Software nutzen und das System belasten, überschaubar ist, liegt das Augenmerk weniger auf Effektivität, sondern mehr auf Effizienz während des Produktverwaltungsprozesses sowie der Bestellverarbeitung.

4.3.1 Preiskalkulation

Einen zentralen Ansatz der Optimierung stellt die Preisberechnung dar. Die Kalkulation im Softwaresystem erfolgt halbautomatisiert. Der Nutzer erhält die Möglichkeit, Preise anhand von Parametern zu beeinflussen. Die Aktualisierung erfolgt durch Kalkulationen in einem festen Ablauf. Zunächst werden die Einkaufspreise bei den verfügbaren Distributoren für die in der Datenbank vorhandenen Produktartikel abgerufen (siehe Abbildung 4.3). Hierzu soll die bereitgestellte Schnittstelle von EGIS verwendet werden. Aus den so gewonnenen Lieferantenpreisen erfolgt durch ein vordefiniertes Kalkulationsschema die Berechnung eines einheitlichen Artikel-Einkaufspreises. Gegebenenfalls auftretende Fehler bei der Aktualisierung werden als E-Mail an den administrierenden Mitarbeiter des Unternehmens gesendet. Neben dem betroffenen Artikel ist in der Nachricht ein Link zu diesem im Softwaresystem vorhanden, welcher die Artikelübersicht mit der entsprechenden Auswahl aufruft. Diese Aktualisierung findet in einem festgelegten Zeitintervall statt oder wird nach Bedarf manuell über einen Klick auf die entsprechende Schaltfläche ausgeführt.

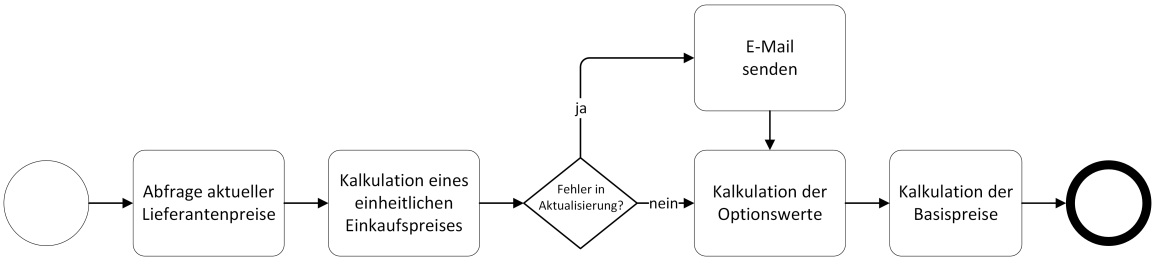

Abbildung 4.3: Preiskalkulation im Softwaresystem

Für Vorlagen, welche die aktualisierten Artikel beinhalten, erfolgt automatisiert eine neue Aufpreiskalkulation für die notwendigen Optionswerte, um für das gesamte Softwaresystem aktuelle Preise sicherzustellen. Für die Kalkulation an den Optionswerten werden Margen verwendet, welche am zugrunde liegenden Artikel festgelegt sind. Der Verkaufspreis des Optionswertes richtet sich an einer eingetragenen Marge am Artikelpreis aus. Der Nutzer hat die Möglichkeit, bei der Kalkulation Aufpreise runden zu las-

sen und kann gegebenenfalls Artikelmargen mit einem individuellen Aufpreis versehen. Nach Änderung eines Parameters wird der Preis ohne erneutes Laden der Ansicht angezeigt und erst nach Bestätigung in der Datenbank gespeichert. Weiterhin soll ersichtlich werden, welcher Bezugswert für die aktuelle Kalkulation als Grundlage verwendet wird. In der Oberfläche des Softwaresystems ist über eine entsprechende Schaltfläche die Darstellung zwischen Netto- und Brutto-Preisen wechselbar.

Die Berechnung des Basispreises erfolgt in einer separaten Ansicht. Grundlage hierfür bilden wie bei der Aufpreiskalkulation die festgelegten Margen der Artikel. Eine Auflistung aller Standardwerte der Vorlage macht die Kalkulation des Basispreises nachvollziehbarer. Anhand der für jeden Standardwert kalkulierten Brutto-Verkaufspreise wird eine Summe gebildet und als empfohlener Basispreis zur Anzeige gebracht. Der Nutzer bestimmt den eingetragenen Basispreis und wird optisch darauf aufmerksam gemacht, ob der so eingetragene Wert der angestrebten Zielmarge entspricht.

Bei der gesamten Preiskalkulation wird auf die aufwendige und fehleranfällige Berechnung über das Excel-Dokument verzichtet. Nähere Anforderungen an die Preiskalkulationen können dem Abschnitt 3.4.2 entnommen werden.

4.3.2 Export ins Live-System

Der Nutzer erhält neben der bisherigen Möglichkeit, die Vorlage als ZIP-Datei herunterzuladen die Wahl, eine Vorlage direkt ins Livesystem des Onlineshops zu überführen. Durch Betätigen der entsprechenden Schaltfläche wird ein neuer Export angelegt und in den justSelling Produkt Konfigurator importiert. Auch der Basispreis, welcher an der Vorlage hinterlegt ist, wird vollautomatisch in das System eingepflegt.

Der Export wird im Softwaresystem dokumentiert und bleibt für zukünftige Überprüfungen nachvollziehbar. Neben dem Zeitpunkt des Exports wird auch dokumentiert, ob die Vorlage direkt ins Livesystem überspielt wurde, der Export erfolgreich war und im Falle eines Fehlschlags eine mögliche Fehlerbeschreibung. Der administrierende Mitarbeiter gibt ein Feedback, ob der Export erfolgreich war und ob die Daten vollständig überspielt wurden.

4.3.3 Bestellüberprüfung

Im Softwaresystem soll es möglich sein, Preise einer Bestellung zu überprüfen. Bestellinformationen lassen sich von Magento zurück in das Softwaresystem überführen. Anhand dieser Informationen werden die Artikel zur Anzeige gebracht, welche Inhalt der Bestellung sind. Neben allgemeinen Artikelinformationen wird ein Ein- und Verkaufspreis ermittelt. Die berechneten Preisdaten orientieren sich nicht an aktuellen Werten, sondern an denjenigen des Bestellzeitpunkts. Eine Vorlagenstruktur sowie Informationen der Komponenten dürfen somit nach einem Export wieder verändert werden.

Die Übersicht bietet außerdem zusätzliche Funktionalitäten. Beispielsweise können Zeilen über Checkboxen markiert werden. Die Teilsumme aller markierten Netto-Verkaufspreise wird dem Nutzer in passender Weise zur Anzeige gebracht. Für jeden Listenpunkt ist ein Button zum Kopieren des Artikels vorhanden. Durch einen Mausklick auf die Schaltfläche wird diese Zeile markiert und die Artikelnummer in die Zwischenablage des Betriebssystems geschrieben.

4.3.4 Laden der Dialoge

Anlegen-, Bearbeiten- und Löschen-Dialoge benötigen keine neuen separaten Ansichten, sondern sind in eine Hauptansicht integriert und werden dem Nutzer in passender Form, beispielsweise über Popup-Fenster, dargestellt. Im Softwaresystem wird es möglich sein, Dialoge nach einem Speichervorgang nicht erneut laden zu müssen.

Die alte Auswahl einer Ansicht bleibt erhalten und wird mit den veränderten Daten lediglich aktualisiert. Über die URL bleibt die Auswahl der aktuellen Ansicht erhalten. Auf diese Weise wird es möglich sein, den Browser zu schließen und bei erneutem Öffnen an derselben Stelle weiterzuarbeiten. Die Internetadresse ist unabhängig vom Endgerät aufrufbar. So können Weiterleitungen definiert werden, um eine vordefinierte Auswahl bereits während des Ladevorgangs der Internetseite zu treffen.

5 Methoden

Dieser Teil der Arbeit erläutert ausgewählte Methoden der Implementation auf Grundlage der in Kapitel 4 erarbeiteten Anforderungen.

5.1 Preiskalkulation

Die Preisberechnung stellt eines der umfangreichsten Optimierungsansätze des Softwaresystems dar, welche in dieser Arbeit erläutert werden. Die Implementation ist in eine Folge von Teilschritten gegliedert, die sequentiell abgearbeitet werden.

5.1.1 Abfrage der aktuellen Lieferanten-Artikelpreise

Nachfolgend wird die Integration der aktuellen Einkaufspreise in das Softwaresystem erläutert. Abbildung 5.1 zeigt den schematischen Ablauf, um die in der Produktdatenbank enthaltenen Preise zu aktualisieren:

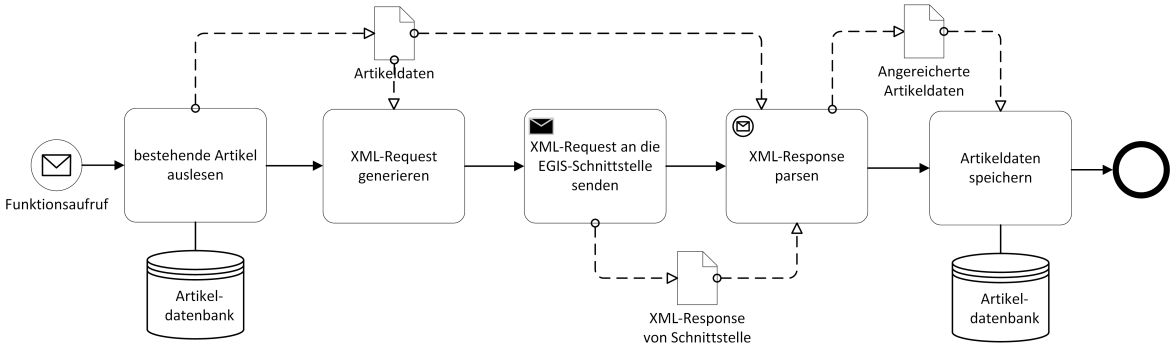

Abbildung 5.1: Schema zur Aktualisierung der Preise über die EGIS-Schnittstelle

Anhand der Artikel wird ein XML-Request generiert, welcher für die Authentifizierung des Nutzers sowie eindeutige Zuordnung der Artikel dient. Wie in Abschnitt 3.3.3 erwähnt wurde, stellt EGIS den *EGIS Business Connector* (EBC) als Schnittstelle bereit, um detaillierte Abfragen aus dem System zu starten. Um die Schnittstelle zu nutzen, muss der XML-Request über einen HTTP-Post an die richtige Schnittstellenfunktion gesendet werden. Die URL ist nach dem folgenden Schema aufgebaut:

```
http://www.egis-online.de/cgi-bin/WebObjects/EBC.woa/wa/
Komponente/Funktion
```

Komponente und Funktion stellen jeweils Platzhalter dar. Jede der angebotenen Komponenten umfasst Funktionen eigener Teilbereiche. Beispielsweise betreffen Funktio-

nen der Komponente *Artikelstamm* alle möglichen Abfragen über Artikelstammdaten. Im Bereich *OrderManagement* hingegen können Bestellungen verarbeitet werden. Die Komponente *ShoppingCart* erlaubt es schließlich, eigene Warenkörbe mit in EGIS bestehenden Lieferantenartikeln zusammenzustellen (siehe Tabelle 5.1).[52]

Komponente	**Inhalt**
Artikelstamm	Artikelstammdaten
OrderManagement	Bestellungen
ShoppingCart	Warenkörbe

Tabelle 5.1: Komponenten der EBC-Schnittstelle

Innerhalb dieser Arbeit wurden Funktionen der Komponente Artikelstamm verwendet. Diese bieten ein umfangreiches Angebot zur Abfrage von Artikelstammdaten. Tabelle 5.2 gibt eine Übersicht über die bereitgestellten Funktionen sowie über ihren Umfang:

Funktionsname	**Beschreibung**
searchQuery	Volltextsuche nach Artikeln
bestpriceQuery	Suche nach günstigstem Lieferantenpreis
priceAndAvailabilityQuery	Preis- und Bestandsabfrage aller Lieferantenartikel
onlineCheckQuery	Preis- und Bestandsabfrage bei einem Lieferanten
productSpecificationQuery	Abruf von Artikel-Informationen
distributorListQuery	Auflistung von Lieferanten

Tabelle 5.2: Funktionen der Komponente Artikelstamm

Die Funktion *priceAndAvailabilityQuery* ermöglicht es, für übergebene Produkte aktuelle Verfügbarkeiten sowie Preise bei Distributoren abzufragen. Die von der Schnittstelle zurückgegebenen Preise und Warenbestände können weiterverarbeitet werden zu einem einheitlichen Einkaufspreis.[53]

Das XML-Dokument, welches für eine Preisabfrage übermittelt werden muss, wird mit Informationen angereichert. Welche Tags und Attribute das Dokument für eine valide Struktur besitzen muss, variiert zwischen den bereitgestellten Funktionen der Schnittstelle. Das übermittelte XML-Dokument muss immer einen *TransactionHeader* besitzen, welcher allgemeine Informationen beinhaltet wie die Version der angesprochenen Schnittstelle, den Zeitpunkt der Erstellung des XML-Dokuments und die erforderlichen Login-Daten (siehe Listing B.1, Zeile 2-7). Um eine valide Struktur sicherzustellen, wird das Dokument mit einem XML-Schema abgeglichen. Dieses steht neben einem funktionsfähigen XML-Dokument als Beispiel für jede Schnittstellenfunktion in der Dokumentation bereit.[54]

[52] vgl. [Synaxon AG 2015a, S. 3]
[53] vgl. [Synaxon AG 2015a, S. 3]
[54] vgl. [Synaxon AG 2015b, S. 2 ff.]

In EGIS lässt sich jedes Produkt anhand einer internen Nummer, der sogenannten *Proprietary Product Number*, kurz PCS-Nummer, identifizieren. Diese wird in der Artikeldatenbank des Softwaresystems für jeden Artikel abgespeichert und ist bereits in der gegebenen Datenbankstruktur enthalten (siehe Abbildung 4.2). Um eine Preisabfrage zu starten, müssen zunächst bestehende Artikel aus der Datenbank ausgelesen werden. Jeder bestehende Produkteintrag erzeugt in dem zu übersendenden XML-Dokument einen weiteren Tag mit der zugeordneten PCS-Nummer als Wert (siehe Listing B.1, Zeile 11-14). Produkte ohne diesen Eintrag werden von der Preisabfrage ausgeschlossen. Das generierte XML-Dokument wird über einen HTTP-Post an die Schnittstelle gesendet. Der HTTP-Response des Servers enthält die angereicherten Produktdaten als XML-Dokument (siehe Listing B.2). Dieses muss geparst werden, um den Datenbestand mit aktuellen Daten zu befüllen. Im *Header* befinden sich allgemeine Informationen wie die Produktbezeichnung, die PCS-Nummer, die Herstellernummer sowie die internationale Artikelnummer. Im *Body* hingegen sind alle Daten zu finden, welche EGIS von den Lieferanten bezieht. In diesem Teil sind alle aktuellen Einkaufspreise und verfügbaren Warenbestände aufgelistet, gruppiert nach Distributoren. Zusätzlich existiert für jeden Distributor ein Tag *End of Life* (EOL). Falls der Wert des Tags auf *true* gesetzt ist, so wird er langfristig nicht mehr von dem Distributor angeboten (siehe Listing B.2, Zeile 37).

5.1.2 Ermittlung eines einheitlichen Einkaufspreises

Aus den gegebenen Daten der Distributoren muss zunächst ein Einkaufspreis ermittelt werden. Für jeden Artikel, welcher über den HTTP-Request an den EBC gesendet wurde, erfolgt eine Filterung nach vorhandenen Lieferantenpreisen. Diese werden als Liste an ein eigenes View-Model gebunden (siehe Listing 5.1).

```
1  foreach (Article dbArticle in dbArticles)
2  {
3    var distriArticles = allDistriArticles
4      .Where(x => x.ArticleId == dbArticle.Id).ToList();
5    var calculatedDistris = PrepareDistributorList(
6      dbArticle, distriArticles, distriList);
7
8    var priceModel = new ArticleDistriPricingViewModel(
9      dbArticle, calculatedDistris)
10
11   dbArticle.LastPriceCheck = DateTime.Now;
12   dbArticle.Price = priceModel.CalculatedPrice;
13   dbArticle.AmountAvailable = priceModel.CalculatedAmount;
14   db.Entry(dbArticle).State = EntityState.Modified;
15 }
16
17 await db.SaveChangesAsync();
```

Listing 5.1: Aktualisierung des einheitlichen Einkaufspreises

Das nun instanziierte View-Model enthält sowohl die kalkulierte Menge bei den Lieferanten als auch einen einheitlichen kalkulierten Einkaufspreis. Als Grundlage für diesen werden die fünf Distributoren mit dem geringsten Preis verwendet. Für Artikel mit einem geringen kumulierten Bestand von unter 100 Stück werden die fünf Distributoren herangezogen, welche dem Median aus Preissicht am Nächsten liegen. Diese Distributoren werden in die Liste *DistrisForCalculation* geschrieben. Mithilfe dieser Variable wird schließlich der Einkaufspreis kalkuliert. Hierbei wird ein gewichteter Mittelwert aus Preis und verfügbarer Anzahl des Artikels gebildet (siehe Listing 5.2). Dem kalkulierten Einkaufspreis wird zusätzlich ein Aufschlag von 0,5 Prozent aufaddiert und auf 0,5 bzw. 1,0 Euro aufgerundet:

```
1  public double CalculatedPrice
2  {
3    get
4    {
5      var middlePrice = DistrisForCalculation.Sum(x =>
6        x.Price * x.Quantity)
7        / DistrisForCalculation.Sum(x => x.Quantity);
8      // inkl. 0,5% genereller Aufschlag
9      middlePrice += middlePrice * 0.005;
10     // Aufrunden auf 0,5 bzw. 1,0
11     return Math.Round(Math.Ceiling(middlePrice * 2)
12       / 2, 2);
13   }
14 }
```

Listing 5.2: Aktualisierung des einheitlichen Einkaufspreises

5.1.3 Benachrichtigung über E-Mail

Bei der Aktualisierung der Einkaufspreise aus dem EGIS-System können Fehler auftreten, welche an den administrierenden Mitarbeiter von XECURIS gesendet werden sollen. Eine denkbare Fehlerquelle bei der Aktualisierung ist eine unterbrochene Verbindung zur Schnittstelle. Des Weiteren muss die übergebene PCS-Nummer nicht im System von EGIS existieren. In diesem Fall liefert die Schnittstelle keine Daten zu dem betroffenen Produktartikel. Gleichermaßen gibt es auch Fälle, die nicht auf Fehler beim Abruf der Daten zurückzuführen sind. Diese werden in der Preiskalkulations-Klasse berücksichtigt. Hierfür existiert die boolesche Variable *IsWarning*. Der Wert der Variablen besitzt den Wahrheitswert *true*, falls der aktuelle Einkaufspreis nicht berechnet werden konnte oder in Zukunft Probleme bereiten wird:

```
1  public bool IsWarning
2  {
3    get
4    {
5      return CalculatedAmount == 0
6        || Distributors.Count == 0
7        || Distributors.All(x => x.IsEol
8          || x.AvailableQuantity == 0);
9    }
10 }
```

Listing 5.3: Die Variable IsWarning

Die Klasse berücksichtigt drei Fehlerquellen: Zum einen ist es möglich, dass bei allen verfügbaren Distributoren kein Bestand mehr vorhanden ist. Der zweite Fall tritt ein, falls keine Distributoren für den angegebenen Artikel gefunden werden konnten. Im dritten Fall sind zwar Distributoren für das Produkt vorhanden, bei diesen ist aber das Produkt als EOL markiert oder sie führen keinen Bestand. Auf diese Weise können die Artikel, deren Aktualisierung fehlschlägt, identifiziert werden. Daraufhin kann ein Nachrichtentext erstellt werden, welcher die Daten der Artikel sowie einen Link zum Produkt beinhaltet.

Anschließend kann die Übermittlung der Nachricht an den administrierenden Mitarbeiter des Unternehmens erfolgen, der daraufhin prüfen muss, ob ein Artikel geändert oder ersetzt werden muss. Listing B.7 zeigt den Quellcode, um eine E-Mail über einen Microsoft Exchange Server zu versenden. Zunächst wird ein Service instanziiert, welcher den Zugang zum Mail-Server auf Grundlage eines hinterlegten Nutzernamens, des dazugehörigen Passworts sowie der Server-Adresse realisiert. Anschließend wird für den Service eine E-Mail-Nachricht erstellt, welchem die Adressaten, der Betreff sowie der Nachrichtentext zugewiesen wird. Abschließend wird die E-Mail versendet und eine Kopie auf dem Exchange-Server hinterlegt.

5.1.4 Aktualisierung der Aufpreise

Bei der Ermittlung der Artikel-Einkaufspreise können, wie in Abschnitt 5.1.2 beschrieben, über die Variable *IsWarning* neben den fehlgeschlagenen Artikeln auch die erfolgreich aktualisierten Artikel identifiziert werden. Ausgehend von diesen muss nun eine Aktualisierung der Vorlagenpreise stattfinden. Die Aufpreise eines Optionswertes sind von dem Standardwert einer Vorlage abhängig. Die Änderung eines Artikels, welcher diesem Optionswert zugeordnet ist, zieht es mit sich, die Preise der gesamten Option in der Vorlage zu aktualisieren. Zunächst müssen über die Referenzen alle nötigen IDs der Optionen identifiziert werden (siehe Listing B.3).

Auf Grundlage der IDs wird eine weitere Methode aufgerufen, welche die Aktualisierung einer gesamten Option vornimmt (siehe Listing B.8). Für jeden Optionswert wird eine individuelle Kalkulation angelegt. Diese erfolgt durch die Klasse *ListOptionValuePricingViewModel*. Inhalt der Klasse ist es zunächst, sowohl für den Standardwert des Optionswertes als auch für den Optionswert selbst einen Netto-Verkaufspreis zu kalkulieren:

```
1  public double GetVkNetto(OptionValue value)
2  {
3    double vkNetto = 0.00;
4    double profit = value.Profit;
5    //Fall: dynamischer Verkaufspreis,
6    //Aufschlag muss aufaddiert werden
7    if(!value.UseRecommendedPrice)
8    {
9      //Marge: Optionswert
10     profit += value.ExtraCharge;
11     //Marge: Option oder Optionsgruppe (optional)
12     profit += value.Option.ExtraCharge
13       ?? value.Option.Group.ExtraCharge ?? 0.00;
14   }
15   foreach (OptionValueArticle ova in optionValue.
       OptionValueArticles)
16   {
17     vkNetto += ova.Quantity * ova.Article.Price
18       * (profit + 1);
19   }
20   return vkNetto;
21 }
```

Listing 5.4: Berechnung des Netto-Verkaufspreises eines Optionswertes in der Klasse ListOptionValuePricingViewModel

Die Berechnung des Verkaufspreis eines Optionswertes geschieht auf Grundlage der aktualisierten Einkaufspreise der Artikel. Jedem Artikel kann eine individuelle Marge zugeordnet werden (siehe Abbildung A.4). Beim Anlegen eines Artikels wird zunächst eine Standardmarge übernommen, welche an der Produktkategorie zugeordnet wird.

Am Optionswert lässt sich konfigurieren, ob der empfohlene oder dynamische Verkaufspreis verwendet werden soll. Der empfohlene Verkaufspreis beruht lediglich auf der festgelegten Artikel-Marge. Ein dynamischer Verkaufspreis wird durch Auf- bzw. Abschläge am Optionswert sowie der Option bzw. der Optionsgruppe ermittelt. Die Aufschläge der Option und der Optionsgruppe können optional gesetzt sein. Von beiden findet maximal ein Aufschlag Anwendung, wobei die Option eine höhere Priorität besitzt (siehe Listing 5.4, Zeile 11-12).

```
1  double vkNetto = GetVkNetto(optionValue);
2  double vkNettoReference = GetVkNetto(optionValue
3    .Option.StandardValue);
4  double vkNettoResult = vkNettoReference - vkNetto;
5
6  if(optionValue.RoundPrice)
7  {
8    double vkBrutto = vkNettoResult * 1.19;
9    vkNettoResult = Math.Ceil(vkBrutto) / 1.19;
10 }
```

Listing 5.5: Berechnung des Aufpreises eines Optionswertes in der Klasse ListOptionValuePricingViewModel

Der Netto-Aufpreis wird schließlich durch die Differenz zwischen dem Netto Verkaufspreis des Standardwertes und des Optionswertes selbst berechnet. Zusätzlich kann die Rundung für den Optionswert aktiviert sein. In Diesem Fall wird der Aufpreis in einen Brutto-Betrag umgewandelt, aufgerundet und in einen Netto-Betrag zurückgerechnet (siehe Listing 5.5).

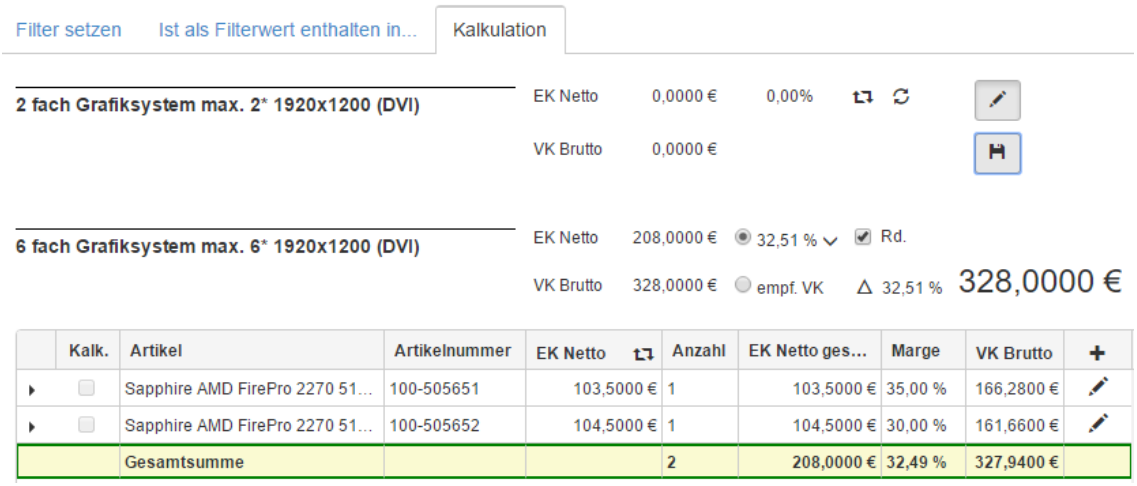

Abbildung 5.2: Kalkulationsansicht Im Vorlagen-Dialog

5.1.5 Berechnung des Basispreises

Auf Grundlage der Einkaufspreise und eingetragenen Margen bzw. Auf- und Abschläge kann ein empfohlener Basispreis ermittelt werden. Der Preis wird im Bearbeiten-Dialog der Vorlage eingetragen:

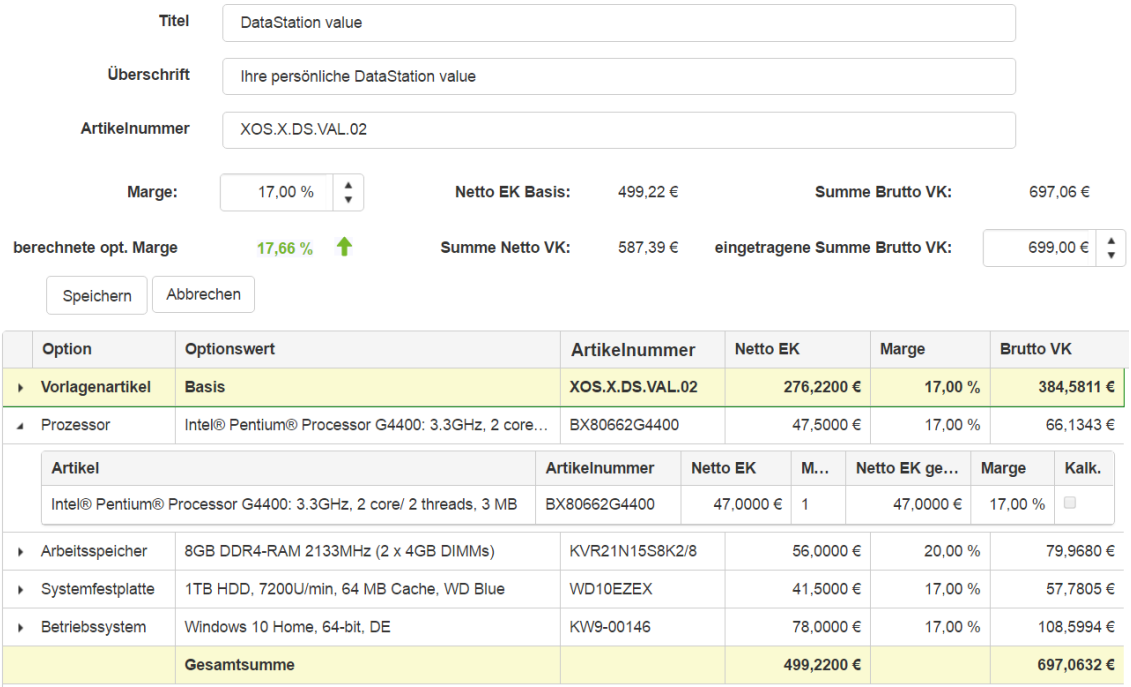

Abbildung 5.3: Ansicht zur Berechnung des Basispreises

In einem Grid im unteren Teil der Ansicht werden die Standardwerte einer Vorlage geladen. Neben den Standardwerten sind auch Vorlagen-Artikel, welche Bestandteil jeder Konfiguration sind, im Basispreis enthalten. Neben einer Auflistung des Netto-Einkaufspreises erfolgt auch die Darstellung des empfohlenen Brutto-Verkaufspreises. Die Grundlage für die Kalkulation bildet, ebenso wie in Abschnitt 5.1.4 für die Aufpreiskalkulation beschrieben wurde, die Margen bzw. Aufschläge an den zugrundeliegenden Artikeln, am Optionswert und der Option bzw. der Optionsgruppe. Die letzte Zeile des Grids stellt die Gesamtsumme der Werte dar. Über dem Grid befinden sich Felder zur Berechnung des Basispreises. Die Felder *Netto EK Basis* und *Summe Brutto VK* werden nach dem Ladevorgang des Grids mit den ausgelesenen Daten der Gesamtsumme aktualisiert. Die *berechnete Summe Brutto VK* kann als empfohlener Basispreis angesehen werden.

Für die Manipulation der entsprechenden Felder lässt sich eine JavaScript-Funktion definieren, die über das DataBound-Event des Grids aufgerufen wird (siehe Listing 5.6). Diese Funktion greift zunächst auf die Datenquelle des Grids zu und filtert die Zeile der Gesamtsumme. Um den Datensatz der Gesamtsumme zu identifizieren, besitzt

das übertragene View-Model den booleschen Wert *IsSum*. Aus dem extrahierten Datenobjekt lassen sich die benötigten Werte auslesen und an den richtigen Stellen als formatierten Währungsstring einfügen:

```
var nettoEk = 0.00;

function dataBound() {
  var grid = $("#TemplateDefaultGrid").data().kendoGrid;
  var sumRow = $(grid.dataSource.data()).filter(function(){
    return this.IsSum === true;
  });

  nettoEk = $(sumRow).attr("NettoEkPrice");
  var bruttoVk = $(sumRow).attr("BruttoVkPrice");

  $(".netto-ek").html(kendo.toString(nettoEk, "c2"));
  $(".brutto-vk-recommended")
    .html(kendo.toString(bruttoVk, "c2"));

  updateCalculation();
}
```

Listing 5.6: Methode beim Auslösen des DataBound-Events im Bearbeiten-Dialog der Vorlage

Der administrierende Mitarbeiter hat nun die Möglichkeit, sich an diesem empfohlenen Verkaufspreis zu orientieren und einen eigenen Preis für die Grundkonfiguration einzutragen. Nach jeder Änderung erfolgt eine Aktualisierung der Felder *Summe Netto VK* und *berechnete opt. Marge*. Die berechnete optimierte Marge stellt die reale Marge dar, welche mit dem eingetragenen Basispreis erzielt wird (siehe Listing 5.7).

```
function updateCalculation() {
  var writtenVkBrutto = $("#SalePriceBrutto")
    .data("kendoNumericTextBox").value();
  var writtenVkNetto = writtenVkBrutto / 1.19;
  var optimizedProfit = writtenVkNetto / nettoEk - 1;
  var aimProfit = $("#Profit").data("kendoNumericTextBox")
    .value() / 100;
  $(".optimizedProfit").html(kendo
    .toString(optimizedProfit, "p2"));
}
```

Listing 5.7: optimierte Marge aktualisieren

Der Nutzer hat die Möglichkeit, neben dem Basispreis eine Zielmarge einzutragen, welche mit der optimierten Marge verglichen wird. Wenn die optimierte Marge unter der Zielmarge liegt, so wird der Wert rot angezeigt und ein nach unten gerichteter Pfeil erscheint hinter der Zahl. Andernfalls erhält der Wert eine grüne Färbung mit einem nach oben gerichteten Pfeil. Die Färbung wird durch die Zuweisung einer Klasse hervorgerufen, welche die Färbung mithilfe von *Cascading Style Sheets* (CSS) realisiert.

Die Pfeilsymbole werden mithilfe von Glyphicons dargestellt. Weiterführende Informationen können der nachfolgenden Quelle entnommen werden.[55] Beide Pfeilsymbole existieren parallel, wobei zu jeder Zeit ein Symbol ausgeblendet ist (siehe Listing B.9).

5.2 Export ins Live-System

Für einen automatisierten Export ins Live-System sind mehrere Schritte erforderlich (siehe Abbildung 5.4). Zunächst muss die zu verwendende Vorlage in das Format überführt werden, womit sie schließich über die Schnittstelle des justSelling Produkt Konfigurators wieder importiert werden kann. In diesem Fall handelt es sich um eine ZIP-Datei. Diese generierte Datei wird nun nicht mehr, wie es vorher der Fall war, als Download durch den administrierenden Mitarbeiter bereitgestellt. Die Datei wird bereits im Arbeitsspeicher des Systems gehalten und anschließend über eine *Secure Copy* (SCP) Verbindung an den Server des Shopsystems übertragen (siehe Listing B.5).

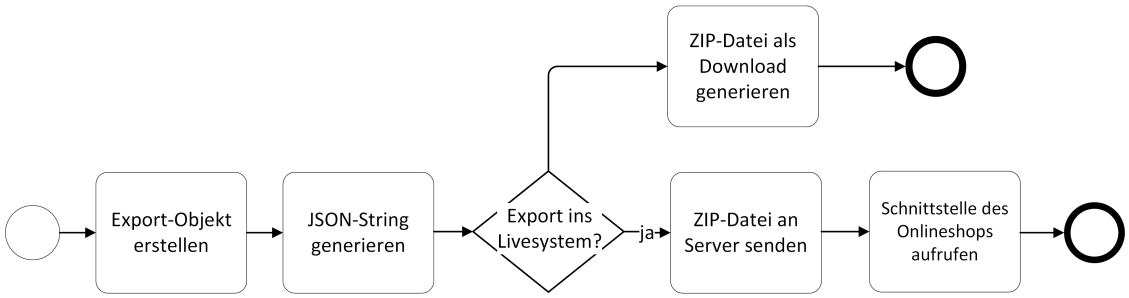

Abbildung 5.4: Schema zum Export einer Vorlage

Um den Import direkt durchzuführen, ohne die manuelle Upload-Funktionalität nutzen zu müssen, ist es notwendig, die auf PHP basierende Schnittstelle des justSelling Produkt Konfigurators so anzupassen, dass der Import der ZIP-Datei lediglich über einen Dateinamen erfolgt und gleichzeitig der Basispreis in Magento eingetragen wird. Nach der erfolgreichen Übertragung wird ein HTTP-Request an die Schnittstelle gesendet, welcher sowohl den Dateipfad der ZIP-Datei als auch den Basispreis übergibt.

5.3 Bestellüberprüfung

Da die manuelle Übertragung aller Einzelpositionen einer Bestellung vergleichsweise viel Zeit kosten würde, wurde im Bereich der Positionsübernahme im Magento Admin Panel ein zusätzliches Feld eingefügt, welches diejenigen IDs der Optionswerte und Vorlagen beinhaltet, die im Umfang der Bestellung enthalten sind. Diese sind identisch mit den IDs des Softwaresystems, da diese beim Import übernommen werden. Auf diese Weise ist es möglich, eine genaue Auflistung derjenigen Artikel zu geben, welche

[55] vgl. [Otto et al. 2016]

durch die ausgewählten Optionswerte benötigt wurden. Listing 5.8 zeigt exemplarisch den validen Code für eine Bestellung:

```
2016-10-13 12:47:29 $ 10302260 $ Versand-Selbstabholung:
0.0000 $ 36|1.0000#562.1849 ~ 6357#0.0000 ~ 6360#0.0000
~ 6365#0.0000 ~ 6366#0.0000 ~ 6373#133.6134 ~ 6550#0.0000
~ 6379#0.0000 ~ 6391#42.8571 ~ 6399#0.0000 ~ 6402#49.5798
~ 6404#32.7731 ~ 6406#0.0000 ~ 1409#0.0000 ~ 1411#0.0000
~ 1413#0.0000 ~ 1420#0.0000 ~ 1427#13.4453 ~ 1457#25.2100
~ 1464#0.0000 ~ 1460#0.0000 ~ 1469#251.2606 ~ 1475#0.0000
~ 1480#0.0000 ~ 1485#0.0000 ~ 1542#0.0000 ~ 1491#326.8907
~ 1529#0.0000 ~ 1532#0.0000 ~ 1545#0.0000
```

Listing 5.8: Beispiel eines generierten Bestellcodes mit einer Vorlage

5.3.1 Historisierung der Preise

Um Exports in der Datenbank abzuspeichern, muss die gegebene Model-Struktur um eine weitere Klasse erweitert werden (siehe Listing 5.9). Diese beinhaltet die Zuordnung zu einer bestimmten Vorlage, für welche ein Export durchgeführt wurde sowie den Zeitpunkt des Exports:

```
1  public class TemplateExport
2  {
3    public long Id { get; set; }
4    [ForeignKey("Template")]
5    public long TemplateId { get; set; }
6    [InverseProperty("Exports")]
7    public Template Template { get; set; }
8    public bool IsInLiveSystem { get; set; }
9    public DateTime TimeStamp { get; set; }
10   public string JsonPath { get; set; }
11   public string ExportPath { get; set; }
12   public bool IsCheckOkay { get; set; }
13   public string ErrorMessage { get; set; }
14 }
```

Listing 5.9: Die Klasse TemplateExport

Durch die ausgeführte Code First Migration wird eine zusätzliche Tabelle *TemplateExports* angelegt. Die Vorlagen- und Export-Tabelle ist durch eine 1:n-Beziehung miteinander verbunden (siehe Abbildung 5.5). Eine verlustfreie Speicherung der gesamten Struktur als Historisierung in der Datenbank hätte den Aufbau einer umfangreichen Parallelstruktur bedeutet. In Absprache mit dem Unternehmen ist die Entscheidung gefallen, nach jedem ausgeführten Export ins Livesystem für gespeicherte Dateien lediglich den Pfad in der Datenbankstruktur zu hinterlegen. Da die generierte ZIP-Datei lediglich die Daten im Format des Konfigurators enthält, sind aus diesen Daten keine Artikelpositionen mehr einsehbar. Folglich wird bei einem durchgeführten Export neben der

ZIP-Datei auch eine aus der Vorlage generierte JSON-Datei auf dem Dateisystem abgespeichert. Wenn eine Bestellüberprüfung ausgeführt wird, so muss diese Datei eingelesen und wieder in die Vorlagen-Objektstruktur überführt werden. Auf diese Weise ist die Abbildung aller zu diesem Zeitpunkt vorhandenen Komponenten möglich. Diese Lösung kostet mehr Ressourcen bei der Verarbeitung, da die komplette Vorlagenstruktur als Objekt eingelesen werden muss. Diese Lösung reduziert aber den Umfang der Datenbank sowohl in Hinsicht auf gespeicherte Datenmenge als auch der nötigen Komplexität. Da ein Export lediglich für die Bestellüberprüfung benötigt wird, kann der höhere Ressourcenverbrauch in diesem Anwendungsfall akzeptiert werden.

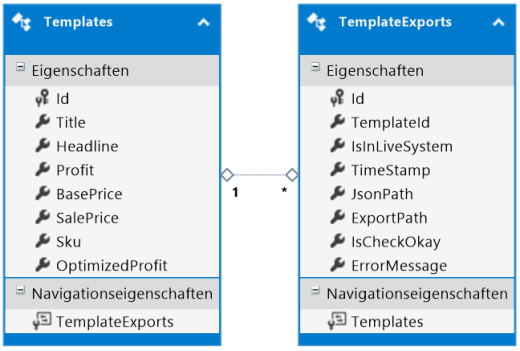

Abbildung 5.5: ER-Diagramm für die Datenbank-Struktur der Klasse TemplateExport

5.3.2 Auslesen des Bestelllinks

Ein generierter Bestelllink muss zunächst von der Positionsübernahme des Magento Onlineshops in das Softwaresystem überführt werden, um diesen verarbeiten zu können. Hierfür gibt es ein separates Feld, in welches der Link eingefügt werden kann (siehe Listing 5.8). Der Bestelllink ist strukturiert aufgebaut: Alle Informationen sind durch festgelegte Zeichen separiert (siehe Listing 5.10). Mithilfe von Split-Funktionen lassen sich die Informationen extrahieren und anschließend in das Zielformat parsen.

```
Bestelldatum $ Bestellnummer. $ Versandtitel:Versandkosten
 $ Vorlagen-Id|Menge#Basispreis ~ Optionswert#Aufpreis
 ~ Optionswert#Aufpreis ~ Optionswert#Aufpreis ~ [...]
 $ Vorlagen-Id|Menge#Basispreis ~ Optionswert#Aufpreis
 ~ Optionswert#Aufpreis ~ Optionswert#Aufpreis ~ [...]
```

Listing 5.10: Schema eines Bestellcodes im Onlineshop

Zunächst sind Informationen, wie der Zeitpunkt der Bestellung sowie die Versandkosten, dargestellt. Anschließend erfolgt eine Auflistung derjenigen Vorlagen, welche bestellt wurden. Bei einer Bestellung ist es möglich, verschiedene konfigurierbare Produkte in jeweils individueller Anzahl in einer Bestellung aufzugeben. Um eine eindeutige Zuordnung zu den gewählten Vorlagen und Optionswerten zu erhalten, sind die IDs aus dem Softwaresystem und dem Onlineshop identisch.

Wie aus Abschnitt 5.3.1 hervorgeht, wird die Historisierung einer Vorlage immer dann vorgenommen, wenn ein Export ins Livesystem geschieht. Der Zeitpunkt der Bestellung und die ID der Vorlage identifizieren eindeutig, welcher Export für die Bestellung Anwendung findet. Hierbei wird der Eintrag des Exports identifiziert, welcher vor dem angegebenen Datum für die Vorlage durchgeführt wurde (siehe Listing 5.11).

```
1  var lastExport = db.Exports.Include(x => x.Template)
2    .Where(x => x.TemplateId == templateId
3      && x.IsInLiveSystem)
4    .Where(x => DateTime.Compare(timeStamp, x.TimeStamp)
5      >= 0)
6    .OrderByDescending(x => x.TimeStamp)
7    .FirstOrDefault();
8
9  Template template;
10
11 //Export-Eintrag gefunden
12 if (lastExport != null)
13 {
14   //Vorlage aus JSON-Datei lesen
15   template = _DeserializeHistory(lastExport.JsonPath);
16   if (template == null)
17   {
18     ViewBag.Message = "Laden der Vorlage fehlgeschlagen";
19     return new HttpStatusResult(HttpStatus.BadRequest);
20   }
21   templates.Add(template);
22 }
```

Listing 5.11: Extraktion des letzten Exports

5.3.3 Berechnung der Verkaufspreise

Wie in Abschnitt 5.3.2 beschrieben wurde, bestehen nach dem Auslesevorgang des Bestelllinks nun alle Vorlagen-Objekte, welche für die Bestellüberprüfung benötigt werden. Die bestellten Optionswerte werden anhand ihrer IDs identifiziert. Diese dienen als Grundlage für die Auflistung aller Artikel der Bestellung. Neben den Artikeln der Optionswerte sind auch Vorlagenartikel Teil der Auflistung:

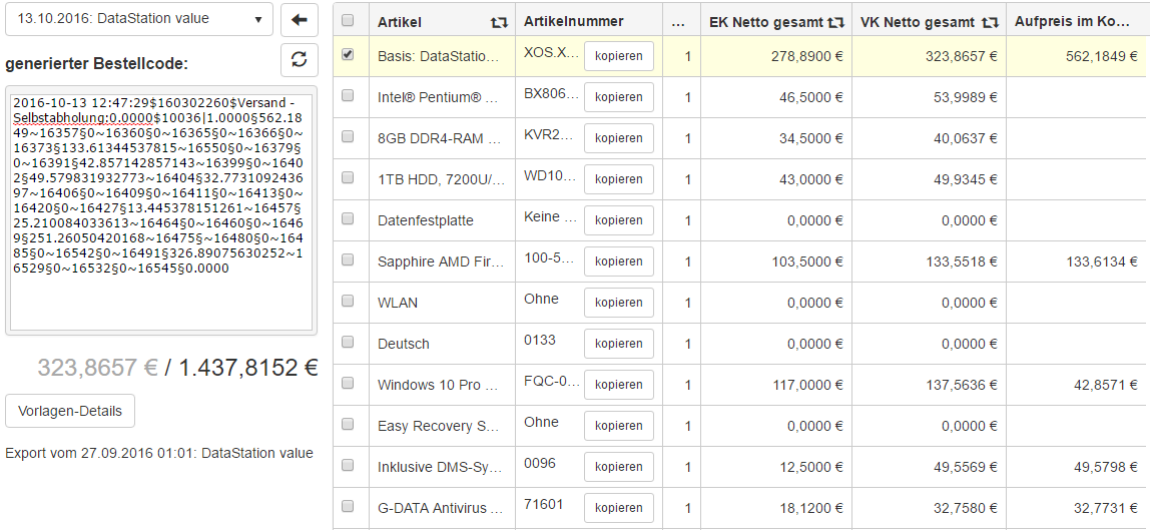

Abbildung 5.6: Die Ansicht der Bestellüberprüfung

Der Netto-Einkaufspreis kann direkt aus dem Artikel abgefragt werden. Die Berechnung des Verkaufspreises erfolgt auf Grundlage der Klasse, die auch für die Aufpreiskalkulation in Abschnitt 5.1.4 angewandt wird. Diese beinhaltet zusätzlich eine Methode, welche die Marge für den gesamten Optionswert berechnet. In Verbindung mit dem Einkaufspreis eines zugrunde liegenden Artikels kann schließlich der Netto-Verkaufspreis ermittelt werden:

```
1  public ListGetValuesByOrderViewModel(OptionValueArticle
2    ova, long templateQuantity)
3  {
4    //Menge = Menge des Artikels
5    //im Optionswert * bestellte Menge der Vorlage
6    Quantity = ova.Quantity * templateQuantity;
7    NettoEkPrice = ova.Article.Price * Quantity;
8
9    //Gesamtmarge des Optionswertes verwenden
10   var pricingModel = new OptionValuePricingViewModel(
11     ova.OptionValue);
12   NettoVkPrice = NettoEkPrice * (pricingModel
13     .ProfitResult + 1);
14 }
```

Listing 5.12: Berechnung des Verkaufspreises

5.3.4 Berechnung der Teilsumme und Kopieren der Artikelnummer

Jeder Checkbox, die Teil der Auflistung in dem Grid sind, wird die Klasse *order-check* zugeordnet. Zusätzlich besitzt das Element den Netto-Verkaufspreis als Attribut:

```
<input type="checkbox" class="order-check" price="55.0349"
  articleId="5"/>
```

Listing 5.13: Checkbox in der Bestellüberprüfung

Zunächst wird eine Funktion definiert, die bei einem Klick auf eine dieser Checkboxen ausgelöst wird. Wenn die Methode ausgeführt wird, lassen sich die gewählten Objekte auslesen und die Preise dynamisch abfragen. Der so ausgelesene Wert muss geparst und zu einer Summenvariable hinzugefügt werden. Anschließend wird das Feld der Teilsumme wieder mit der formatierten Summe als Währungsstring überschrieben:

```
$(document).on("click", ".order-check", function(){
  var sum = 0.00;
  $("input.order-check:checked").each(function(){
    var price = $(this).attr("price");
    sum += kendo.parseFloat(price, "en-US");
  });
  $(".checked-prices").html(kendo.toString(sum, "c4"));
})
```

Listing 5.14: Setzen der Summen-Variable in der Übersicht

Die Kopierfunktion des Grids funktioniert nach einem ähnlichen Prinzip. Jedem Button ist ein Artikelnummer-Attribut unter dem Namen *sku* und eine Klasse zugeordnet:

```
<button type="button" class="btn btn-default
  copy-to-clipboard" sku="55123">kopieren</button>
```

Listing 5.15: Kopier-Button in der Bestellübersicht

Auch für diesen Button wird wieder ein entsprechendes Event hinterlegt. Zunächst wird die Zeile markiert, indem von dem Button aus die überliegende Zeile gesucht und anschließend die Checkbox identifiziert wird. Ist diese noch nicht gesetzt, so wird die Methode ausgeführt, welche bereits für das Klick-Event hinterlegt ist.

Anschließend wird eine weitere Methode ausgeführt, welche die Artikelnummer in die Zwischenablage kopiert:

```
$(document).on("click", ".copy-to-clipboard", function() {
  var input = $(this).closest("tr")
    .find("input:not(:checked)");
  if(input.length) {
    $(input).click();
  }
  copyTextToClipboard($(this).attr("sku"));
});
```

Listing 5.16: Kopierfunktion der Bestellübersicht

Das Kopieren eines Textes funktioniert mithilfe von JavaScript nicht auf direktem Weg. Um den gewünschten Effekt zu erzielen, muss der aktuellen Ansicht der Internetseite ein Eingabefeld hinzugefügt werden. Mit diesem Element lässt sich ein Kopier-Befehl ausführen, wenn der genutzte Internetbrowser diese Funktionalität unterstützt. Nach der Ausführung der Funktion kann das TextArea wieder entfernt werden (siehe Listing 5.17)

```
function copyTextToClipboard(text) {
  var textArea = document.createElement("textarea");
  [...]
  document.body.appendChild(textArea);
  textArea.select();

  try {
    var successful = document.execCommand("copy");
    var msg = successful ? "kopieren erfolgreich"
      : "kopieren fehlgeschlagen";
  } catch (err) {
    console.log("kopieren fehlgeschlagen");
  }
  document.body.removeChild(textArea);
}
```

Listing 5.17: Methode zum Kopieren in die Zwischenablage

5.4 Laden der Dialoge

Nachfolgend wird der Ladevorgang der Webansichten beschrieben. Hierbei werden neben dem strukturellen Aufbau der Hauptübersichten die Verarbeitung der aktuellen Auswahl sowie die Editierdialoge thematisiert.

5.4.1 Struktureller Aufbau der Hauptübersichten

Im Softwaresystem bestehen zwei Ansichten, die den größten Funktionsumfang bereitstellen: Zum einen die allgemeine Artikelübersicht (siehe Abbildung A.2) und zum anderen die Übersicht einer spezifischen Vorlage (siehe Abbildung A.1). Jede dieser Übersichten besteht aus einer Hauptansicht, welche lediglich die Informationen beinhaltet, die nach jedem Editiervorgang erhalten bleiben. Alle weiteren Informationen werden über Teilansichten, sogenannte *PartialViews*, nachgeladen. In der Vorlagenansicht bilden Optionsgruppen, Optionen und Optionswerte jeweils getrennte PartialViews, da der Inhalt dieser Ansichten von der übergeordneten ausgewählten Komponente abhängig ist. Bei der Auswahl einer Optionsgruppe beispielsweise werden in der nächsten Ebene lediglich die Optionen zur Anzeige gebracht, welche auch dieser Optionsgruppe zugeordnet sind.

5.4.2 Laden der aktuellen Auswahl

Wenn die Übersicht geladen wird, so kann dem Controller zur Verarbeitung eine initiale Auswahl übergeben werden. Wenn der Aufruf über HTTP-Get erfolgt, so können diese direkt als Parameter über den URL-Link gesendet werden. Diese bilden die Grundlage für den Fortbestand der aktuellen Auswahl, auch wenn die Internetseite neu geladen oder durch eine Referenz auf die Auswahl verlinkt werden soll. Wird eine Seite über einen Hauptdialog geladen, so enthält diese zu Beginn noch keine Parameter. Erst durch Auswahl einer gewünschten Komponente erweitert bzw. verändert sich der Link um die nötigen Informationen. Im Beispiel der Vorlagen-Detailansicht handelt es sich bis auf die Vorlagen-ID um optionale Parameter als Nullable-Typen (siehe Listing 5.18).

```
1 public ActionResult TemplateOverview(long id,
2   long? groupId, long? optionId, long? valueId,
3   bool? showNetto)
4 {
5   return View(new DetailViewModel(id, groupId, optionId,
6     valueId, showNetto));
7 }
```

Listing 5.18: Die Controller-Methode für den Aufruf der Vorlagen-Detailansicht

Die übergebenen Werte werden an ein ViewModel gebunden und können in der Webansicht mithilfe von Razor-Code verwendet werden. Für diese Werte werden unsichtbare

Felder mit jeweils einer individuellen ID angelegt, welche sich in der Hauptansicht befinden. Diese Felder können verändert werden und dienen dem zentralen Zugriff für alle PartialViews. Um den Zugriff auf die Felder zu vereinfachen, wurde für jedes Feld eine individuelle Getter- und Setter-Methode definiert (siehe Listing 5.19).

```
@model Konfigurator.ViewModels.DetailViewModel

@Html.HiddenFor(x => x.TemplateId)
@Html.HiddenFor(x => x.GroupId)
@Html.HiddenFor(x => x.OptionId)
@Html.HiddenFor(x => x.ValueId)
@Html.HiddenFor(x => x.ShowNetto)

<script>
  //Beispielzugriff auf die aktuelle Auswahl
  function getGroupId() {
    return $("#GroupId").val();
  }
  function setGroupId(value) {
    $("#GroupId").val(value);
  }
</script>
```

Listing 5.19: Hidden-Felder mit der aktuellen Auswahl

Auf Grundlage dieser Felder kann die nächste untergeordnete Sicht nachgeladen werden. Listing B.11 zeigt exemplarisch den Code für das Nachladen der Optionen in Abhängigkeit der ausgewählten Optionsgruppe. Wenn die aktuelle Auswahl noch nicht gesetzt ist, so wird die erste mögliche Option gewählt und geladen. Nach jeder Änderung der aktuellen Auswahl wird zusätzlich eine Methode ausgeführt, die Parameter der URL auf Basis der Felder aktualisiert. Hierzu wird ein String generiert, welcher die benötigten Informationen enthält. Zu Beginn steht die Vorlagen-ID, die direkt als ID ohne Bezeichner an den Controller gesendet wird. Die übrigen Parameter sind durch ein *?*- bzw. ein *&*-Zeichen getrennt und enthalten jeweils einen Bezeichner sowie den dazugehörigen Wert, separiert durch ein =-Zeichen:

```
function updateLink() {
  var templateId = getTemplateId();
  var groupId = getGroupId();
  var optionId = getOptionId();
  var valueId = getValueId();
  var showNetto = getShowNetto();

  var urlString = templateId;
  urlString += "&groupId=" + groupId;
  urlString += "&optionId=" + optionId;
  urlString += "&valueId=" + valueId;
  urlString += "&showNetto=" + showNetto;
  window.history.pushState("", "", urlString);
}
```

Listing 5.20: Methode zum Aktualisieren der URL-Parameter

5.4.3 Editierdialoge

Editierdialoge können so gestaltet werden, dass diese bei Betätigung der entsprechenden Schaltfläche als Popup-Fenster erscheinen. Auf diese Weise sind die Dialoge deutlich von der Hauptansicht getrennt. Zur Gestaltung von Fenster-Dialogen wird von Kendo ein *Window* angeboten, das sich mithilfe von Razor-Code definieren lässt. Hierbei stehen umfangreiche vordefinierte Funktionalitäten zur Verfügung (siehe Listing B.10). Das Popup kann mithilfe von JavaScript aufgerufen werden. Hierfür wurde eine eigene Methode geschrieben:

```
1  function openDialog(windowName, url, data, type) {
2    if (!type) type = "GET";
3    var dialog = $("#" + windowName).data("kendoWindow");
4    dialog.refresh({
5       url: url,
6      data: data,
7      type: type
8    });
9    dialog.center();
10   dialog.open();
11 }
```

Listing 5.21: Methode für einen Popup-Aufruf

Diese Methode kann beispielsweise bei einem Klick auf einen Button ausgeführt werden. Listing 5.22 zeigt den Code, um den Editier-Dialog einer Vorlage aufzurufen. Bei dem Aufruf des Windows wird eine URL übergeben, welche beim Aufruf geladen wird. Hierbei handelt es sich um eine vom Controller zurückgegebene Teilansicht.

```
1 $(document).off("click", ".edit-template")
2   .on("click", ".edit-template", function () {
3     openDialog("EditTemplateWindow", "/Template/Edit",
4     { id: getTemplateId() });
5 });
```

Listing 5.22: Klick-Event für einen Popup-Aufruf

Wenn Editierdialoge gespeichert werden, aber die Ansicht nach Absenden der Daten nicht erneut geladen werden soll, so existieren zwei Möglichkeiten. Bei der ersten Möglichkeit bleibt die Formular-Gestaltung erhalten. Felder können über entsprechende Razor-Elemente definiert werden, welche bei Absenden der Daten vom Controller als Model interpretiert werden. Hierbei kann mithilfe einer Success- bzw. Fail-Methode bestimmt werden, welche Aktion nach einem Speichervorgang erfolgen soll. Formulare können aber nicht für jede Speicherung angewandt werden, da es diese Struktur lediglich erlaubt, statische Daten in Form von Feldern an den Server zu schicken.

Die zweite Variante besteht darin, dass Daten asynchron über AJAX abgesendet werden. Hierbei müssen jedoch die Daten, welche an den Server gesendet werden, aus

jedem Objekt manuell ausgelesen und dem AJAX-Aufruf übergeben werden. Diese Variante ist jedoch flexibler, da Daten dynamisch vorverarbeitet und verschiedenen Quellen entnommen werden können, ohne ein Eingabefeld mit der exakten Benennung hierfür berücksichtigen zu müssen. Diese Variante kommt unter anderem bei der Speicherung von Filtern zum Einsatz, da hierbei mehrere Checkboxen für einen bestimmten Index und einen bestimmten Optionswert dynamisch ausgelesen werden müssen (siehe Listing 5.23)

```
function onSave(e)
{
  var valueId = $(".filterCheckBox").attr("assigned-id");
  var index = $(".filterCheckBox").attr("index");
  var checkedIds = $(".filterCheckBox:checked")
    .map(function()
  {
    return $(this).attr("data-id");
  }).get();

  var dataRequest = { ValueId: valueId,
    CheckedIds: checkedIds, Index: index };

  $.ajax({
    data: JSON.stringify(dataRequest),
    type: "POST",
    contentType: "application/json; charset=utf-8",
    url: "/Template/FilterWrite",
    success: function() {
      loadFilterData(valueId, index, false);
      window.updateFilterInformation();
    }
  });
}
```

Listing 5.23: Filter abspeichern mithilfe von AJAX

6 Fazit

Innerhalb der Arbeit wurden auf Basis von vermittelten Grundlagen wichtige Aspekte des Produktverwaltungs- und Bestellverarbeitungsprozesses näher gebracht. Anhand der Ausgangssituation konnten ineffiziente Workflows im Arbeitsablauf von XECURIS aufgezeigt werden. Der Einsatz schlecht anpassbarer Insellösungen hat zu redundanten, fehleranfälligen und zeitintensiven Prozessabläufen geführt. Auf Grundlage der Mängel wurden Optimierungsansätze erarbeitet, deren Implementation schließlich in den Fokus gesetzt wurde. Die Arbeit hat dazu beigetragen, dass das Softwaresystem effektiv im Unternehmen eingesetzt wird. Die Automatisierung regelmäßiger, aufwendiger und strukturierter Abläufe wie die Preiskalkulation führt zu einer deutlichen Effizienzsteigerung der eingesetzten Workflows. Alle Verwaltungsprozesse für die hoch anpassbaren Produkte laufen über das Softwaresystem ab und sind auf die unternehmenseigenen Bedürfnisse abgestimmt.

Nach der Erstellung von Vorlagen besitzt der administrierende Mitarbeiter lediglich die Aufgabe, Artikel hinsichtlich ihrer Verfügbarkeit zu überprüfen und nach Bedarf einen Export in das Livesystem des Onlineshops durchzuführen. Der Basispreis sowie die Vorlage werden automatisiert in den Onlineshop importiert. Vorlagen müssen nicht mehr manuell heruntergeladen und über die vorgesehene Schaltfläche hochgeladen werden. Mithilfe des Kalkulationsablaufes wird nun auf eine aufwendige Excel-Tabellenkalkulation verzichtet. Die Oberfläche des justSelling Produkt Konfigurators machte es nötig, die gesamte Preiskalkulation manuell durchzuführen und händisch einzutragen. Eine Aktualisierung aller Produkte bedeutete einen Aufwand von mehreren Tagen. Gleichzeitig musste eine umfangreiche Nachbearbeitung der Optionswerte stattfinden, in denen Fehler beim Übertragen der Preise passiert sind. Stark fluktuierende Preise stellen mithilfe des Softwaresystems kein Problem dar, da eine vollständige Aktualisierung aller Vorlagen innerhalb von wenigen Minuten durchführbar ist.

Die bestehende Artikeldatenbank wird durch die Integration zur EGIS-Schnittstelle ständig um aktuelle Einkaufspreise angereichert. Preise werden regelmäßig durch einen externen Task oder nach Bedarf manuell aktualisiert. Auf Grundlage der verfügbaren Lieferantenpreise wird ein einheitlicher Einkaufspreis berechnet. Anschließend erfolgt eine Aufpreiskalkulation für zugeordnete Optionswerte. Wenn eine Preisaktualisierung fehlschlägt, so erhält der zuständige Mitarbeiter eine E-Mail. Folglich ist es nicht mehr notwendig, Verfügbarkeiten manuell über das EGIS-Portal zu überprüfen. Auch die Bestellverarbeitung wird mithilfe des Softwaresystems abgebildet. Der Nutzer erhält anhand eines Bestelllinks eine Auflistung aller Artikel, inklusive des Verkaufspreises zum Bestellzeitpunkt. Durch die Einführung einer Historisierung lassen sich Preise der Vergangenheit zurückverfolgen. Ohne das Softwaresystem mussten alle gewählten Optionswerte in die Excel-Tabelle übertragen und in Artikel aufgeschlüsselt werden. Eine Historisierung ist im Produkt Konfigurator nicht möglich.

Weiterhin bleiben die Produkte, welche über das Softwaresystem erstellt wurden, unabhängig von der Magento Infrastruktur. Sollte zukünftig auf ein anderes System umgestiegen werden, so ist eine Integration zu einer weiteren Schnittstelle möglich, welche den Export der Daten in das gewünschte Format vornimmt. Auch die Daten bleiben beliebig erweiterbar und können bei Bedarf um zusätzliche Strukturen erweitert werden.

7 Ausblick

Weiterführend ist angestrebt, eine Integration des eigen entwickelten Konfigurators zum X1-System von XECURIS zu durchzuführen. Das ganzheitlich eigen programmierte ERP-System verfügt über eine separate Artikeldatenbank, welche bereits einen Eintrag mit der zugehörigen PCS-Nummer besitzt. Eine Aktualisierung der dort vorhandenen Artikel-Einkaufspreise ist folglich denkbar. Auch eine Vorlagenverwaltung sowie eine Historisierung der Preise lässt sich in das System integrieren. Auch eine Verschmelzung beider Projekte zu einem Gesamtprojekt kann hierbei in Betracht gezogen werden. In dieser Hinsicht ist es möglich, eine Datenintegration durchzuführen, sodass eine gemeinsame Artikeldatenbank geführt wird. Auch auf Prozess- und Funktionsebene ist die Integration anwendbar.

Weiterhin können am Softwaresystem strukturelle Optimierungen vorgenommen werden. Beispielsweise sind bestehende Ansichten derzeit nur für hohe Displayauflösungen ausgelegt. Weiterführend kann ein Responsive Webdesign eingeführt werden, sodass jede Ansicht adäquat für jedes Gerät nutzbar ist. Hierbei ist zu entscheiden, welche Funktionalitäten auf welchen Endgeräten zur Verfügung stehen müssen, da eine Menge an Informationen zur Anzeige gebracht werden müssen. Bei einer adäquaten Implementation ist es Mitarbeitern auch ohne Nutzung eines Computers möglich, das Softwaresystem zu bedienen.

Neben EGIS existieren weitere Preisportale, welche aktuelle Lieferantenpreise liefern. Auch bei diesen Systemen ist es bei einer verfügbaren Schnittstelle möglich, aktuelle Referenzdaten in das Softwaresystem zu importieren. Das Kalkulationsschema zur Berechnung des einheitlichen Einkaufspreises kann weiterführend variiert werden, sodass dem Nutzer eine größere Flexibilität bei der Wahl des Einkaufspreises gewährt wird.

Anhang A: Screenshots

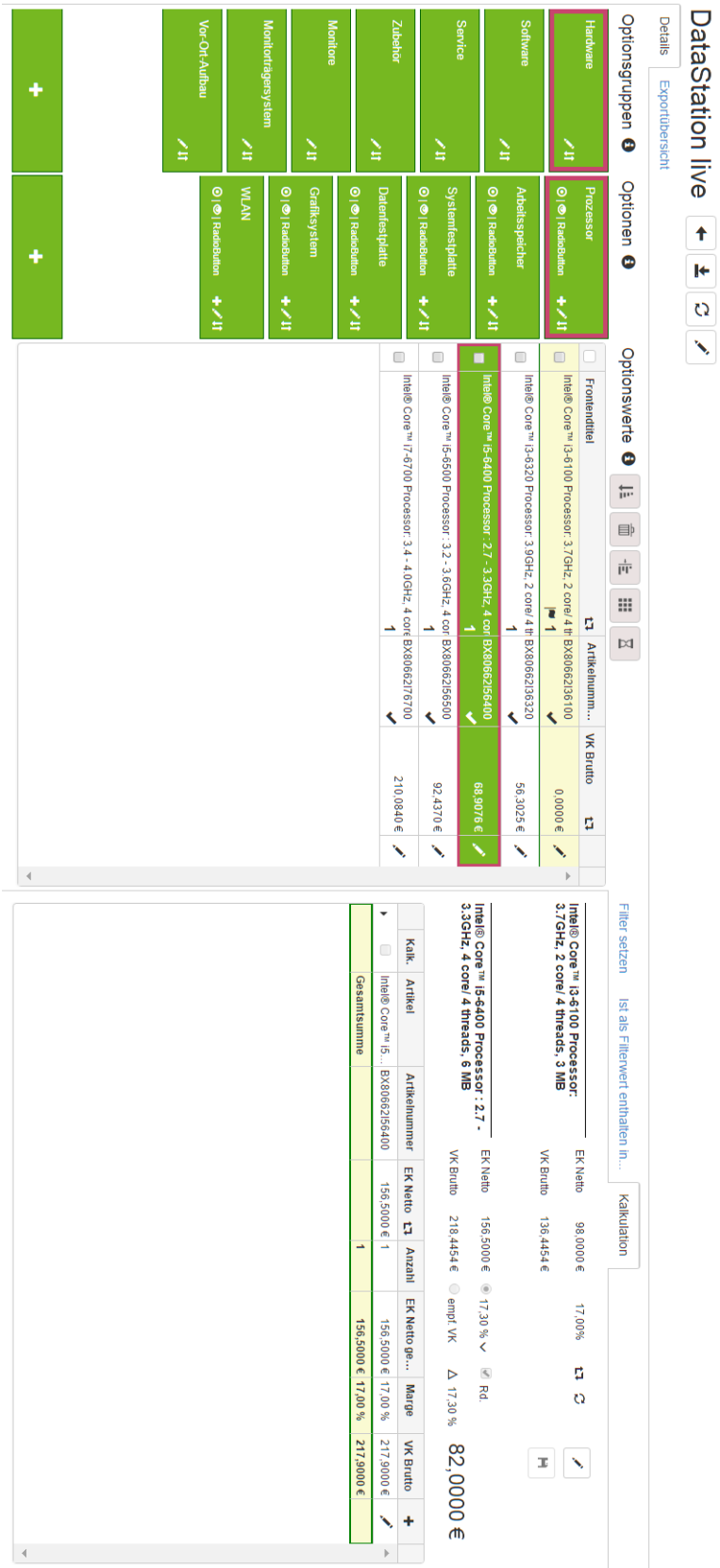

Abbildung A.1: Vorlagen-Details

Abbildung A.2: Artikel-Übersicht

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Gruppe	Art.Nr.	Bezeichnung	EK	17,00%	aktiv	Brutto				netto EK	netto VK	netto vk opt.	Auswahl	Summe	Gruppenpreis
2	Basis	XDS.X.DS.ACT.02	DataStation activ	486,92 €	✔	0,00 €	677,94 €	674,00 €	566,3866	-3,94 €	486,92 €	569,70 €	571,75 €	✔	571,75 €	571,75 €
3	CPU	BX80662i56400	Intel® Core™ i5-6400 Processor : 2.7 - 3.3GHz, 4 core(4 threads, 6 MB	153,00 €	✔	153,00 €	- €		0,0000	0,00 €	153,00 €	179,01 €	179,65 €		179,65 €	
4	CPU	BX80662i56600K	Intel® Core™ i5-6600K Processor: 3.5 - 3.9GHz, 4 core/ 4 threads, 6 MB	205,00 €	!	153,00 €	72,40 €	79,00 €	66,3866	6,60 €	0,00 €	0,00 €	0,00 €		0,00 €	
5	CPU	BX80662i76700	Intel® Core™ i7-6700 Processor: 3.4 - 4.0GHz, 4 core/ 8 threads, 8 MB	264,00 €	!	153,00 €	154,55 €	159,00 €	133,6134	4,45 €	0,00 €	0,00 €	0,00 €	✔	133,61 €	
6	CPU	BX80662i76700K	Intel® Core™ i7-6700K Processor: 4.0 - 4.2GHz, 4 core/ 8 threads, 8 MB	299,00 €	!	153,00 €	203,28 €	200,00 €	168,0672	-3,28 €	0,00 €	33,00 €	38,61 €		38,75 €	313,27 €
7	LowNoise	BK010	LowNoise II	33,00 €	✔	33,00 €	- €		0,0000	0,00 €	33,00 €	38,61 €	38,75 €	✔	38,75 €	
8	LowNoise	BK018 + 3x BL062	LowNoise III	97,00 €	!	33,00 €	89,11 €	89,00 €	74,7899	-0,11 €	0,00 €	0,00 €	0,00 €		0,00 €	38,75 €
9	RAM	KVR21N15D8K2/16	16GB DDR4-RAM 2133MHz (2 x 8GB DIMMs)	59,00 €	✔	59,00 €	- €		0,0000	0,00 €	59,00 €	69,03 €	69,28 €		69,28 €	
10	RAM	2x KVR21N15D8K2/16	32GB DDR4-RAM 2133MHz (4 x 8GB DIMMs)	118,00 €	!	59,00 €	82,15 €	105,00 €	88,2353	22,85 €	0,00 €	0,00 €	0,00 €	✔	88,24 €	157,51 €
11	Systemfestplatte	SSDSC2KW120H6X1	120GB SSD: Intel 540s series	42,00 €	✔	42,00 €	- €		0,0000	0,00 €	42,00 €	49,14 €	49,32 €		49,32 €	
12	Systemfestplatte	2x SSDSC2KW120H6X1	2x 120GB SSD: Intel 540s series (RAID 1)	104,00 €	!	42,00 €	86,32 €	89,00 €	74,7899	2,68 €	0,00 €	0,00 €	0,00 €		0,00 €	
13	Systemfestplatte	SSDSC2KW240H6X1	240GB SSD: Intel 540s series	67,00 €	!	42,00 €	34,81 €	36,00 €	30,2521	1,19 €	0,00 €	0,00 €	0,00 €	✔	30,25 €	
14	Systemfestplatte	MZ-V5P256BW	256GB SSD M.2 SAMSUNG 950 Pro Series NVMe	152,00 €	!	42,00 €	153,15 €	155,00 €	130,2521	1,85 €	0,00 €	0,00 €	0,00 €		0,00 €	
15	Systemfestplatte	2x SSDSC2KW240H6X1	2x 240GB SSD: Intel 540s series (RAID 1)	154,00 €	!	42,00 €	155,94 €	159,00 €	133,6134	3,06 €	0,00 €	0,00 €	0,00 €		0,00 €	
16	Systemfestplatte	SSDSC2KW480H6X1	480GB SSD: Intel 540s series	119,00 €	!	42,00 €	107,21 €	119,00 €	100,0000	11,79 €	0,00 €	0,00 €	0,00 €		0,00 €	
17	Systemfestplatte	MZ-V5P256BW	512GB SSD M.2 SAMSUNG 950 Pro Series NVMe	263,00 €	!	42,00 €	307,70 €	309,00 €	259,6639	1,30 €	0,00 €	0,00 €	0,00 €		0,00 €	
18	Systemfestplatte	2x SSDSC2KW480H6X1	2x 480GB SSD: Intel 540s series (RAID 1)	278,00 €	!	42,00 €	328,58 €	329,00 €	276,4706	0,42 €	0,00 €	0,00 €	0,00 €		0,00 €	
19	Systemfestplatte	MZ-75E1T0B/EU	1TB SSD: Samsung 850 EVO series	242,00 €	!	42,00 €	278,46 €	279,00 €	234,4538	0,54 €	0,00 €	0,00 €	0,00 €		0,00 €	79,57 €
20	Storagefestplatte		keine Storagefestplatte	0,00 €	✔	0,00 €	- €		0,0000	0,00 €	0,00 €	0,00 €	0,00 €		0,00 €	
21	Storagefestplatte	WD10EFRX	1TB IntelliPower, 64MB Cache, WD red	53,00 €	!	0,00 €	73,79 €	73,00 €	61,3445	-0,79 €	0,00 €	0,00 €	0,00 €	✔	61,34 €	
22	Storagefestplatte	WD20EFRX	2TB IntelliPower, 64MB Cache, WD red	75,00 €	!	0,00 €	104,42 €	109,00 €	91,5966	4,58 €	0,00 €	0,00 €	0,00 €		0,00 €	
23	Storagefestplatte	WD30EFRX	3TB IntelliPower, 64MB Cache, WD red	95,00 €	!	0,00 €	132,27 €	139,00 €	116,8067	6,73 €	0,00 €	0,00 €	0,00 €		0,00 €	
24	Storagefestplatte	MZ-75E500B/EU	500GB SSD: Samsung 850 EVO series	129,00 €	!	0,00 €	179,61 €	185,00 €	155,4622	5,39 €	0,00 €	0,00 €	0,00 €		0,00 €	
25	Storagefestplatte	MZ-75E1T0B/EU	1TB SSD: Samsung 850 EVO series	242,00 €	!	0,00 €	336,94 €	339,00 €	284,8739	2,06 €	0,00 €	0,00 €	0,00 €		0,00 €	61,34 €
26	Grafiksystem		2 fach Grafiksystem max. 2* 1920x1200 (DVI)	0,00 €	✔	0,00 €	- €		0,0000	0,00 €	0,00 €	0,00 €	0,00 €		0,00 €	
27	Grafiksystem	100-505651	4 fach Grafiksystem max. 4* 1920x1200 (DVI)	110,00 €	!	0,00 €	153,15 €	159,00 €	133,6134	5,85 €	0,00 €	0,00 €	0,00 €		0,00 €	
28	Grafiksystem	VCQK1200DP-PB	4 fach Grafiksystem max. 4* 3840x2160 @ 60Hz (DP)	320,00 €	!	0,00 €	445,54 €	449,00 €	377,3109	3,46 €	0,00 €	0,00 €	0,00 €		0,00 €	
29	Grafiksystem	100-505651 + 100-505652	6 fach Grafiksystem max. 6* 1920x1200 (DVI)	220,00 €	!	0,00 €	306,31 €	314,00 €	263,8655	7,69 €	0,00 €	0,00 €	0,00 €	✔	263,87 €	
30	Grafiksystem	2x 100-505651 + 100-505652	8 fach Grafiksystem max. 8* 1920x1200 (DVI)	330,00 €	!	0,00 €	459,46 €	469,00 €	394,1176	9,54 €	0,00 €	0,00 €	0,00 €		0,00 €	
31	Grafiksystem	2x VCQK1200DP-PB	8 fach Grafiksystem max. 8* 3840x2160 @ 60Hz (DP)	640,00 €	!	0,00 €	891,07 €	889,00 €	747,0588	-2,07 €	0,00 €	0,00 €	0,00 €		0,00 €	
32	Grafiksystem	2x 100-505651 + 2x 100-505652	10 fach Grafiksystem max. 10 * 1920x1200 (DVI)	440,00 €	!	0,00 €	612,61 €	619,00 €	520,1681	6,39 €	0,00 €	0,00 €	0,00 €		0,00 €	
33	Grafiksystem	2x 100-505651 + 3x 100-505652	12 fach Grafiksystem max. 12* 1920x1200 (DVI)	550,00 €	!	0,00 €	765,77 €	769,00 €	646,2185	3,24 €	0,00 €	0,00 €	0,00 €		0,00 €	
34	Grafiksystem	100-505610 + 100-505651 + 3x 100-505652	14 fach Grafiksystem max. 14* 1920x1200 (DVI)	660,00 €	!	0,00 €	918,92 €	929,00 €	780,6723	10,08 €	0,00 €	0,00 €	0,00 €		0,00 €	263,87 €
35	WLAN		keine W-LAN Adapter	0,00 €	✔	0,00 €	- €		0,0000	0,00 €	0,00 €	0,00 €	0,00 €	✔	0,00 €	
36	W-LAN	TL-WDN4800	W-Fi 802.11 a/b/g/n, dual band (2.4GHz, 5GHz), 450Mbit/s	29,00 €	!	0,00 €	40,38 €	45,00 €	37,8151	4,62 €	0,00 €	0,00 €	0,00 €		0,00 €	0,00 €
37	Betriebssystem	KW9-00146	Windows 10 Home, 64-bit, DE	77,00 €	✔	77,00 €	- €		0,0000	0,00 €	77,00 €	90,09 €	90,41 €		90,41 €	
38	Betriebssystem	FQC-08922	Windows 10 Pro 64-bit, DE	115,00 €	!	77,00 €	52,91 €	53,00 €	44,5378	0,09 €	0,00 €	0,00 €	0,00 €	✔	44,54 €	
39	Betriebssystem	FQC-08291	Windows 7 Professional 64-bit	125,00 €	!	77,00 €	66,83 €	69,00 €	57,9832	2,17 €	0,00 €	0,00 €	0,00 €		0,00 €	134,95 €
40	Garantie		Herstellergarantie 24 Monate Bring-In	0,00 €	✔	- €			0,0000	0,00 €	0,00 €	0,00 €	0,00 €	✔	0,00 €	
41	Garantie	GAR-36	Herstellergarantie 36 Monate Bring-In	70,00 €	!	0,00 €	97,46 €	99,00 €	83,1933	1,54 €	0,00 €	0,00 €	0,00 €		0,00 €	0,00 €
42																
43											850,92 €	995,58 €	1.184,74 €	144,66 €	1.621,01 €	1.621,01 €
44												999,1597 €	1.189,00 €	148,24 €		
45												17,00%				
46												17,42%				

Abbildung A.3: Eine Excel-Preiskalkulation

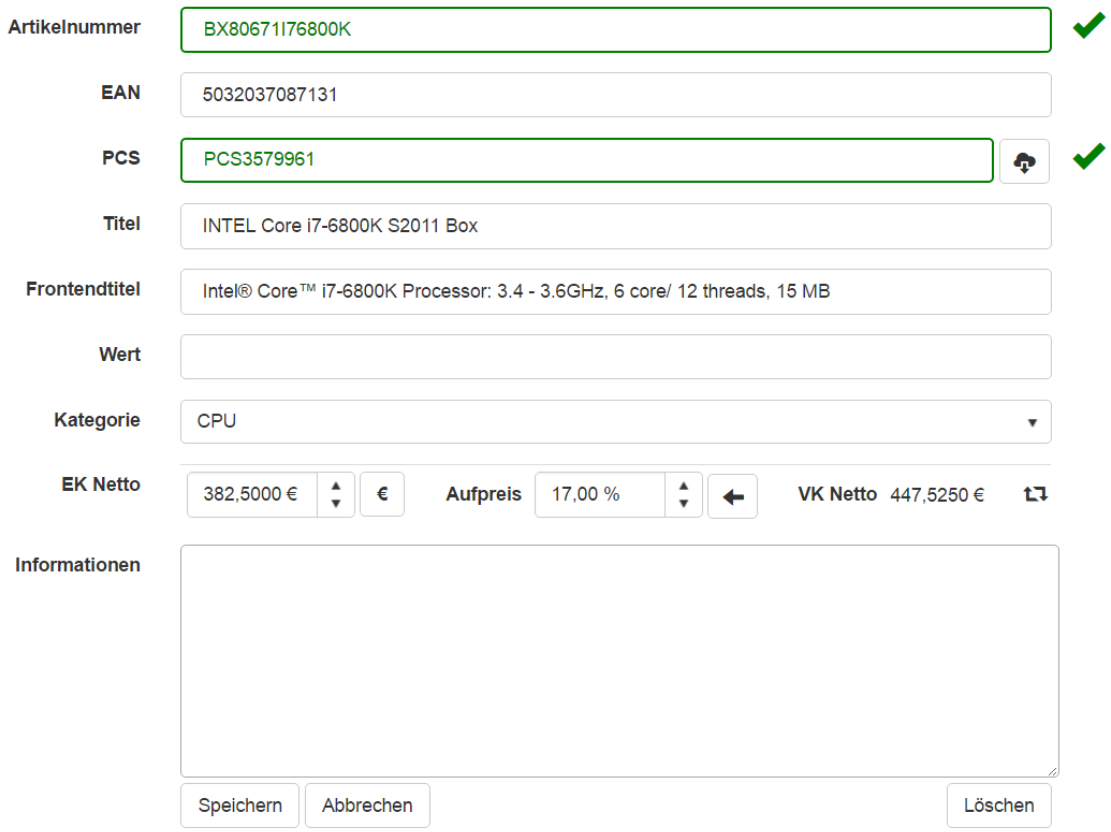

Abbildung A.4: Bearbeiten-Dialog eines Artikels

Anhang B: Quellcode

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PriceAndAvailabilityQuery>
  <TransactionHeader>
    <VersionId>1.00</VersionId>
    <GenerationDateTime>2017-01-06T14:29:00
      </GenerationDateTime>
    <Login>Login</Login>
    <Password>Password</Password>
  </TransactionHeader>
  <PriceAndAvailability>
    <Query>
      <ProductReference>
        <ProprietaryProductIdentifier>
          <ProprietaryProductNumber>PCS3209692
            </ProprietaryProductNumber>
          <ProprietaryProductNumber>PCS3209693
            </ProprietaryProductNumber>
          <ProprietaryProductNumber>PCS3369697
            </ProprietaryProductNumber>
        </ProprietaryProductIdentifier>
      </ProductReference>
    </Query>
  </PriceAndAvailability>
</PriceAndAvailabilityQuery>
```

Listing B.1: beispielhaftes XML-Request-Dokument für die Nutzung der EBC-Schnittstelle

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PriceAndAvailabilityQueryResponse>
  <TransactionHeader>
    <VersionId>1.00</VersionId>
    <GenerationDateTime> 2017-02-01T09:48:32
      </GenerationDateTime>
  </TransactionHeader>
  <PriceAndAvailability>
    <Header>
      <ProductReference>
        <ProductIdentifier>
          <ProductNumber>PCS3209692</ProductNumber>
        </ProductIdentifier>
        <ManufacturerProductNumber>BX80662I565
          </ManufacturerProductNumber>
        <GlobalProductIdentifier>
          <GlobalProductNumber> 5032037076500
            </GlobalProductNumber>
        </GlobalProductIdentifier>
        <ProductDescription>Intel Core i5-6500
          </ProductDescription>
      </ProductReference>
    </Header>
    <Body>
      <DistributorProductItem>
        <DistributorDescription>
          <DistributorName>ACTIONIT
            </DistributorName>
          <DistributorIdentifier> 71009
            </DistributorIdentifier>
        </DistributorDescription>
        <Quantity>
          <AvailableQuantity>48</AvailableQuantity>
        </Quantity>
        <UnitPrice>
          <PurchasePrice> 178.50 </PurchasePrice>
          <NetSalesPrice> 112.12 </NetSalesPrice>
          <RetailPrice> 265.52 </RetailPrice>
          <CurrencyCode>EUR</CurrencyCode>
        </UnitPrice>
        <EOL>false</EOL>
      </DistributorProductItem>
       [...]
    </Body>
  </PriceAndAvailability>
  [...]
</PriceAndAvailabilityQueryResponse>
```

Listing B.2: beispielhaftes XML-Response-Dokument der EBC-Schnittstelle

```
//Alle aktualisierten Artikel
var articleIds = priceModels
  .Where(x => !x.IsWarning)
  .Select(x => x.ArticleId)
  .ToList();

//Alle Optionen von Optionswerten
//mit modifizierten Artikeln
var optionIds = dbArticles
  .Where(x => articleIds.Contains(x.Id))
  .Select(a => a.OptionValueArticles
  .SelectMany(ova => ova.OptionValue.OptionId))
  .Distinct()
  .ToList();

foreach(var optionId in optionIds){
  UpdateValuePricesForOption(optionId);
}
```

Listing B.3: Extrahieren der zu aktualisierenden Optionen

```
string errorMessage;

if (!isInLiveSystem)
{
  errorMessage = _CreateZipDownload(zipFileName,
     exportData, imagePaths);
}
else
{
  errorMessage = _UploadToServer(Path.Combine(folderPath,
     zipFileName));
  if(errorMessage == null)
  {
    errorMessage = _SendRequest(template.Id, zipFileName,
       template.SalePrice);
  }
}

var export = new TemplateExport() {
  IsInLiveSystem = isInLiveSystem,
  TemplateId = template.Id,
  IsCheckOkay = errorMessage == null,
  ErrorMessage = errorMessage
};

db.Exports.Add(export);
await db.SaveChangesAsync();
return Json(new {Success=export.IsCheckOkay});
```

Listing B.4: Ablauf des Exports

```
ConnectionInfo connNfo = new ConnectionInfo(host, port,
    saveFilePath, new PasswordAuthenticationMethod(username,
     password));
try
{
  using (var scp = new ScpClient(connNfo))
  {
    string uploadfn = filePath;
    scp.Connect();
    var fileInfo = new FileInfo(uploadfn);

    using (FileStream upfileStream = System.IO.File.
       OpenRead(uploadfn))
    {
      scp.OperationTimeout = new TimeSpan(0, 1, 0);
      scp.Upload(fileInfo, fileInfo.Name);
    }
    scp.Disconnect();
    return null;
  }
}
catch (Exception x)
{
  return x.Message;
}
```

Listing B.5: Senden einer Datei über eine SCP-Verbindung

```
var warningModels = priceModels.Where(x => x.IsWarning).
    ToList();

if (warningModels.Any())
{
  string subject = "Fehler bei " + warningModels.Count + "
      Artikeln";
  string message = "";

  foreach (var warningModel in warningModels)
  {
    //<a href='path/Article/Index/?articleId=5'>Intel i5
       3600</a>
    message += "<a href='" + path;
    message += "/Article/Index/?articleId=" + warningModel
       .ArticleId;
    message += "'>";
    message += warningModel.Title;
    message += "</a> <br/>";
  }
}
```

Listing B.6: Generieren des Nachrichtentextes

```
1  //Service zur Verbindung mit dem Exchange-Server aufbauen
2  ExchangeService service = new ExchangeService(
       exchangeVersion)
3  {
4    Credentials = new WebCredentials(username, password),
5    Url = new Uri(exchangeAddress)
6  };
7
8  //E-Mail-Nachricht erstellen
9  EmailMessage msg = new EmailMessage(service);
10
11 // Addressaten der E-Mail zuweisen
12 foreach (var address in addresses)
13 {
14   msg.ToRecipients.Add(new EmailAddress(address));
15 }
16 // Betreff und Nachricht in HTML-Struktur zuweisen
17 msg.Subject = subject;
18 msg.Body = new MessageBody(BodyType.HTML, message);
19
20 //Nachricht absenden
21 msg.SendAndSaveCopy(WellKnownFolderName.SentItems);
```

Listing B.7: Quellcode zum Senden einer Nachricht über einen Exchange Server

```
1  public bool UpdateValuePricesForOption(long optionId)
2  {
3    var values = db.OptionValues.Where(x => x.OptionId ==
       optionId)
4      .Include(x => x.Option.Group.Template)
5      .Include(x => x.Option.StandardValue.
         OptionValueArticles
6      .Select(y => y.Article))
7      .Include(x => x.OptionValueArticles.Select(y => y.
         Article))
8      .ToList();
9
10   if(!values.Any()) return false;
11
12   foreach (var value in values)
13   {
14     var model = new ListOptionValuePricingViewModel(value)
         ;
15     value.Price = model.VkNettoResult;
16     db.Entry(value).State = EntityState.Modified;
17   }
18   await db.SaveChangesAsync();
19   return true;
20 }
```

Listing B.8: Quellcode zur Aktualisierung aller Optionswerte einer Option

```
function updateColour(){
  if (optimizedProfit > aimProfit) {
    $(".arrow-down").hide();
    $(".arrow-up").show();
    $(".optimizedProfit").removeClass("highlightRed");
    $(".optimizedProfit").addClass("highlightGreen");
    return;
  }
  $(".arrow-down").show();
  $(".arrow-up").hide();
  $(".optimizedProfit").removeClass("highlightGreen");
  $(".optimizedProfit").addClass("highlightRed");
}
```

Listing B.9: Aktualisieren der Farbe und Pfeile für Bearbeiten-Dialog der Vorlage

```
@(Html.Kendo().Window()
  .Name("EditTemplateWindow")
  .Title("Vorlage bearbeiten")
  .Visible(false)
  .Scrollable(true)
  .Modal(true)
  .Draggable()
  .Resizable()
  .Events(e => {
    e.Refresh("OnRefreshWindow");
    e.Close("OnCloseWindow");
  })
)
```

Listing B.10: Kendo-Window

```
$(document).ready(function(){
  var groupId = getGroupId();
  //falls Wert nicht gesetzt
  //und keine Optionsgruppe vorhanden -> Abbruch
  if(isNaN(groupId)){
    if(!$(".group-panel").length) return;
    groupId = $(".group-panel:first").attr("groupId");
    setGroupId(groupId);
  }

  $(".optionPanels").load("/Template/OptionsByGroup/" +
     groupId);
});
```

Listing B.11: Laden der Option nach Laden der Optionsgruppen-Ansicht

Literaturverzeichnis

[a3 Wirtschaftsverlag Gesellschaft m.b.H. 2012] A3 WIRTSCHAFTSVERLAG GESELLSCHAFT M.B.H.: *wiso - Jeder Zweite hat Onlineshop*. https://www.wiso-net.de/document/AAA__065306906707909520120802l%20151100009. Version: 2012

[Afsari et al. 2017] AFSARI, Kereshmeh ; EASTMAN, Charles M. ; CASTRO-LACOUTURE, Daniel: JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. In: *Automation in Construction* 77 (2017), S. 24–51

[Anderson u. Mullen 2016] ANDERSON, Rick ; MULLEN, Taylor: *Razor Syntax Reference*. https://docs.microsoft.com/en-us/aspnet/core/mvc/views/razor. Version: 2016

[Brich u. Hasenbalg 2013] BRICH, Stefanie (Hrsg.) ; HASENBALG, Claudia (Hrsg.): *Kompakt-Lexikon Wirtschaftsinformatik: 1.500 Begriffe nachschlagen, verstehen, anwenden*. Wiesbaden : Springer Gabler, 2013 http://dx.doi.org/10.1007/978-3-658-03029-2. – ISBN 978–3–658–03028–5

[combit GmbH 2002] COMBIT GMBH: *combit factura manager 2002 Benutzerhandbuch*. 2002

[EHI Retail Institute 2013] EHI RETAIL INSTITUTE: *Einsatz von Eigenentwicklungen und Standardsoftware bei Warenwirtschaftssystemen im deutschsprachigen Handel im Jahr 2013*. 2013

[EntityFrameworkTutorial 2016] ENTITYFRAMEWORKTUTORIAL: *What is Entity Framework?* http://www.entityframeworktutorial.net/what-is-entityframework.aspx. Version: 2016

[Handelsverband Deutschland 2016] HANDELSVERBAND DEUTSCHLAND: *Umsatz durch E-Commerce (B2C) in Deutschland in den Jahren 1999 bis 2015 sowie eine Prognose für 2016 (in Milliarden Euro)*. https://de.statista.com/statistik/daten/studie/3979/umfrage/e-commerce-umsatz-in-deutschland-seit-1999/. Version: 2016

[Heinemann 2010] HEINEMANN, Gerrit: *Der neue Online-Handel: Erfolgsfaktoren und Best Practices*. 3., überarbeitete Auflage. Wiesbaden : Gabler Verlag / Springer Fachmedien Wiesbaden GmbH Wiesbaden, 2010 http://dx.doi.org/10.1007/978-3-8349-8944-4. – ISBN 978–3–8349–2312–7

[jQuery Foundation 2016a] JQUERY FOUNDATION: *jQuery*. https://jquery.com/. Version: 2016

[jQuery Foundation 2016b] JQUERY FOUNDATION: *jQuery API Documentation*. http://api.jquery.com/. Version: 2016

[justSelling GmbH 2016a] JUSTSELLING GMBH: *Funktionen - justSelling E-Commerce*. http://www.product-configurator.biz/funktionen. Version: 2016

[justSelling GmbH 2016b] JUSTSELLING GMBH: *Wer sind wir - justselling Ihr Spezialist für E-Commerce*. http://www.justselling.de/ueber-uns/wer-sind-wir. Version: 2016

[Kohnke 2015] KOHNKE, Oliver: *Anwenderakzeptanz unternehmensweiter Standardsoftware*. Wiesbaden : Springer Fachmedien Wiesbaden, 2015. – ISBN 978–3–658–08205–5

[Magento Inc. 2014] MAGENTO INC.: *Magento Features List*. http://info2.magento.com/rs/magentosoftware/images/Magento%20Full%20Features%20List%20052714.pdf. Version: 2014

[Magento Inc. 2016a] MAGENTO INC.: *Market & Merchandise Products More Effectively*. https://magento.com/products/enterprise-edition. Version: 2016

[Magento Inc. 2016b] MAGENTO INC.: *Open Source eCommerce Software | Magento*. https://magento.com/products/community-edition. Version: 2016

[Mertens et al. 2017] MERTENS, Peter ; BODENDORF, Freimut ; KÖNIG, Wolfgang: *Grundzüge der Wirtschaftsinformatik*. 12., grundlegend überarbeitete Auflage. 2017 (Lehrbuch). – ISBN 978–3–662–53362–8

[Mertens et al. 2012] MERTENS, Peter ; BODENDORF, Freimut ; KÖNIG, Wolfgang ; PICOT, Arnold ; SCHUMANN, Matthias ; HESS, Thomas: *Grundzüge der Wirtschaftsinformatik*. 11. Aufl. 2012. Berlin and Heidelberg : Springer, 2012 (Springer-Lehrbuch). http://dx.doi.org/10.1007/978-3-642-30515-3. – ISBN 978–3–642–30514–6

[Microsoft Corporation 2008] MICROSOFT CORPORATION: *Link to Everything: A List of LINQ Providers*. https://blogs.msdn.microsoft.com/charlie/2008/02/28/link-to-everything-a-list-of-linq-providers/. Version: 2008

[Microsoft Corporation 2016] MICROSOFT CORPORATION: *Entity Framework 6.x*.

https://github.com/aspnet/EntityFramework6/blob/master/README.md. Version: 2016

[MID GmbH] MID GMBH: *Methodisches Vorgehen für Business-Analysten mit BPMN*. http://www.mid.de/fileadmin/user_upload/products/flyer/MM5_Methodisches_Vorgehen_fuer_BusinessAnalysten_mit_BPMN.pdf

[Mozilla Developer Network 2016] MOZILLA DEVELOPER NETWORK: *Einführung in objektorientiertes JavaScript*. https://developer.mozilla.org/de/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript. Version: 2016

[Nurseitov et al. 2009] NURSEITOV, Nurzhan ; PAULSON, Michael ; REYNOLDS, Randall ; IZURIET, Clemente: *Comparison of JSON and XML Data Interchange Formats: A Case Study*. 2009

[Object Management Group Inc. 2016] OBJECT MANAGEMENT GROUP INC.: *BPMN Specification - Business Process Model and Notation*. http://www.bpmn.org/. Version: 2016

[Otto et al. 2016] OTTO, Mark ; THORNTON, Jacob ; CONTRIBUTORS ; BOOTSTRAP: *Components · Bootstrap*. http://getbootstrap.com/components/. Version: 2016

[Schneider et al. 2014] SCHNEIDER, Gabriel ; GEIGER, Ingrid K. ; SCHEURING, Johannes: *Prozess- und Qualitätsmanagement: Grundlagen der Prozessgestaltung und Qualitätsverbesserung mit zahlreichen Beispielen, Repetitionsfragen und Antworten*. 2., überarbeitete Aufl. Zürich : Compendio Bildungsmedien, 2014 (Betriebswirtschaftslehre). – ISBN 978–3–7155–9954–0

[Schwichtenberg 2016] SCHWICHTENBERG, Holger: *Eine Einführung in das Microsoft .NET Framework (DOTNET)*. http://www.dotnetframework.de/dotnet/DOTNET_Framework_Einfuehrung.aspx. Version: 2016

[Sjurts 2011] SJURTS, Insa: *Gabler Lexikon Medienwirtschaft*. Wiesbaden : Springer Fachmedien, 2011 http://gbv.eblib.com/patron/FullRecord.aspx?p=751554. – ISBN 978–3–8349–6487–8

[Steyer 2014] STEYER, Ralph: *JavaScript: Die universelle Sprache zur Web-Programmierung*. 1. Aufl. München : Carl Hanser Fachbuchverlag, 2014 http://www.hanser-elibrary.com/isbn/9783446439429. – ISBN 3446439420

[Synaxon AG 2015a] SYNAXON AG: *EGIS Business Connector (EBC) Technische Dokumentation: Allgemeines*. 2015

[Synaxon AG 2015b] SYNAXON AG: *EGIS Business Connector (EBC) Technische Dokumentation: Artikelstammdaten*. 2015

[Synaxon AG 2016a] SYNAXON AG: EGIS Business Connector (EBC): EGIS Data-Pack Artikelstammdaten. (2016). https://soc.wiki.synaxon.de/images/9/9a/Synaxon_EGIS-Flyer_neu.pdf

[Synaxon AG 2016b] SYNAXON AG: *EGIS ist effiziente Beschaffung*. http://www.egis-online.de/prozessoptimierung.html. Version: 2016

[Telerik 2017a] TELERIK: *ASP.NET MVC Grid control example*. https://demos.telerik.com/aspnet-mvc/grid. Version: 2017

[Telerik 2017b] TELERIK: *Kendo UI HTML Framework*. http://www.telerik.com/kendo-ui. Version: 2017

[Tresch 1996] TRESCH, Markus: Middleware: Schlüsseltechnologie zur Entwicklung verteilter Informationssysteme. In: *Informatik-Spektrum* 19 (1996), Nr. 5, S. 249–256

[W3Schools 2016] W3SCHOOLS: *AJAX Introduction*. http://www.w3schools.com/xml/ajax_intro.asp. Version: 2016

[W3Schools 2017] W3SCHOOLS: *JSON Introduction*. https://www.w3schools.com/js/js_json_intro.asp. Version: 2017

[XECURIS GmbH & Co. KG 2016a] XECURIS GMBH & CO. KG: *Impressum - Trading-PC*. http://www.trading-pc.de/tpc/impressum.html. Version: 2016

[XECURIS GmbH & Co. KG 2016b] XECURIS GMBH & CO. KG: *Wir über uns - Trading-PC*. http://www.trading-pc.de/tpc/wir-ueber-uns.html. Version: 2016

[XECURIS GmbH & Co. KG 2016c] XECURIS GMBH & CO. KG: *XECURIS GmbH & Co. KG*. https://xecuris.com/. Version: 2016

[XECURIS GmbH & Co. KG 2016d] XECURIS GMBH & CO. KG: *XECURIS GmbH & Co. KG*. http://www.trading-pc.de/xecuris/. Version: 2016

Glossar

.NET Infrastruktur für Systementwicklung auf Windows-Systemen.

AJAX Technologie zum asynchronen Datenaustausch.

ASP.NET MVC Framework für webbasierte Softwaresysteme, folgt dem MVC-Entwurfsmuster.

Basisoption Methode zur Definition von Abhängigkeiten zwischen Optionswerten.

Basispreis wird berechnet aus dem VK Netto der Standardwerte einer Vorlage.

BPMN Spezifikationssprache für Prozesse.

Code First Migration Technologie für den automatisierten Aufbau einer objektrelationalen Struktur aus Model-Klassen.

Controller Enthält die Logik eines Softwaresystems.

Customizing Anpassung eines Softwaresystems an die Bedürfnisse des Endkunden.

Distributor auch Lieferant, Großhändler für den Einkauf von Gütern.

DOM Schnittstelle für den Datenzugriff auf zugrundeliegende Strukturen, wird unter anderem in JavaScript implementiert.

E-Commerce Online-Handel, Ein- und Verkaufsvorgänge über das Internet.

EBC EGIS Business Connector, Schnittstelle des Online-Portals von EGIS.

EGIS Online-Portal, um den Zugang zu aktuellen Marktpreisen zu erhalten.

Entity Framework Technologie, welche mithilfe eines O/R-Mappers Zugriffsoperationen auf die Datenbank vereinfacht.

ERP-System ganzheitliches Softwaresystem für Unternehmensprozesse mit hohem Integrationsgrad.

Expression JavaScript-Programmierschnittstelle des justSelling Produkt Konfigurators.

Factura ERP-System aus dem Jahr 2002.

HTML Webseitenformat zur Verarbeitung von digitalen Dokumenten.

HTTP Protokoll zur Übertragung von Daten über ein Rechnernetz.

Individualsoftware unternehmensintern entwickeltes Softwaresystem.

JavaScript webbasierte objektorientierte Skriptsprache.

jQuery Bibliothek zur vereinfachten Nutzung von JavaScript.

JSON Datenformat zum Datenaustausch an Schnittstellen.

Kendo UI Framework, welches grafische Benutzerelemente zur Verfügung stellt.

LINQ Teil von .NET, stellt einen einheitlichen Satz an Methoden für listenähnliche Datenstrukturen zur Verfügung.

Magento verbreitetes Onlineshop-System, basiert auf PHP.

Microsoft Exchange Server dient der zentralen Verwaltung von E-Mails, Kontakten und Tasks.

Middleware Individualentwicklung, welches eine Integration bzw. Synchronisation zwischen verschiedenen Systemen realisiert.

Model enthält die Daten.

Option Komponente des justSelling Produkt Konfigurators, enthält Optionswerte der gleichen Art.

Optionsgruppe Komponente des justSelling Produkt Konfigurators, ist eine übergeordnete Kategorie für Optionen und Optionswerte.

Optionswert Komponente des justSelling Produkt Konfigurators, ist eine auswählbare Komponente im Frontend.

PHP Scriptsprache zur serverseitigen Verarbeitung, wird verwendet um dynamische Webseiten zu erzeugen.

Post eine Request-Methode des Übertragungsprotokolls HTTP.

Razor Syntax von ASP.NET MVC für dynamische Webansichten.

Responsive Webdesign Anpassung einer Webseite, sodass das Layout an das Endgerät angepasst wird.

Schnittstelle Teil eines Systems, welches dem Datenaustausch zu anderen Systemen gewährleistet.

SEO Suchmaschinenoptimierung, Optimierung einer Webseite, um unbezahlten organischen Traffic aus Suchmaschinen zu erhöhen.

Softwaresystem modular gestaltete Software.

SQL Sprache zur Definition und Bearbeiten von Datenbanken.

Standardsoftware Softwareprodukt, welches für eine breite Zielgruppe entwickelt wurde.

View dient der Definition von Webansichten.

XML deskriptive Auszeichnungssprache, dient zur Abbildung von hierarchisch strukturierten Daten.

ZIP Dateiformat zur verlustfreien Kompression von einem oder mehreren Datenobjekten.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 30.03.2017