Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

Fachbereich IV
Angewandte Informatik

Hausarbeit

über das Thema

Versuch einer *'lege artis' Analyse* der Software Device Registration

Hendrik Hofmann
Matr.Nr.: 539721
7.Semester
5. Januar 2017

Prüfer:
Prof. Dr.-Ing. habil.
Dierk Langbein,
Onur Yavuz

I Inhaltsverzeichnis

1 Grundlagen

1.1 iTool3

iTool3 ist eine auf dem CakePHP 3.3 - Framework basierende eCommerce Software Lösung zur Steuerung von Produktsortimenten auf verschiedenen Marktplätzen mit dem Ziel, den Vertriebsprozess zu automatisieren. Es ermöglicht dem Benutzer über eine einzelne Benutzeroberfläche Produkte auf Marktplätzen wie eBay, Amazon oder einem Magento Store zu verwalten. Produkte können dabei händisch erstellt oder aus bestehenden Datenquellen in die Software eingepflegt werden. Im Anschluss ist es möglich diese Produkte auf einem oder mehreren Marktplätzen anzubieten. Die Verwaltung und Abwicklung der eingehenden Bestellungen läuft dabei komplett über das iTool. Da für jeden Marktplatz unterschiedliche Daten benötigt werden um auf ihm erfolgreich zu verkaufen, können für jedes Produkt unterschiedliche Attribute mit wiederum unterschiedlichen Werten angelegt werden. Die Produktverwaltung der Software folgt daher dem Entity-Attribute-Value Modell.

1.1.1 Verkäufer und Benutzer

Es wird unterschieden zwischen Verkäufern (Core-Seller) und Benutzern (Core-User). Ein Verkäufer ist z.B. "Markisenshop 2000". Diesem Verkäufer werden Benutzer zugeordnet, die mit mehr oder weniger Rechten ausgestattet, die Produkte z.B. nur einsehen können oder Kontrolle über die gesamte Produkt- und Bestellverwaltung haben.

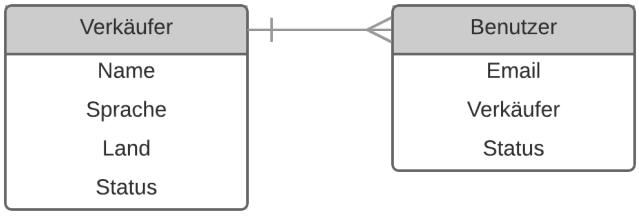

Abbildung 1: ER-Diagramm: Verkäufer - Benutzer

Benutzerdaten werden in der Tabelle `core_users` gespeichert, Verkäuferdaten in `core_sellers`.

1.1.2 Produktverwaltung

Die Produktverwaltung ist aufgeteilt in Produkte und Kategorien. **Produkte** besitzen Attribute wie Titel, Preis, Beschreibung etc. die für jeden Marktplatz auf denen diese angeboten werden sollen unterschiedlich ausfallen können. Es kann gewählt werden ob ein Produkt auf einem bestimmten Marktplatz angeboten werden soll oder nicht. Ein Produkt kann dabei mehreren

Kategorien zugeordnet sein.

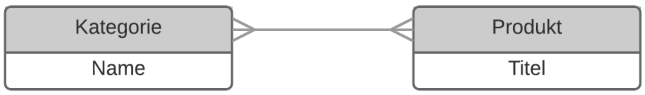

Abbildung 2: ER-Diagramm: Kategorie - Produkt

Eine Kind-Kategorie hat jeweils genau eine Eltern-Kategorie. Eine Eltern-Kategorie kann aber mehrere Kinder haben. Produkte sind genau einem Verkäufer zugeordnet.

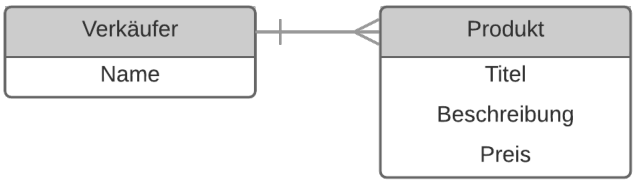

Abbildung 3: ER-Diagramm: Verkäufer - Benutzer

Alle Produktdaten sind in der Tabelle `core_products` und den damit verknüpften Tabellen hinterlegt.

1.1.3 Dashboard

Auf dem Dashboard werden Informationen über die Anzahl der insgesamt eingegangenen Bestellungen, den durchschnittlichen Bestellwert, die Gesamtzahl der Kunden, den insgesamt erwirtschafteten Umsatz, eine Übersicht der zuletzt eingegangenen Bestellungen sowie eine graphische Übersicht der während eines Jahres erwirtschafteten Umsätze angezeigt.

1.2 Der BMECat

Der BMECat ist ein vom 'Bundesverband Materialwirtschaft, Einkauf und Logistik e.V' in Zusammenarbeit mit dem 'eBusiness Standardization Committee' entwickelter XML Standard mit dem Ziel den Austausch von Produktkatalogen zwischen Lieferanten und beschaffenden Organisationen zu standardisieren und somit zu vereinfachen.[1].

[1] BMECat V1.2 Spezifikation, Seite 5

1.2.1 Terminologie

Ein **Produktkatalog** ist die Menge aller benötigten Daten, welche vom katalogerzeugenden Unternehmen an das katalogempfangende Unternehmen übermittelt werden sollen.
Ein **Katalogdokument** ist eine XML-Datei, in der der Produktkatalog im BMECat-Format gespeichert und zum Katalogemfänger übermittelt wird.
Eine **Kataloggruppe** ist ein Datenbereich, der eine Gruppe definiert, welcher gleichartige Artikel zugeordnet werden können. Diese wird im BMEcat-Format durch das Element **CATALOG_STRUCTURE** abgebildet.
Ein **Kataloggruppensystem** ist ein hierarchischer Baum von verknüpften Kataloggruppen. Es wird im BMEcat-Format durch das Element **CATALOG_GROUP_SYSTEM** abgebildet.[2]

1.2.2 Transaktionen

Im BMECat wird zwischen den 3 verschiedenen Transaktionsarten

- **T_NEW_CATALOG** - Übertragung eines neuen Produktkataloges
- **T_UPDATE_PRODUCTS** - Aktualisierung von Produktdaten
- **T_UPDATE_PRICES** - Aktualisierung von Preisinformationen

unterschieden. Die Unterscheidung geschieht um die Größe eines Katalogdokumentes zu reduzieren. Es muss so z.B. nicht ein kompletter Produktkatalog übertragen werden, falls sich bei einem *odermehreren* Artikel*n* der Preis ändert.

1.2.3 Aufbau

Ein BMECat-Dokument besteht aus einer Folge von KANN und MUSS Feldern, den dazugehörigen Datentypen und Feldlängen und ist folgendermaßen aufgebaut:

1. XML-Deklaration und Header-Bereich (mit Informationen über Kataloganbieter und Empfänger, Bezeichnung und Erstellungsdatum des Kataloges etc.)
   Bsp. für einen Header:

```
<HEADER>
  <GENERATOR_INFO> Kann </GENERATOR_INFO>
  <CATALOG> Muss </CATALOG>
  <BUYER> Kann </BUYER>
  <SUPPLIER> Muss </SUPPLIER>
</HEADER>
```

   Bsp. für XML Deklaration:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE BMECAT SYSTEM "bmecat_new_catalog.dtd">
```

[2] BMECat V1.2 Spezifikation, Seite 7

```
<BMECAT version="1.2" xml:lang="de" xmlns="http://www.
    bmecat.org/bmecat/1.2/bmecat_new_catalog">
```

2. Produktgruppensystem (Baumstruktur der Produktgruppen mit den Attributwerten Root, Node und Leaf)

```
<CATALOG_STRUCTURE type="root">
    <GROUP_ID>1</GROUP_ID>
    <GROUP_NAME>Katalog</GROUP_NAME>
    <PARENT_ID>0</PARENT_ID>
    <GROUP_ORDER>1</GROUP_ORDER>
</CATALOG_STRUCTURE>
  <CATALOG_STRUCTURE type="node">
    <GROUP_ID>2</GROUP_ID>
    <GROUP_NAME>Spiele &amp;amp; Konsolen</GROUP_NAME>
    <PARENT_ID>1</PARENT_ID>
 </CATALOG_STRUCTURE>
 <CATALOG_STRUCTURE type="leaf">
    <GROUP_ID>7</GROUP_ID>
    <GROUP_NAME>PlayStation 4</GROUP_NAME>
    <PARENT_ID>2</PARENT_ID>
 </CATALOG_STRUCTURE>
```

3. Artikel (mit Attributen und Werten)

```
<ARTICLE mode="new">
  <SUPPLIER_AID>9057320097280</SUPPLIER_AID>
    <ARTICLE_DETAILS>
      <DESCRIPTION_SHORT>GTA 5</DESCRIPTION_SHORT>
      <DESCRIPTION_LONG>Das tolle neue Spiel</
          DESCRIPTION_LONG>
      <EAN>87126723434</EAN>
        ... weitere Attribute ...
    </ARTICLE_DETAILS>
        ...weitere Felder ...
</ARTICLE>
```

4. Zuordnung der Artikel zu den Produktgruppen.

```
<ARTICLE_TO_CATALOGGROUP_MAP>
  <ART_ID>9057320097280</ART_ID>
  <CATALOG_GROUP_ID>7</CATALOG_GROUP_ID>
</ARTICLE_TO_CATALOGGROUP_MAP>
```

— Übersicht der im BMECat verwendeten Datentypen — noch einfügen —

Im folgenden Abschnitt wird jeder Teilbereich mit seinen Unterelementen, wie sie in vorliegender Arbeit verwandt wurden, graphisch dargestellt und kurz erläutert. Rot hervorgehoben sind jeweils die MUSS-Felder, welche zwingend in einem gültigen BMECat Dokument vorkommen müssen, grün die KANN-Felder. Ein Plus + Zeichen hinter dem Elementnamen indiziert, dass dieses Element mehrfach an dieser Stelle vorkommen kann, jedoch mindestens einmal. Ein Asterisk $*$ zeigt an, dass dieses Element einmal, mehrfach oder nicht vorkommen kann.

Header

Im Header werden allgemeine Informationen über das Katalogdokument hinterlegt und Default Werte gesetzt. Das Element **CATALOG** enthält dabei Informationen zur Identifikation und Beschreibung des Produktkataloges, wie z.B. die Katalog Id, die Katalogversion oder die für das Dokument geltende Sprache sowie Elemente zum setzten von Standard-Werten wie z.B. die für das Katalogdokument geltende Währungsangabe [3]

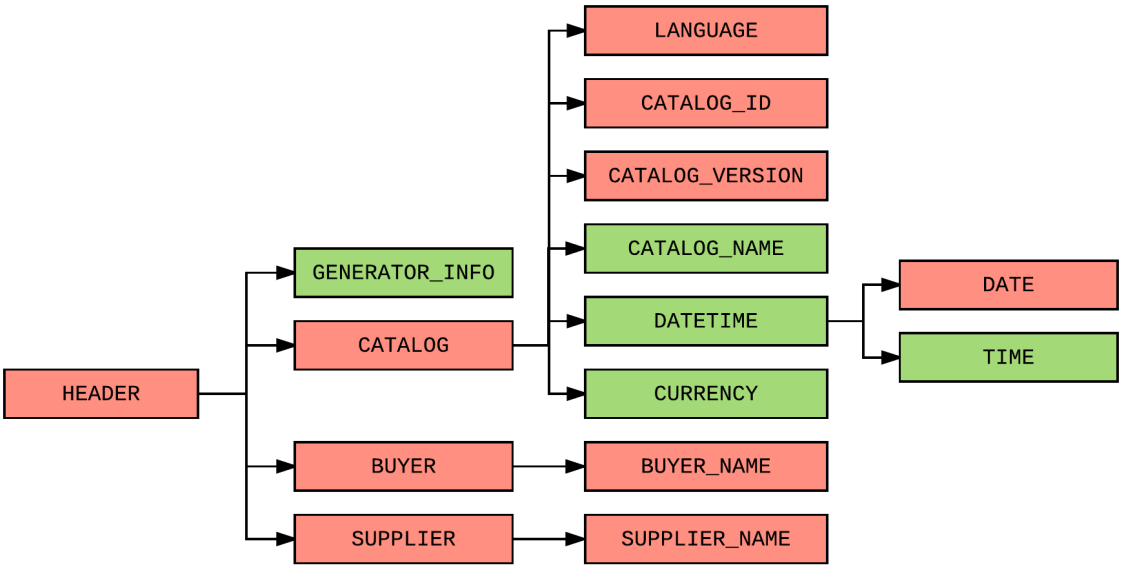

Abbildung 4: Headerstruktur

Die Transaktion T_NEW_CATALOG

Diese Transaktion wird verwandt, um einen Produktkatalog neu zu übertragen. Das empfangende System reagiert dabei je nach übertragener `CATALOG_ID`, `CATALOG_VERSION` und `LANGUAGE` unterschiedlich. Dieser Zusammenhang wir später noch erläutert.

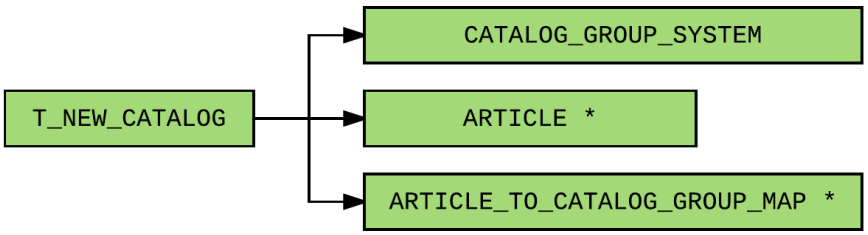

Abbildung 5: T_NEW_CATALOG

[3] BMECat V 1.2 Spezifikation, Seite 27,29

Die Transaktion T_UPDATE_PRODUCTS

Bei dieser Transaktion werden Artikeldaten übertragen und gegebenenfalls einer Kataloggruppe zugeordnet.Je nach Kennung des Artikels (s.u.) werden die übertragenen Artikel im Zielsystem entweder hinzugefügt, gelöscht oder die Artikeldaten werden komplett ersetzt. Der Artikel wird immer komplett ausgetauscht, eine Änderung von einzelnen Datenfeldern innerhalb eines Artikels ist nicht möglich. Wie der Grafik entnommen werden kann ist bei dieser Transaktion nur die Übertragung von Produktdaten und die Zuordnung von Produkten zu Kataloggruppen möglich. [4]

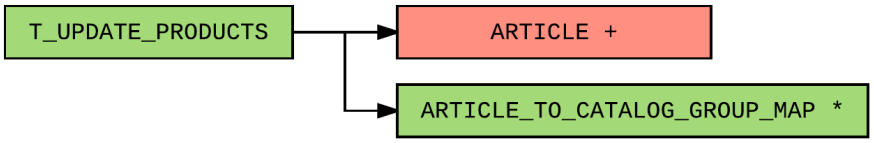

Abbildung 6: T_UPDATE_PRODUCTS

Das Element `T_UPDATE_PRODUCTS` verfügt zusätzlich über das Attribut `prev_version`, welches die Anzahl der vorausgegangenen Updates bzw. die Nummer des übertragenen Updates enthält. Der Wer dieses Attributes wird nach jedem Katalogupdate um 1 erhöht.

```
<T_UPDATE_PRODUCTS prev_version="91">...</T_UPDATE_PRODUCTS>
```

Die Elemente CATALOG_GROUP_SYSTEM und CATALOG_STRUCTURE

Im Element `CATALOG_GROUP_SYSTEM` werden die `GROUP_SYSTEM_ID` und der `GROUP_SYSTEM_NAME` bekannt gemacht sowie die Katalogstruktur `CATALOG_STRUCTURE` beschrieben. Dabei gibt es genau ein Wurzelelement, sowie beliebig viele Knoten und Blätter. Jedes Element hat dabei eine als `GROUP_ID` bezeichnete ID und wird über `PARENT_ID` die dem jeweiligen Elternelement zugeordnet. Die Zuordnung der Artikel zu den Artikelgruppen erfolgt mit dem Element `ARTICLE_TO_CATALOG_GROUP_MAP` das weiter unten beschrieben wird.

[4] vgl. BMECat V 1.2 Spezifikation, Seite 52

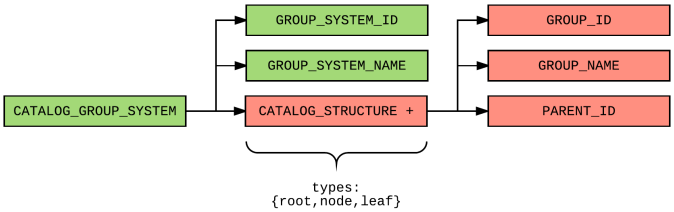

Abbildung 7: CATALOG_GROUP_SYSTEM und CATALOG_STRUCTURE

Das Element ARTICLE

Das Artikelelement schließlich enthält Informationen über einen Artikel, wie Überschrift, Beschreibung, Bilder, Preisinformationen, eine **eindeutige** Artikelnummer usw. Die Artikelnummer wird über das Element `SUPPLIER_AID` bekanntgegeben, handelt es sich um einen Variantenartikel, so bildet sich die Artikelnummer aus der `SUPPLIER_AID` und der `SUPPLIER_AID_SUPPLEMENT`. Dies ist hier jedoch nicht umgesetzt. Die als *eCl@ass* und *Zolltarifnummer* zusammengefassten `ARTICLE_FEATURES` werden explizit von Mercateo verlangt.

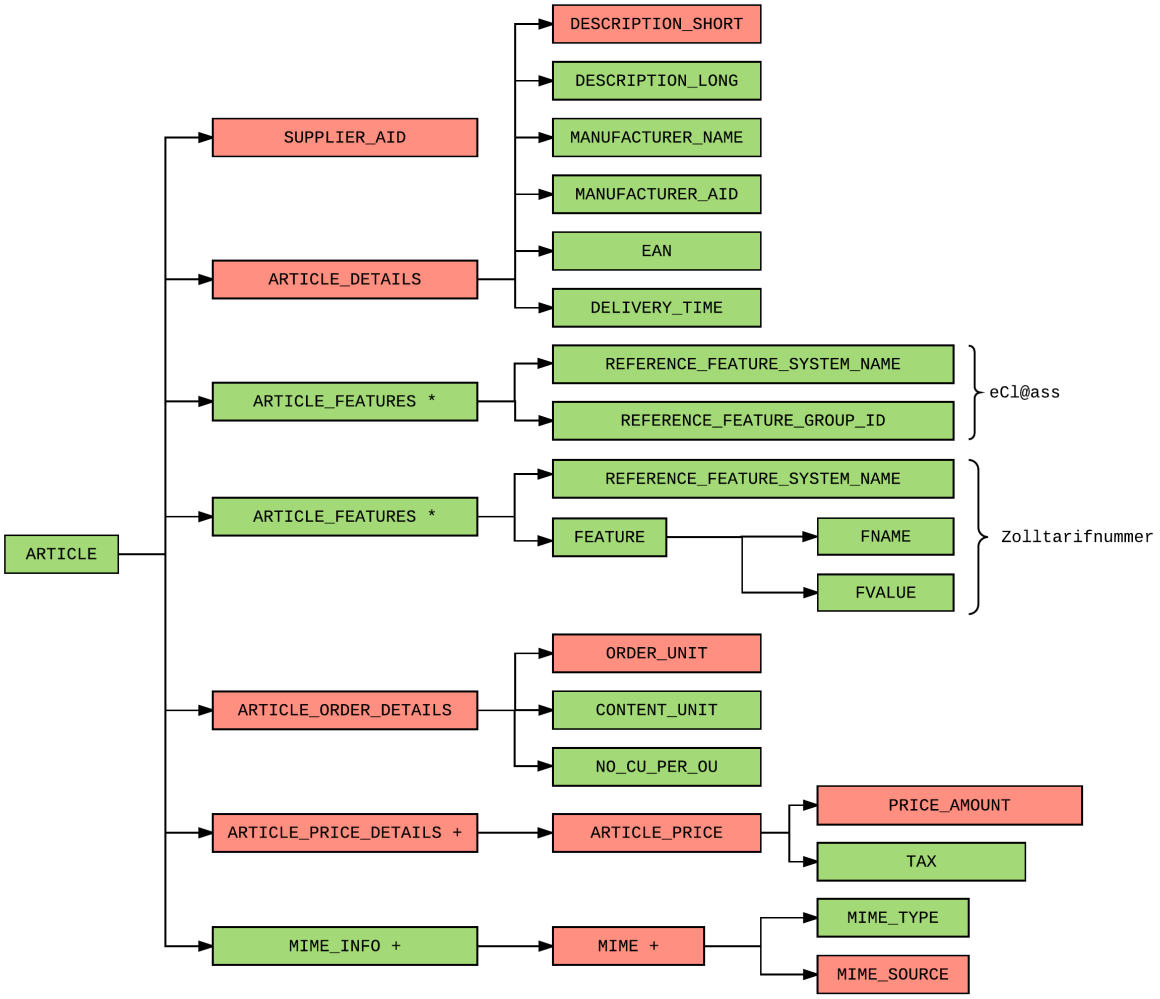

Abbildung 8: Article

Das Element ARTICLE verfügt über das Attribut mode, welches Informationen darüber enthält, ob es sich um die Anlage eines neuen Artikel, ein Update der Artikelinformationen oder die Löschung eines Artikels handelt.

```
<ARTICLE mode="new">...</ARTICLE>
<ARTICLE mode="update">...</ARTICLE>
<ARTICLE mode="delete">...</ARTICLE>
```

Das Element ARTICLE_TO_CATALOG_GROUP_MAP

Um Produkte ihren Kategorien zuordnen zu können wird das Element ARTICLE_TO_CATALOG-GROUP_MAP verwandt. Es erfolgt hier eine Verknüfung aus der eindeutigen Artikelnummer und der GROUP_ID welcher der Artikel zugeordnet werden soll. Eine Mehrfachzuordnung ist möglich, d.h. ein Artikel kann in unterschiedliche Kategorien ëingehängt"werden.

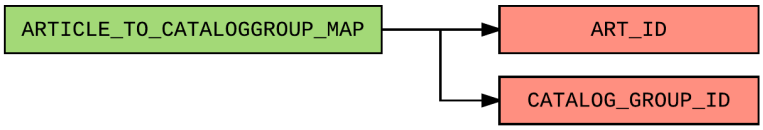

Abbildung 9: ARTICLE_TO_CATALOG_GROUP_MAP

Im Kontext der Transaktion `T_UPDATE_PRODUCTS` verfügt das Element zusätzlich über das Attribut `mode`, mit welchem angegeben wird, ob es sich um eine Neuzuweisung zu einer Kategorie handelt oder der Artikel aus einer Kategorie entfernt werden soll.

```
<ARTICLE_TO_CATALOGGROUP_MAP mode="new">...</<ARTICLE_TO_CATALOGGROUP_MAP>
<ARTICLE_TO_CATALOGGROUP_MAP mode="delete">...</<ARTICLE_TO_CATALOGGROUP_MAP>
```

Zusammenspiel verschiedener Transaktionen

Die folgende Grafik zeigt, wie das empfangende System bei der Transaktion `T_NEW_CATALOG` je nach übergebener `CATALOG_ID`, `CATALOG_VERSION` und `LANGUAGE` reagiert.

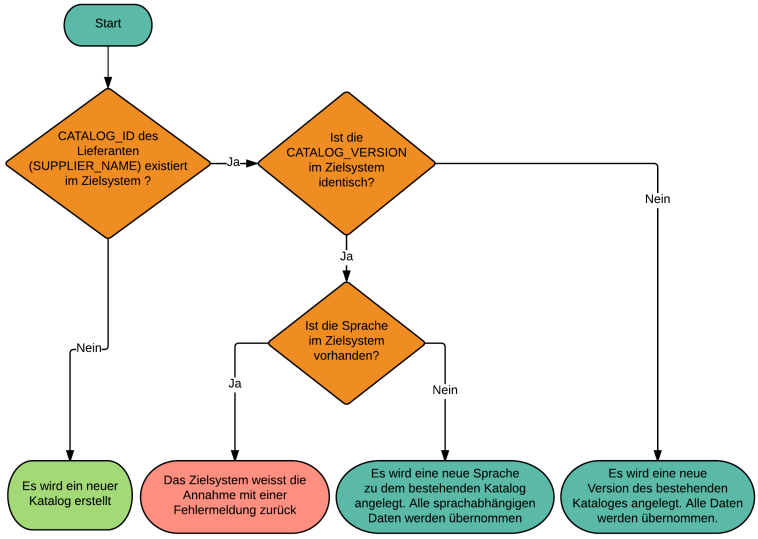

Abbildung 10: Flowchart T_NEW_CATALOG

Kommt die Transaktion `T_UPDATE_PRODUCTS` zur Anwendung, gilt es folgendes zu beachten:[5]

- Die übertragene `CATALOG_ID` des jeweiligen Lieferanten und die dazugehörige `CATALOG_VERSION` müssen im Zielsystem bereits vorhanden sein.

[5] vgl. hierzu: BMECat V 1.2 Spezifikation, Seite 52

- Das Attribut `prev_version` muss bei der ersten anderen Transaktionsart nach `T_NEW_CATALOG`, (`T_UPDATE_PRODUCTS`,`T_UPDATE_PRICES`) auf '0' gesetzt werden.
- Danach wird es bei jeder solchen Transaktion um '1' erhöht.

—- Übersicht, tabellarisch oder nicht über die wichtigsten Felder und ihre Einschränkungen, vor allem die von Mercateo —-

1.3 Das Cake-PHP Framework

Cake PHP ist ein Webframework, das dem MVC (Model-View-Controller) Schema folgt und dabei die Softwaredesignparadigmen DRY (Don't repeat yourself) und 'Convention over configuration' umsetzt.

1.3.1 Convention over Configuration in CakePHP

In CakePHP wird das Softwaredesign-Paradigma der 'Konvention vor Konfiguration' konsequent umgesetzt.

Die Klassennamen von **Controllern** sind im Plural verfasst, 'CamelCased' und enden auf *Controller*. `UsersController` und `ArticleCategoriesController` sind Beispiele dafür. Eine öffentliche Methode eines solchen Controllers kann über einen Webbrowser aufgerufen werden. Per Konvention werden URLs klein geschrieben und mit Bindestrich verbunden. `http://samplesite.com/article-categorie/view` ruft demnach die öffentliche `view()` Methode des ArticleCategoriesControllers auf.

Die Namen von **Model** Klassen sind 'CamelCased' und im Plural. Der Name der zum Model gehörenden Tabelle ist im Plural verfasst und mit einem Unterstrich verbunden.
`article_categories` ist die dem Model `ArticleCategories` zugrundeliegende Tabelle. Um einen Fremdschlüssel auf eine Tabelle zu vergeben genügt es das Suffix `_id` an den kleingeschriebenen Namen dieser Tabelle anzuhängen. Wenn Users eine hasMany Beziehung zu Articles hat, kann mit dem Fremdschlüssel `user_id` in der `articles`-Tabelle auf den entsprechenden Eintrag in der `users`-Tabelle verwiesen werden.

Die Template Datei einer **View** ist nach der entsprechenden Methode im Controller benannt die sie darstellen soll. Die `view()` Methode der `ArticlesController` Klasse würde demnach unter `src/Template/Articles/view.ctp` nach einem View-Template suchen[6].

[6] vgl. hierzu: `http://book.cakephp.org/3.0/en/intro/conventions.html`

1.3.2 Model

Das Backend einer CakePHP Anwendung wird von einer SQL Datenbank gebildet. Das Model repräsentiert die Daten einer Anwendung und enthält die Geschäftslogik zur Datenmanipulation. Nach der CakePHP Konvention wird die Datenbankverbindung einmal in der `config/app.php` konfiguriert. Die Model-Klasse stellt dabei Methoden zur Verfügung, über die es möglich ist, den Zustand der Daten abzufragen, die Daten zu filtern und zu verändern. Die CRUD-Funktionalität (CREATE-READ-UPDATE-DELETE) ist so direkt im Model integriert.[7]. Die Beziehungen einzelner Models zueinander werden über *Associations* hergestellt. Die vier Assoziationstypen in CakePHP sind:

Nr.	Beziehung	Typ	Beispiel
1.	one to one	hasOne	Ein Museum hat eine Adresse.
2.	one to many	hasMany	In einem Museum hängen mehrere Kunstwerke.
3.	many to one	belongsTo	Mehrere Bilder gehören zu einem Museum.
4.	many to many	belongsToMany	Ein Student hat mehrere Professoren. Ein Professor hat mehrere Hörer.

Es ist möglich ein *Model* um ein oder mehrere *Behavior* zu erweitern. Dabei handelt es sich um Klassen, in denen, ähnlich einem Trait, Funktionen zur Erweiterung des Models gekapselt sind. Ein Beispiel hierfür ist das Tree-Behavior, das es ermöglicht hierarchische Datenstrukturen in der Datenbank zu pflegen. Anwendung hierfür kann z.B. die Abbildung einer Kategoriestruktur sein [8].

Mit Hilfe von im Model definierten Validatoren können zu speichernde Daten auf Vollständigkeit und Konsistenz geprüft werden.

Später bei der Beschreibung des Codes auf eigenen Validator verweisen. (MercateoAccountsTable::validateCatalogVersionFormat)

```
    $validator
    ->requirePresence('catalog_name', 'create')
    ->notEmpty('catalog_name')
    ->add('catalog_name', [
            'maxLength' => [
                'rule' => ['maxLength', 100],
                'message' => 'maxLength = 100.'
            ]
]);
```

[7] vgl. hierzu: Webentwicklung mit CakePHP, 2. Auflage, O'Reilly, Seite 7
[8] vgl. hierzu: `http://book.cakephp.org/3.0/en/orm/behaviors/tree.html`

1.3.3 View

Die View ist für die Darstellung der Daten in der Anwendung zuständig. Eine View ist in CakePHP immer auf ein bestimmtes Model bezogen und wird nicht für die Darstellung anderer Daten verwendet[9]. CakePHP View Template Dateien Enden auf '.ctp' und bedienen sich der alternativen PHP Syntax für Kontrollstrukturen und Ausgabe. In einer View kann direkt auf Variablen zugegriffen werden die in der entsprechenden Controller Methode gesetzt wurden:

```
$this->set('articleCategories', $articleCategories);
```

Die Codebeispiele zeigen, wie die Variable `$articleCategories` im Controller für die View freigegeben wird und dort z.B. mit einer foreach-Schleife durchlaufen werden kann um ihren Inhalt auszugeben.

```
<ul>
<?php foreach ($todo as $item): ?>
  <li><?= $item ?></li>
<?php endforeach; ?>
</ul>
```

Listing 1: Alternative PHP Syntax

Eine View ist dabei nicht auf das Anzeigen von HTML Inhalten beschränkt, sondern kann auch dazu verwandt werden XML- oder JSON- Repräsentationen der angefragten Daten zurückzuliefern.

1.3.4 Controller

Der Controller regelt den Ablauf der Benutzerinteraktion. Er ist dafür zuständig, dass das richtige Model aufgerufen und die entsprechende Antwort oder View erzeugt wird. Er dient dabei als eine Art Vermittler zwischen dem Model und der View. Normalerweise ist in CakePHP ein Controller für ein Model verantwortlich, es ist dennoch möglich, oft auch nötig, dass ein Controller mit mehreren Models arbeitet.

Der Controller enthält eine Reihe von Methoden die HTTP Anfragen verarbeiten. Diese Methoden werden in CakePHP *actions* genannt. Per Definition ist jede öffenliche Methode in einem Controller eine *action* und über eine URL der Form `http://samplesite.com/article-categorie/view` erreichbar. Eine *action* ist für die Verarbeitung der Anfrage und das zurückliefern einer Antwort zuständig. Im normalfall wird dabei eine View erzeugt, es können aber auch (wie im Abschnitt Model erläutert) auch XML oder JSON Daten zurückgeliefert werden.[10]

[9] vgl. hierzu: Webentwicklung mit CakePHP, 2. Auflage, O'Reilly, Seite 7
[10] vgl. hierzu: `http://book.cakephp.org/3.0/en/controllers.html`

1.3.5 Component

Komponenten (Components) sind in sich geschlossene Bereiche innerhalb einer Applikation, die eine bestimmte Funktionalität kapseln und über die Grenzen eines Controllers hinaus verfügbar machen. Sollen bestimmte logische Prozesse in verschiedenen Teilen einer Anwendung zur Verfügung stehen - insbesondere in unterschiedlichen Controllern- so ist es sinnvoll diese in eine Komponente auszulagern.[11] Die Möglichkeit mit Komponenten zu arbeiten setzt das DRY Paradigma konsequent um.

1.3.6 Shell

CakePHP bietet die Möglichkeit Konsolenanwendungen zu schreiben. Dies ist nützlich für Anwendungen die per Cronjob ausgeführt werden sollen oder für solche die nicht aus einem Browser erreicht werden müssen bzw. sollen. `vgl. hierzu http://book.cakephp.org/3.0/en/console-and-shells.html` Eine der wichtigsten Funktionalität der Cake Shell ist das 'Backen' (Baking). Gemeint ist damit die automatische Generierung von Code. Der Befehl `bin/cake bake` erstellt, je nach gewählter Option, ganze MVC Grundgerüste, Controller- oder Model-Klassen, Plugin Verzeichnisstrukturen oder Shell-Klassen. Einzelne Funktionalitäten einer Shell Klasse können in Tasks ausgelagert werden.

1.3.7 Einschätzung

2 Analyse der Aufgabe und der Anforderungen

2.1 Bewertung von theoretischen Ansätzen, Konzepten, Methoden, Verfahren

Im folgenden sollen die verwendeten Technologien bewertet werden. Wie gut sind sie jeweils für den Einsatzzweck geeignet? Bewertungskriterien sind:

2.1.1 CakePHP-Framework

2.1.2 SQL Datenbank

2.1.3 BMECat Format

Allgemeine Vorteile die sich aus dem XML-Format ergeben sind die gleichzeitige Mensch- und Maschinenlesbarkeit sowie die Möglichkeit das Dokument gegen ein XML-Schema testen zu können. So kann schon direkt nach der Erzeugung des BMECat Dokumentes überprüft werden,

[11] vgl hierzu: CakeBuch Webentwcik, Seite 223

ob die geschriebenen Elemente vom richtigen Datentyp sind und das Dokument der in der XSD Datei festgelegten Struktur folgt. Weitere Vorteile speziell des BMECat Standards sind[12]:

- konfigurierbare Produkte sind abbildbar
- mehrsprachige Kataloge sind in einem Katalogdokument abbildbar
- Übermittlung multimedialer Datenelemente ist möglich (z.B. Produktvideos)
- gilt zumindest in Deutschland als etabliertes Katalogaustauschformat

2.1.4 Datenübertragung zu Mercateo

Die Übertragung der Katalogdatei zum Mercateo-Server geschieht über FTP. Neue Dateien werden alle 30 Minuten vom Mercateo-System verarbeitet.
Vorteile:

- einfach anzuwenden.
- Eine korrekte Datenübertragung ist durch die Fehlerbehandlung von TCP gewährleistet.

Nachteile:

- Datenübertragung nicht nach außen abgesichert.
- Übertragene Daten können mitgelesen und manipuliert werden.
- Benutzerkennung und Passwort können abgefangen werden

Fazit:
Nicht optimal, vor allem aus Sicherheitsgründen. Zudem Fehleranfällig, wenn die Ordnerstruktur- und Dateinamenskonventionen von Mercateo nicht eingehalten werden [13].

2.2 Funktionale und nichtfunktionale Anforderungen

-

2.3 Informelle Aufgabenbeschreibung

Ziel der Arbeit ist es die von der Software iTool aus verwaltbaren, in verschiedenen Tabellen einer SQL-Datenbank gehaltenen Produkt-, Katalog- Kategorie- und Herstellerdaten in ein von Mercateo verarbeitbares Format (dem BMECat) zu bringen. Dabei gilt es, den Anforderderungen der Spezifikationen sowohl das BMECat, als auch der besonderen Anforderungen seitens Mercateo zu genügen. Es soll möglich sein, die erwähnten Daten aus dem UI des iTool heraus nach dem CRUD-Prinzip zu bearbeiten. Die eigentliche Erstellung der unterschiedlichen Kataloge (neuer Katalog bzw. Produktupdatekatalog) erfolgt dabei (automatisiert) über die

[12] vgl. hierzu: `http://wiki.prozeus.de/index.php/BMEcat`
[13] vgl. hierzu `http://www.mercateo.com/support/verkaufen/katalog-allgemeine-informationen/datenuebertragung-per-ftp/`

CakePHP Shell. Kataloge können dabei für unterschiedliche Verkäufer erstellt werden. Zudem soll es Meracteo ermöglicht werden Bestandsdaten zu einer bestimmten Artikelnummer über ein Webinterface abzurufen.

2.4 Zielstellung

Folgende Funktionalitäten sollen implementiert werden:

- Die in iTool hinterlegten Produkt- bzw. Herstellerdaten sollen in ein gültiges und vollständiges BMECat Dokument entsprechend der Mercateo Anfoderungen überführt werden. Dabei ist insbesondere auf die Unterschiede und Besonderheiten der notwendigen beiden Transaktionsarten `T_NEW_CATALOG` - also die Erstellung eines neuen Kataloges - und `T_UPDATE_PRODUCTS` - also der Änderungen von Produktdaten, sowie dem löschen und neu erstellen von Produkten - zu achten .
  - Gültig bedeutet in diesem Fall, dass Struktur und Inhalt des Dokuments fehlerfrei gegen die entsprechende XSD Datei laufen, d.h. die Felder müssen in der richtigen Reihenfolge unter Beachtung der Datentypen und Längenbegrenzungen sowie Formatlimitierungen (z.B. keine Sonderzeichen in der SKU (o.ä.)) geschrieben werden.
  - Vollständig heißt, dass zum einen mindestens jene Felder im BMECat Dokument vorkommen, die die BMECat Spezifikation verlangt.Zusätzlich müssen jene Felder vorkommen, die die Mercateo Spezifikation erfordert und zwar unter zusätzlicher Beachtung der Limitierungen bzw. Besonderheiten jener Spezifikation.
- Die Produktkategoriestruktur des iTool soll in das Kataloggruppensystem des BMECat überführt werden.
- die Implementierung der Katalogerstellungslogik erfolgt in einer Cake Shell. Liegt noch kein Katalog vor, wird ein neuer Katalog erstellt; ein Updatekatalog wird erstellt, wenn es Änderungen bei den Produktdaten gab.
- Kataloge können für unterschiedliche Verkäufer erstellt werden.
- Die Produkt und Katalogdaten können über die Benutzeroberfläche des iTool eingesehen bzw. verändert werden.
- Es soll Mercateo ermöglicht werden Bestandsdaten zu den im Katalog vermerkten Produkten über ein Webinterface abzurufen.
- Bei der Katalogerstellung ist darauf zu achten, dass es zu keinen Arbeitsspeicherüberläufen kommen kann.

3 Entwurf

3.1 Katalogerstellung

Die Erstellung der BMECat Dokumente wird über eine CakePHP Shell realisiert. Das hat den Vorteil, dass diese über einen Cronjob automatisiert werden kann werden kann. Um die Shell

im Bedarfsfall leicht erweitern zu können und nicht zu überladen wird die Logik in einen Shell-Task `prepareCatalog` ausgelagert.Es soll möglich sein beim Aufruf des Tasks die *id* des `core_sellers` zu übergeben, dessen Produkte exportiert werden sollen. Die Spezifikation des BMECat verlangt, dass im Katalogdokument bestimmte Informationen zur Katalogversion, dem Katalognamen, der Währung *etc.* aufgeführt werden. Diese Daten werden in der Tabelle `mercateo_accounts` gespeichert und können über die GUI des iTool eingesehen, erstellt, gelöscht und geändert werden.

3.1.1 Die Tabelle `mercateo_accounts`

Eine Übersicht der in `mercateo_accounts` gespeicherten Werte bietet folgende Tabelle, wenn nicht anders angegeben entspricht das BMECat Element der Spaltenbezeichnung (`catalog_id` == <CATALOG_ID >)

Spalte	Erläuterung	BMECat Element
id	Primärschlüssel	×
core_seller_id	Fremdschlüssel auf `core_sellers`	×
core_marketplace_id	Fremdschlüssel auf `core_marketplaces`	×
core_currency_id	Fremdschlüssel auf `core_marketplaces`	<CURRENCY >
supplier_name	Name des verkaufenden Unternehmens	✓
catalog_id	Eindeutiger Bezeichner des Produktkataloges	✓
catalog_version	Version des Produktkataloges	✓
catalog_name	Beliebiger Name, der den Produktkatalog beschreibt	✓
group_system_id	Kennung des Kataloggruppensystems	✓
group_system_name	Name des Kataloggruppensystems	✓
group_system_description	Beschreibung des Kataloggruppensystems	✓
created	Timestamp, wann der Eintrag erzeugt wurde	×
modified	Timestamp, wann der Eintrag geändert wurde	×

3.1.2 Die Tabelle `mercateo_products`

Im Zentrum der Katalogerstellung steht die Tabelle mercateo_products. Sie dient als Zwischenspeicher für die in `core_products` gespeicherten Daten und gibt Auskunft darüber, ob (und wann) Produkte geändert, gelöscht oder neu hinzugefügt wurden. Dadurch wird sie zum zentralen Element zur Umsetzung der Transaktionen `T_NEW_CATALOG` und `T_UPDATE_PRODUCTS`.

Spalte	Erläuterung
id	Primärschlüssel
core_product_id	Fremdschlüssel auf core_products Tabelle
core_categorie_id	Kategorie ID
status	Der Status des Eintrages
sku	Die SKU des Artikels
title	Die 'DESCRITION_SHORT' des Artikels
created	Timestamp, wann der Eintrag erzeugt wurde
modified	Timestamp, wann der Eintrag geändert wurde

Die Spalten 'sku','core_category_id' & 'title' sind notwendig einen Artikel in einem BMECat Dokument als *gelöscht* auszeichnen zu können. Die Spalte 'status' akzeptiert vier *Zustände*:

Zustand	Erläuterung
`new`	Produktdaten wurden neu in `core_products` angelegt.
`update`	Produktdaten wurden geändert.
`delete`	Produktdaten wurde aus `core_products` gelöscht.
`active`	Produktdaten wurden in das aktuelle BMECat Dokument übernommen.

Diese Zustände sind die Werte die das Attribut `mode` des BMECat Elements `ARTICLE` annehmen kann. Gleichzeitig geben sie an dieser Stelle Auskunft darüber ob ein in `core_products` gespeicherter Datensatz neu ist bzw. gelöscht oder verändert wurde. Jene Datensätze die in das aktuelle BMECat Dokument geschrieben wurden, werden mit 'active' markiert.

3.1.3 Allgemeiner Programmablauf

3.1.4 Katalogerstellungslogik bei der Transaktion `T_NEW_CATALOG`

Mit dem ersten Programmaufruf wird die `mercateo_products` Tabelle mit den entsprechenden Werten aus `core_products` initialisiert. Allen Einträgen wird dabei zunächst der Status `new` zugewiesen. Zugleich werden Datum & Uhrzeit der Initialisierung in die Tabelle `core_configurations` geschrieben. Diese Tabelle ist bereits in iTool angelegt und dient als zentraler Ort um Konfigurationsdaten abzulegen. Anschließend wird eine BMECat Datei mit der Transaktion `T_NEW_CATALOG` erstellt, die Katalogversion in `core_configurations` geschrieben und der Status aller Einträge in `mercateo_products` auf `active` gesetzt.

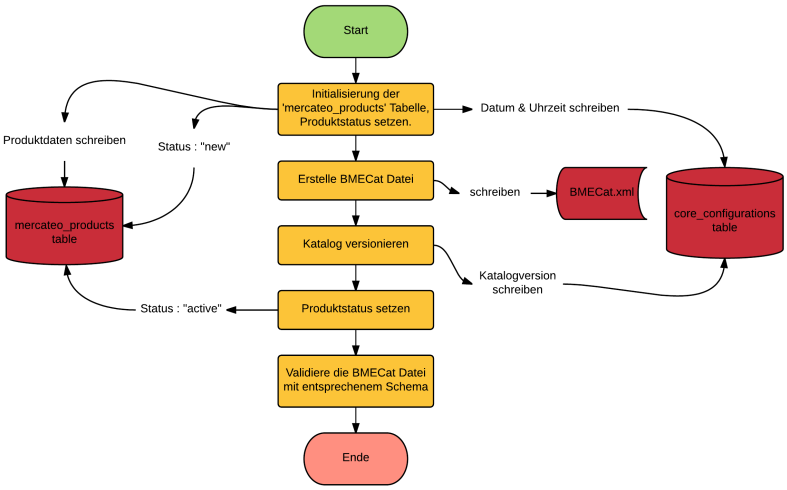

Abbildung 11: Programmlogik bei der Transaktion `T_NEW_CATALOG`

Nachdem das Katalogdokument geschrieben wurde wird es mithilfe des XSD-Schemas (`bmecat_new_catalog_1_2.xsd`) überprüft.

3.1.5 Katalogerstellungslogik bei der Transaktion `T_UPDATE_PRODUCTS`

Bei jedem erneuten Aufruf des Programmes wird zunächst überprüft ob Einträge in der `core_products` Tabelle gelöscht, neu hinzugefügt oder geändert wurden. Letzteres geschieht mit Hilfe der Tabelle `core_product_updates` in der jede Änderung an einem `core_product` mit dem Zeitstempel der Änderung erfasst wird. Ist einer der Fälle eingetreten wird der Status des Eintrages in der `mercateo_products` Tabelle entsprechend gesetzt.

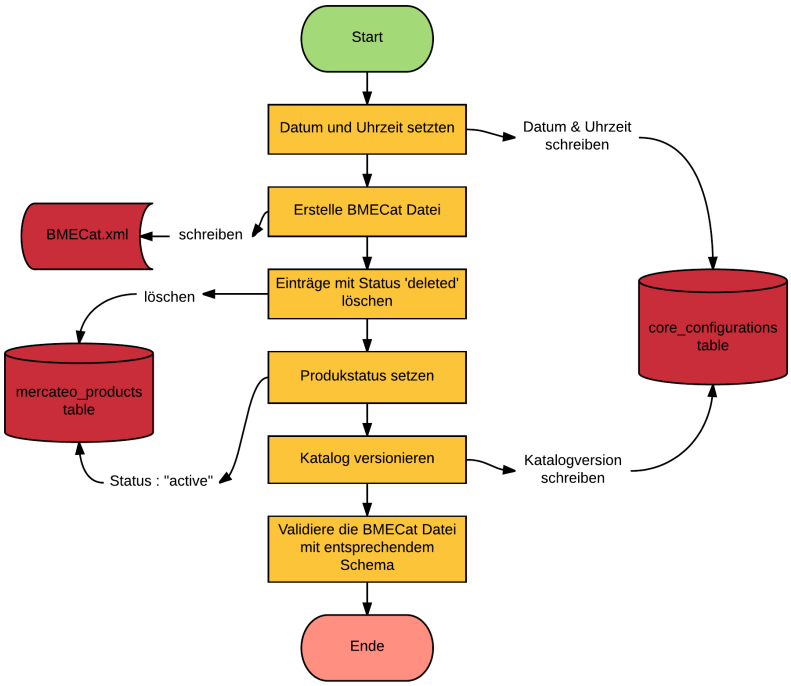

Abbildung 12: Programmlogik bei der Transaktion `T_UPDATE_PRODUCTS`

Anschließend werden wiederum Datum & Uhrzeit der Erstellung des Katalogdokumentes gesetzt und selbiges erstellt. Jene Einträge in `mercateo_products`, die den Status `delete` haben, werden gelöscht, danach wird der Status der Übrigen auf `active` gesetzt und das Attribut `prev_version` des Elementes `T_UPDATE_PRODUCTS` mit dem initialen Wert von '0' in die `core_configurations` Tabelle geschrieben. Dieser Wert wird bei jedem Updatevorgang um '1' erhöht. Abschließend erfolgt die Validierung des Dokumentes mithilfe des Schemas `bmecat_update_products_1_2.xsd`.

3.2 Bestandsdatenabfrage

Die Bestandsdatenabfrage wird mit einem einfachen Controller realisiert dessen `index()` Funktion als Parameter die angefragte SKU hat. Von Seiten Mercateo kann so eine URL der Form `http://itool.local/mercateo/availability/12` aufgerufen werden. Ist die SKU System vorhanden, wird die Bestandsmenge als Integer Wert zurückgeliefert, ist die angefragte SKU nicht im System, wird eine Fehlermeldung ausgegeben.

4 Implementierung

Die Implementierung gliedert sich in 2 Bereiche, die Konsolenanwendung zur Generierung des Katalogdokumentes und die Bestandsdatenabfrage.

4.1 Die BMECat Creator Shell

Mit der BMECat Creator Shell wird der Entwurf zur Erzeugung eine BMECat Katalogdokumentes umgesetzt. Die eigentliche Shell Klasse `BmeCatCreatorShell` dient in der Implementierung nur dazu das Subcommando `prepareCatalogTask` aufzurufen und sicherzustellen, dass die notwendigen Argumente übergeben werden.

```
Usage:
cake mercateo.bme_cat_creator [subcommand] [-h] [-q] [-v] <Core Seller Id> [<Delete
    flag>]
Subcommands:
prepareCatalog  Creates or updates a BMECat Catalog file.

To see help on a subcommand use `cake mercateo.bme_cat_creator [subcommand] --help`

Options:
--help, -h     Display this help.
--quiet, -q    Enable quiet output.
--verbose, -v  Enable verbose output.

Arguments:
Core Seller Id  The ID of the Seller for whom the BMECat shall be
                created
Delete flag     If "del" is set as second Argument, the
                mercateo_products table will be truncated
                (optional)
```

Dabei liegt dem Programm folgende Aufrufhierarchie zugrunde:

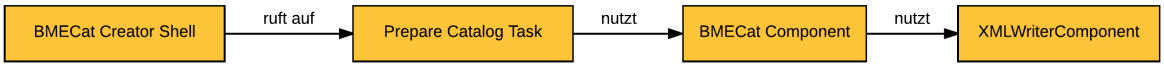

Abbildung 13: Aufrufhierarchie BMECatCreatorShell

Im Folgenden werden die einzelnen Klassen vorgestellt.

4.2 XMLWriterComponent

Die XMLWriterComponent Klasse ist insofern wichtiger Bestandteil der Implementierung, als das ohne sie nur auf umständlicherem Wege XML geschrieben werden kann. Hier seien nun in Kürze jene Methoden vorgestellt, die von der Klasse BMECatComponent genutzt werden:

1. public function openXmlWriter($filePath$,rootElement, $attributes = null$,doctype = null) Ermöglicht das Anlegen einer neuen XML Datei mit der Option Attribute (z.B. die BMECat Version) zu übergeben, sowie über den DOCTYPE eine entsprechende .dtd Datei zu referenzieren.

2. public function closeXmlWriter()
   Schließt das XML Dokument ab.
3. writeXmlElement(*name*,value, *type* = "*text*",attributes = [])
   Schreibt ein XML Element mit dem übergebenen Wert und den dazugehörigen Attributen und schließt es sogleich ab. (Schreibt Start- und End Tag)
4. public function writeStartXmlElement(*name*,attributes = [])
   Öffnet ein XML Element und setzt die übergebenen Attribute. (Schreibt den Start-Tag)
5. public function writeEndXmlElement()
   Schließt das zuvor geöffnete Element ab. (Schreibt den End-Tag)

VORTEILE?

4.3 BMECatComponent

Die Klasse BMECatComponent dient dazu ein wohlgeformtes und gültiges XML Dokument entsprechend den BMECat- und Mercateo Vorgaben zu erstellen. Wie im Kapitel Grundlagen bereits beschrieben gliedert sich ein BMECat Dokument in 4 Bereiche und zwar den Header, das Kataloggruppensystem, die Auflistung der einzelnen Artikel und die Zuordnung der Artikel zu ihren Kategorien. Der BMECat Komponent stellt für jeden dieser Teilbereiche Funktionen bereit die in Folge erläutert werden sollen. Zur Orientierung dient eine Übersicht der Aufrufhierarchie.

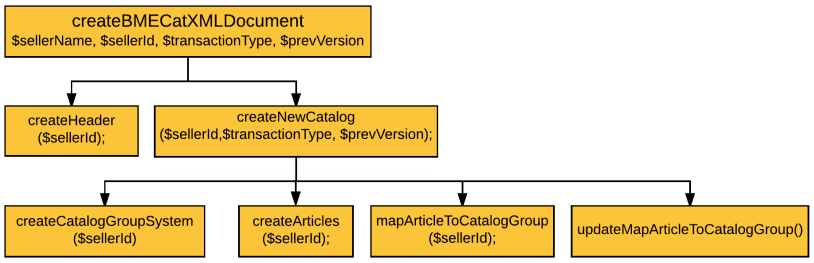

Abbildung 14: Aufrufhierarchie createBMECatXMLDocument

4.3.1 Erstellen des BMECat Dokumentes

Die Funktion `createBMECatXMLDocument($sellerName, $sellerId, $transactionType, $prevVersion)` dient der Erzeugung eines BMECat Dokumentes. Sie legt die die XML Datei mit der Namenskonvention `'Verkäufername_Erzeugungsdatum_Erzeugungszeit.xml'` an und schreibt Informationen zum XML Namensraum (xmlns) und zur Dokumenttypdefinition (dtd) in die XML-Deklaration. Von ihr werden die Methoden zur Erzeugung des Headers und der restlichen Abschnitte des BMECat Dokumentes aufgerufen. Anhand des

Parameters `$transactionType` wird entschieden ob bei dem zu erzeugenden Dokument die Transaktion `T_NEW_CATALOG` oder `T_UPDATE_PRODUCTS` umgesetzt werden soll.

4.3.2 Schreiben der Header Sektion

Die Methode `createHeader($sellerId)` schreibt die BMECat-Header-Sektion des Dokumentes. Alle benötigten Informationen, wie z.B. Herstellername oder Katalogversion werden dabei anhand der `sellerId` aus der `mercate_accounts` Tabelle geladen.

4.3.3 Schreiben des Kataloggruppensystems

Die Methode `createCatalogGroupSystem($sellerId)` steuert die Erzeugung des Kataloggruppensystems. Die von ihr aufgerufenen Methoden sind im wesentlichen für das Schreiben bestimmter XML Elemente zuständig.

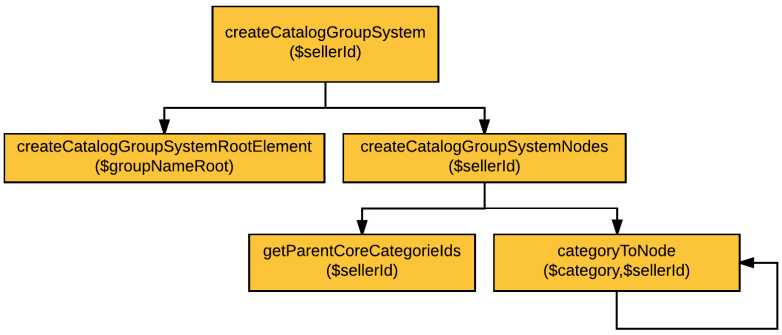

Abbildung 15: Aufrufhierarchie createCatalogGroupSystem

Das tatsächliche Abbilden der Katalogstruktur erfolgt in der Methode `categoryToNode( $category,$sellerId)`.

> Dazu ein kleiner Exkurs:
> CakePHP bietet die Möglichkeit einem Model ein sogenanntes Tree-Behaviour hinzuzufügen. Dieses basiert auf dem Nested-Set-Konzept, das es ermöglicht hierarchische Strukturen in relationalen Datenbanken abzubilden[14]. Die `core_categories` Tabelle bedient sich dieses 'Behavious', was es ermöglicht diesen Kategoriebaum rekursiv zu durchlaufen und dadurch das Kataloggruppensystem des BMECat abzubilden.

Die Methode `categoryToNode($category,$sellerId)` durchläuft, ausgehend vom Wurzelelement, alle Kindelemente und schreibt die entsprechenden Daten in das Dokument. Solange dabei die Anzahl der Kindelemente des gerade traversierten Elementes größer 0 ist wird dabei

[14] vgl. hierzu `https://www.sitepoint.com/hierarchical-data-database-2/`

dem Attribut `type` der Wert `node` zugewiesen. Gibt es keine Kindelemente mehr, wird der Wert auf `leaf` gesetzt.

```
<CATALOG_STRUCTURE type="node">
 <GROUP_ID>207</GROUP_ID>
 <GROUP_NAME>Auto -Motorrad - Flugzeug</GROUP_NAME>
 <PARENT_ID>202</PARENT_ID>
</CATALOG_STRUCTURE>
<CATALOG_STRUCTURE type="leaf">
 <GROUP_ID>210</GROUP_ID>
 <GROUP_NAME>Oldtimer</GROUP_NAME>
 <PARENT_ID>207</PARENT_ID>
</CATALOG_STRUCTURE>
```

4.3.4 Artikelerstellung

Die Methode `createArticles($sellerId)` aggregiert die zu schreibenenden Artikeldaten. Über einen `INNER JOIN` werden die Tabellen `mercateo_products` und `core_products` verbunden, so dass über die in `mercateo_products` hinterlegte `core_product_id` die entsprechenden Daten aus der `core_products` Tabelle nachgeladen werden können. Ist der Artikelstatus gleich *'new'* oder *'update'* wird die Methode `writeArticle($product, $articleMode)` aufgerufen, die die entsprechenden XML Elemente schreibt.

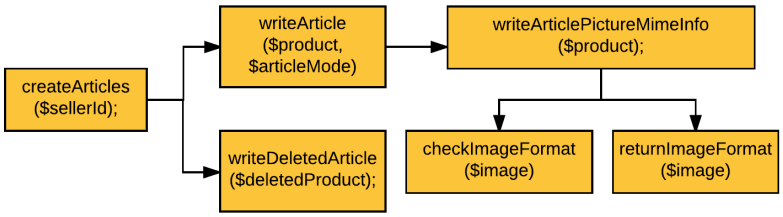

Abbildung 16: Aufrufhierarchie createArticle

Zusätzlich wird geprüft ob die dem Artikel zugeordneten Bilder der Mercateo Spezifikation entsprechen. Diese gestattet als Bildformate nur '.jpeg' und '.png'. Entsprechen die Bilddateien nicht diesem Format wird eine entsprechende Ausnahmebehandlung durchgeführt. Dabei wird der komplette Pfad der beanstandeten Datei zurückgeliefert, um es dem Anwender zu erleichtern diesen Fehler zu beheben.

```
2017-01-05 16:30:55 Info: Product with CoreProductId: 7147 updated
Exception: 'png' ist not allowed as File Extension.
Only .gif & .jpg Files are accepted by Mercateo-Marketplace
Check Image Path: https://bild-im-rahmen.com/wp-content/uploads/2016/03/s21r.png
```

Handelt es sich um einen gelöschten Artikel, wird die Methode `writeDeletedArticle($product)` aufgerufen. Diese schreibt die in `mercateo_products` hinterlegten, um einen Artikel als gelöscht auszeichnen zu können notwendigen Informationen - Die SKU & den Titel- in das Dokument.

4.3.5 Kategoriemapping

Mit der Methode `mapArticleToCatalogGroup($sellerId)` werden bei der Transaktion `T_NEW_CATALOG` die Artikel ihren Kategorien zugewiesen. Alle dazu notwendigen Informationen finden sich in der `mercateo_products` Tabelle.

Wird ein Update Katalog erstellt wird die Funktion `updateMapArticleToCatalogGroup()` aufgerufen. Sie setzt den bei der Transaktion `T_UPDATE_PRODCUTS` geforderten Attributwert für `mode` entsprechend der Angaben in der `mercateo_products` Tabelle. Mögliche Werte sind *'new'* für neu erstellte und *'delete'* für gelöschte Produkte.

5 prepareCatalogTask

6 Test

Hier kann vielleicht auch die Validation rein (MercateoAccountsTable)

1. Fowler, Martin ; Beck, Kent ; Brant, John ; Opdyke, William ; Roberts, Don ; Gamma, Erich: Refactoring : Improving the Design of Existing Code. 1. Aufl.. Amsterdam: Addison-Wesley, 2012.
2. Martin, Robert C.: Clean Code - Refactoring, Patterns, Testen und Techniken für sauberen Code : Deutsche Ausgabe. 1. Aufl.. Heidelberg: MITP-Verlags GmbH & Co. KG, 2013.
3. Hauer, Phillip ; philliphauer.de; http://www.philipphauer.de/study/se/design-pattern.php; Abgerufen am 29.3.2016;