

BERLINER HOCHSCHULE FÜR TECHNIK
FACHBEREICH VI - INFORMATIK UND MEDIEN
STUDIENGANG MEDIENINFORMATIK ONLINE
LUXEMBURGER STR. 10
13353 BERLIN

**Konzeption und Implementierung einer TYPO3-Extension zur
Analyse und Darstellung von statistischen Daten gegebener
Content Management Systeme**

BACHELORARBEIT ZUR ERLANGUNG DES GRADES EINES BACHELOR OF SCIENCE
(B.Sc.)

- *Bachelorarbeit* -

eingereicht von
Alexander Ullrich
Matrikel-Nr.: 770481

Betreuer:
Sebastian Kreideweiß
Prof. Dr. Tramberend

26. Dezember 2021

Inhaltsverzeichnis

1	Einleitung	1
1.1	Motivation	1
1.2	Zielsetzung	1
1.3	Struktur der Arbeit	2
2	Fachlicher und Technischer Hintergrund	3
2.1	WhatCMS.org - Wer ist das?	3
2.2	WhatCMS.org - Was kann man damit tun?	3
2.3	WhatCMS.org - API Schnittstelle	4
2.3.1	API Schnittstelle - Anfrage	4
2.3.2	API Schnittstelle - Antwort	5
2.4	CMScensus - Grundgedanke zur Idee	5
2.5	CMScensus - Namensgebung	5
3	Anforderungen	6
3.1	Funktionale Anforderungen	6
3.1.1	Mehrsprachigkeit	6
3.1.2	Vorschläge	6
3.1.3	Sicherheit	6
3.1.4	Import	6
3.1.5	Ausgabe	6
3.2	Technische Anforderungen	7
3.2.1	Grundsystem	7
4	Architektur und technische Konzeption	8
4.1	Domain-Driven Design	8
4.1.1	Das Domänenmodell	8
4.1.2	Ubiquitous Language (UL)	8
4.2	Bausteine des DDD	9
4.2.1	Entity	9
4.2.2	Value Object	9
4.2.3	Service	10
4.2.4	Factory	10
4.2.5	Repository	10
4.2.6	Aggregate	10
4.2.7	Relation	11
4.2.8	MVC	11
5	Umsetzung	12
5.1	Grundkonzept	12
5.2	Das Glossar	12
5.3	Domain Model ("Domänenmodell")	13

5.4	Vorbereitung ("Entwicklungsumgebung")	14
5.4.1	TYPO3 v11 installieren	14
5.4.2	Extension Builder installieren	16
5.4.3	Introduction Package installieren	16
5.4.4	External Data Import installieren	17
5.4.5	Cache leeren	18
5.5	Entwicklung der Extension	19
5.5.1	Erstellung Grundgerüst	19
5.5.2	Erstellung Datei Import	22
5.5.3	Erstellung Import über API Schnittstelle	29
5.5.4	Plugins für Content verfügbar machen	32
5.5.5	Erstellung Proposal Funktionalität	34
5.5.6	Erstellung Chart Funktionalität	37
6	Fazit und Ausblick	40
6.1	Fazit	40
6.2	Ausblick	41
7	Quellen und Literaturverzeichnis	42

Abbildungsverzeichnis

Abb. 1	WhatCMS.org - Screenshot Batch Hochschulen	3
Abb. 2	WhatCMS.org - Screenshot Chart Hochschulen	4
Abb. 3	WhatCMS.org - Aufbau API Request URL	4
Abb. 4	WhatCMS.org - JSON Antwort API Request	5
Abb. 5	TYPO3 - Roadmap	7
Abb. 6	Bausteine des Domain-Driven Designs	9
Abb. 7	Glossar - CMScensus	12
Abb. 8	Domänenmodell - CMScensus	13
Abb. 9	CMScensus - DDEV config.yaml	14
Abb. 10	CMScensus - ddev start	15
Abb. 11	CMScensus - TYPO3 Installation Schritt 1	15
Abb. 12	TYPO3 - Extension Builder	16
Abb. 13	TYPO3 - Introduction Package	17
Abb. 14	TYPO3 - External Data Import	18
Abb. 15	TYPO3 - Extension Builder new Model	19
Abb. 16	TYPO3 - Extension Builder Domänenmodell	20
Abb. 17	TYPO3 - Extension Builder Relation	20
Abb. 18	TYPO3 - lokal installierte Extensions	21
Abb. 19	CMScensus - ext_conf_template.txt	22
Abb. 20	CMScensus - Extension Configuration Fileimport	23
Abb. 21	CMScensus - external im TCA des URL Model	23
Abb. 22	CMScensus - mapping im TCA des URL Model	24
Abb. 23	CMScensus - Data Import Columns Mapping	24
Abb. 24	CMScensus - Data Import Process Steps	25
Abb. 25	CMScensus - CustomFileImportSetupStep Konstruktor	25
Abb. 26	CMScensus - CustomFileImportSetupStep Extends	26
Abb. 27	CMScensus - CustomFileImportSetupStep setUriParameter()	26
Abb. 28	CMScensus - CustomFileImportSetupStep setCategory()	27
Abb. 29	CMScensus - CustomizeStoragePIDStep setStoragePID()	28
Abb. 30	CMScensus - Data Import Synchronize	28
Abb. 31	CMScensus - external API im TCA des URL Model	29
Abb. 32	CMScensus - API Import PlanningAutoUpdateStep	30
Abb. 33	CMScensus - API Import CustomizeReferenceUIDMappingStep	31
Abb. 34	CMScensus - Content Plugin Registration	32
Abb. 35	CMScensus - Content Element Plugins	32
Abb. 36	CMScensus - Configure Plugin	33
Abb. 37	CMScensus - ProposalController addUrlFormAction()	34
Abb. 38	CMScensus - Fluid Template Proposal	35
Abb. 39	CMScensus - ProposalController createUrlAction()	36
Abb. 40	CMScensus - ChartController und AjaxController	37
Abb. 41	CMScensus - Fluid Template Show und Ajax	38
Abb. 42	CMScensus - Fluid Template Show und Chart Darstellung	39

1 Einleitung

In dieser Arbeit soll die Konzeption, Entwicklung und Implementierung einer TYPO3 Extension zur Analyse und Darstellung von statistischen Daten gegebener Content Management Systeme durchgeführt und anschaulich vorgestellt werden. Die entwickelte Extension soll anschließend im Onlinebetrieb produktiv zur Anwendung kommen und helfen, auf Grundlage der statistischen Daten über die Verteilung von benutzten CMS in einer bestimmten Kategorie entsprechende Entscheidungen zu treffen.

1.1 Motivation

Um digitale Inhalte einer Webseite unabhängig vom Design, der verwendeten Technologie, sowie dem technischen Aufbau gemeinsam erstellen, bearbeiten und organisieren zu können, wurden sogenannte **”CMS - Content Management System”**¹ als Softwarelösung entwickelt.

Wie beim Onlineanbieter **”CMS Matrix”** [2] ersichtlich, gibt es heutzutage mehr als 1300 verschiedene CMS, welche die unterschiedlichsten Ansätze in Technologie, Komplexität, der Größe der Community, sowie der Weiterentwicklung und dem Support des bestehenden Systems beinhalten und verfolgen. Es wird schnell erkennbar, daß man nicht jedes CMS in seiner Funktionalität beherrschen und kennen kann, was die Auswahl des richtigen CMS für den eigenen Anwendungsfall erschwehrt.

Da verschiedene CMS für die unterschiedlichsten Anwendungsfälle entwickelt worden sind, dort auch ihren Schwerpunkt finden und diese CMS bereits auf online Plattformen stabil laufen, wäre die Information über das jeweils eingesetzte CMS eine große Entscheidungshilfe bei der Suche nach dem geeigneten System für den eigenen Anwendungsfall.

1.2 Zielsetzung

Im Rahmen dieser Arbeit wird eine TYPO3 Extension als Prototyp entwickelt, wobei die einzelnen Entwicklungsschritte und die eingesetzten Technologien dokumentiert und vorgestellt werden.

Die Extension wird für TYPO3 (Version 11) [3] erstellt und anschließend installier- und ausführbar sein. Sie wird die Möglichkeit zur Eingabe neuer Kategorien und URL's, sowie die visuelle Auswertung der statistischen Daten über die Verteilung von benutzten CMS einer Kategorie in Tabellen- und Chartform für Frontendbenutzer zur Verfügung stellen.

Die Grundlage für die Auswertung der statistischen Daten liefern die von WhatCMS.org [4] übertragenen Informationen, welche in der zu entwickelnden Extension über API Schnittstelle oder eine entsprechende Datei im JSON Format importiert werden können.

¹Englisch für **”Inhaltsverwaltungssystem”** [1]

1.3 Struktur der Arbeit

Diese Arbeit gliedert sich in sechs Kapitel, welche die Entwicklung des Prototyp einer TYPO3 Extension von der Idee bis zur einsatzfähigen Anwendung beschreiben.

Nachdem in der Einleitung Informationen zur Motivation und Zielsetzung benannt worden, wird im zweiten Kapitel auf den fachlichen und technischen Hintergrund der Anwendung eingegangen. Besonderes Augenmerk fällt dabei auf die Quelle der Daten, sowie die Beschreibung der zugrunde liegenden Schnittstelle.

Im dritten Kapitel der Arbeit wird auf die funktionalen und technischen Anforderungen der Anwendung eingegangen. Es wird neben "Mehrsprachigkeit" und "Sicherheit" auch die "Mindestanforderung" an das jeweilige Grundsystem beschrieben.

Das vierte Kapitel widmet sich dem theoretischen Teil der Arbeit. Es wird in seiner ausführlichen Beschreibung die Architektur und technische Konzeption des bekannten "Domain-Driven Design", inklusive der Betrachtung aller einzelnen Bausteine beinhalten.

Ab dem fünften Kapitel wird die Umsetzung des praktischen Teils der Arbeit, vom Grundkonzept über das Domänenmodell bis hin zum fertigen Code der Anwendung, mit dem kompletten Entwicklungszyklus beschrieben und entsprechend vorgestellt.

Der abschließende Teil wird sich im sechsten Kapitel mit dem kritischen Rückblick auf das gesamte Projekt beschäftigen. Mit Fazit und Ausblick werden gesammelte Erkenntnisse und Ideen für die zukünftige Weiterentwicklung des TYPO3 Extension Prototyps aufgeführt.

2 Fachlicher und Technischer Hintergrund

Im folgenden wird die Quelle der zugrundeliegenden Daten zur statistischen Erhebung, die verwendete Technologie zum Austausch der Daten, sowie der Grundgedanke zur Idee und Festlegung des Namen für die Extension beschrieben.

2.1 WhatCMS.org - Wer ist das?

WhatCMS.org [5] beschreibt sich selbst als Anbieter, welcher Anfragen von Benutzern aus der ganzen Welt bearbeitet, die mehr über die von ihnen verwendeten Webseiten erfahren möchten.

Um diese Informationen so genau wie möglich zur Verfügung stellen zu können, verwendet WhatCMS.org eigene Erkennungsalgorithmen und hat eine Reihe von Qualitätskontrollsystemen eingerichtet.

Ein besonderes Markenzeichen des von WhatCMS.org verwendeten Erkennungsalgorithmus ist die Fähigkeit, Content Management Systeme pro Seite, statt pro Domain zu erkennen.

2.2 WhatCMS.org - Was kann man damit tun?

Nachdem man sich als Nutzer auf der Webseite registriert und anschließend angemeldet hat, besteht die Möglichkeit sogenannte "Batches" anzulegen. Eine Batch ist eine Art Kategorie (siehe Abbildung 1), welcher man die verschiedensten URL [6] zuweisen kann, um anschließend durch den Erkennungsalgorithmus von WhatCMS.org das dahinter laufende CMS und die jeweilige Technologie ermitteln zu lassen.

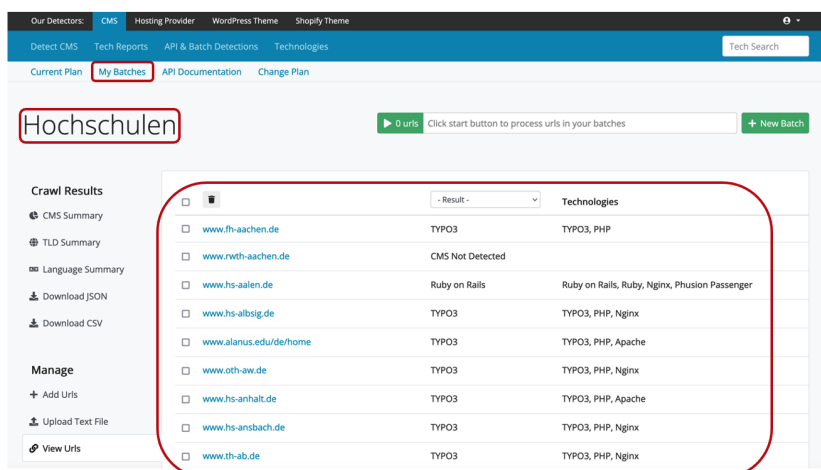


Abbildung 1: WhatCMS.org - Screenshot Batch Hochschulen

Sobald der Erkennungsalgorithmus alle CMS einer Batch erkannt hat, besteht die Möglichkeit die Daten in verschiedenen Dateiformaten zu exportieren oder auch visuell in einem Verteilungschart (siehe Abbildung 2) anzeigen zu lassen.

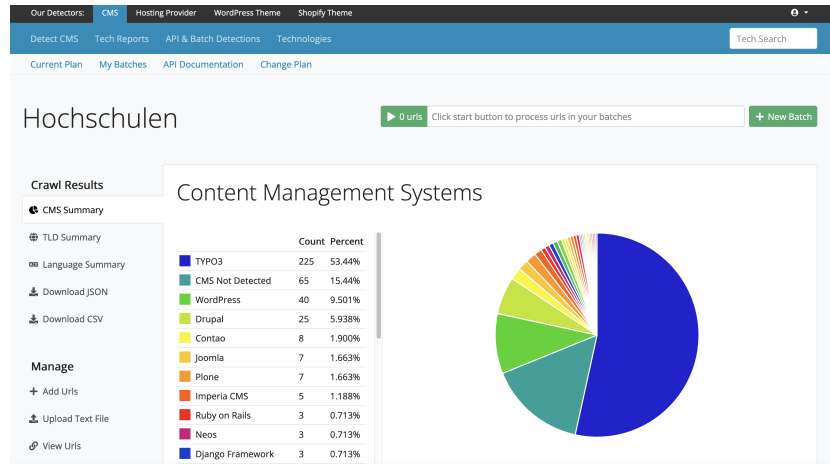


Abbildung 2: WhatCMS.org - Screenshot Chart Hochschulen

2.3 WhatCMS.org - API Schnittstelle

Neben der Visualisierung der Daten und der Erkennung des jeweiligen CMS, sowie der jeweiligen Technologie hinter einer URL bietet WhatCMS.org noch die Möglichkeit der Abfrage über eine API Schnittstelle.

2.3.1 API Schnittstelle - Anfrage

Wie in der Dokumentation [7] von WhatCMS.org beschrieben, wird die Anfrage über einen GET Request an den API Server von WhatCMS.org gestellt. Die Abfrage URL des API Server hat dabei der Struktur wie in Abbildung 3 ersichtlich zu erfolgen. Zuerst kommt die URL des API Server, gefolgt vom privaten API Schlüssel und der abschließenden URL, für welche man das CMS ermitteln möchte.



Abbildung 3: WhatCMS.org - Aufbau API Request URL

2.3.2 API Schnittstelle - Antwort

Wurde die Anfrage erfolgreich vom API Server entgegengenommen und bearbeitet, so erhält man eine Antwort im JSON [8] Format mit allen wichtigen Informationen zur angefragten URL zurück (siehe Abbildung 4). Diese strukturierten Daten können nun ausgewertet und weiterverarbeitet werden.

```
{
  "request": "https://whatcms.org/API/CMS?key=YOUR-API-KEY&url=en.wikipedia.org",
  "request_web": "https://whatcms.org/?s=en.wikipedia.org",
  "result": {
    "code": 200,
    "msg": "CMS Found",
    "id": 8,
    "name": "MediaWiki",
    "confidence": "high",
    "cms_url": "https://whatcms.org/c/8_MediaWiki"
  },
  "private": false
}
```

Abbildung 4: WhatCMS.org - JSON Antwort API Request

2.4 CMSensus - Grundgedanke zur Idee

Möchte man die Ergebnisse seiner WhatCMS.org Batches gern der breiten Öffentlichkeit zur Analyse und Auswertung zugänglich machen, so ist dies nur möglich, wenn man seine privaten Zugangsdaten von WhatCMS.org ebenfalls öffentlich macht.

Eine weitere Schwierigkeit liegt in der Tatsache begründet, daß jeder öffentliche Besucher andere Interessen hat und ein wildes Bearbeiten der Batches früher oder später zu großem Chaos führt.

CMSensus wird eine TYPO3 Extension, welche dieses Problem auf elegante Art und Weise löst, indem es die Daten von WhatCMS.org verwendet, weiter verarbeitet und auf jeder beliebigen TYPO3 Website darstellen kann. Desweiteren wird CMSensus die Möglichkeit der Vorschläge neuer Kategorien und URLs durch entsprechende Frontendbenutzer beinhalten und damit die Funktionalität erweitern.

2.5 CMSensus - Namensgebung

Da Census für Volkszählung [9] steht und eine gesetzlich angeordnete Erhebung statistischer Bevölkerungsdaten ist, bot sich der Name zur Verwendung in Kombination mit CMS als **”CMSensus”** an und wird damit als Name für die neue TYPO3 Extension verwendet.

3 Anforderungen

Nachfolgend werden die Anforderungen an die TYPO3 Extension beschrieben, welche sich aus den im Vorfeld geführten Meetings mit Herrn Kreideweiß, in der Rolle als "Product Owner" von CMSensus ergaben. Sie wurden mit dem Gedanken erstellt, ein Minimal-Viable-Product² zu entwickeln, welches anschließend auch eine produktive Anwendung finden soll.

3.1 Funktionale Anforderungen

3.1.1 Mehrsprachigkeit

Da die Extension nicht nur im deutschsprachigen Raum, sondern auch international Anwendung finden soll, muss die Möglichkeit der Mehrsprachigkeit implementiert sein.

3.1.2 Vorschläge

Frontendbenutzer sollen Vorschläge für neue Kategorien und dazugehörige URLs unterbreiten können. Damit die vorgeschlagenen URLs der neuen Kategorie zugeordnet werden, muss diese nach absenden des Kategorievorschlages mit auswählbar sein. Bei Eingabe einer fehlerhaften URL soll der Nutzer durch eine entsprechende Warnung darauf hingewiesen werden.

3.1.3 Sicherheit

Der Administrator muss die Möglichkeit haben, die Extension so zu konfigurieren, daß neue Vorschläge nur von angemeldeten Frontendbenutzern erstellt werden können. Damit soll die Möglichkeit des SPAM, durch Eingabe unzähliger unnützer Kategorien und URLs von Spaßnutzern unterbunden werden.

3.1.4 Import

Der Import von den WhatCMS.org gelieferten Daten soll sowohl über eine Datei im JSON Format, als auch über eine Anfrage an die WhatCMS.org API Schnittstelle mit Rückgabe im JSON Format funktionieren.

3.1.5 Ausgabe

Der Redakteur hat die Möglichkeit für jede Kategorie eine Seite zu erstellen, welche die prozentuale Verteilung der CMS der einzelnen URLs visuell als Tabelle, sowie als Chart darstellt.

²In einem Minimal Viable Product (MVP) wird ein Minimalumfang für ein Produkt (z.B. eine Software) über eine Anzahl von "Musst-Have-Features" definiert. Wenn das Produkt weniger Funktionen als diese definierten hat, erwirtschaftet es keinen Nutzen. Folgend kann das MVP dann schrittweise um weitere Funktionen erweitert werden. [34]

3.2 Technische Anforderungen

3.2.1 Grundsystem

Da die Extension nach Fertigstellung auf einer öffentlichen Webseite innerhalb eines CMS laufen soll, wurde sich für TYPO3 als Grundsystem entschieden. Dabei soll die Entwicklung auf der neuesten TYPO3 (Version 11) durchgeführt werden. Ein wichtiger Grund für diese Entscheidung liegt unter anderem darin begründet, daß TYPO3 (Version 11) wie in Abbildung 5 ersichtlich durch den sog. **”Long Term Support”** (LTS) [10] bis 2027 unterstützt wird. Daraus ergibt sich für die Extension eine entsprechende Langlebigkeit.

Support Times

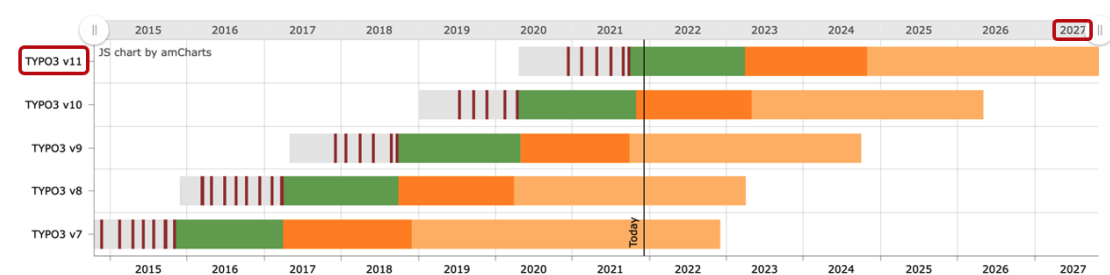


Abbildung 5: TYPO3 - Roadmap

Eine weitere wichtige Systemanforderung an die Extension beinhaltet, daß sie mindestens unter PHP7.4 [11] lauffähig und mit Composer v2 [12], sowie der TYPO3 Console [13] installierbar sein sollte.

Für die Versionierung soll das freie Open Source Kontrollsystem Git [14] zum Einsatz kommen. Das entsprechende ”Git Repository” wird auf ”GitHub” eingerichtet und ist unter der URL **”https://github.com/usadmin77/bachelorarbeit.git”** für ausgewählte Benutzer öffentlich erreichbar.

4 Architektur und technische Konzeption

Für die Umsetzung der Entwicklung der Extension wird das von Eric Evans in seinem 2003 veröffentlichten Buch "Domain-Driven Design: Tackling Complexity in the Heart of Software" [35] geprägte und vorgestellte "**Domain-Driven Design**" (DDD) verwendet.

4.1 Domain-Driven Design

Das Hauptziel der Verwendung liegt darin begründet, Probleme und Erwartungen des Kunden von Anfang an vollständig zu verstehen und die gesamte Bandbreite der jeweiligen Geschäftsaktivitäten in all seiner Komplexität als "**Domain**" zu bezeichnen und zu betrachten.

Wie in dem von Michael Schams im Jahr 2010 veröffentlichten Buch "TYPO3 Extbase: Moderne Extension-Entwicklung mit Extbase und Fluid" [36] (S. 35 ff.), basiert das Domain-Driven Design auf den folgenden zwei wichtigen Annahmen:

- Der Schwerpunkt des Softwaredesigns liegt auf der Sachlichkeit und der Fachlogik.
- Der Entwurf komplexer Zusammenhänge sollte auf einem Fachmodell basieren.

4.1.1 Das Domänenmodell

Es ist eine wichtige Anforderung an DDD, daß der Entwurf der Software über ein Model zu erfolgen hat [36] (S. 37). Ein Model ist dabei die vereinfachte Beschreibung der Wirklichkeit, was bedeutet, daß das Domänenmodell (engl. domain model) einen Plan für die darin enthaltenen Objekte, deren Eigenschaften und ihren Relationen untereinander darstellt.

4.1.2 Ubiquitous Language (UL)

Da die Sprache der Domäne die entscheidende Sprache ist, muss sie als zentrales Element beim Modellieren eindeutig festgelegt werden. Dies geschieht am besten über ein "Ubiquitous Language"³ Glossar, welches zu Beginn der Projektphase durch alle Projektbeteiligten gemeinsam erstellt wird. Ziel ist es, daß alle Projektmitglieder genau wissen, worum es sich bei den verschiedensten Begriffen der Domäne handelt und diese von den einzelnen Mitgliedern nicht mehr falsch verstanden und interpretiert werden.

³Englisch für "allgegenwärtige Sprache"

4.2 Bausteine des DDD

Wie in Abbildung 6 ersichtlich, setzt sich das Domain-Driven Design aus den verschiedensten Bausteinen zusammen, welche nachfolgend detailliert vorgestellt und beschrieben werden.

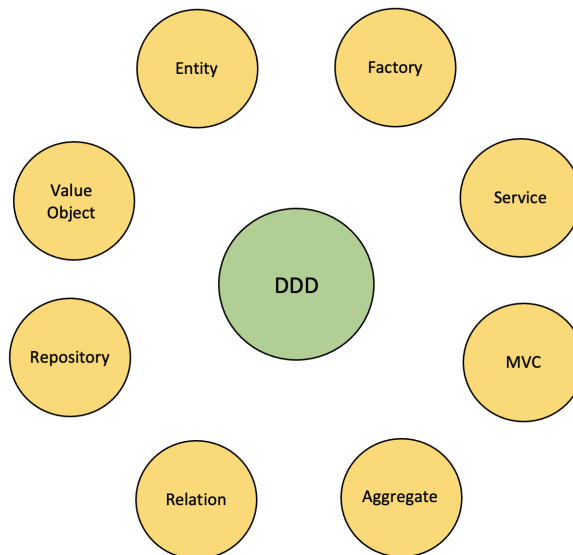


Abbildung 6: Bausteine des Domain-Driven Designs

4.2.1 Entity

Eine Entity (Entität) ist ein Domänenobjekt, welches eine globale und beständige Identität besitzt. Um es mit dem realen Leben zu vergleichen, könnte dies zum Beispiel die eindeutige Zuordnung einer Person sein, welche trotz Heirat oder Umzug immer die gleiche Nummer im Reisepass behält und darüber jederzeit identifiziert werden kann. Auch der Fingerabdruck ändert sich niemals und könnte in diesem Beispiel zur Bestimmung der Identität herangezogen werden.

4.2.2 Value Object

Ein Value Object wird durch die Summe der Eigenschaften definiert und hat im Gegensatz zur Entity keine globale Identität. Es wird über seine Eigenschaften angesprochen und ist immutable (nicht änderbar). Als Beispiel könnte hier die Farbe "Rot" als Value Object mit dem Farbcode "c00" genannt werden, welche nicht einfach im Namen in "Blau" geändert werden kann, ohne eine andere oder neue Farbe zu erhalten.

4.2.3 Service

Damit Entities und Value Objects als Grundbausteine des DDD untereinander kommunizieren können, existieren sogenannte Service-Klassen, welche auch mit dem Begriff **”Services”** bezeichnet werden. Als Beispiel aus der realen Welt kann man hier die Überweisung zwischen zwei Konto-Objekten nennen.

4.2.4 Factory

Ist die Erstellung der Domänobjekte selbst zu komplex (wenn z.B. Assoziationen notwendig sind, welche vom Domänobjekt nicht benötigt werden), oder soll die spezifische Erzeugung des Domänobjektes zur Laufzeit ausgetauscht werden, so kommen sogenannte **”Factories”** zum Einsatz. Sie dienen dazu, Domänobjekte als dezentrale, spezifische Factory-Objekte zu erstellen. Ein gutes Beispiel für den Einsatz einer Factory wäre die Instanziierung von vier Rad-Objekten im ersten Schritt, mit der anschließenden Verknüpfung des Auto-Objekts im zweiten Schritt und der abschließenden Rückgabe dieses Auto-Objektes an den jeweiligen Aufrufer.

4.2.5 Repository

Da Datenbanken und Speichermechanismen im DDD nicht existieren und DDD die Infrastruktur streng trennt, wurden sog. **”Repositories”** eingeführt. Ein Repository ist dabei als eine Art Blackbox zu betrachten, welches Objekte persistieren und lediglich die Suche nach Objekten ermöglichen kann. Es wird immer genau einem Objekt zugeordnet und erlaubt auch nur den Zugriff auf das entsprechende Objekt. Möchte man ein Objekt über ein anderes Repository ermitteln, so muss eine Beziehung zu dem gesuchten Objekt bestehen.

4.2.6 Aggregate

Ein Aggregate ist eine Zugriffskontrolle und wird im DDD als **”Aggregate Root”** bezeichnet. Wie ein Türsteher den Eingang eines Nachtclubs bewacht und entscheidet, welche Person Zutritt zu den Objekten (z.B. Tanzfläche oder alkoholische Getränke) innerhalb des Nachtclubs hat und wem der Zutritt verweigert wird, entscheidet der Aggregate Root über den Zugriff auf das jeweilige Objekt. In Extbase gibt es die klare Vorgabe, daß alle Objekte die **”Aggregate Root”** sind, ein Repository und einen Controller inklusive entsprechender Actions haben müssen.

4.2.7 Relation

Die Relation stellt eine Verbindung zwischen zwei Objekten dar. In Extbase gibt es dafür genau vier verschiedene **Relationen**, die sich wie folgt zusammensetzen:

1:1 Bei der **1:1 Relation** gibt es für ein Objekt nur genau ein anderes Objekt, mit dem eine Verbindung bestehen kann. Ein Beispiel aus der realen Welt wäre die Sozialversicherungsnummer, welche immer nur genau zu einer Person gehört.

1:n Bei der **1:n Relation** können von einem Objekt beliebig viele Verbindungen zu anderen Objekten bestehen. So hat zum Beispiel ein Blog beliebig viele Beiträge, wobei ein Beitrag immer nur genau zu einem Blog gehören kann.

n:1 Die **n:1 Relation** gibt an, daß es für beliebig viele Objekte genau ein Objekt gibt, mit dem eine Verbindung besteht. Als Beispiel kann der Beitrag eines Blogs genannt werden, welcher immer nur einen Author hat, wobei der Author im Gegensatz beliebig viele Beiträge schreiben kann.

m:n Bei der **m:n Relation** können von beliebig vielen Objekten Verbindungen zu beliebig vielen anderen Objekten bestehen. Ein gutes Beispiel dafür ist die Verwendung in CMSensus, wo einer Kategorie mehrere URLs zugewiesen werden können und im Gegensatz eine URL auch in mehreren Kategorien vorkommen kann.

4.2.8 MVC

MVC steht für das **Model-View-Controller** [15] Design Pattern und entspricht einer **Layered Architecture**⁴, welche im Domain-Driven Design gefordert ist. Die Aufgaben der einzelnen Komponenten des MVC sind dabei wie folgt gegliedert:

M Model ("das Modell"):

Diese Ebene kann man auch als Domain ("Domäne") bezeichnen, da sie die komplette Geschäftslogik und die Geschäftsdaten beinhaltet.

V View ("die Ansicht"):

Diese Ebene wird als sog. Ansichts- oder Template-Ebene bezeichnet, da sie für die Ausgabe der Daten zuständig ist. Sie wird in Extbase durch Templates und die dahinterliegende Template Engine "Fluid" realisiert.

C Controller ("die Steuereinheit"):

Diese Ebene darf weder Geschäfts-, noch View-Logik enthalten und kümmert sich ausschließlich um die Steuerlogik. Extbase setzt hier auf das "Slim Controller" Konzept, wobei der Controller selbst in Actions unterteilt ist, welche die Steueraufgaben übernehmen.

⁴Englisch für "Mehrschichtige Architektur" [37]

5 Umsetzung

5.1 Grundkonzept

Damit die zu entwickelnde Extension dem Sinn und Zweck gerecht wird, sollte sie mindestens folgende Funktionen bereitstellen:

- Ein Frontendbenutzer kann eine unbegrenzte Anzahl von neuen Kategorien und URLs vorschlagen.
- Beim vorschlagen der Kategorie oder URL kann zu dieser eine Beschreibung hinzugefügt werden.
- Jede vorgeschlagene Kategorie oder URL wird automatisch als Vorschlag markiert.
- Wird eine URL vorgeschlagen, so können zu dieser alle zur Verfügung stehenden Kategorien ausgewählt werden.
- Jede URL kann für die automatische Abfrage bei WhatCMS.org sowohl aktiviert, wie auch deaktiviert werden, wobei sie standardmäßig erst einmal deaktiviert ist.
- Neben dem URL Namen und der Beschreibung muss außerdem die Möglichkeit der Speicherung des verwendeten CMS gegeben sein.

5.2 Das Glossar

Damit alle Projektbeteiligten das gleiche Verständnis für die verwendete Terminologie haben, wird das in Abbildung 7 ersichtliche Glossar erstellt und für die Domänenobjekte die englische Sprache benutzt.

Terminologie	Deutsche Übersetzung	Beschreibung	Relevant für das Model
Batch	Charge	ist die bei WhatCMS.org verwendete Bezeichnung für die Sammlung aller URLs in einer bestimmten Kategorie.	Nein
Category	Kategorie	entspricht der Batch von WhatCMS.org und beinhaltet alle URLs, welche zu dieser Kategorie zusammengefasst werden sollen.	Ja
URL	URL	Standard für die Adressierung einer Website im World Wide Web.	Ja
Description	Beschreibung	Beschreibender Text zur Verwendung des jeweiligen Objektes.	Ja
WhatCMSType	WhatCMS Type	ist die von WhatCMS.org verwendete CMS Bezeichnung.	Ja
Proposal	Vorschlag	wird zur Markierung einer Kategorie oder URL als entsprechender Vorschlag verwendet.	Ja
Auto Update	Automatisches Update	wird zur Markierung einer URL für das ermitteln des dazugehörigen CMS beim nächsten Import Lauf zu WhatCMS.org verwendet.	Ja

Abbildung 7: Glossar - CMSensus

5.3 Domain Model ("Domänenmodell")

Auf Grundlage des Glossar wird nun das Domänenmodell erstellt. Wie in Abbildung 8 ersichtlich, ergeben sich daraus die folgenden Details:

- Es gibt zwei Domänenobjekte: *Url* und *Category*, welche vom Typ *Entity* sind (erkennbar an dem Buchstaben E in der rechten oberen Ecke).
- Bei beiden Objekten handelt es sich um *Aggregate Root* (erkennbar an den Buchstaben AR in der rechten unteren Ecke).
- Das Domänenobjekt *Url* besitzt die sieben Eigenschaften ("properties"): *name*, *description*, *is_proposal*, *only_next_auto_update*, *is_auto_update_planned*, *every_update* und *whatcmstype*.
- Das Domänenobjekt *Category* hat die Eigenschaften: *name*, *description* und *is_proposal*.
- Zwischen beiden Domänenobjekten besteht eine *m:n - Beziehung*, was so viel bedeutet wie, daß eine *Url* eine beliebige Anzahl von *Categories* enthalten kann. Außerdem kann einer *Category* eine beliebige Anzahl von *Urls* zugewiesen werden.

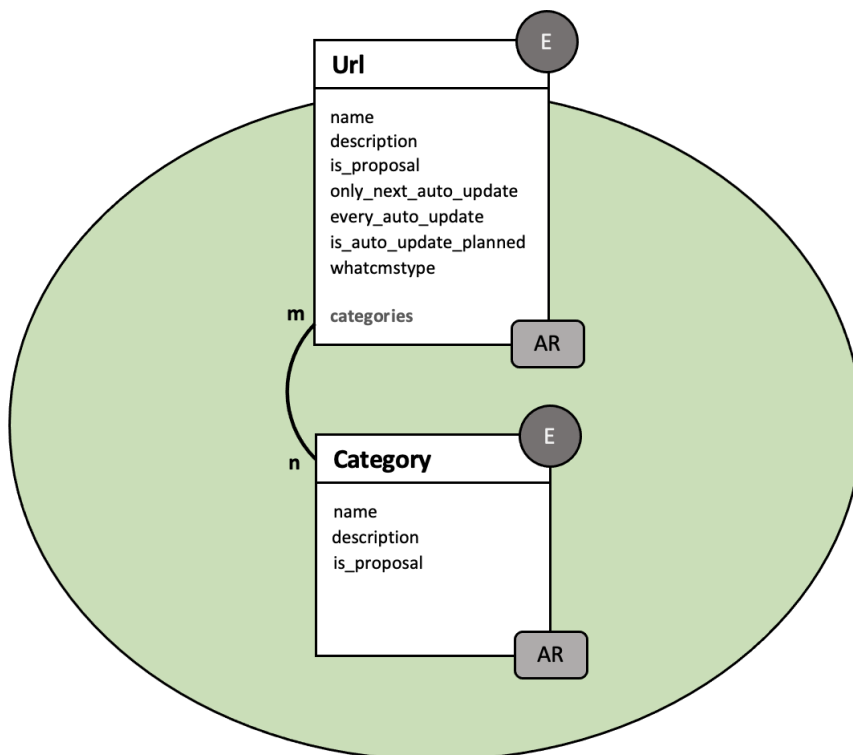


Abbildung 8: Domänenmodell - CMSensus

5.4 Vorbereitung ('Entwicklungsumgebung')

Um mit der Entwicklung der Extension beginnen zu können, wird in diesem Abschnitt die Einrichtung der Entwicklungsumgebung in kurzer Form beschrieben. Es wird angenommen, daß auf dem Entwicklungssystem PHP 7.4 [11], Composer v2 [12], DDEV [16] sowie Docker [17] installiert sind und in lauffähiger Form vorliegen.

5.4.1 TYPO3 v11 installieren

Wie in der Dokumentation [18] von TYPO3.org ausführlich beschrieben, wird im ersten Schritt der Ordner erstellt, in welchem unsere neue TYPO3 Installation liegen soll. Dazu wird die Shell geöffnet und folgender Befehl eingegeben:

```
mkdir TYPO3v11AUBA
```

nun wechseln wir in den eben erstellten Ordner durch Eingabe von:

```
cd TYPO3v11AUBA
```

und erzeugen das entsprechende DDEV TYPO3 Projekt mit nachfolgendem Befehl:

```
ddev config --project-type=typo3 --docroot=public --create-docroot
```

Da es festgelegte Anforderungen an das System gibt, müssen diese in der **config.yaml** Datei, welche im **Ordner .ddev** zu finden ist, so wie in Abbildung 9 ersichtlich eingestellt und angepasst werden.

```
1 name: TYPO3v11AUBA
2 type: typo3
3 docroot: public
4 php_version: "7.4"
5 webserver_type: nginx-fpm
6 router_http_port: "80"
7 router_https_port: "443"
8 xdebug_enabled: false
9 additional_hostnames: []
10 additional_fqdns: []
11 mariadb_version: "10.2"
12 mysql_version: ""
13 provider: default
14 use_dns_when_possible: true
15 composer_version: "2"
```

Abbildung 9: CMSscensus - DDEV config.yaml

Um mit der Einrichtung fortzufahren, wird der DDEV Container anschließend durch Eingabe des folgenden Befehls gestartet:

```
ddev start
```

Nach erfolgreichem Start des DDEV Container sollte so wie in Abbildung 10 ersichtlich die URL angezeigt werden, welche später in die Browser Adresszeile zum Starten der Installation eingefügt werden muss.

```
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
Recreating ddev-TYPO3v11AUBA-db ...
Recreating ddev-TYPO3v11AUBA-db ... done
Recreating ddev-TYPO3v11AUBA-web ...
Recreating ddev-TYPO3v11AUBA-dba ...
Recreating ddev-TYPO3v11AUBA-dba ... done
Recreating ddev-TYPO3v11AUBA-web ... done
ddev-router is up-to-date
Generating AdditionalConfiguration.php file for database connection.
Successfully started TYPO3v11AUBA
Project can be reached at http://typo3v11auba.ddev.site http://127.0.0.1:62511
usadmin@Alexanders-MacBook-Pro TYPO3v11AUBA %
```

Abbildung 10: CMSensus - ddev start

Nun wird die komplette "TYPO3 v11 CMS Base Distribution" via Composer geholt und alle Abhängigkeiten durch Eingabe des nachfolgenden Befehls aufgelöst:

```
ddev composer create 'typo3/cms-base-distribution:^11'
```

Bevor mit der Installation von TYPO3 über den Browser begonnen werden kann, muss noch die leere Datei "FIRST_INSTALL" im public Ordner durch Eingabe des touch Befehls erstellt werden:

```
ddev exec touch public/FIRST_INSTALL
```

Wurden alle Schritte erfolgreich durchgeführt, so sollte nun die Installation von TYPO3 über den Browser möglich sein und wie in Abbildung 11 zu sehen, durch Eingabe der beim Start generierten URL in die Browser Adresszeile zum ersten Installationschritt führen.

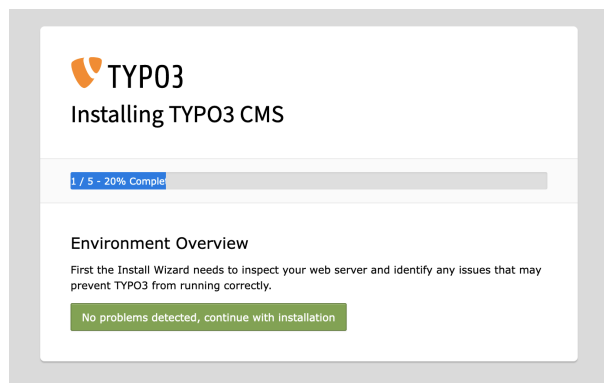


Abbildung 11: CMSensus - TYPO3 Installation Schritt 1

Da die restlichen Schritte der Installation selbsterklärend sind, wird auf die weitere Erläuterung im Rahmen dieser Arbeit verzichtet.

5.4.2 Extension Builder installieren

Der ”**Extension Builder**” [19] ist selbst eine Extension, welche die Entwicklung einer neuen eigenen Extension erleichtern soll. Er bietet nach der Installation die Möglichkeit, im Backend via GUI⁵ seine eigene Extension vorzubereiten und ein erstes kleines Gründerüst automatisch erstellen zu lassen.

Um den Extension Builder via Composer in die aktuelle TYPO3 Instanz zu installieren, wird der nachfolgende Befehl in die Shell im Hauptverzeichnis der TYPO3 Installation eingegeben:

```
ddev composer req friendsoftypo3/extension-builder
```

Nach erfolgreicher Installation sollte nun wie in Abbildung 12 zu sehen der Extension Builder im Backend in dem Modulbereich der ”**Admin Tools**” sicht- und auswählbar sein.

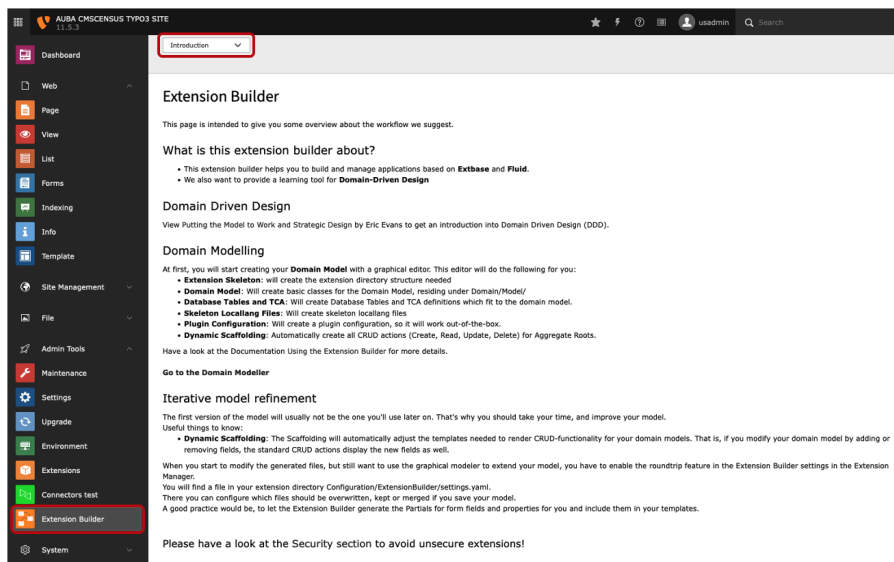


Abbildung 12: TYPO3 - Extension Builder

5.4.3 Introduction Package installieren

Damit die visuelle Ausgabe einer lauffähigen TYPO3 Webseite schneller realisiert werden kann, wird das offizielle ”**TYPO3 Introduction Package**” [21] installiert. Wurde die Installation erfolgreich abgeschlossen, so findet man eine komplette TYPO3 Beispiel Webseite vor, in welcher anschließend eigene Änderungen vorgenommen werden können.

⁵Graphical User Interface - Englisch für ”Grafische Benutzeroberfläche” [20]

Um die Installation des TYPO3 Introduction Package via Composer zu starten, muss der folgende Befehl ausgeführt werden:

```
ddev composer req typo3/cms-introduction
```

Damit die Einrichtung durchgeführt wird, ist anschließend folgender Befehl auszuführen:

```
ddev typo3cms extension:setup
```

Wurde die Installation erfolgreich abgeschlossen, so ist der Seitenbaum im Backend wie in Abbildung 13 unter dem Modulbereich **”Web”** im Modul **”Page”** ersichtlich.

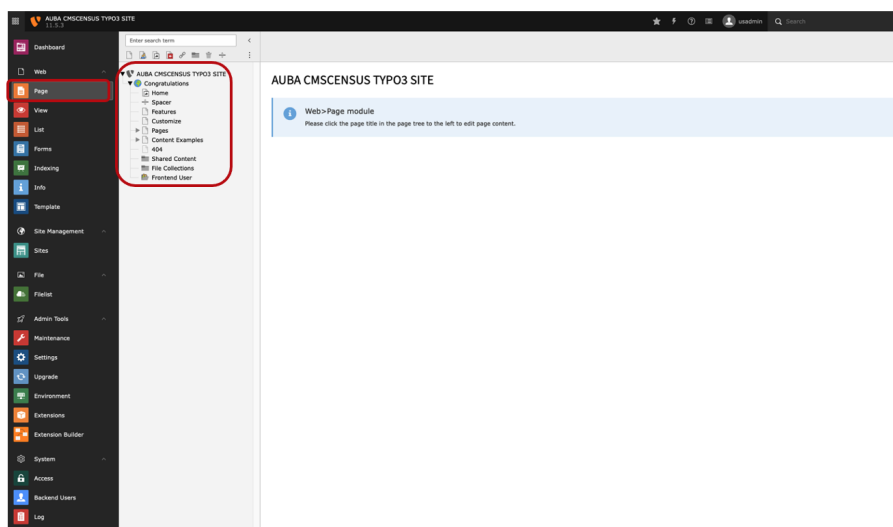


Abbildung 13: TYPO3 - Introduction Package

5.4.4 External Data Import installieren

Da eine Anforderung an die CMSensus Extension der Import von Daten im JSON Format ist und man bei der Entwicklung von Anwendungen auf die Wiederverwendung von bereits existierender Funktionalität zur Vermeidung von Redundanzen⁶ achten sollte, wird an dieser Stelle für die Entwicklung der CMSensus Extension die bereits existierende Extension **”External Data Import”** [23] verwendet und an die eigenen Anforderungen durch entsprechende Erweiterungen angepasst.

⁶von lateinisch redundare, „überlaufen, sich reichlich ergießen“ [22]

Für die Installation der Extension "External Data Import" ist der nachfolgende Befehl einzugeben und auszuführen:

```
ddev composer req cobweb/external_import
```

Da die Extension den Scheduler von TYPO3 benötigt, muss auch hier zur abschließenden Einrichtung der folgende Befehl ausgeführt werden:

```
ddev typo3cms extension:setup
```

Nach erfolgreichem Abschluß der Installation, sollten jetzt wie in Abbildung 14 zu sehen, die Module "Connectors test", "Data Import" und "Log" im Backend sichtbar geworden sein.

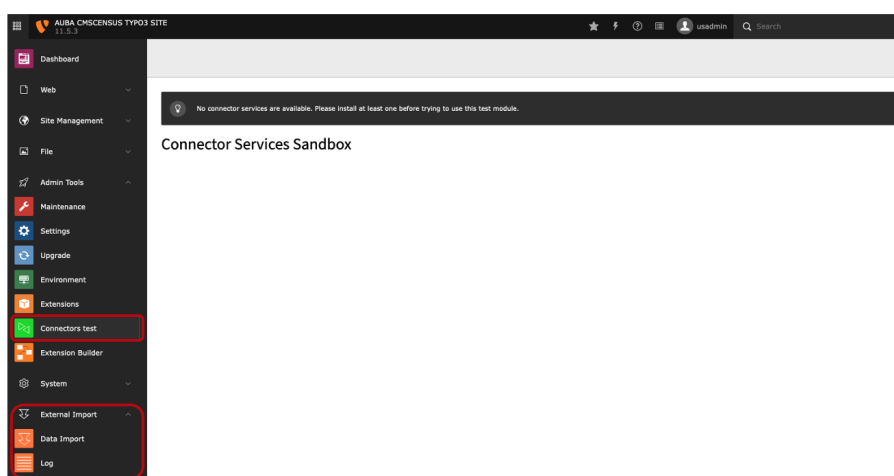


Abbildung 14: TYPO3 - External Data Import

5.4.5 Cache leeren

Eine gute Vorgehensweise ist das regelmäßige leeren des kompletten Cache der TYPO3 Installation, um eventuelle Artefakte vom System selbständig wegräumen zu lassen. Besonders nach der Installation neuer Module oder Extensions bietet sich das leeren des Cache an.

Damit man das leeren des Cache über die Konsole startet, ist nachfolgender Befehl in der Shell einzugeben:

```
ddev typo3cms cache:flush
```

5.5 Entwicklung der Extension

In diesem Abschnitt wird die komplette Entwicklung der CMSensus Extension beschrieben. Es wird sowohl die technische, als auch die konzeptionelle Umsetzung in der Beschreibung zu finden sein.

5.5.1 Erstellung Grundgerüst

Im ersten Schritt der Entwicklung wird das Grundgerüst der CMSensus Extension mit Hilfe des "Extension Builder" erstellt. Wie in Abbildung 15 zu sehen, werden zuerst alle Informationen der zu erstellenden Extension in die Konfigurationsfelder des Extension Builder eingetragen und anschließend abgespeichert.

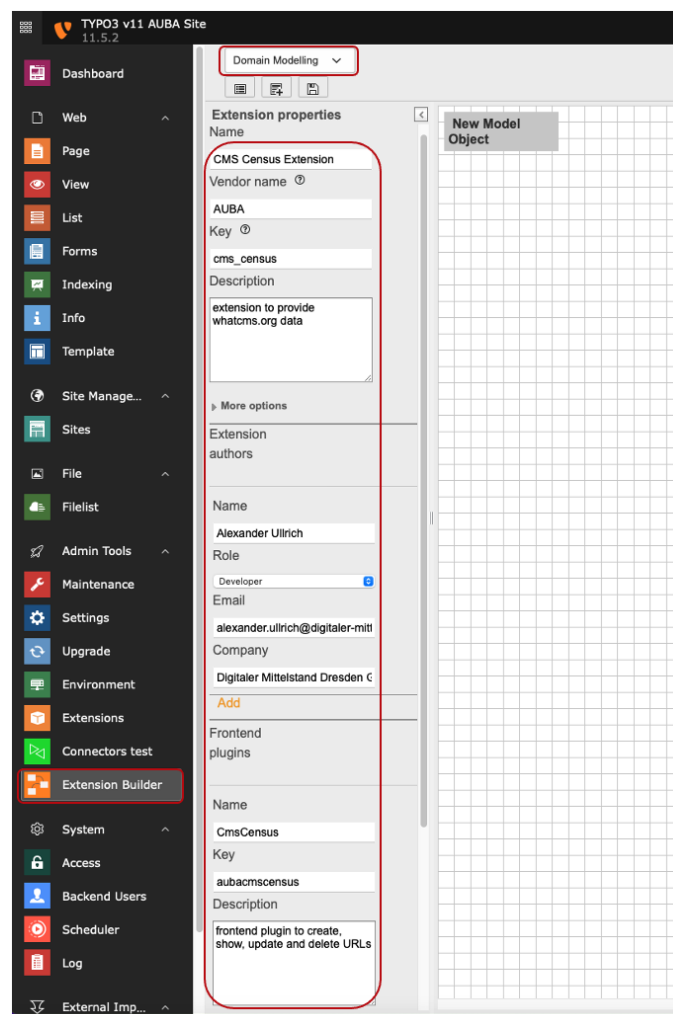


Abbildung 15: TYPO3 - Extension Builder new Model

Im nächsten Schritt wird das aus **Abschnitt 5.3** (Seite 13) theoretisch erstellte Domänenmodell, so wie in Abbildung 16 ersichtlich, modelliert und mit den entsprechenden Eigenschaften versehen.

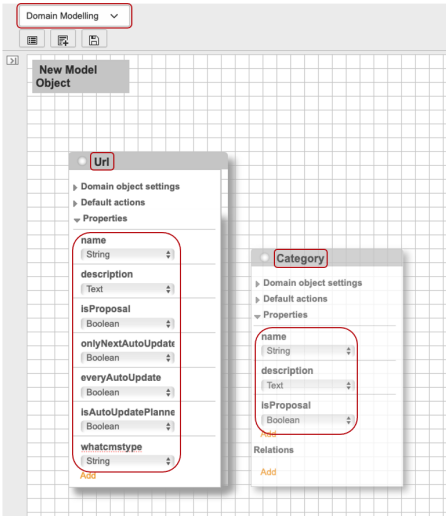


Abbildung 16: TYPO3 - Extension Builder Domänenmodell

Im letzten Schritt der Modellierung werden die beiden Objekte "URL und Category" in ihrer entsprechend dem Domänenmodell vorgesehenen "m:n - Relation", so wie in der Abbildung 17 erkennbar, miteinander verbunden.

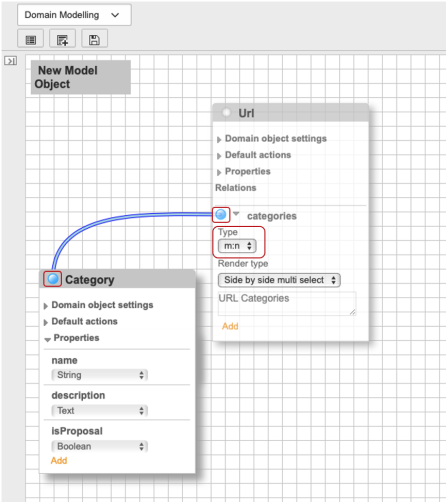


Abbildung 17: TYPO3 - Extension Builder Relation

Nachdem die Modellierung abgeschlossen ist, muss die neue Extension gespeichert werden, wobei auch ein komplettes Grundgerüst durch den Extension Builder erstellt wird.

Bevor dies jedoch geschieht, wird mit den nachfolgenden Befehlen in der Shell erst ein neuer Ordner mit dem Namen "packages" angelegt und anschließend in der Konfigurationsdatei "composer.json" ein Link zum Ordnerpfad hinzugefügt:

```
mkdir -p packages &&  
ddev composer config repositories.local path "packages/*"
```

Nun kann die neue Extension abgespeichert und in die lauffähige TYPO3 Installation durch den folgenden Befehl entsprechend eingebunden werden:

```
ddev composer require auba/cms-census:@dev
```

Um zu überprüfen ob die Installation und Einbindung der Extension erfolgreich durchgeführt wurde, kann man sich wie in Abbildung 18 ersichtlich unter dem entsprechenden Modul "Extensions" alle lokal installierten Extensions anzeigen lassen und sollte damit auch die soeben installierte CMSsensus Extension finden.

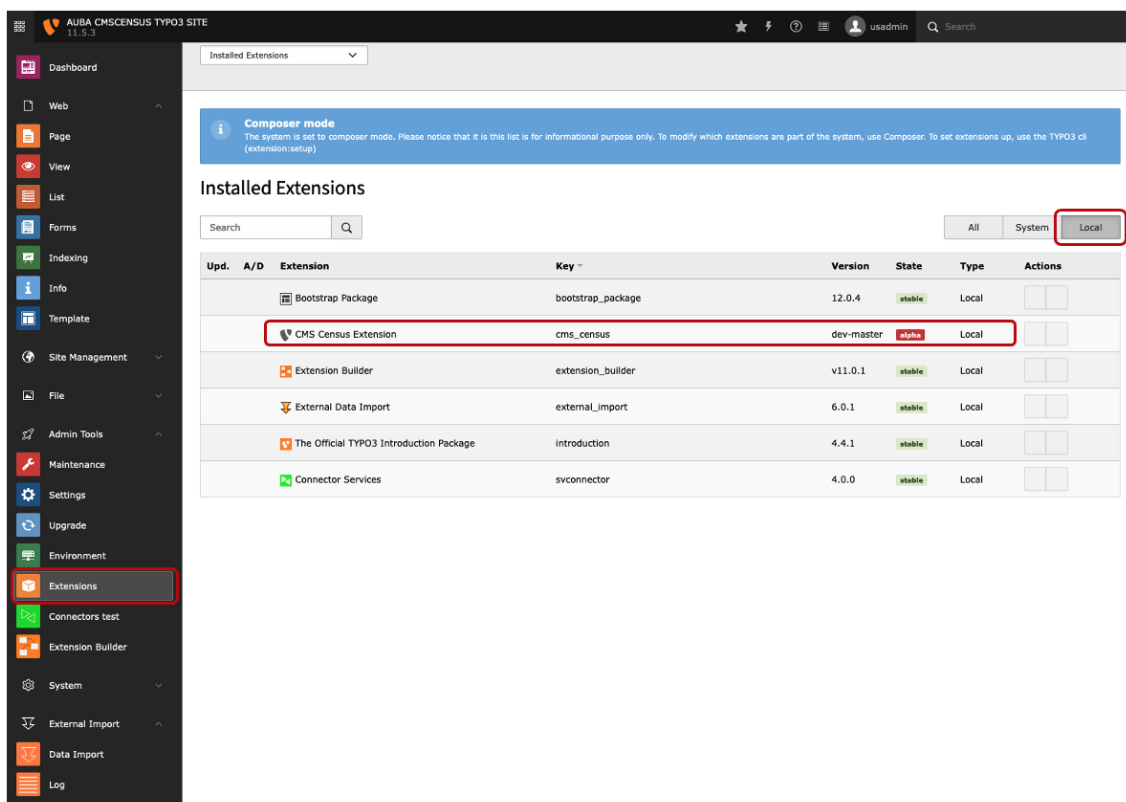


Abbildung 18: TYPO3 - lokal installierte Extensions

5.5.2 Erstellung Datei Import

Damit zur weiteren Entwicklung entsprechende Daten verfügbar sind, wird an dieser Stelle mit der Entwicklung des Datei Import begonnen. Dazu wird die Extension ”**External Data Import**”, so wie in **Abschnitt 5.4.4** (Seite 17) bereits beschrieben verwendet und um benötigte Funktionalitäten erweitert.

Um den Datei Import zu realisieren, ist ein entsprechender Dateiablageort von nöten. Auf diesen hat der jeweilige Dateizugriff beim ausführen des Importszenario zu erfolgen. Er wird über das TYPO3 Backend für die CMSscensus Extension, wie in Abbildung 19 zu sehen, durch anlegen der ”ext_conf_template.txt” Datei konfigurier- und speicherbar gemacht. Neben dem **Dateipfad**, wird der **Dateiname** und die **Kategorie**, welcher der Inhalt zugewiesen werden soll, abgefragt und später unter den entsprechenden Variablenamen in der Datenbank gespeichert.

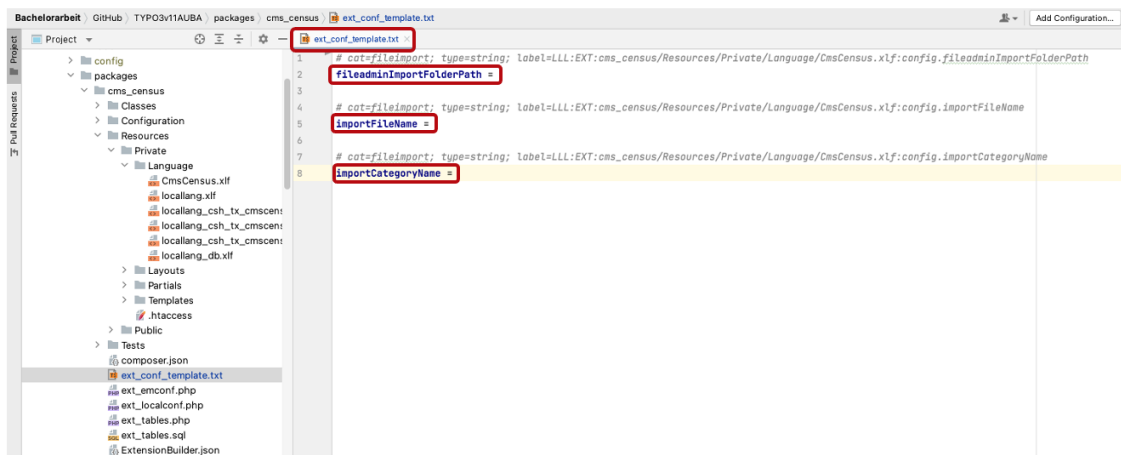


Abbildung 19: CMSscensus - ext_conf_template.txt

Wie in Abbildung 20 ersichtlich, weisen die Kommentarinhalte über den Variablen in der ”ext_conf_template.txt” Datei TYPO3 an, in welchem ”Tab” sie in den ”Settings” für die CMSscensus Extension angezeigt werden sollen und wie die entsprechende Übersetzung der jeweiligen Ausgabertexte durch die Verknüpfung zur ”CMSscensus.xlf” Datei lautet. Es wird außerdem der Typ definiert, welchen die jeweilige Variable hat. Später kann der Zugriff auf den Inhalt der Variablen in der Applikation über die ”**ExtensionConfiguration::class**” der ”**TYPO3\CMS\Core\Configuration\ExtensionConfiguration**” durchgeführt werden.

Durch diesen Zugriff können die entsprechenden Werte zur Auswertung und Programmsteuerung in den jeweiligen Klassen verwendet werden, was damit die Flexibilität der Anwendung erhöht.

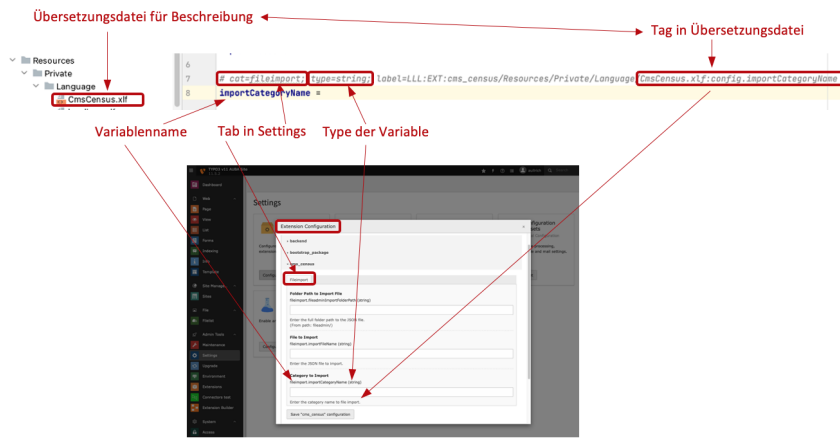


Abbildung 20: CMScensus - Extension Configuration Fileimport

So wie in der Dokumentation [24] von **External Data Import** beschrieben, hat die Konfiguration und das Mapping auf die speziellen Datenbankfelder über die jeweilige **TCA**⁷ des zu konfigurierenden Objektes zu erfolgen. Abbildung 21 zeigt die Konfiguration im **external** Bereich der TCA des **URL** Models. Zu sehen ist der Konfigurationsname **whatCmsJsonFileImport**, der Import Connector Typ **json**, die aufzurufende **uri** und die speziellen **CustomSteps**, welche die eigenen Klassen zur Modifikation beinhalten. Desweiteren ist die jeweilige Position der **CustomSteps** im geplanten Ablauf der späteren Importschritte zu erkennen.

```

tx_cmscensus_domain_model_url.php
14 'external' => [
15   'general' => [
16     'whatCmsJsonFileImport' => [
17       'connector' => 'json',
18       'parameters' => [
19         'uri' => 'fileadminImportFolderPath/importFileName',
20         'encoding' => 'utf8'
21       ],
22       'data' => 'array',
23       'arrayPath' => '',
24       'referenceUid' => 'name',
25       'disabledOperations' => 'delete',
26       'customSteps' => [
27         [
28           'class' => AUBA\CmsCensus\Step\CustomizeFileImportSetupStep::class,
29           'position' => 'before:'. \Cobweb\ExternalImport\Step\CheckPermissionsStep::class
30         ],
31         [
32           'class' => AUBA\CmsCensus\Step\CustomizeStoragePIDStep::class,
33           'position' => 'before:'. \Cobweb\ExternalImport\Step\StoreDataStep::class
34         ]
35       ],
36       'priority' => 5300,
37       'description' => 'Import whatcms.org JSON File'
38     ],
  
```

Abbildung 21: CMScensus - external im TCA des URL Model

⁷Table Configuration Array - Englisch für "Tabellenkonfigurations-Array" [25]

Es ist dabei klar zu sehen, an welchen Punkten (1. und 9.) die eigenen CMSscensus Klassen zur Modifikation des "Import Szenario" aufgerufen werden.

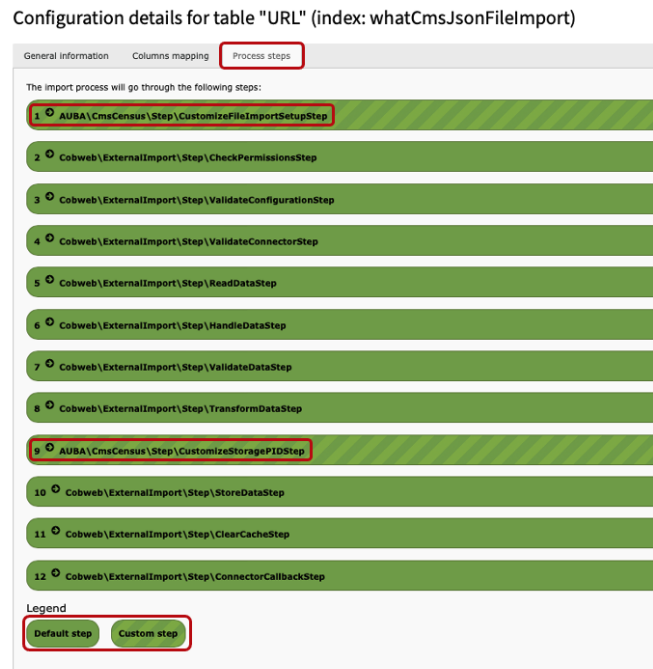


Abbildung 24: CMSscensus - Data Import Process Steps

Im Konstruktor der Klasse "CustomizeFileImportSetupStep" (siehe Abbildung 25) von CMSscensus, werden durch Verwendung des Entwurfsmuster "DI"⁸ zuerst nacheinander Objekte der Klassen "ExtensionConfiguration", "PersistenceManager" und der Klasse "CategoryRepository" erstellt. Diese Objekte werden benötigt, um auf die Konfigurationsfelder der CMSscensus Extension zuzugreifen, die Datensätze des "Category Model" zu bearbeiten und entsprechende Daten persistieren [27] zu können.

```

40  /**
41   * CustomizeSetupStep constructor.
42   *
43   * @param CategoryRepository $categoryRepository
44   * @throws \TYPO3\CMS\Core\Configuration\Exception\ExtensionConfigurationExtensionNotConfiguredException
45   * @throws \TYPO3\CMS\Core\Configuration\Exception\ExtensionConfigurationPathDoesNotExistException
46   */
47  public function __construct(CategoryRepository $categoryRepository)
48  {
49      $this->extensionConfiguration = GeneralUtility::makeInstance(ExtensionConfiguration::class)
50          ->get('cms_census');
51
52      $this->persistenceManager = GeneralUtility::makeInstance(PersistenceManager::class);
53
54      $this->categoryRepository = $categoryRepository;
55  }

```

Abbildung 25: CMSscensus - CustomFileImportSetupStep Konstruktor

⁸Dependency Injection - Englisch für "Abhängigkeitsinjektion" [26]

Da die Klasse "CustomizeFileImportSetupStep" wie in Abbildung 26 ersichtlich von der Klasse "AbstractStep" erbt, stehen ihr alle Eigenschaften von "Importer" zur Verfügung. Sie muss die Funktion "run()" implementieren, da diese automatisch aufgerufen wird.

```
14  /**
15  * Class for adapting the connector URI parameter.
16  *
17  * @package AUBA\CmsCensus\Step
18  */
19  class CustomizeFileImportSetupStep extends AbstractStep
20  {
```

Abbildung 26: CMScensus - CustomFileImportSetupStep Extends

Durch den Aufruf von "run()", wird als nächstes die Funktion "setUriParameter()" aufgerufen. Diese holt sich wie in Abbildung 27 zu erkennen, im ersten Schritt den Pfad zum "fileadmin" Ordner und speichert ihn in der Variablen "\$destinationAbsolutePath" ab. Im nächsten Schritt wird das "GeneralConfiguration" Array des Importer geholt und der Variablen "\$generalConfiguration" zugewiesen, um aus dieser das "Parameters" Array zu extrahieren und in der Variablen "parameters" für die weitere Bearbeitung zu speichern. Im Anschluß wird das Parameters Array in einer "foreach" Schleife durchlaufen und beim Key "uri" der default Eintrag für den Pfad und die Datei durch die in den Settings der "extensionConfiguration" angegeben Werte ersetzt. Im letzten Schritt werden diese neuen Werte den "Parameters" der "GeneralConfiguration" des Importer zugewiesen und damit die alten Werte überschrieben.

```
CustomizeFileImportSetupStep.php x
73  protected function setUriParameter(): void
74  {
75      $destinationAbsolutePath = GeneralUtility::getFileAbsFileName($GLOBALS['TYPO3_CONF_VARS']['BE']['fileadminDir']);
76      $generalConfiguration = $this->getImporter()->getExternalConfiguration()->getGeneralConfiguration();
77      $parameters = $generalConfiguration['parameters'];
78
79      foreach ($parameters as $key => $value) {
80          // Update the "whatCmsJsonFilePath" string from the "uri" parameter, if it exists
81          if ($key === 'uri' && strpos($value, needle: 'fileadminImportFolderPath') !== false) {
82              $parameters[$key] = str_replace( search: 'fileadminImportFolderPath',
83              replace: $destinationAbsolutePath . $this->extensionConfiguration['fileadminImportFolderPath'], $value);
84              $parameters[$key] = str_replace( search: 'importFileName', $this->extensionConfiguration['importFileName'],
85              $parameters[$key]);
86              $parameters[$key] = str_replace( search: '//', replace: '/', $parameters[$key]);
87          }
88      }
89
90      $generalConfiguration['parameters'] = $parameters;
91      $this->getImporter()->getExternalConfiguration()->setGeneralConfiguration($generalConfiguration);
92  }
```

Abbildung 27: CMScensus - CustomFileImportSetupStep setUriParameter()

Nachdem die "setUriParameter()" Funktion erfolgreich abgearbeitet wurde, wird als nächstes die "setCategory()" Funktion aufgerufen. Diese holt sich zu Beginn, wie in Abbildung 28 ersichtlich, die "ColumnConfiguration" des Importer und speichert sie in der Variable "\$columnConfiguration" ab. Anschließend wird überprüft, ob der Wert in der "extensionConfiguration" für den "importCategoryName" nicht leer ist. Ist dies der Fall, so wird im nächsten Schritt geprüft, ob die Kategorie schon in der Datenbanktabelle des "Category" Model existiert, um sie bei nicht Existenz an dieser Stelle neu anzulegen.

```

99     protected function setCategory(): void
100    {
101        $columnConfiguration = $this->getImporter()->getExternalConfiguration()->getColumnConfiguration();
102
103        if($this->extensionConfiguration['importCategoryName'] != '') {
104            $query = $this->categoryRepository->createQuery();
105            $queryResult = $query->statement(
106                statement: 'SELECT *
107                            FROM tx_cmsscensus_domain_model_category
108                            WHERE name = ?
109                            AND deleted = 0',
110                [$this->extensionConfiguration['importCategoryName']]
111            )->execute();
112
113            if(count($queryResult) == 0){
114                $newCategory = GeneralUtility::makeInstance(Category::class);
115                $newCategory->setName($this->extensionConfiguration['importCategoryName']);
116                $newCategory->setPid((int)$this->extensionConfiguration['storagePID']);
117                $this->categoryRepository->add($newCategory);
118                $this->persistenceManager->persistAll();
119
120                $categoriesTransformations[10] = array('value' => $newCategory->getUid());
121            }else{
122                $categoriesTransformations[10] = array('value' => $queryResult[0]->getUid());
123            }
124
125            $categories['transformations'] = $categoriesTransformations;
126            $columnConfiguration['categories'] = $categories;
127            $this->getImporter()->getExternalConfiguration()->setColumnConfiguration($columnConfiguration);
128        }
129    }
130 }

```

Abbildung 28: CMSscensus - CustomFileImportSetupStep setCategory()

Im letzten Schritt der "setCategory()" Funktion wird ein neues "Transformation" Array erzeugt, welchem die entsprechende "uid" der Kategorie in einem eigenen "Key-Value" Pair⁹ Array zugewiesen wird. Dieses neu erzeugte Array wird anschließend in dem "categories" Eintrag der "ColumnConfiguration" des Importer gespeichert, um die Verbindung des importierten Wertes zur festgelegten Kategorie zu sichern.

Ist der "importCategoryName" leer, so wird die gesamte Logik zur Erzeugung der Kategorieverbindung übersprungen und einfach mit dem nächsten Schritt der Importfunktion weitergemacht.

⁹Englisch für "Paar"

Die nächsten Schritte im Ablauf sind die Punkte 2 bis 8 (siehe Abb. 24 / Seite 25), welche zum Standard Szenario der "External Data Import" Extension gehören. Im anschließenden Schritt 9 wird die CMSscensus Klasse "**CustomizeStoragePIDStep**" aufgerufen, welche durch Vererbung ebenfalls vom Typ "AbstractStep" ist und für die Abspeicherung der in der Konfiguration festgelegten "pid" genutzt wird.

In Abbildung 29 ist zu erkennen, daß durch den Aufruf der "run()" Funktion, lediglich die "setStoragePID()" Funktion aufgerufen wird. Diese schaut nach ob ein Eintrag für die "storagePID" in der "extensionConfiguration" vorhanden ist. Ist dies der Fall, so wird die Default "pid" der "GeneralConfiguration" des Importer durch diesen Wert überschrieben und neu gesetzt.

```

CustomizeStoragePIDStep.php x
35  /**
36   * Performs the actual tasks of the step.
37   *
38   * @return void
39   * @throws \Doctrine\DBAL\Driver\Exception
40   */
41  public function run(): void
42  {
43      $this->setStoragePID();
44  }
45
46  /**
47   * Set pid to StoragePID if StoragePID is not 0.
48   *
49   * @return void
50   */
51  protected function setStoragePID(): void
52  {
53      $generalConfiguration = $this->getImporter()->getExternalConfiguration()->getGeneralConfiguration();
54      if((int)$this->extensionConfiguration['storagePID'] != 0){
55          $generalConfiguration['pid'] = (int)$this->extensionConfiguration['storagePID'];
56          $this->getImporter()->getExternalConfiguration()->setGeneralConfiguration($generalConfiguration);
57      }
58  }
59  }

```

Abbildung 29: CMSscensus - CustomizeStoragePIDStep setStoragePID()

Möchte man den Import testen, so hat man die Möglichkeit, dies über das Backend Modul "Data Import" für "[whatCmsJsonFileImport]" durch drücken der "Synchronize" Taste (siehe Abb. 30) entsprechend zu starten.

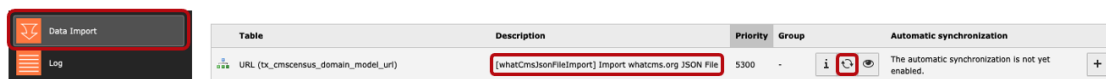
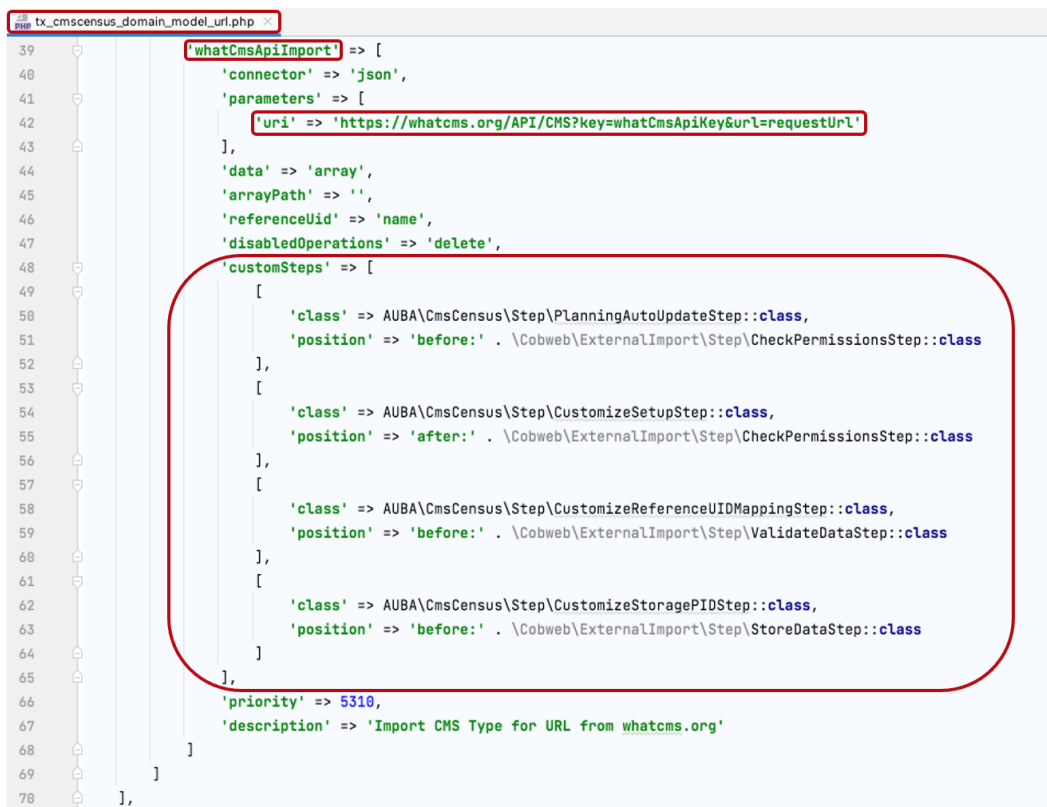


Abbildung 30: CMSscensus - Data Import Synchronize

5.5.3 Erstellung Import über API Schnittstelle

Da als Grundlage für den Import die Erweiterung der "External Data Import" Extension vorgenommen wird, entspricht der Import über die "API Schnittstelle" zum großen Teil der des "Datei Import" Szenarios, welches im vorangegangenen Kapitel beschrieben wurde. Aus diesem Grund wird auf die ausführliche Erklärung der bereits beschriebenen Funktionalitäten verzichtet und nachfolgend lediglich auf die wichtigsten Unterschiede eingegangen.

Während das Mapping in der "TCA" dem gleichen Muster wie beim "Datei Import" entspricht, weicht der Konfigurationsteil im "external" Bereich etwas ab. Es ist deutlich zu erkennen (siehe Abbildung 31), daß die "uri" in der "parameters" Sektion dem zuvor in **Abschnitt 2.3.1** (Seite 4) beschriebenen "API Request URL" Muster folgt. Die darin eingefügten Platzhalter werden später in der "setUriParameter()" Funktion der "CustomizeSetupStep" Klasse überschrieben. Desweiteren wird ersichtlich, daß es vier "CustomSteps" gibt, wobei "CustomizeStoragePIDStep" derselbe Schritt wie beim Datei Import ist und auch hier zur Speicherung der "pid" dient, so wie bereits im vorigen Abschnitt (Seite 28) ausführlich beschrieben.



```
39 'whatCmsApiImport' => [
40     'connector' => 'json',
41     'parameters' => [
42         'uri' => 'https://whatcms.org/API/CMS?key=whatCmsApiKey&url=requestUrl'
43     ],
44     'data' => 'array',
45     'arrayPath' => '',
46     'referenceUid' => 'name',
47     'disabledOperations' => 'delete',
48     'customSteps' => [
49         [
50             'class' => AUBA\CmsCensus\Step\PlanningAutoUpdateStep::class,
51             'position' => 'before:' . \Cobweb\ExternalImport\Step\CheckPermissionsStep::class
52         ],
53         [
54             'class' => AUBA\CmsCensus\Step\CustomizeSetupStep::class,
55             'position' => 'after:' . \Cobweb\ExternalImport\Step\CheckPermissionsStep::class
56         ],
57         [
58             'class' => AUBA\CmsCensus\Step\CustomizeReferenceUIDMappingStep::class,
59             'position' => 'before:' . \Cobweb\ExternalImport\Step\ValidateDataStep::class
60         ],
61         [
62             'class' => AUBA\CmsCensus\Step\CustomizeStoragePIDStep::class,
63             'position' => 'before:' . \Cobweb\ExternalImport\Step\StoreDataStep::class
64         ]
65     ],
66     'priority' => 5310,
67     'description' => 'Import CMS Type for URL from whatcms.org'
68 ],
69 ],
70 ],
```

Abbildung 31: CMSsensus - external API im TCA des URL Model

Da man über die API von WhatCMS.org nur einen Request aller 10 Sekunden stellen darf, musste für CMSsensus eine spezielle Lösung erarbeitet werden. Im Lösungsansatz wurde die Idee der Planung von Updates nach dem Muster des "Bubblesort" [28] Algorithmus gefunden. Dabei sollen im ersten Schritt die URLs, für welche das jeweilige CMS bei WhatCMS.org ermittelt werden soll, mit einem **"is_auto_update_planned"** Flag und dem Wert **"TRUE"** versehen und anschließend bei jedem Aufruf des Importskript wie die Blasen beim Bubblesort Algorithmus einer nach dem anderen abgearbeitet werden. Nach erfolgreicher Abarbeitung eines URL Updates wird für diese URL das **"is_auto_update_planned"** Flag auf **"FALSE"** gesetzt und damit beim nächsten Import die nächste URL verwendet.

Schaut man sich in Abbildung 32 die Funktionsweise der "PlanningAutoUpdateStep" Klasse genauer an, so ist zu erkennen, daß im ersten Schritt in der "run()" Funktion geprüft wird, ob überhaupt eine URL mit dem "is_auto_update_planned" Flagwert "TRUE" in der Datenbank existiert. Ist dies der Fall, so wird **keine** Planung durchgeführt und direkt mit dem nächsten Schritt des Importablaufes fortgefahren.

Ist dies hingegen **nicht** der Fall, so wird die Funktion "planningAutoUpdate()" aufgerufen, welche alle URLs in der Datenbank einzeln durchläuft und prüft, ob die Updates nur einmalig (only_next_auto_update == 1) oder immer (every_auto_update == 1) erfolgen sollen und ob es sich bei dem Eintrag um keinen Vorschlag handelt (is_proposal == 0). Ist dies für die jeweilige URL der Fall, so wird für diese der "is_auto_update_planned" Flagwert auf 1 (TRUE) gesetzt. Um alle nur einmal geplanten Updates für die nächste Planung zu deaktivieren, wird der Wert für "only_next_auto_update" pauschal mit 0 (FALSE) versehen.

```

42 public function run(): void
43 {
44     if (!$this->isAutoUpdatePlanned()) {
45         $this->planningAutoUpdate();
46     }
47 }
48
49 /**
50  * Check is auto update planned
51  *
52  * @return bool
53  */
54 protected function isAutoUpdatePlanned(): bool
55 {
56     $resultArray = $this->queryBuilder
57         ->count('*')
58         ->from('tx_cmscensus_domain_model_url')
59         ->where(
60             $this->queryBuilder->expr()->eq(
61                 'is_auto_update_planned',
62                 '1'
63             )
64         )
65         ->execute()
66         ->fetchFirstColumn();
67
68     return $resultArray[0] > 0;
69 }
76 protected function planningAutoUpdate(): void
77 {
78     // query all entries where:
79     // only_next_auto_update = 1 or every_auto_update = 1 and is_proposal = 0
80     $query = $this->queryBuilder
81         ->select('uid')
82         ->from('tx_cmscensus_domain_model_url')
83         ->where(
84             $this->queryBuilder->expr()->eq('only_next_auto_update', '1'),
85         )
86         ->orWhere(
87             $this->queryBuilder->expr()->eq('every_auto_update', '1'),
88         )
89         ->andWhere(
90             $this->queryBuilder->expr()->eq('is_proposal', '0')
91         )
92         ->execute();
93
94     // set is_auto_update_planned to 1 on all query entries
95     while ($row = $query->fetch()) {
96         $this->queryBuilder
97             ->update('tx_cmscensus_domain_model_url')
98             ->where(
99                 $this->queryBuilder->expr()->eq('uid', $row['uid'])
100             )
101             ->set('is_auto_update_planned', '1')
102             ->set('only_next_auto_update', '0')
103             ->execute();
104     }
105 }
106

```

Abbildung 32: CMSsensus - API Import PlanningAutoUpdateStep

Da die "External Data Import" Extension durch die CMSensus Extension modifiziert wurde, kann sie den Datensatz der jeweiligen URL nicht automatisch zuordnen. Aus diesem Grund wird durch die "**CustomizeReferenceUIDMappingStep**" Klasse das entsprechende Mapping für das Feld "**name**" zum jeweiligen URL Eintrag vorgenommen.

Zum Zeitpunkt des Aufruf der "CustomizeReferenceUIDMappingStep" Klasse, welche ebenfalls von "AbstractStep" erbt, stehen die bereits bei WhatCMS.org abgefragten Daten zur URL im "**RawData**" Array des Importer zur Verfügung. Durch den Aufruf der "run()" Funktion, wird die "addRequestUrlToRawData()" Funktion ausgeführt. Im ersten Schritt wird das "RawData" Array des Importer (siehe Abbildung 33) in der Variablen "\$rawData" gespeichert. Diesem wird anschließend durch ein "Key-Value" Pair¹⁰ Array die URL Adresse als "requestURL" Eintrag hinzugefügt. Konnte WhatCMS.org kein passendes CMS für die abgefragte URL finden, so ist das "Records" Array des Importer leer, was im "if() - Zweig" der "addRequestUrlToRawData()" Funktion überprüft wird. Wie in der Abbildung 33 zu erkennen ist, wird im Falle eines gefüllten "Records" Array diesem ebenfalls ein "Key-Value" Pair Array mit der URL Adresse als "name" hinzugefügt. Ist das "Records" Array hingegen leer, so wird ein Default Eintrag mit dem Inhalt "**CMS Not Detected**" für den "**whatcmstype**" gesetzt. Desweiteren werden die Einträge für "is_auto_update_planned" und "every_auto_update" auf 0 (FALSE) gesetzt, damit diese URL aus der zukünftigen, weiteren Updateplanung entsprechend ausgeschlossen wird.

```

66 protected function addRequestUrlToRawData(): void
67 {
68     $rawData = $this->getData()->getRawData();
69     $rawData['result'] = $rawData['result'] + array('requestURL' => $this->getRequestUrl());
70     $this->getData()->setRawData($rawData);
71
72     // prepare the record to match name with requestURL when record not empty
73     $records = $this->getData()->getRecords();
74     if (empty($records)) {
75         $records[0] = $records[0] + array('name' => $this->getRequestUrl());
76         $this->getData()->setRecords($records);
77     } else {
78         $this->removeUrlFromPlanning();
79     }
80 }
81
82 /**
83  * Supplies the URL to be store referenceId.
84  *
85  * @return string
86  */
87 protected function getRequestUrl(): string
88 {
89     // query all entries where is_auto_update_planned = 1
90     $query = $this->urlRepository->createQuery();
91     $queryResult = $query->statement(
92         statement: 'SELECT name
93                   FROM tx_cmsensus_domain_model_url
94                   WHERE is_auto_update_planned = ?', [1]
95     )->execute();
96
97     return $queryResult[0]->getName();
98 }

```

```

105 protected function getRequestUrlId(): int
106 {
107     // query all entries where is_auto_update_planned = 1
108     $query = $this->urlRepository->createQuery();
109     $queryResult = $query->statement(
110         statement: 'SELECT uid
111                   FROM tx_cmsensus_domain_model_url
112                   WHERE is_auto_update_planned = ?', [1]
113     )->execute();
114
115     return $queryResult[0]->getUid();
116 }
117
118 /**
119  * Remove the URL from planning, delete every auto update and set whatcmstype to unknown.
120  *
121  * @return void
122  */
123 protected function removeUrlFromPlanning(): void
124 {
125     // update is_auto_update_planned with 0 where name is requestURL
126     $requestUrlId = $this->getRequestUrlId();
127
128     $this->queryBuilder
129         ->update('tx_cmsensus_domain_model_url')
130         ->where(
131             $this->queryBuilder->expr()->eq('uid', $requestUrlId)
132         )
133         ->set('is_auto_update_planned', '0')
134         ->set('every_auto_update', '0')
135         ->set('whatcmstype', 'CMS Not Detected')
136         ->execute();
137 }
138 }

```

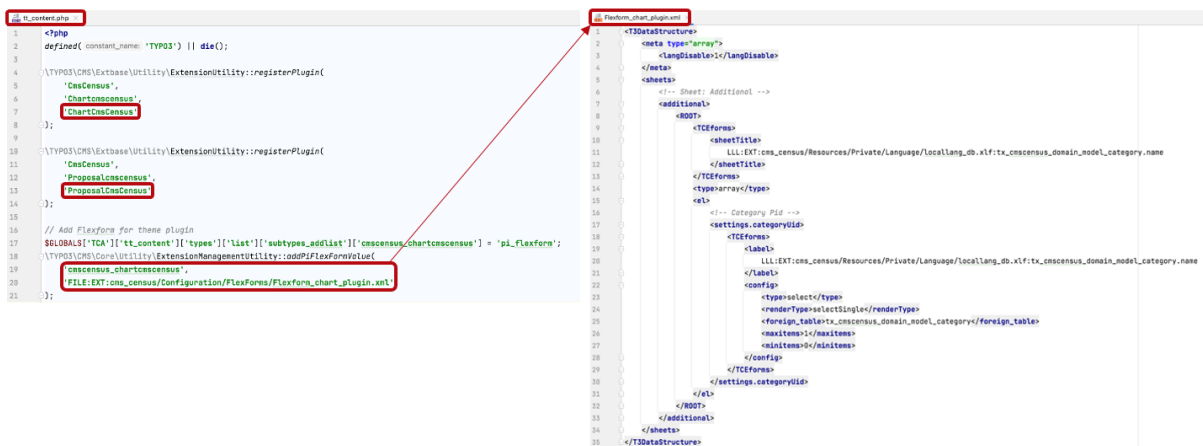
Abbildung 33: CMSensus - API Import CustomizeReferenceUIDMappingStep

¹⁰Englisch für "Paar"

5.5.4 Plugins für Content verfügbar machen

In der CMScensus Extension existieren zwei Plugins, welche im Contentbereich auswählbar sein sollen. Es handelt sich zum einen um das Plugin **”ProposalCmsCensus”**, welches für den Prozess neuer Vorschläge verantwortlich ist und zum anderen um das für die Chartdarstellung verantwortliche **”ChartCmsCensus”** Plugin, welches sich um die Darstellung der ausgewerteten Daten einer zuvor ausgewählten Kategorie kümmert.

Wie in Abbildung 34 zu erkennen ist, wird die Registrierung der Plugins in der **”tt_content”** PHP Datei durchgeführt. Neben den beiden Plugins wird noch eine Flexform [29] registriert, welche die Möglichkeit der Auswahl einer Kategorie im **”cmscensus_chartcmscensus”** Plugin hinzufügt. Die Flexform selbst bezieht ihre Struktur dabei aus einer **”XML”** [30] Datei.



```
1 <?php
2 defined( 'constant_name' 'TYPO3' ) || die();
3
4 \TYPO3\CMS\ExtensionUtility\ExtensionUtility::registerPlugin(
5     'CmsCensus',
6     'ChartCmsCensus',
7     'ChartCmsCensus'
8 );
9
10 \TYPO3\CMS\ExtensionUtility\ExtensionUtility::registerPlugin(
11     'CmsCensus',
12     'ProposalCmsCensus',
13     'ProposalCmsCensus'
14 );
15
16 // Add Flexform for these plugin
17 $GLOBALS['TCA']['tt_content']['types']['list']['subtypes_addlist']['cmscensus_chartcmscensus'] = 'pi_flexform';
18 \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addPiFlexFormValue(
19     'cmscensus_chartcmscensus',
20     'FILE:EXT:cms_census/Configuration/FlexForms/Flexform_chart_plugin.xml'
21 );
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583

```

Damit nach dem Einfügen des jeweils ausgewählten Plugins im Content auch der richtige Controller mit der jeweiligen "Action-Funktion" ausgeführt wird, muss in der Datei "ext_localconf.php" die entsprechende Konfiguration durchgeführt werden.

Es wird wie in Abbildung 36 ersichtlich, der "**configurePlugin()**" Funktion, sowohl der Name der Extension, als auch der Name des Plugin, für welches die Konfiguration zu erfolgen hat übergeben. Desweiteren werden die Controller des Plugin, sowie ihre aufrufbaren "Action-Funktionen" TYPO3 bekannt gemacht. Die erste Action Funktion wird dabei als **Default** für den jeweiligen Controller definiert und beim Aufruf ohne Action ausgeführt. In Abbildung 36 ist beispielhaft zu erkennen, daß für die Default Action des "ChartController" die Action "show" angegeben wird, welche als Funktion in der "ChartController" Klasse mit der Bezeichnung "showAction()" existieren muss.

```
13 (static function() {
14     \TYPO3\CMS\Extbase\Utility\ExtensionUtility::configurePlugin(
15         'CmsCensus',
16         'Chartcmsensus',
17         [
18             ChartController::class => 'show',
19             AjaxController::class => 'cmsPerCategoryUrLs'
20         ],
21         // non-cacheable actions
22         [
23             ChartController::class => 'show',
24             AjaxController::class => 'cmsPerCategoryUrLs'
25         ]
26     );
27
28     \TYPO3\CMS\Extbase\Utility\ExtensionUtility::configurePlugin(
29         'CmsCensus',
30         'Proposalcmsensus',
31         [
32             ProposalController::class => 'addUrLForm, addCategoryForm, createUrL, createCategory'
33         ],
34         // non-cacheable actions
35         [
36             ProposalController::class => 'addUrLForm, addCategoryForm, createUrL, createCategory'
37         ]
38     );

```

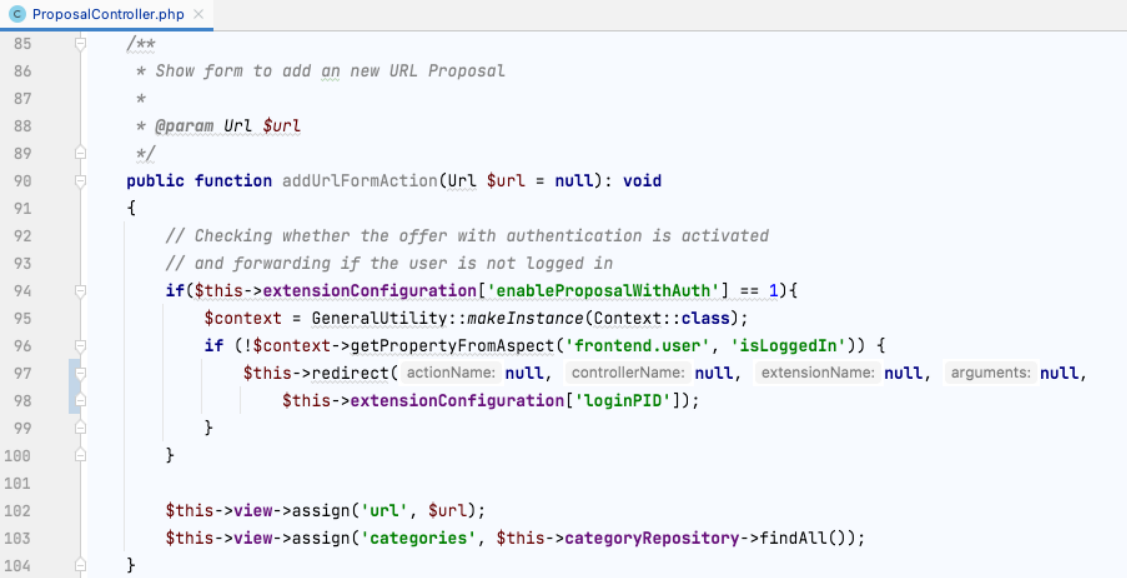
Abbildung 36: CMSensus - Configure Plugin

Es ist möglich, weitere **Konfigurationseinstellungen** in der "ext_localconf.php" Datei vorzunehmen, um die Verhaltensweise der Plugins zu steuern. Für "CMSensus" wurden zum Beispiel noch weitere Einstellungen für die anzuzeigenden **Icons**, die **Titel** und die **Beschreibungen** vorgenommen.

5.5.5 Erstellung Proposal Funktionalität

Wie im **Abschnitt 5.5.4** (Seite 33 / Abbildung 36) in der abgebildeten Konfiguration zu erkennen ist, wird als Defaultfunktion des "ProposalController" beim Aufruf einer Seite, welche im Inhalt das Content Plugin "**proposalcmsscensus**" verwendet, als erstes die "addUrlFormAction()" Funktion ausgeführt.

In der "addUrlFormAction()" Funktion wird wie in Abbildung 37 ersichtlich im ersten Schritt geprüft, ob das "enableProposalWithAuth" Flag im "extensionConfiguration" Array auf "**TRUE**" (1) gesetzt ist und wenn ja, ob es sich bei dem Besucher der Seite um einen angemeldeten Frontendbenutzer handelt. Ist das Flag auf 1 und der Besucher nicht angemeldet, so wird er zu der Seite mit der im "extensionConfiguration['loginPID']" Array angegebenen "pid" durch ein ausgelöstes "redirect()" weitergeleitet. Diese Seite sollte den Benutzer entsprechend zur Anmeldung auffordern.



```
85  /**
86   * Show form to add an new URL Proposal
87   *
88   * @param Url $url
89   */
90  public function addUrlFormAction(Url $url = null): void
91  {
92      // Checking whether the offer with authentication is activated
93      // and forwarding if the user is not logged in
94      if($this->extensionConfiguration['enableProposalWithAuth'] == 1){
95          $context = GeneralUtility::makeInstance(Context::class);
96          if (!$context->getPropertyFromAspect('frontend.user', 'isLoggedIn')) {
97              $this->redirect( actionName: null, controllerName: null, extensionName: null, arguments: null,
98                           $this->extensionConfiguration['loginPID']);
99          }
100     }
101
102     $this->view->assign('url', $url);
103     $this->view->assign('categories', $this->categoryRepository->findAll());
104 }
```

Abbildung 37: CMSscensus - ProposalController addUrlFormAction()

Da der "ProposalController" von der Klasse "ActionController" erbt, steht ihm die Funktionalität eines View Objektes, welches zur Verarbeitung der darzustellenden Daten verwendet wird, zur Verfügung. Diesem wird durch Benutzung der "view()" Funktion unter dem Schlüssel "url" das URL Objekt und unter dem Schlüssel "categories" alle Kategorieinträge der Rückgabe der "findAll()" Funktion des "categoryRepository" übergeben (siehe Abbildung 37).

Bevor es zur Browserausgabe kommt wird, wie in Abbildung 38 zu erkennen ist, das Fluid [31] Styled Template "AddUrlForm.html" geladen, welches dem Namen der zuvor ausgeführten Controller Action "addUrlForm" entspricht. In dem Template wird wiederum das Layout "Default.html" geöffnet und in diesem die Section "content" des "AddUrlForm.html" Template gerendert. Beim Prozess des Rendering wird das angegebene Partial "UrlForm.html" eingebunden und alle darin enthaltenen Variablen mit den dazugehörigen Werten befüllt. Nach erfolgreichem Abschluß des kompletten Vorgangs wird das in Abbildung 38 ersichtliche Formular im Browser ausgegeben.

The image displays the Fluid template rendering process for CMSensus. It consists of three main parts:

- AddUrlForm.html:** This template includes a layout call for 'default.html' and a section call for 'content'. It also defines variables for 'headline' and 'submitmessage' and uses the 'render partial' function to include 'Proposal/UrlForm.html'.
- Default.html:** This layout template renders the 'content' section from 'AddUrlForm.html'.
- UrlForm.html:** This partial template contains the HTML for the form, including labels for 'name', 'description', and 'categories', and form controls like text fields, a text area, and a select menu. It also includes submit and link actions.

Red arrows labeled "Aufruf" indicate the rendering flow: 'AddUrlForm.html' calls 'Default.html', which renders the 'content' section, which then renders the 'UrlForm.html' partial. A red arrow labeled "Ausgabeformular" points from the 'UrlForm.html' code to a rendered browser form. The form includes fields for 'name', 'description', and 'categories', along with 'Submit URL Proposal' and 'Proposal a new Category' buttons.

Abbildung 38: CMSensus - Fluid Template Proposal

Nach ausfüllen des Formulars und anschließender Betätigung des zum absenden vorgesehenen Button **Submit URL Proposal**, wird die Action **createUrl** an den Proposal Controller gesandt, wo sie die direkte Ausführung der `createUrlAction()` Funktion zur Folge hat.

Durch Aufruf der `createUrlAction()` Funktion (siehe Abbildung 39), wird im ersten Schritt der Wert des Namen von dem an die Funktion übergebenen `URL` Objekt auf Korrektheit geprüft. Dies geschieht durch Verwendung der von PHP zur Verfügung stehenden `filter_var()` [32] Funktion. Ist der Name der URL korrekt, so wird das in dem übergebenen URL Objekt vorhandene `is_proposal` Flag auf `TRUE` gesetzt und anschließend das komplette URL Objekt dem `urlRepository` hinzugefügt. Dieses hinzufügen hat die automatische Speicherung des URL Objektes in der Datenbank zur Folge, um welche sich das `UrlRepository` selbständig kümmert. Im letzten Schritt wird die Controller eigene Funktion `finishCreation()` aufgerufen, welche die Weiterleitung zur `addUrlFormAction()` Funktion zur Folge hat.



```
110 public function createUrlAction(\AUBA\CmsCensus\Domain\Model\Url $newUrl)
111 {
112     if (filter_var($newUrl->getName(), filter: FILTER_VALIDATE_URL)) {
113         $newUrl->setIsProposal(true);
114         $this->urlRepository->add($newUrl);
115         $this->finishCreation(redirectDestination: 'addUrlForm');
116     } else {
117         $errorMessageBody = LocalizationUtility::translate(
118             'tx_cmscensus_flashmessage_proposal-error.url',
119             'CmsCensus'
120         );
121         $this->addFlashMessage( messageBody: $errorMessageBody . ' => [' . $newUrl->getName() . ']', messageTitle: '',
122             severity: AbstractMessage::ERROR);
123         $this->redirect( actionName: 'addUrlForm');
124     }
125 }
```

Abbildung 39: CMScensus - ProposalController createUrlAction()

Ist die Validierung der URL in der `createUrlAction()` Funktion fehlgeschlagen, so wird eine eigene Fehlermeldung erzeugt und dem `FlashMessage` Stack des Controllers hinzugefügt (siehe Abbildung 39). Anschließend wird durch Aufruf der `redirect()` Funktion zur `addUrlFormAction()` Funktion weitergeleitet. Der zuvor mit der Fehlermeldung gefüllte `FlashMessage` Stack wird im View des `AddFormUrl.html` Template an der Stelle `flashMessages` (siehe Abbildung 38 / Seite 35) eingesetzt und dadurch als Hinweis im gerenderten Formular des Browser ausgegeben.

Der Vorschlagprozess für die Kategorien ist technisch sehr ähnlich dem der bereits ausführlich beschriebenen Funktionsweise der URLs. Aus diesem Grund wird auf die weitere genaue Beschreibung im Zuge der vorliegenden Arbeit verzichtet.

5.5.6 Erstellung Chart Funktionalität

Auch für die Chartausgabe ist wie im **Abschnitt 5.5.4** (Seite 33 / Abbildung 36) in der abgebildeten Konfiguration zu erkennen, das als Defaultfunktion beim Aufruf einer Seite des "ChartController", welche im Inhalt das Content Plugin "chartcmscensus" verwendet, als erstes die "showAction()" Funktion ausgeführt wird. Es ist außerdem zu sehen, das als Defaultfunktion für den "AjaxController" beim Aufruf die Funktion "cmsPerCategoryUrlsAction()" entsprechend Anwendung findet.

Schaut man sich die beiden Controller genauer an (siehe Abbildung 40), so ist zu erkennen, das beide Defaultfunktionen zu Beginn versuchen die "uid" der Kategorie zu ermitteln und entsprechend in der Variablen "\$categoryId" zu speichern. Im nächsten Schritt wird via "DI"¹¹ ein Objekt der "CategoryUrls" Klasse erzeugt und in der Klassenvariablen "categoryUrls" gespeichert. Danach wird die "countCmsOfCategoryUrls()" Funktion des "CategoryUrls" Objekt mit der "categoryId" als Funktionsparameter aufgerufen. Ihr Rückgabewert ist ein Array mit den durch die Funktion berechneten statistischen Daten zur zuvor übergebenen "categoryId". Die im Array gespeicherten Daten werden in der Defaultfunktion des "ChartController" dem View Objekt zur Ausgabe mit verschiedenen "Key-Value" Pairs¹² zugewiesen. In der Funktion des "AjaxController" werden die Werte des Array erst in ein JSON Objekt gepackt und anschließend dem Aufrufer der Funktion zurückgegeben.



```
ChartController.php
56 //
57 * @param CategoryRepository $categoryRepository
58 */
59 public function injectCategoryRepository(CategoryRepository $categoryRepository)
60 {
61     $this->categoryRepository = $categoryRepository;
62 }
63 //
64 //
65 //
66 //
67 public function showAction() void
68 {
69     $categoryId = (int)$this->settings['categoryId'];
70     $category = $this->categoryRepository->findById($categoryId);
71
72     $this->categoryUrls = GeneralUtility::makeInstance(CategoryUrls::class);
73     $countCmsOfCategoryUrls = $this->categoryUrls->countCmsOfCategoryUrls($categoryId);
74
75     $this->view->assign('category', $category);
76     $this->view->assign('tableCmsCountOfCategoryUrls', $countCmsOfCategoryUrls['cmsTable']);
77     $this->view->assign('totalCountOfCMSUrls', $countCmsOfCategoryUrls['cmsSumCmsUrlCounts']);
78 }
79 }

AjaxController.php
23 //
24 *
25 * @return ResponseInterface
26 */
27 public function cmsPerCategoryUrlsAction(Category $category = null) ResponseInterface
28 {
29     // Do all urls, if category is empty
30     if ($category == null || $category->getName() == "") {
31         $categoryId = 0;
32     } else {
33         $categoryId = $category->getUid();
34     }
35
36     $this->categoryUrls = GeneralUtility::makeInstance(CategoryUrls::class);
37
38     $data = [
39         'cmsPerCategoryUrls' => $this->categoryUrls->countCmsOfCategoryUrls($categoryId)
40     ];
41
42     return (new JsonResponse())->setPayload($data);
43 }
44 }
```

Abbildung 40: CMScensus - ChartController und AjaxController

Die Hauptaufgabe der "countCmsOfCategoryUrls()" Funktion des "CategoryUrls" Objekt liegt darin, die Anzahl der verwendeten "CMS" pro Kategorie prozentual ins Verhältnis zu setzen und als entsprechende Werte zurückzugeben. Desweiteren wird im Rückgabearray die Liste der verwendeten Farben und CMS Namen gespeichert. Somit können die Daten an den unterschiedlichsten Stellen individuell ausgegeben oder weiterverarbeitet werden.

¹¹Dependency Injection - Englisch für "Abhängigkeitsinjektion" [26]

¹²Englisch für "Paare"

Genau wie beim "Proposal" Plugin (siehe Abschnitt 5.5.5 / Seite 35), wird auch beim "Chart" Plugin vor der Browserausgabe ein Fluid [31] Styled Template geladen. Dieses entspricht im Namen wieder der zuvor ausgeführten Action. Da beim "ChartController" die Defaultaction "show" lautet, wird entsprechend das "Show.html" Template geladen.

Wie in Abbildung 41 zu erkennen ist, wird im ersten Schritt des "Show.html" Template das "Default.html" Layout eingebunden, welches wiederum die Sektion "content" des "Show.html" Template zum Rendering ausführt. In diesem wird danach die Script Konstante "ajaxUrl" erstellt und mit der für die Anfrage des "AjaxController" zuständigen "URL" befüllt, um sie in der dazugehörigen "cmscensus.js" JavaScript Datei für den nachfolgenden "AjaxRequest" zur Erstellung der Chartdarstellung zu verwenden. Im nächsten Schritt wird die Überschrift im Bereich der Codezeilen 15 bis 17 durch die in der "locallang.xlf" Übersetzungsdatei für den Schlüssel "tx_cmscensus_chart_url_headline" passende Übersetzung hinzugefügt.

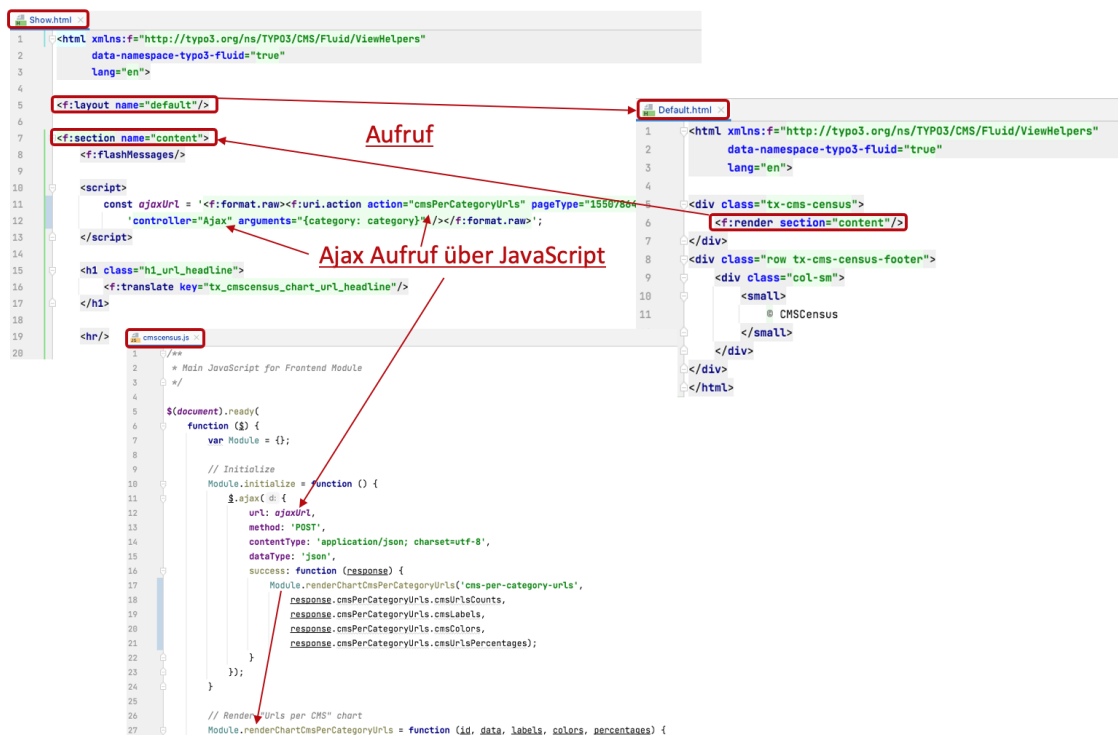


Abbildung 41: CMScensus - Fluid Template Show und Ajax

In den Codezeilen 20 bis 45 wird der Inhalt des Kopfbereiches für die Ausgabe des "Chart" Plugin mit den zusammenfassenden Werten für die ausgewählte "Kategorie", sowie die Gesamtzahl aller betrachteten "URLs" erzeugt.

Im letzten Teil des "Show.html" Template wird, wie in Abbildung 42 erkennbar ist, zuerst die Tabelle mit den statistischen Daten erzeugt, welche die Verteilung der "CMS" aller "URLs" ausgibt. Dazu wird das Datensatz Array "tableCmsCountOfCategoryUrls" in einer "For-Schleife" durchlaufen und jeder Wert Zeile für Zeile der Tabelle hinzugefügt. Die letzte Zeile der Tabelle beinhaltet die Zusammenfassung aller Teilwerte, um sie als "Total" auszugeben.

Das in der Zeile 85 eingefügte "canvas" Tag wird benötigt, um die Ausgabe des Chart zu erzeugen. Durch die angegebene ID "cms-per-category-urls" wird eine Verbindung zwischen dem "Canvas"¹³ Objekt und der JavaScript Datei "cmscensus.js" hergestellt, um anschließend die statistischen Daten des ausgeführten JavaScript in Chartform zur Anzeige zu bringen.

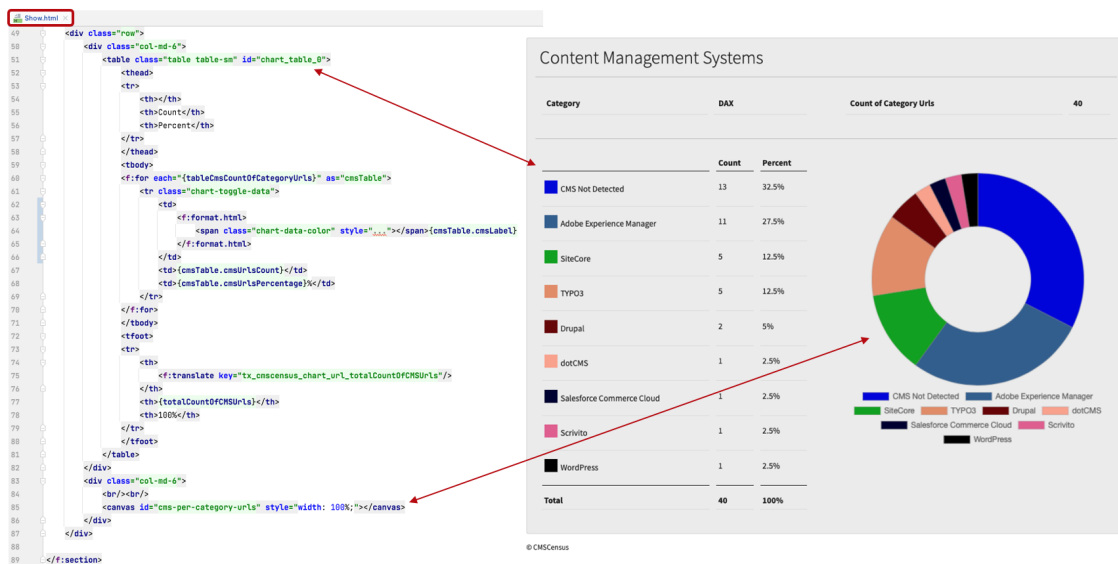


Abbildung 42: CMScensus - Fluid Template Show und Chart Darstellung

Da die Ausführung des "Ajax-Request" in der JavaScript Datei "cmscensus.js" erst bei die Browerausgabe des "Show.html" Template angeschoben wird, ist die Anzeige der Chartform im Gegensatz zur Ausgabe der Datentabelle je nach Antwortzeit des Servers verzögert zu sehen.

¹³Englisch für "Leinwand oder Gemälde"

6 Fazit und Ausblick

Im Folgenden wird der Entwicklungsprozess von der Idee, über die Konzeption, bis hin zur Implementierung des Prototyp der TYPO3 Extension "CMScensus" kritisch betrachtet und Erfahrungen, sowie Erkenntnisse, welche während des Prozesses auftraten vorgestellt und beschrieben. Im Anschluß werden Vorschläge und Anregungen zur weiteren Optimierung oder Weiterentwicklung des Prototyps aufgezeigt.

6.1 Fazit

Das Ziel der Arbeit war die Entwicklung einer Extension für TYPO3 (Version 11), welche erstellt und anschließend installier-, sowie ausführbar sein muss. Der entwickelte Prototyp soll für den Onlinebetrieb einsatzbereit und für die Öffentlichkeit im Internet erreichbar sein. Desweiteren sollte die Extension die Möglichkeit zur Eingabe neuer Kategorien und URL's, sowie die visuelle Auswertung der statistischen Daten über die Verteilung von benutzten CMS einer Kategorie in Tabellen- und Chartform für Frontendbenutzer zur Verfügung stellen. Dieses Ziel konnte im Rahmen der vorliegenden Arbeit erfolgreich umgesetzt werden. Der Prototyp der Extension läuft aktuell¹⁴ auf einem Server mit installiertem TYPO3 (Version 11), welcher öffentlich im Internet unter der Domain "**www.cmscensus.eu**" erreichbar ist.

Der gesamte Entwicklungsprozess folgte einer im Vorfeld selbst definierten und festgelegten Struktur. Während zu Beginn erst das Verständnis für die Datenquelle, sowie für die Funktionsweise der API Schnittstelle von "WhatCMS.org" aufgebaut werden musste, folgten im Anschluß eigene Tests zur Implementierung und Weiterverarbeitung. Dabei wurde die Erkenntnis gewonnen, daß es für die Umsetzung der Importfunktionalität besser ist, die bereits bestehende TYPO3 Extension "External Data Import" zu verwenden und diese anschließend für die eigenen Zwecke zu erweitern. Auch die Umsetzung des "Domain-Driven Design" Pattern erforderte im Vorfeld die Wissenserweiterung durch verschiedene Quellen (Bücher und Internet), damit es im Anschluß Schritt für Schritt nach einem festen Muster abgearbeitet werden konnte.

Für die Entwicklung wurde zur Versionierung das freie Open Source Kontrollsystem Git [14] verwendet und das dazugehörige "Git Repository" auf "GitHub" eingerichtet. Unter der URL "**https://github.com/usadmin77/bachelorarbeit.git**" ist für ausgewählte Benutzer das Repository zugänglich und somit der komplette Entwicklungszyklus durch die automatisch geführten Logeinträge jederzeit lückenlos nachvollziehbar. Im Zuge der Entwicklung konnte die Erfahrung gemacht werden, daß die Funktionalität der Wiederherstellung eines zuvor gespeicherten Projektzustandes aus der Version im "Git Repository" absolute Flexibilität und Sicherheit mit sich bringt und damit die Qualität der Entwicklung deutlich verbessert.

¹⁴Stand Dezember 2021

Die funktionalen und technischen Anforderungen an die Extension konnten im Ergebnis erfolgreich umgesetzt und implementiert werden. Im Laufe der Entwicklung kamen neue Erkenntnisse für besseren Programmierstil hinzu. So wurden zum Beispiel zu Beginn Datenbankabfragen, welche das Model betreffen, via "QueryBuilder" durchgeführt. Auf diese Art der Abfrage sollte allerdings gemäß der TYPO3 Dokumentation [33] verzichtet werden und stattdessen die Abfrage über ein "Repository" erfolgen.

6.2 Ausblick

Der aktuelle Stand des Prototyps der TYPO3 Extension "CMSscensus" bietet eine gute Ausgangslage für eine entsprechende Weiterentwicklung. So empfiehlt es sich zum Beispiel beim "Proposal" Plugin die Überprüfung des angemeldeten Frontendbenutzer auf den Vorschlag der Kategorie zu erweitern. Da die Klassen der "CustomSteps" zu Anfang mit dem "QueryBuilder" entwickelt wurden und dies bewusst beispielhaft im Code an einigen Stellen erhalten blieb, sollten diese durch die "Repository" Abfrage ersetzt werden.

Für die bessere Bedienung empfiehlt es sich die Extension um weitere "Backend" Funktionalität zu erweitern. So könnte zum Beispiel eine getrennte Ansicht aller URLs einer Kategorie erzeugt werden. Auch eine Filterung nach verschiedensten Gesichtspunkten könnte die Übersichtlichkeit für Redakteure verbessern. Die Importfunktionalität kann durch ein Upload Formular mit anschließender Ausführung des Import erweitert werden.

Da die "CMSscensus" Extension auf jedem TYPO3 (Version 11) Server installierbar ist, besteht somit auch die Möglichkeit diese nach eigenen Anforderungen individuell anzupassen, zu erweitern oder entsprechend weiterzuentwickeln.

7 Quellen und Literaturverzeichnis

Literatur

- [1] Wikipedia. *Content-Management-System – Wikipedia*. URL: <https://de.wikipedia.org/wiki/Content-Management-System> (besucht am 30.11.2021).
- [2] CMS Matrix. *The CMS Matrix - cmsmatrix.org - The Content Management Comparison Tool*. URL: <http://www.cmsmatrix.org> (besucht am 30.11.2021).
- [3] TYPO3.org. *Development Roadmap for TYPO3 CMS*. URL: <https://typo3.org/cms/roadmap> (besucht am 30.11.2021).
- [4] WhatCMS.org. *Detect which CMS a site is using - What CMS?* URL: <https://whatcms.org/> (besucht am 30.11.2021).
- [5] WhatCMS.org. *About - What CMS?* URL: <https://whatcms.org/About> (besucht am 01.12.2021).
- [6] Wikipedia. *Uniform Resource Locator – Wikipedia*. URL: https://de.wikipedia.org/wiki/Uniform_Resource_Locator (besucht am 01.12.2021).
- [7] WhatCMS.org. *Documentation - What CMS?* URL: <https://whatcms.org/Documentation> (besucht am 01.12.2021).
- [8] Wikipedia. *JavaScript Object Notation – Wikipedia*. URL: https://de.wikipedia.org/wiki/JavaScript_Object_Notation (besucht am 01.12.2021).
- [9] Wikipedia. *Volkszählung – Wikipedia*. URL: <https://de.wikipedia.org/wiki/Volksz%C3%A4hlung> (besucht am 01.12.2021).
- [10] DEV Insider. *Was bedeutet LTS?* URL: <https://www.dev-insider.de/was-bedeutet-lts-a-959219> (besucht am 04.12.2021).
- [11] The PHP Group. *PHP: PHP 7 ChangeLog*. URL: <https://www.php.net/ChangeLog-7.php> (besucht am 04.12.2021).
- [12] The Composer Group. *Composer*. URL: <https://getcomposer.org/changelog/2.1.14> (besucht am 04.12.2021).
- [13] Helhum. *TYPO3-Console/TYPO3-Console: Console command for TYPO3 CMS*. URL: <https://github.com/TYPO3-Console/TYPO3-Console> (besucht am 04.12.2021).
- [14] git. *Git*. URL: <https://git-scm.com> (besucht am 04.12.2021).
- [15] Wikipedia. *Model View Controller – Wikipedia*. URL: https://de.wikipedia.org/wiki/Model_View_Controller (besucht am 05.12.2021).
- [16] Localdev Foundation. *DDEV-Local Documentation*. URL: <https://ddev.readthedocs.io/en/stable> (besucht am 06.12.2021).
- [17] Docker. *Empowering App Development for Developers — Docker*. URL: <https://www.docker.com> (besucht am 06.12.2021).

- [18] TYPO3.org. *Installing TYPO3 — Getting Started main documentation*. URL: <https://docs.typo3.org/m/typo3/tutorial-getting-started/main/en-us/Installation/Install.html#install> (besucht am 06.12.2021).
- [19] TYPO3.org. *TYPO3 Extension 'Extension Builder'*. URL: https://extensions.typo3.org/extension/extension_builder (besucht am 08.12.2021).
- [20] Wikipedia. *Grafische Benutzeroberfläche – Wikipedia*. URL: https://de.wikipedia.org/wiki/Grafische_Benutzeroberfl%C3%A4che (besucht am 08.12.2021).
- [21] TYPO3.org. *TYPO3 Extension 'The Official TYPO3 Introduction Package'*. URL: <https://extensions.typo3.org/extension/introduction> (besucht am 08.12.2021).
- [22] Wikipedia. *Redundanz (Informationstheorie) – Wikipedia*. URL: [https://de.wikipedia.org/wiki/Redundanz_\(Informationstheorie\)](https://de.wikipedia.org/wiki/Redundanz_(Informationstheorie)) (besucht am 08.12.2021).
- [23] TYPO3.org. *TYPO3 Extension 'External Data Import'*. URL: https://extensions.typo3.org/extension/external_import (besucht am 08.12.2021).
- [24] TYPO3.org. *General TCA configuration — External Import 6.0 documentation*. URL: https://docs.typo3.org/p/cobweb/external_import/6.0/en-us/Administration/GeneralTca/Index.html (besucht am 10.12.2021).
- [25] TYPO3.org. *TCA Reference — TCA Reference main documentation*. URL: <https://docs.typo3.org/m/typo3/reference-tca/main/en-us/Index.html> (besucht am 10.12.2021).
- [26] Wikipedia. *Dependency Injection – Wikipedia*. URL: https://de.wikipedia.org/wiki/Dependency_Injection (besucht am 12.12.2021).
- [27] Wikipedia. *Persistenz (Informatik) – Wikipedia*. URL: [https://de.wikipedia.org/wiki/Persistenz_\(Informatik\)](https://de.wikipedia.org/wiki/Persistenz_(Informatik)) (besucht am 12.12.2021).
- [28] Wikipedia. *Bubblesort – Wikipedia*. URL: <https://de.wikipedia.org/wiki/Bubblesort> (besucht am 14.12.2021).
- [29] TYPO3.org. *Flexforms — TYPO3 Explained main documentation*. URL: <https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/ApiOverview/FlexForms/Index.html> (besucht am 16.12.2021).
- [30] Wikipedia. *Extensible Markup Language – Wikipedia*. URL: https://de.wikipedia.org/wiki/Extensible_Markup_Language (besucht am 14.12.2021).
- [31] TYPO3.org. *Fluid Template Engine*. URL: <https://typo3.org/fluid> (besucht am 21.12.2021).
- [32] PHP. *PHP: filter_var - Manual*. URL: <https://www.php.net/manual/de/function.filter-var.php> (besucht am 21.12.2021).
- [33] TYPO3.org. *Individual database queries — Developing TYPO3 Extensions with Extbase and Fluid main documentation*. URL: <https://docs.typo3.org/m/typo3/book-extbasefluid/main/en-us/6-Persistence/3-implement-individual-database-queries.html> (besucht am 23.12.2021).

- [34] J. Böhm. *Erfolgsfaktor Agilität*. Hrsg. von Springer Vieweg. 2019, S. 119 ff.
- [35] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Hrsg. von Addison Wesley. 2003.
- [36] Michael Schams. *TYPO3 Extbase: Moderne Extension-Entwicklung mit Extbase und Fluid*. Hrsg. von Independently published. 2010.
- [37] Mark Richards. *Software Architecture Patterns*. Hrsg. von Inc. Reilly Media. URL: <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html> (besucht am 05.12.2021).