

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor

Technische Hochschule Wildau

Fachbereich Wirtschaft, Informatik, Recht

Studiengang Wirtschaftsinformatik (B. Sc.)

Thema (deutsch): Digitalisierung der Kundenkommunikation eines kleinen
Autohändlers mit Hilfe von TYPO3 CMS

Thema (englisch): Digitalization of customer communication of a small car dealer with
the help of TYPO3 CMS

Autor/in: Juljano Aliaj

Seminargruppe: I2/18

Betreuer/in: Prof. Dr. rer. nat. Alexander Lübke

Zweitgutachter/in: Dipl.-Informatiker (FH) Henning Almus (auch als Betreuer/in tätig)

Spätestmögliche Abgabe: 29.08.2022

Abstract

Mit der Digitalisierung in den letzten Jahren hat sich das Kaufverhalten der Kunden verändert. Das Verwenden von digitalen Kommunikationsgeräten fordert immerhin neue Formen der automatisierten Kundeninteraktion. Doch digitale Strategien werden bisher in den meisten kleinen Unternehmen gehindert. In einem Markt, in dem alle versuchen, digitale Prozesse voranzutreiben, scheint die Schwierigkeit der Umstellung bestehender Arbeitsprozesse eine große Herausforderung zu sein. Teil dieser Unternehmensgruppe sind in Deutschland auch zahlreiche Autohändler, die noch den klassischen Weg der Kundenkommunikation nutzen, was den Kundengewohnheiten inkonsistent steht. Verschiedenen Recherchen zufolge betreiben ein Großteil der Fahrzeughändler keine eigenen oder nur statischen Websites, sondern sind abhängig und bieten ihre Artikel über gängige Internetplattformen wie Social Media oder Online-Marktplätze an. Als Ausgangspunkt dieser Thesis wird ein solches Autohaus angenommen, dessen Digitalisierung zum Teil bereits durch die Nutzung von solchen B2C-Internetplattformen erfolgt ist. Die Arbeit wird einen Einstieg in ein neues Content-Management-System darstellen mit dem Ziel der praktischen Umsetzung unterschiedlicher Konzepte der Kundenkommunikation entsprechend dem neuen digitalen Verhalten des Kunden. Das zu erwartende Ergebnis ist die Steuerung einer Webanwendung, die kleinere Autohändler dazu anregen könnte, ein CMS in ihr Geschäft zu integrieren, was erhebliche Vorteile sowohl für Autokäufer als auch für das Unternehmen bringen könnte.

With digitalization in recent years, the consumer's way of buying has also changed. The use of digital devices in everyday life requires new forms of digital customer interaction. However, digital development remains stagnant in most small businesses, caused by the difficulty of converting existing work processes being a major challenge. Numerous car dealers in Germany, as a part of this group of businesses, continue to follow the classic way of client communication, which is inconsistent with the new buying behaviors of the fast-developing digital society. According to different research, most car dealers either do not apply their own websites or use static ones, and offer their products via other internet platforms such as social media or online marketplaces. As a starting point for this work, a car dealer will be considered, whose digitalization may be partly accomplished through the use of B2C platforms including social media. The work will demonstrate an entry into a new Content-Management-System for the business, with the aim of implementing different customer communication concepts, corresponding to the new digital habits of the customer. The expected result is the ability to develop and manage a web application that could encourage smaller car dealers to integrate a CMS in their business, which could be beneficial to both the car buyer and the company.

Inhaltsverzeichnis

1 Einleitung	6
1.1 Motivation	6
1.2 Zielstellung	7
2 Website Kundenkommunikation Strategien	8
3 Grundlagen des TYPO3 Content-Management-Systems	10
3.1 Systemarchitektur	11
3.1.1 MySQL	12
3.1.2 PHP	14
3.1.2.1 Objektorientierte Programmierung	14
3.1.2.2 Datenbank Verbindung	16
3.1.2 Extbase	17
3.1.2.1 MVC-Architecture	17
3.1.2.2 Fluid	18
3.1.2.3 TypoScript	20
3.1.2.4 TCA	22
3.1.2.5 Configurationsdateien und ext_tables.sql	23
3.2 Backend Aufbau	26
3.2.1 Module und Page Tree	26
3.2.2 Content Area	28
4 Projekt Realisierung	29
4.1 Setup	29
4.2 Entwicklung	32
4.2.1 Erweiterung von Kernfunktionen	32
4.2.2 Fahrzeugpräsentationen	35
4.2.3 Online Anfragen	39
4.2.4 Social Media Widgets	45
4.2.5 Feedback und FAQ	46
4.2.6 TYPO3 Blog Extension	48
5 Fazit	48
5.1 Zusammenfassung	48
5.2 Zukünftige Arbeit	49
6 Quellenverzeichnis	50
7 Anhang	55

Abbildungsverzeichnis

Abbildung 1: TYPO3 Architektur.....	12
Abbildung 2: Relative Tabellen in der Datenbank.....	13
Abbildung 3: Beispiel einer SQL-Abfrage.....	13
Abbildung 4: Beispiel einer PHP-Anweisung.....	14
Abbildung 5: PHP Klasse in Extbase.....	15
Abbildung 6: PHP-Abfrage.....	16
Abbildung 7: MVC-Modell.....	18
Abbildung 8: Link Action.....	18
Abbildung 9: HTML Link.....	18
Abbildung 10: If Viewhelper.....	19
Abbildung 11: For Viewhelper.....	20
Abbildung 12: Initiales TypoScript Page-Objekt.....	20
Abbildung 13: Entsprechendes PHP-Array des TypoScript Page-Objekts.....	21
Abbildung 14: TypoScript Konstantendefinition.....	21
Abbildung 15: Verwendung der TypoScript Konstanten in Markup.....	22
Abbildung 16: Registrierung des Plugins in tt_content.php.....	23
Abbildung 17: Plugin Konfiguration in ext_localconf.php.....	24
Abbildung 18: Registrierung des Moduls in ext_tables.php.....	24
Abbildung 19: Tabellenerstellung in ext_tables.sql.....	25
Abbildung 20: TYPO3 Admin Panel.....	26
Abbildung 21: Inhaltselement-Wizard.....	28
Abbildung 22: Der General Tab von Text & Media Inhaltselement.....	28
Abbildung 23: Der General Tab von Seiteneinstellungen.....	29
Abbildung 24: TYPO3-Instanz Ordnerstruktur in Linux.....	30
Abbildung 25: Erstellen einer neuen Datenbank und neuen Benutzer in MySQL.....	32
Abbildung 26: ext_emconf.php von Autohaus Extension.....	33
Abbildung 27: Ordnerstruktur für Autohaus Extension.....	33
Abbildung 28: Composer Konfiguration für Autohaus Extension.....	34
Abbildung 29: Root-Composer Konfiguration für das lokale Extension.....	34
Abbildung 30: Definition von Media Properties in der Autoklasse.....	35
Abbildung 31: Definition des TCA-Arrays für die Autoklasse.....	36
Abbildung 32: TCA-Aufzeichnungen für das Automodell im List Modul.....	37
Abbildung 33: Definition und Injizieren der Auto Repository-Klasse in Controller.....	38
Abbildung 34: Erstellen einer Action Methode für das Zuweisen von Auto Objekte ins View.....	38
Abbildung 35: Markup-Template der listAction.....	38
Abbildung 36: Einfügen des Auto-Plugins als Inhaltselement.....	39
Abbildung 37: Tabelle Relation für die Buchungsfunktion.....	40

Abbildung 38: Hinzufügen eines TCA Inline-Elements in der Autoklasse	40
Abbildung 39: TCA Inline-Element in TCA-Array	40
Abbildung 40: Definition von TCA-Array der Timeslot Klasse	41
Abbildung 41: Hinzufügen des Buchungsmonats in tx_autohaus_domain_model_auto	42
Abbildung 42: String in Timestamp umwandeln	42
Abbildung 43: Definieren von Markup-Templates für das Backend-Modul in TypoScript	43
Abbildung 44: Markup für die Anzeige von Zeitfenstern	43
Abbildung 45: Anzeige von Buchungszeitfenstern im Frontend	44
Abbildung 46: General Tab von Social Media Widget Plugin	45
Abbildung 47: Anzeige von Social Media Widget im Frontend	45
Abbildung 48: Zuordnung von Fremdtabelle in Autohaus Extension	47

Abkürzungsverzeichnis

API	Application Programming Interface
CMS	Content Management System
CSS	Cascading Style Sheets
DAT	Deutsche Automobil Treuhand
DB	Datenbank
DBAL	Database Abstraction Layer
DDD	Domain Driven Design
EM	Extension Manager
FAQ	Frequently Asked Questions
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	Javascript
MVC	Model View Controller
MySQL	My Structured Query Language
NW-Portale	Neuwagen Portale
OOP	Objektorientierte Programmierung
PHP	Hypertext Preprocessor
RDBMS	Relational database management system
SQL	Structured Query Language
TCA	Table Configuration Array
TER	TYPO3 Extension Repository
TPM	Technical Project Manager
URL	Uniform Resource Locators
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language

1 Einleitung

1.1 Motivation

Digitalisierung und digitale Kommunikationswege bieten ein breites Spektrum von Geschäftsoportunitäten an. Doch scheinen viele Autohändler die Opportunitäten des Internets beim Autoverkauf nicht bestmöglich auszunutzen. Um sich über den Kauf eines Fahrzeugs zu informieren, wenden sich viele Verbraucher bei ihrer Recherche neben großen Marken-Websites auch an Händler, die keine voll funktionsfähigen Websites haben.¹ Vielen Autohaus-Websites fehlen grundlegende digitale Funktionen, die die heutigen Autokäufer erwarten, daher bleibt diese Vertriebsstrategie noch mangelhaft.

DAT Report² zufolge hatten im Jahr 2021 99% der Autokäufer Zugang zum Internet, 87% davon haben das Internet als Informationsmittel im Kaufprozess genutzt und nur 18% haben ihren PKW über NW-Portale gekauft. Das zeigt, weit mehr als die Hälfte der Autokäufer waren eingeschränkt, ihr Wunschfahrzeug über analoge Kommunikationskanäle anzuschaffen. Ein Beispiel dafür ist das Verwenden des Telefons als häufigster Kommunikationskanal, welches trotz des Großteils der Umsatzgenerierung ausmacht, oft von manchen Unternehmen übersehen wird.³ Als Folge müssen viele interessierte Autokäufer, die innerhalb oder außerhalb der Öffnungszeiten anrufen, in vielen Fällen auf eine Probefahrt oder einen Werkstatttermin warten.

Mangelnde Produktinformationsanzeige kann Kunden zwingen, mehrmals in die Werkstatt zu gehen, bevor sie sich für ihr Lieblingsauto entscheiden. Die heutigen Generationen wenden sich in ihren Alltagssituationen überwiegend dem Internet zu, was die Suche nach Informationen mit wenigen Klicks erleichtert. Bei der Suche nach einem bestimmten Produkt werden gut strukturierte Websites mit besserer Suchmaschinenoptimierung auf den Suchergebnissen zuerst aufgelistet, und die große Vielfalt an Produktpräsentationen und interaktiven Features von solchen Websites könnte die Aufmerksamkeit des Kunden von den kleineren Händlern ablenken, die zu bestimmten Zeiten eventuell bessere Angebote

¹ Vgl. Coleman, Brian [Youtube]: Digitalization of the Car Industry: Following the Consumer, 2017, 17:00-17:25.

² Vgl. DAT Pressemitteilung, 2021, S. 5.

³ Vgl. LDB Gruppe: Vielen Dank für Ihren Anruf. Das Telefon im Autohaus., in: LDB Gruppe, 2020.

anbieten könnten. Trotz Datenverfügbarkeit und Fassungsvermögen stellt sich heraus, dass viele Fahrzeughändler diese nicht angemessen und strategisch nutzen.

1.2 Zielstellung

In der heutigen Gesellschaft ist die Online-Präsenz eines Unternehmens sehr wichtig. Ein Autohändler mit einer starken Internet-Präsenz kann mehr Autointeressierte begeistern als ein Händler mit mangelndem Fortschritt. Durch die Nutzung alter Websites kann ein Kunde zu dem Schluss kommen, dass sich das Unternehmen nicht genug um die Kunden kümmert, um sie mit einer besseren interaktiven Website zu begrüßen. Darüber hinaus, wenn ein Besucher kurz nach dem Webseitenaufruf zurückspringt, werden Suchmaschinen benachrichtigt, dass die Seite nicht das beste Ergebnis für die Suchanfrage war, was der Suchmaschinenoptimierung für die Website schaden könnte.⁴ Somit ist das übergeordnete Ziel dieser Arbeit, durch Umsetzung verschiedener digitaler Kommunikationsfeatures den Ankaufsweg für den Benutzer zu erleichtern und ihn davon zu überzeugen, mehr Zeit auf der Seite zu verbringen.

Da der Hauptfokus des kleinen Autohändlers mehr auf Verkauf und Kundenzufriedenheit liegt, muss die Pflege von Webinhalten und digitaler Kommunikationsinfrastruktur so zeitsparend und kosteneffektiv wie möglich sein. Darum empfiehlt diese Thesis die Integration eines Content-Management-Systems (CMS) in das Unternehmen, mithilfe dessen das Autohaus Kunden über neue Waren und Dienstleistungen auf dem Laufenden halten kann, eine bessere Präsenz im Internet schafft und als Folge neue Leads über eine strategische Kundenkommunikation generieren kann.

Für eine erfolgreiche Kundenkommunikation mit CMS werden in dieser Arbeit verschiedene Website Funktionen ausgewählt und in folgenden Abschnitten erläutert.

⁴ Vgl. Tyler Smith: How Your Current Traffic Impacts Your SEO, in: SmartBug., 2019.

2 Website Kundenkommunikation Strategien

Die betrachteten Features mithilfe einer kurzen Vorteilanalyse für deren Auswahl sind als wesentlich für die Kundenkommunikation einer Autohandel-Webseite ausgedacht. Für eine optimale Kundeninteraktion werden hier Publishing Tool, Online Anfragen, Social Media Links und Kunden-Feedback betrachtet. Für ein besseres Kundenerlebnis helfen Features wie Media-Produktpräsentationen sowie Blogs für ein steigendes Besucherengagement.

Publishing Tool

Eine Website zu erstellen und diese konsistent zu aktualisieren, erfordert Programmierkenntnisse und kann für ein Unternehmen, das nicht auf Software spezialisiert ist, sehr zeit- und kostenaufwändig sein. Dazu finden in der Gegenwart Website-Baukasten immer noch ein großes Ausmaß. Unternehmen müssen in der Lage sein, eine Vielzahl von Content wie Blogs oder Produktseiten mit aufgebauten Ressourcen schnell zu veröffentlichen und zu aktualisieren. Die Nutzer aus verschiedenen Bereichen müssen die Inhalte aus dem jeweiligen Sektor wie Fotos, Videos, neue Artikel usw. anpassen und den Online-Stand der Firma auf dem Laufenden halten. Neue Produktseiten oder andere digitale Leistungen müssen getestet und den Kunden zu bestimmten Zeiten präsentiert werden. Mit dieser Hinsicht werden hier als erste Anforderung einige der wesentlichen Veröffentlichungsfunktionen für die Website-Verwaltung gesetzt. Darunter zählen:

- Inhalt editieren (Text-Editor, Video/Image u. Artikel hochladen/entfernen usw.)
- Frontend Editor (Content Styling)
- Content/Page Staging (Inszenierung vor Veröffentlichung)
- Arrangierte (scheduled) Veröffentlichung
- Publishing Controls und Workflow (Zuweisen verschiedener Rollen und Zugriffsebenen innerhalb des CMS)

Fahrzeugpräsentationen

Für einen Käufer kann es sehr hilfreich sein, wenn die wichtigsten Details mit Bildern und Videos veranschaulicht werden. Lediglich die Rechtschreibung auf das Spezifische mag für Menschen, die keine grundsätzliche Vorstellung haben, immer noch sehr verwirrend sein, daher wird heutzutage Bild- und Videopräsentationen als Kommunikationsmittel große Bedeutung beigemessen.⁵ Dementsprechend wird betrachtet, dass die Anwendung neben

⁵ Vgl. Dragon2000: Why car dealers should be combining finance calculators with sales presentation videos, in: Dragon2000, 2020

den anzuzeigenden Informationen und der Bildoption eine Möglichkeit bieten kann, Videopräsentationen auf die Detailseite eines Autoartikels hochladen zu können.

Online Anfragen

Ein Autohaus kann ungeduldige Kunden verlieren, wenn seine Antwortzeit auf Probefahrt Anfragen unabhängig vom Kommunikationskanal länger dauert.⁶ Deshalb ist die kurze Reaktionszeit des Unternehmens ein wichtiger Aspekt der Kundenkommunikation. Die Suche nach einem neuen Auto beginnt am häufigsten mit dem Internet⁷ und eine solche Suche kann jederzeit innerhalb oder außerhalb der Öffnungszeiten anfangen. Ist der Website Besucher überzeugt, will er sich so schnell wie möglich in Kontakt mit dem Händler setzen und das Wunschfahrzeug testen. Mehr als die Hälfte der deutschen Verbraucher erwartet noch am selben Tag eine Antwort auf ihre Anfrage, jüngere unter 35 Jahren dagegen erwarten eine Antwort innerhalb von sechs Stunden⁸.

Um dieses Problem zu beheben und die Reaktionszeit komplett zu minimieren, kann das Anbieten einer Form für die Auflistung freier Autohaus-Termine helfen. Die Anwendung sollte in der Lage sein, eine Liste mit Übereinstimmungen von freien Mitarbeitern mit zum Zeitpunkt freien Fahrzeugen anzubieten. Wenn ein Online-Besucher für einen freien Termin auf der Autodetailseite interessiert ist, kann er jederzeit einen der generierten Terminergebnisse ohne unnötige Warteschlangen buchen und ein Terminformular ausfüllen. Dafür müssen Fahrzeugbestandsdaten immer aktuell für Kunden zur Verfügung stehen. Redakteure andererseits sollen in der Lage sein, die Mitarbeiter- und Fahrzeugverfügbarkeiten an die Anwendung weiterzugeben.

Social Media Links

Indem das Unternehmen seine Kommunikationskanäle erweitert, bleibt es in dem Bewusstsein dessen Verbraucher und kann sie gegebenenfalls bei späteren Gelegenheiten überzeugen.⁹ Mit dieser Einstellung bleibt die Verwendung von bestehenden Social Media Accounts für den Händler weiterhin eine gute Marketingstrategie. Außerdem kann man durch die Verlinkung von sozialen Netzwerken auf der Website Inhalte austauschen und darüber hinaus die Suchmaschinenoptimierung verbessern. Daher sollte die Anwendung eine Liste von Social Media Links auf der Website verwalten, die den Besucher auf den Social Media Kanal des Autohauses weiterleitet.

⁶ Vgl. LDB Gruppe: So geht Kundenkommunikation 4.0 im Autohaus., in: LDB Gruppe, 2017

⁷ Vgl. DAT Pressemitteilung, 2021, S. 5

⁸ Vgl. LDB Gruppe, 2017

⁹ Vgl. Jay Kang: Social Media Links & Your Website, in: SEOptimer, 2021.

Kundenfeedback

Unternehmen können ihren Wettbewerbsvorteil bewerten und Geld für Verbesserungen entlang entscheidender Situationen investieren, indem sie die Auswirkungen von Kundenanforderungen analysieren.¹⁰ Daher ist es wichtig, eine Möglichkeit für Kundenfeedback auf der Website anzubieten, um die aktuellen Anforderungen des Marktes zu erkennen. Website-Besucher können auf veröffentlichte Artikel zugreifen und ihre Meinung äußern. Das Autohaus kann diesen Informationspool für zukünftige Strategien nutzen, um seinen Autokäufern Anforderungen nachzukommen.

Blog

Nach der Empfehlung von dem Form4 GmbH TPM, um mehr Besucher begeistern zu können, kann das Autohaus technische Informationen zu fahrzeugbezogenen Problemen bereitstellen, die seinen Kunden mit weniger Wissen dabei helfen (D. Weingart, persönliche Kommunikation, 14.06.2022). Somit kann der Redakteur Schlüsselwörter in seinem Blog verwenden, sodass mehr Benutzer auf der Website landen, indem sie nach relativen Informationen suchen. Ein Beispiel könnte sein: „Unterschied zwischen Automatik- und Manuellgetriebe“. Auf diese Weise kann durch mehr Website-Traffic auch das Suchmaschinenranking verbessert werden.

3 Grundlagen des TYPO3 Content-Management-Systems

Die Umsetzung der oben genannten Features mit TYPO3 Content-Management-System kann eine komplexe Aufgabe sein, daher erfordert es ein gewisses Verständnis, bevor man mit der Entwicklung beginnt. Da es sich in dieser Arbeit nur um die interaktiven Features der Kundenkommunikation handelt, werden daher hauptsächlich die relevanten Backend-Technologien und Komponenten untersucht, die den Kernzweck bedienen.

Content-Management-System oder auf Deutsch Inhaltsverwaltungssystem ist ein zentrales System für die Verwaltung einer Website von dem innerlichen Inhalt bis hin zu ihrer Einordnung auf dem Netzwerk, und abhängig von der Relation zwischen Backend (Admin Oberfläche), Datenbanken und Frontend sind Content-Management-Systeme in 3 Kategorien aufgeteilt: klassisches CMS (volle Abhängigkeit), headless CMS (REST-API übernimmt Datenübertragung an mehrere Frontends) und hybrides CMS (Mischung von klassischem und headless).¹¹

¹⁰ Vgl. McAdams, D./ Stone, R./ Wood, K.: Functional Interdependence and Product Similarity Based on Customer Needs . *Res Eng Des* 11, 1–19 (1999), S. 2

¹¹ Vgl. SoftGuide: Content-Management-System, in: SoftGuide, o.D.

Mit der Einstellung, dass in der Ausgangslage ein kleiner Autohändler mit gewissen kleineren Kapazitäten steht, wird davon ausgegangen, dass das Unternehmen noch nicht bereit ist, weitere Anwendungen mit dem CMS außer seiner Website zu pflegen. Trotzdem werden die zukünftigen Opportunitäten nicht ausgeschlossen und hier ist ein ursprünglich klassisches CMS ausgewählt, das sich mithilfe von entsprechender Extension auch als Hybrid verhalten kann.

TYPO3 wurde im Jahr 1997 von Kasper Skårhøj initiiert und dann im Jahr 2000 unter der GNU Public Licence veröffentlicht.¹² D. h. TYPO3 steht heute als Open-Source zur Verfügung und kann jederzeit kostenlos heruntergeladen werden.

Es gibt verschiedene Methoden, eine Webanwendung mit TYPO3 zu entwickeln. Es wird in diesem Fall an die manuelle Vorgehensweise herangegangen und mit Hilfe von einem Code Editor werden die Konfigurationen an die benötigten Stellen angepasst.

3.1 Systemarchitektur

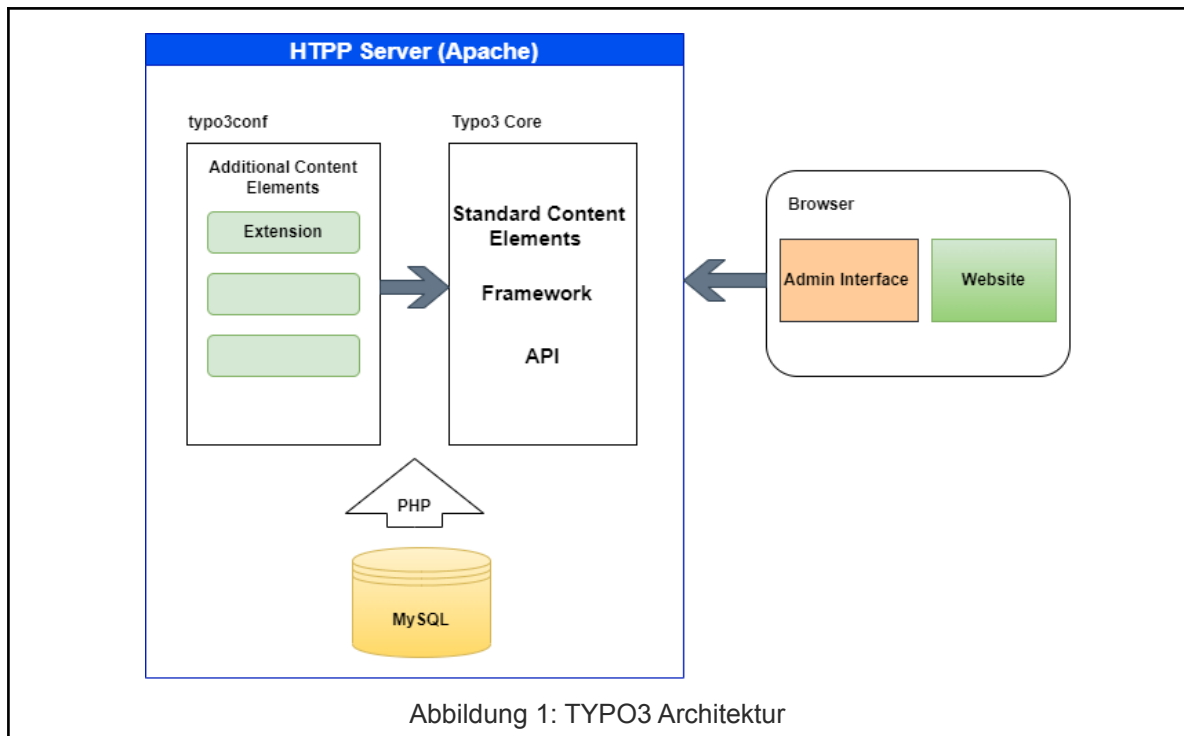
Das CMS kann auf einem HTTP Server deployt werden, und das Benutzerinterface sowie das Frontend können mit Hilfe von einem Browser aufgerufen werden. Die CMS Sprache ist PHP und es ist kompatibel mit verschiedenen Datenbankenverwaltungssystemen. In dieser Arbeit wird ein Apache Server mit MySQL Datenbanken verwendet, weil beide Schnittstellen besonders für TYPO3 geeignet sind.¹²

Unter anderem wird TYPO3 als ein umfangreiches und flexibles Framework bezeichnet, weil es eine Anwendungsumgebung zur Verfügung stellt, in der man die Kernfunktionalität des TYPO3 Core weiterentwickeln kann.¹³ Mit dieser Einstellung ist TYPO3 eine Zusammensetzung von verschiedenen Extensions, die jeweils für bestimmte Aufgaben zuständig sind. Der Kern von TYPO3 selbst besteht aus Extensions, die auch als Systemerweiterungen bezeichnet¹⁴ und automatisch zusammen mit der Plattform installiert werden. Die zweite Art von Erweiterungen sind entweder allgemeine Funktionserweiterungen, die von der TYPO3-Community entwickelt werden und in TYPO3

¹² Vgl. Falkenberg, Karen: TYPO3 CMS 6.2 Grundlagen: für Redakteure und Integratoren, 1.Ausgabe, 1. Aktualisierung, Herdt, 2015, S. 8.

¹³ Vgl. Lammenett, Erwin: TYPO3 Online-Marketing-Guide: Affiliate- und E-Mail-Marketing, Keyword-Advertising, Suchmaschinen-Optimierung mit TYPO3, Wiesbaden, Deutschland: Gabler, 2007 S.13-14.

¹⁴ Vgl. Falkenberg, 2015, S.112.



Extension Repository (TER) verfügbar sind oder benutzerdefinierte Erweiterungen, die von anderen Entwicklern lokal erstellt werden, um die einzigartigen Anforderungen von bestimmten Websites zu erfüllen.

Das Thema der TYPO3-Systemarchitektur ist ziemlich weit gefasst; es gibt diverse Konfigurationen und mögliche Funktionen, die das CMS bereitstellt. In dieser Arbeit werden die wesentlichen Elemente behandelt, die notwendig sind, um eine grundlegende, anfangs funktionsfähige Website basierend auf den vorausgesetzten Anforderungen aufzubauen.

3.1.1 MySQL

Janet Valade¹⁵ zufolge ist die Datenbank das Herz einer datenbankgestützten Webanwendung. Es enthält alle wichtigen Informationen, die für das Handeln über die Website im Allgemeinen erforderlich sind. Bei einem Autohaus sind Datenbanken dafür zuständig, Produktinformationen und Bestände für Kunden bereitzustellen. Besucherrouten können an bestimmten Prozeduren in der Datenbank protokolliert werden und Anbieter auf potenzielle Verwaltungs- und Unterstützungssituationen aufmerksam machen.

¹⁵ Vgl. Valade, Janet: PHP and MySQL Für Dummies, USA: John Wiley & Sons, Incorporated, 2017, S. 34

MySQL, das von der Oracle Corporation angeboten und von mehreren großen Webanwendungen wie Facebook und Twitter verwendet wird, ist heutzutage die beliebteste Wahl für relationale Datenbankverwaltungssysteme (RDBMS)¹⁶.

Wie eine Tabellenrelation von RDBMS in MySQL funktioniert, ist mit Hilfe von MySQL Tutorial¹⁷ anhand der unteren Abbildung kurz zusammengefasst.

table1			table2				
id	marke	beschreibung	id	marke	model	jahr	preis
1	BMW	Ein Auto mit schönen Design.	351	BMW	Serie 3	2010	€19.900,-
2	Volkswagen	Deutsches qualitatives Auto.	352	BMW	760	2014	€49.990,-

Abbildung 2: Relative Tabellen in der Datenbank

In einem RDBMS werden einzelne Daten in verschiedenen Tabellen gespeichert, die miteinander basierend auf übereinstimmenden Spalten in Beziehung stehen.

Eine Tabelle besteht aus Zeilen, die jeweils einen bestimmten Datensatz für einen Datenbankeintrag enthalten. Es umfasst eine ID, eine spezielle numerische Eigenschaft, die jeden Datensatz unterscheidet und das Auffinden erleichtert. Alle speziellen Informationen dieses Datensatzes sind in Spalten enthalten und jede einzelne Spalte repräsentiert einen speziellen SQL Datentyp. Die Standardsprache von RDBMS und darüber hinaus von MySQL heißt SQL. Damit werden die Datenbanken ebenso wie die Abfragen verwaltet, die für bestimmte Aufgaben erforderliche Daten liefern.¹⁸ Als Beispiel sieht man, dass die Spalte *marke* in beiden Tabellen enthalten ist. Diese Spalte steht dann als Relation zwischen den beiden Tabellen. Man kann hier eine allgemeine Beschreibung für das Modell der Serie 3 nachschlagen, indem man auf die erste Tabelle zugreift und mit Hilfe von Attribut *marke* die *beschreibung* abrufen. Die SQL-Abfrage, um diese Information herauszuholen, wäre:

```
SELECT beschreibung FROM table1 WHERE marke = "BMW";
```

Abbildung 3: Beispiel einer SQL-Abfrage

¹⁶ Vgl. Mehta, Chintan/ Bhavsar, Antik/ Oza, Hetal/ Shah, Subhash:MySQL 8 Administrator's Guide:Effective Guide to Administering High-Performance MySQL 8 Solutions, Birmingham-Mumbai: Packt Publishing, Limited, 2018, S. 6.

¹⁷ Vgl. MySQL Tutorial, MySQL RDBMS, in: W3schools, o.D..

¹⁸ Vgl. Valade, 2018, S. 38.

3.1.2 PHP

PHP steht für Pre-Processor Hypertext und ist meistens im Webdesign als Server-Side Skriptsprache bekannt.¹⁹ Es wird in HTML eingebettet, bekommt im Vergleich die Erweiterung *.php* oder *.phtml*, um zu zeigen, dass die Datei PHP Anweisungen enthält, welche für dynamische Contents innerhalb HTML sorgen.²⁰ PHP verfügt über umfangreiche Funktionen und Einstellungsmöglichkeiten. PHP Anweisungen werden immer innerhalb von PHP-Tags definiert, die abhängig vom Framework innerhalb des HTML-Codes oder auch in bestimmten einzelnen Dateien dargestellt werden können.

```
<?php echo "das ist ein PHP Befehl"; ?>
```

Abbildung 4: Beispiel einer PHP Anweisung

3.1.2.1 Objektorientierte Programmierung

Eines der nützlichsten Features von PHP ist die OOP (objektorientierte Programmierung), deren Bedeutung für TYPO3 Komponente namens Extbase in den nächsten Abschnitten erläutert wird. Wichtig ist, dass PHP Objekte in Webanwendungen eng mit Datenbanken verbunden sind und eine Menge von Funktionen zur Datenbank Interaktion bereitstellen. Ein Objekt in PHP kann innerhalb einer PHP-Klasse mit entsprechenden Eigenschaften und Methoden definiert werden. Diese letzten führen bestimmte Funktionen auf diese Objekte aus, was Objekteigenschaften ändern kann oder sie beispielsweise im Falle von Getter und Setter außerhalb der Klasse zugreifbar machen etc.

Um eine Vorstellung davon zu bekommen, wie Objekte in PHP dargestellt werden, dient die folgende Abbildung als Beispiel, und es wird versucht mithilfe von Lobacher und Schams²¹ zu erklären:

¹⁹ Vgl. Carr, David/ Gray, Marcus: Beginning PHP: Master the latest features of PHP7 and fully embrace modern PHP development, Birmingham-Mumbai: Packt Publishing, Limited 2018, S. 1.

²⁰ Vgl. Valade, 2018, S. 40.

²¹ Vgl. Lobacher, Patrick/ Schams, Michael: TYPO3 Extbase, Modern Extension Development for TYPO3 CMS with Extbase & Fluid: Open Source Press GmbH, 2015, S. 37-42.

```

namespace VENDOR\Autohaus\Domain\Model;

Class Auto {
    protected $marke;

    public function getMarke() {
        return $this->marke;
    }

    public function setMarke() {
        $this->marke = $marke;
    }
}

```

Abbildung 5: PHP Klasse in Extbase

Klassennamen werden immer mit großem Anfangsbuchstabe geschrieben, sowie alle anderen in dem Namen Bedeutungseinheiten. Diese Schreibweise nennt man auch UpperCamelCase. Die Eigenschaften oder Properties, sowie Methoden stehen innerhalb von den geschweiften Klammern. Geschützte Variablen sind praktisch, wenn eine Eigenschaft nicht einfach geändert werden muss und ihr Zugriff nur auf ihre eigene Klasse erlaubt ist. Properties und Methoden werden nach dem LowerCamelCase geschrieben, wobei der Anfangsbuchstabe immer klein ist und jede andere Bedeutungseinheit folgend mit Großbuchstabe anfängt. Vor der Property im Vergleich steht Charakter \$. Die Getter und Setter sind Methoden, um den Zugriff auf diese Properties außerhalb der Klasse weiterhin haben zu können. In diesem Fall sind die Methoden öffentlich, sodass sie auch außerhalb der Klasse leicht verwendbar sind. Der Methodenname wird von runden Klammern gefolgt, wobei gegebenenfalls auch Parameter übergeben werden können. Die Funktionalität der Methode wird innerhalb von geschweiften Klammern dargestellt. Schlüsselwort *\$this* gefolgt von Pfeiloperator ist eine Pseudovariablen, die nur innerhalb von Methoden verfügbar ist und für den Zugriff auf eine Property oder Methode innerhalb der aktuellen Instanz (Klasse) verwendet wird.

Weiter dazu ist die erste Deklaration auf der oberen Abbildung der Namespace. Es weist auf den Pfad, wo die Klasse zu finden ist, hin. Seit Version 6.0 verwendet TYPO3 Namespaces für alle Klassen in Core, dementsprechend ist das Verwenden von solchen Pfaden wichtig bei der Erstellung neuer Klassen. Dafür sind die Quellen aus der TYPO3-Dokumentation empfehlenswert.²²

²² Vgl. TYPO3 Contributors: TYPO3 Explained / Namespaces, in: typo3.org.

3.1.2.2 Datenbank Verbindung

PHP wird ebenfalls für die Kommunikation mit der Datenbank verwendet, nämlich für das Auslesen und Übermittlung der von Nutzern angegebenen Daten über das Front-/Backend. In TYPO3 wird eine Datenbankverbindung mithilfe der Doctrine DBAL hergestellt, einer API mit Query-Building Features für SQL, die es ermöglicht, unabhängige DB Abfragen zu erstellen²³, bzw. durch PHP-Code SQL-Abfragen im Hintergrund durchzuführen.

In der unteren Abbildung wird eine Methode, um die SQL-Abfrage aus Abbildung 3 in Kapitel 3.1.1 in PHP-Code darzustellen, gezeigt. Die Anleitung aus der TYPO3 Community unter Querybuilder²⁴ dient hier als Orientierung und ist für weitere Informationen über dieses Thema weiterzuempfehlen.

```
//use TYPO3\CMS\Core\Utility\GeneralUtility;
//use TYPO3\CMS\Core\Database\ConnectionPool;

public function findBeschreibungFromMarke() {

$queryBuilder =
GeneralUtility::makeInstance(ConnectionPool::class)->getQueryBuilderForTable('table1');

return $queryBuilder->select('beschreibung')
    ->from('table1')
    ->where('marke', 'Serie3');
}
```

Abbildung 6: PHP-Abfrage

Zuerst wird die *ConnectionPool* Klasse in eine Variable namens *\$queryBuilder* instanziiert. *ConnectionPool* ist eine Doctrine-PHP-Klasse und hilft, eine bestimmte Verbindung zu Daten in der Datenbank herzustellen. Die *getQueryBuilderForTable* Methode bekommt den Tabellennamen, in dem abgefragt wird, als Parameter. Weitere Methoden werden die Beschreibung zurückgeben. Für die Instanziierung der *ConnectionPool*, wird *GeneralUtility* verwendet. Das ist eine Klasse von den TYPO3 Klassen, die APIs für bestimmte TYPO3-Funktionalitäten bereitstellen.²⁵ Eine Methode dieser Klasse ist die *makeinstance*, die ein Objekt aus der eingegebenen Parameterklasse erstellt.

²³ Vgl. TYPO3 Contributors: TYPO3 Explained / Database (Doctrine DBAL) / Introduction, in: typo3.org.

²⁴ Vgl. TYPO3 Contributors: TYPO3 Explained / Database (Doctrine DBAL) / Querybuilder, in: typo3.org.

²⁵ Vgl. TYPO3 Contributors: TYPO3 Explained / Main classes and methods / High priority functions, in: typo3.org.

3.1.2 Extbase

Extbase ist eine Zusammensetzung aus dem MVC-Framework, dem Domain Driven Design (DDD) und der Templating-Engine namens Fluid und wurde initiiert, um die OOP in der Webentwicklung als die häufigste Neigung jeder Software zu unterstützen.²⁶

3.1.2.1 MVC-Architecture

Zunächst wird versucht mithilfe von TYPO3 Community Dokumentation unter Model-View-Controller²⁷ die strukturellen Grundlagen von diesem Abschnitt zusammenzufassen. Model-View-Controller in der TYPO3-Entwicklungsumgebung ist ein Framework, das die OOP von domänengetriebenem Design mit den Templates bündelt. Es ermöglicht die Abbildung von Objekten aus der Datenbank und deren Präsentation im Frontend durch ein definiertes Muster. MVC unterteilt die Anwendung in drei Schichten:

Model Schicht

In dieser Schicht werden Objektklassen definiert. Das domänengesteuerte Design ermöglicht in dieser Schicht die Funktionalität des Repositorys, wo man PHP-Abfragen als Methoden erstellen kann, um über die definierten Objekte mit der Datenbank zu interagieren, sowie anderer Dienste, um Regeln für diese Objekte zu implementieren. Um die in letzteren definierten Methoden aufzurufen, müssen die Repositories und die Dienste in dem Controller konstruiert werden. Ein typische PHP Abfrage aus einer Repository Klasse wurde in Abbildung 6 gezeigt.

Controller Schicht

Das ist das Gehirn dieses Frameworks. Es nimmt die definierten Methoden, d. h. DB Abfragen und Restriktionen aus der Domäne (Model Schicht) und stellt sie für die View bereit. Der Controller besteht aus verschiedenen Methoden, auch als Actions bezeichnet. Jeder von ihnen repräsentiert eine View Vorlage, an die die Daten nach jeder vom Server gesendeten Anfrage übergeben werden sollen.

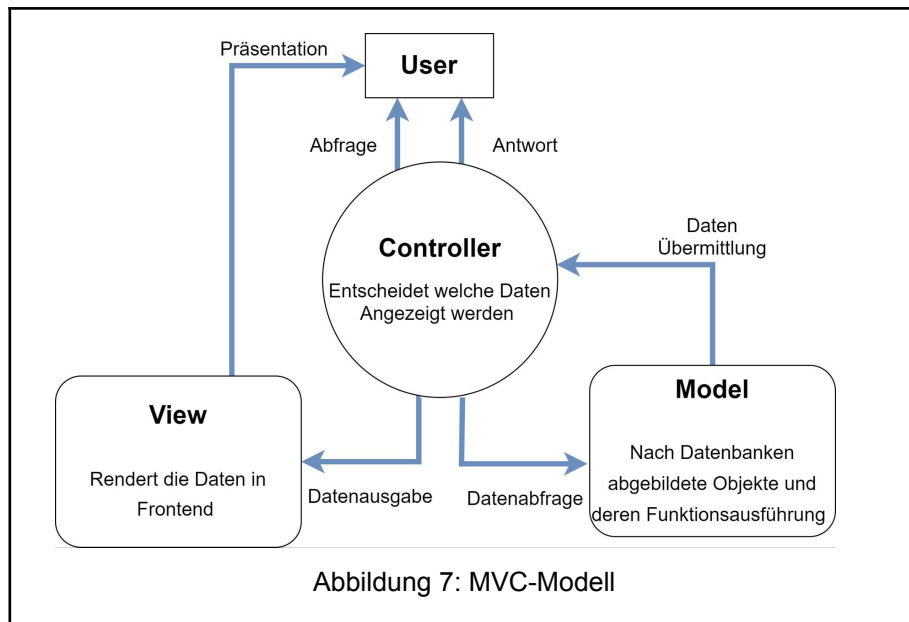
View Schicht

Die View Schicht kapselt die eingeschränkten Daten vom Controller und stellt sie für den Benutzer bereit.

Eine strukturelle Abbildung für das Abrufen von Daten und deren Präsentation für den Benutzer ist in der folgenden Abbildung veranschaulicht.

²⁶ Vgl. Lobacher und Schams, 2015, S. 34

²⁷ Vgl. Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Model-view-controller (MVC) in Extbase, in: typo3.org.



3.1.2.2 Fluid

Fluid ist die Standard-Rendering-Sprache für TYPO3, die auf XML und HTML-Auszeichnungssprachen basiert.²⁸ Mit Fluid werden innerhalb von HTML-Tags zusätzlich XML Komponenten eingebunden, die für bestimmte Funktionen verwendet werden, die HTML allein innerhalb des Extbase-Frameworks nicht leisten kann. Damit sie mit dem Browser kompatibel sind, werden diese XML Elemente schließlich komprimiert und vom CMS, mit Hilfe der PHP Konfigurationskomponente für Fluid, namens Viewhelper, in HTML-Code umgewandelt.²⁸ Dieses Konzept lässt sich mithilfe des folgenden Fluid-Snippets besser verstehen:

```
<f:link.action extension="autohaus" controller="auto" action="list"> Autoliste anzeigen</f:link.action>
```

Abbildung 8: Link Action

Die *f:link.action* baut einen Verweis auf die in der *listAction* definierte Objektliste auf und leitet den Benutzer zu ihrer Vorlage. Dies entspricht dann dem folgenden HTML-Hyperlink:

```
<a href="index.php?id=4569&tx_autohaus_auto[action]=list&tx_autohaus_auto[controller]=Autod &chash=xyz1324">Autoliste anzeigen</a>
```

Abbildung 9: HTML Link

²⁸ Vgl. TYPO3 Contributors: TYPO3 Explained / Fluid, in: typo3.org.

Die Verzeichnisse in Extbase, die für die Verwaltung von Fluid Templates zuständig sind, sind in 3 Kategorien unterteilt: Partial, Layout und Template, und befinden sich unter Resources/Private Verzeichnis.²⁹ Das Partial-Verzeichnis umfasst kleinere charakteristische Teile, die entweder in das Layout Markup oder in Templates importiert und wiederholend²⁹ gerendert werden können. Das Layout-Verzeichnis dient dazu, das grundlegende Markup der Seite zu definieren, und das Template-Verzeichnis enthält das gesamte Markup zum Rendern der im Controller definierten Inhaltselemente oder sogenannten Plugins.

Der Unterordner des Template Verzeichnisses sollte den gleichen Namen wie der Controller, und die HTML-Datei darin sollte den gleichen Namen wie die Action haben, wobei die Namenskonventionen³⁰ zu beachten sind. Dadurch können die von der Controller Action zugewiesenen Daten in dem Template gerendert werden. Daher müssen für so viele Controller-Klassen, die in der Extension vorhanden sind, so viele Template-Unterordner erstellt werden.

Zunächst wird veranschaulicht, wie die in das Template übergebenen Controller-Daten verwendet werden, im Beispiel eines *if* Viewhelpers:

```
<f:if condition="{auto.booked} == 0">
  <f:then>Auto ist verfügbar</f:then>
  <f:else if="{auto.booked} == 1">
    Auto ist nicht mehr verfügbar
  </f:else>
</f:if>
```

Abbildung 10: If Viewhelper

Dies ist eine einfache If-Bedingung, die basierend auf den übergebenen Daten verschiedene Optionen rendert. Das zugewiesene Argument kann eine Property, ein Array oder ein Objekt sein. Daten müssen innerhalb von geschweiften Klammern eingegeben werden, um von Fluid gelesen zu werden. Die Punkttrennzeichen dienen dafür, um auf die Hierarchieebenen innerhalb dieser Daten zuzugreifen. In diesem Fall greift man auf die Property *booked* von Objekt *auto* zu.

Das nächste Beispiel in dem unteren Ausschnitt zeigt ein Foreach-Konstrukt, das die Iteration innerhalb eines Arrays oder Objekts ausführt. In diesem Fall gibt das Konstrukt für jedes *auto* Objekt automatisch die Marke und das Fahrzeugmodell aus.

²⁹ Vgl. TYPO3 Contributors: TYPO3 Explained / Extension Development / File Structure / Resources / Private, in: typo3.org.

³⁰ Vgl. TYPO3 Contributors: TYPO3 Explained / Extension Development / Best practices and conventions / Naming conventions, in: typo3.org.

```

<f:for each="{auto}" as="item" key="itemkey"
    <div class="marke">
        {item.marke}
    </div>
    <div class="model">
        {item.model}
    </div>
</f:for>

```

Abbildung 11: For Viewhelper

Dies war ein kurzer Überblick über Fluid. Um ein tieferes Verständnis zu bekommen, findet man weitere Informationen unter TYPO3-Dokumentation.³¹

3.1.2.3 TypoScript

TypoScript ist eine Konfigurationssprache, in der definiert wird, wie eine Website gerendert werden soll.³² Daher ist es wichtig, jeder Seite im Backend Page-Tree (in Kapitel 3.2.1 erläutert) ein TypoScript Template zuzuweisen. Unterseiten können dasselbe Template wie ihre Parent Seiten verwenden, man könnte trotzdem auch unterschiedliche Templates basierend auf eigenen Anforderungen einbinden. Die initiale TypoScript-Einstellung, die mit der TYPO3-Installation kommt, sieht wie folgt aus:

```

page = PAGE
page.10 = TEXT
page.10.value = Hello World

```

Abbildung 12: Initiales TypoScript Page-Objekt

Jedes TypoScript Template sollte ein PAGE-Objekt unter *Configuration/TypoScript/setup.typoscript* deklarieren, das zur Strukturierung der Seite verwendet wird. Es gibt keine Beschränkung für die Anzahl der nummerierten Elemente, die verwendet werden, um weitere Objekte in der Seite zu definieren, die auch als Sortiereigenschaften bezeichnet werden.³³ Diese Letzten können verschiedene Typen sein, wie Texte, Inhaltselemente, Templates usw. Innerhalb davon wird gesteuert, wie die Inhaltskomponenten der Website angezeigt werden, welche Vorlagen in welchen Bereichen der Seite, welche CSS- und Javascript-Dateien auf jeglichen Inhalt angewendet werden usw.

³¹ Vgl. TYPO3 Contributors: TYPO3 Explained / Fluid, in: typo3.org.

³² Vgl. TYPO3 Contributors: TypoScript in 45 Minutes / TypoScript - A quick overview / Why TypoScript, in: typo3.org.

³³ Vgl. TYPO3 Contributors: TypoScript in 45 Minutes / TypoScript - A quick overview / First steps, in: typo3.org.

Das erste Element auf der Seite im obigen Beispiel ist ein Text mit dem Wert „Hello World“. Die Konfiguration wird dann in einem PHP-Array gespeichert, das wie folgt aussieht:

```
$data = [
    'page' => 'PAGE',
    'page.' => [
        '10' => 'TEXT',
        '10.' => [
            'value' => 'Hello World',
        ],
    ],
],
```

Abbildung 13: Entsprechendes PHP-Array des TypoScript Page-Objekts³⁴

Constants

TypoScript ermöglicht die Verwendung von Konstanten, die in der Programmiersprache als Variablen gelten und an verschiedenen Stellen³⁵ wie in *setup.typoscript*, Controller und Fluid verwendet werden können. Die Konstanten können unter

Configuration/TypoScript/constants.typoscript definiert werden. Sie können hier einen Anfangswert zugewiesen bekommen, der später, wenn es nötig ist, über das Backend überschrieben werden kann. Extbase erkennt die definierten Konstanten, sodass die Datei nicht importiert werden soll. Hier ist ein einfaches Beispiel für die Definition des Website-Logo-Bildpfads:

```
page {
    logo {
        path = EXT:my_extension/Resources/Public/Images/Logo.svg
        width = 50px
        height = 50px
    }
}
```

Abbildung 14: TypoScript Konstantendefinition³⁵

Um die definierten Konstanten weiter in Extbase zu verwenden, müssen sie zuerst in *setup.typoscript* in dem Page-Objekt unter *settings* Array weitergegeben werden, bzw.:

³⁴ Vgl. TYPO3 Contributors: TypoScript in 45 Minutes / TypoScript - A quick overview / TypoScript is just an array, in: typo3.org.

³⁵ Vgl. TYPO3 Contributors: TypoScript Reference / Using and setting TypoScript / Constants, in: typo3.org.

`settings['logoPath'] = {$page.logo.path}`. In Fluid stehen dann die entsprechenden Array-Indices als Argumente in geschweiften Klammern:

```

```

Abbildung 15: Verwendung der TypoScript Konstanten in Markup

Wenn die Konstante im Controller genutzt werden soll, könnte

```
$this->settings['logoPath'] verwendet werden und für TypoScript  
{settings.logoPath}.
```

Um TypoScript besser zu verstehen, bietet die Contributor-Dokumentation für TYPO3 weitere Konfigurationsmöglichkeiten.³⁶

3.1.2.4 TCA

TCA oder Table Configuration Array ermöglicht es, mit der Datenbank über das Backend zu interagieren.³⁷ Unter Verwendung des List Moduls im Backend (Backend Module werden in Kapitel 3.2.1 erläutert) kann man eine Schnittstelle mit der Datenbank herstellen, um neue Dateneinträge hinzuzufügen, sie nachher zu bearbeiten oder zu löschen. Um die DB Felder im Backend zugänglich zu machen, muss ein spezifisches PHP-Array für die entsprechende DB Tabelle, bzw. eine PHP-Datei im *Configuration/TCA* Verzeichnis mit demselben Namen wie die SQL-Tabelle erstellt werden. Wenn eine Tabelle *tx_autohaus_domain_model_auto* heißt, sollte die PHP-Datei für das TCA *tx_autohaus_domain_model_auto.php* heißen.³⁸ Die Felder innerhalb des Arrays müssen auch die gleichen Namen wie die Tabellenspalten haben. Durch TCA werden die SQL-Datentypen an das Backend weitergegeben und in Texte, Prüffunktionen und sogar Text-Mediendateien umgewandelt. Beispielsweise ist *check* der entsprechende TCA Typ für den SQL Datentyp einer *boolean* Spalte, der im List Modul ein Kontrollkästchen generiert, um den Wert dieses Felds bzw. von *true* auf *false* zu ändern. Eine andere nützliche Funktion von TCA für die Anwendung ist die Unterordnung von Tabellen zu anderen Tabellen. Um dies zu erreichen, kann der Spaltentyp als *Inline* definiert werden. Dies setzt voraus, dass die untergeordnete Tabelle immer mit ihrer übergeordneten

³⁶ Vgl. TYPO3 Contributors: TypoScript Reference, in: typo3.org.

³⁷ Vgl. TYPO3 Contributors: TCA Reference / Introduction, in: typo3.org.

³⁸ Vgl. TYPO3 Contributors: TYPO3 Explained / Extension Development / Best practices and conventions / Naming conventions, in: typo3.org.

verbunden ist und ihre Existenz ohne diese letzte sinnlos ist.³⁹ Für weitere Informationen zeigt die Doku aus der TCA Referenz die vollständige Auswahl an TCA Feldtypen.⁴⁰

Eine weitere wichtige TCA Datei befindet sich in *Configuration/TCA/Overrides* und heißt *tt_content.php*. Diese Datei muss ergänzt werden, um einzelne Inhaltselemente zu registrieren und zu speichern. Um beispielsweise ein Plugin zu registrieren, dient die folgende Abbildung als Beispiel:⁴¹

```
<?php
Use TYPO3\CMS\Extbase\Utility\ExtensionUtility;

(static function () {
    ExtensionUtility::registerPlugin('Autohaus', 'AutoPlugin', 'Auto Plugin');
})
```

Abbildung 16: Registrierung des Plugins in tt_content.php

Die hier an das *registerPlugin* von der *ExtensionUtility* Klasse übergebenen Parameter sind Extensionname, Pluginname und Plugintitel.

Zudem wird im Kapitel Projektrealisierung eine praktische Anwendung des TCA angezeigt.

3.1.2.5 Configurationsdateien und ext_tables.sql

Die unten beschriebenen Dateien befinden sich auf der obersten Ebene des Extension Verzeichnisses.

ext_emconf.php

In TYPO3 Admin Interface gibt es eine Sektion (Extension Manager), um Extensions entweder zu aktivieren oder zu deaktivieren. Beim Erstellen von neuen lokalen Extensions muss diese Konfigurationsdatei ergänzt werden. Die Datei beinhaltet ein assoziatives Array, das bestimmte Aspekte wie Extension-Titel, -Kategorie, -Autor, -Abhängigkeiten usw. zusammenfasst. Anhand dieser Datei kann der EM die neue Extension identifizieren und sie in seine Extensionliste mitnehmen.⁴²

³⁹ Vgl. Lobacher und Schams, 2015, S. 302.

⁴⁰ Vgl. TYPO3 Contributors: TCA Reference, ['columns']['*']['config'], in: typo3.org.

⁴¹ Vgl. Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Creating a First extension / Configuring the Plugin, in: typo3.org.

⁴² Vgl. TYPO3 Contributors: TYPO3 Explained / Extension Development / File Structure / ext_emconf.php, in: typo3.org.

ext_localconf.php

Diese Datei ist dafür zuständig, Plugins im Admin-Interface verfügbar zu machen.

```
<?php
defined('TYPO3_MODE') or die();

\TYPO3\CMS\Extbase\Utility\ExtensionUtility::configurePlugin(
    $_EXTKEY,
    'DisplayAvailableVehicles',
    [
        'DisplayAvailableVehicles' => 'index, show',
    ],
    [
        'DisplayAvailableVehicles' => 'index, show',
    ]
);
```

Abbildung 17: Plugin Konfiguration in ext_localconf.php

Hierfür kommt die Methode *configurePlugin* aus der *ExtensionUtility* Klasse zur Nutze. Im Grunde genommen bezieht sich eine Extension auf verschiedene Plugins, welche gemeinsame Anwendungsziele anstreben. Jedes Plugin repräsentiert einen bestimmten Controller, in dem definiert wird, was für eine Funktion dieses Plugin erfüllen soll. Mit der *\$EXT_KEY* wird der Extensionname, wonach die Anwendung für ein bestimmtes Plugin suchen soll, bekannt gemacht. Nachfolgend kommen Controllername und die zwei letzten assoziativen Arrays mit den Controller Actions. Das erste Array registriert die gelisteten Actions und das zweite Array sorgt dafür, dass sie von dem CMS ungecached bleiben.⁴³

ext_tables.php

Damit die Admins verschiedene Plugins auf der Admin-Oberfläche verwalten und mit deren Inhalten interagieren können, kann diese Datei verwendet werden, um neue Backend-Module zu registrieren. Die Konfiguration kann wie im folgenden Beispiel implementiert werden:

```
\TYPO3\CMS\Extbase\Utility\ExtensionUtility::registerModule(
    'VENDOR.autohaus', 'web', 'tx_Bookingdisposalcontroller', 'bottom',
    [
        'BookingDisposal' => 'index, createBookings',
    ],
    [
        'access' => 'admin',
        'labels' => 'Generate Booking Availabilities',
    ]
);
```

Abbildung 18: Registrierung des Moduls in ext_tables.php⁴⁴

⁴³ Vgl. Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Creating a First extension / Configuring the Plugin, in: typo3.org.

⁴⁴ Vgl. TYPO3 Contributors: TYPO3 Explained / Extension Development / File Structure / ext_tables.php, in: typo3.org.

Der erste Parameter, der der registerModule Methode übergeben wird, ist der Name des Anbieters, gefolgt vom Namen der Extension, in der dieses Modul zur Verfügung steht. Der zweite Parameter ist der Dashboard-Abschnitt, zu dem das Modell gehören wird. Der dritte Parameter definiert den dafür zuständigen Controller und der vierte Parameter setzt fest, wo das Modul im Dashboard angezeigt wird. Der nächste Parameter ist ein Array mit dem Controller und entsprechenden Actions, und der letzte Parameter-Array gibt ein Label sowie den Zugriff auf eine bestimmte Benutzergruppe im Backend.

ext_tables.sql

In Extbase bekommt diese Datei SQL-Building-Queries, die benötigt werden, um Tabellen in der Datenbank zu erstellen und bestehende verändern zu können.⁴⁵ Nachdem der Code auf dem Server deployt wird, muss eine Datenbankanalyse in dem Maintenance Modul ausgeführt werden, um die Änderungen in der Datenbank zuzulassen. Eine SQL-Einstellung kann wie folgt aussehen:

```
CREATE TABLE tx_autohaus_domain_model_auto (  
    uid int(11) unsigned DEFAULT 0 NOT NULL auto_increment,  
    pid int(11) DEFAULT 0 NOT NULL,  
    model varchar(255) NOT NULL,  
    year int(4) NOT NULL,  
    producer varchar(80) NOT NULL,  
    transmission varchar(80) NOT NULL,  
    price varchar(80) NOT NULL,  
    seats int(4) NOT NULL,  
    image varchar(255) NOT NULL,
```

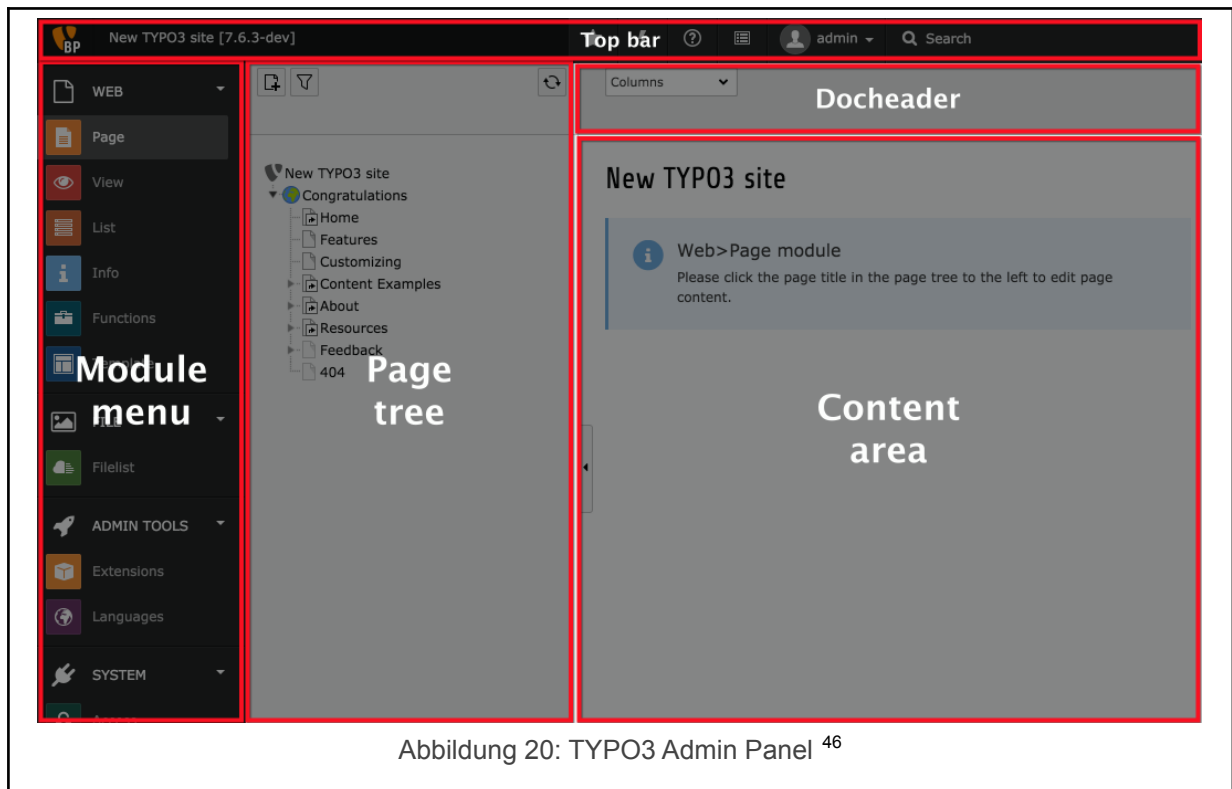
Abbildung 19: Tabellenerstellung in ext_tables.sql

⁴⁵ Vgl. TYPO3 Contributors: TYPO3 Explained / Extension Development / File Structure / ext_tables.sql, in: typo3.org.

3.2 Backend Aufbau

Das folgende Bild ist ein Beispiel für die Backend-Oberfläche, die gemäß der TYPO3-Dokumentation in 5 Bereiche aufgeteilt ist:

Top Bar - Module - Page Tree - Docheader - Content Bereich



3.2.1 Module und Page Tree

Mit TYPO3 kann man mehr als eine Website pflegen und auch Unterseiten für jede davon einrichten. Dazu ermöglicht TYPO3 als CMS Benutzern, durch die Seitenhierarchie zu navigieren und mithilfe von Backend-Modulen verschiedene Änderungen an ihrer Konfiguration und ihrem Inhalt vorzunehmen. Die Backend-Module befinden sich im rechten Bereich unter dem Dashboard und sind in mehrere Sektionen aufgeteilt. Die folgende Auflistung erläutert meistens die relevantesten Module aus eigenen Feststellungen, die in der Anwendung von Relevanz sein könnten.

➤ Web

- Page - Ermöglicht die Inhaltselemente für die selektierte Seite zu verwalten.
- View - Zeigt den auf der selektierten Seite gerenderten Inhalt an, bevor sie veröffentlicht wird.

⁴⁶ TYPO3 Contributors: Getting Started / General Principle / General Backend Structure, in: typo3.org.

- List - Gibt einen Einblick in die TCA-Aufzeichnungen. Im Pagetree können auch Ordner erstellt werden, die verschiedene TCA-Datensätze speichern, sodass Editoren wichtige Datenbankeinträge verwalten können.
- Form - Enthält eine Liste aller gespeicherten Formulare, die auf bestimmten Seiten für verschiedene Zwecke verwendet werden.
- Info - Zeigt die Administratoraktivität auf der ausgewählten Seite.
- Template - Hier können TypoScript Templates für die ausgewählte Seite erstellt oder verwaltet werden. Extensions müssen eingefügt werden, um ihre TypoScript-Konfiguration zu importieren und ihre darin definierten Layouts und Inhalte zu verwenden. Nach dem Einbinden des Templates hat man dann einen Einblick in Konstanten, die in den eingebundenen Extensions zur Verfügung stehen, und kann diesen unterschiedliche Werte zuweisen, ohne auf die dafür zuständige TypoScript-Datei zugreifen zu müssen.
- **Site Management**
 - Sites - Bietet eine Auswahl über den Website-Titel, die Haupt-URL zum Aufrufen des Website-Frontends im Browser, die Sprache usw.
- **File**
 - Filelist - Ermöglicht es, verschiedene Mediendateien hochzuladen und stellt Speicherpfade zur Verfügung, um weiter in verschiedenen Inhalten verwendet zu werden.
- **Admin Tools**
 - Maintenance - Analysiert Tabellenänderungen aus *ext_tables.sql* und nimmt sie in die Datenbank mit. Hier kann man auch einen neuen Backend-Administrator einrichten.
 - Extension - Gibt eine Liste aller installierten Extensions und ermöglicht es, sie zu aktivieren oder zu deaktivieren. Dies liegt daran, dass es Extensions gibt, deren Inhalte nur zu bestimmten Zeiten gerendert werden sollen und/oder deren zweckloses Aktivhalten nur die Ladezeit der Anwendung verlängert.
- **System**
 - Access - Der Hauptadministrator kann die Berechtigungen innerhalb des Backends für verschiedene Gruppen verwalten. Z. B. verschiedenen Benutzern erlauben, Seiten zu bearbeiten, zu erstellen oder zu löschen usw.
 - Backend Users - Speichert Informationen für Backend Benutzer wie E-Mail, Benutzername und Passwort.
 - Configuration - Gibt einen Einblick in jede einzelne Konfiguration in Form eines Arrays. Hier können TCA-Einträge, Seiten, Benutzer, Extensions, Plugins und mehr in ihrer jeweiligen Hierarchie überprüft werden.

3.2.2 Content Area

Der Inhaltsbereich ist die Ansicht des Page Moduls. Wenn ein TypeScript Template für die selektierte Seite vorhanden ist, wird in diesem Bereich auch die Layoutstruktur angezeigt. Mit dem Klick auf den + *Content* Button auf einer Layoutsektion erscheint ein Fenster (Abbildung 21), in dem man zwischen Inhaltselementen wie Headern, Texten, Bildern und definierten Plugins auswählen kann.

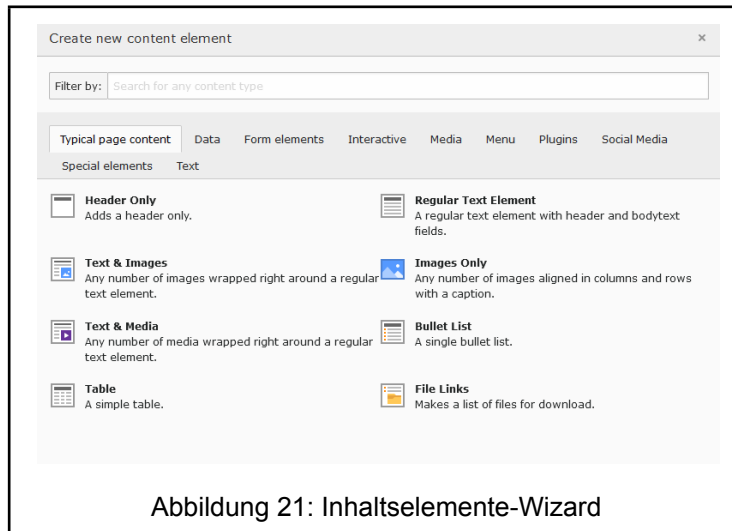


Abbildung 21: Inhaltselemente-Wizard

Die anfänglichen Inhaltselemente sind begrenzt, daher ist das Erstellen neuer Plugins an entsprechenden Extensions sehr wichtig, um eine multifunktionale Website zu haben. Nachdem ein Inhaltselement ausgewählt ist, kann es weiter auf der folgenden Registerkarte angepasst werden:

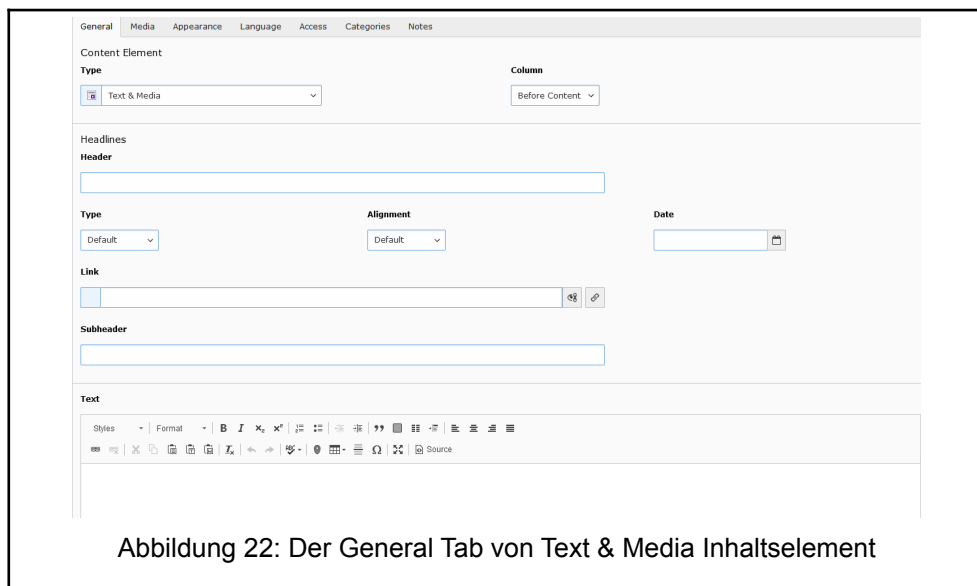


Abbildung 22: Der General Tab von Text & Media Inhaltselement

Das obige Fenster stellt die Anpassungsmöglichkeiten für ein Text & Media Inhaltselement dar, das eine Kopfzeile, einen Text und eine Datei enthält. Der letzte Abschnitt unter dem

General Tab ist eine Funktion des Rich-Text-Editors, mit der Texte ohne Verwendung von HTML und CSS angepasst werden können und eine Vorabanzeige des zu veröffentlichenden Frontend-Textes angezeigt wird.⁴⁷ Die nächste Media Tab ermöglicht es, eine Mediendatei hochzuladen, die ein Bild oder ein Video sein kann, und auch ihre Ausrichtung mit dem Text zu wählen. Im Appearance kann man den Rahmen oder den Hintergrund dieses Inhalts definieren. Das Veröffentlichungsdatum und das Ablaufdatum, bzw. wann der Inhalt auf der Website verfügbar sein wird, werden in dem Access Tab festgestellt. Dabei können diese Inhalte auch bestimmten Nutzergruppen zugänglich oder für jeden Webseitenbesucher sichtbar gemacht werden.

Nach dem Speichern eines Inhaltselements kann man mit dem Klick auf das Bearbeiten-Symbol im Doheader der ausgewählten Seite die Seiteneigenschaften anpassen. Hier stehen verschiedene Optionen zur Verfügung, unter anderem ob die Seite eine Stammseite sein soll, ihre Layouts, Veröffentlichungszeiten, Benutzerzugriff usw.

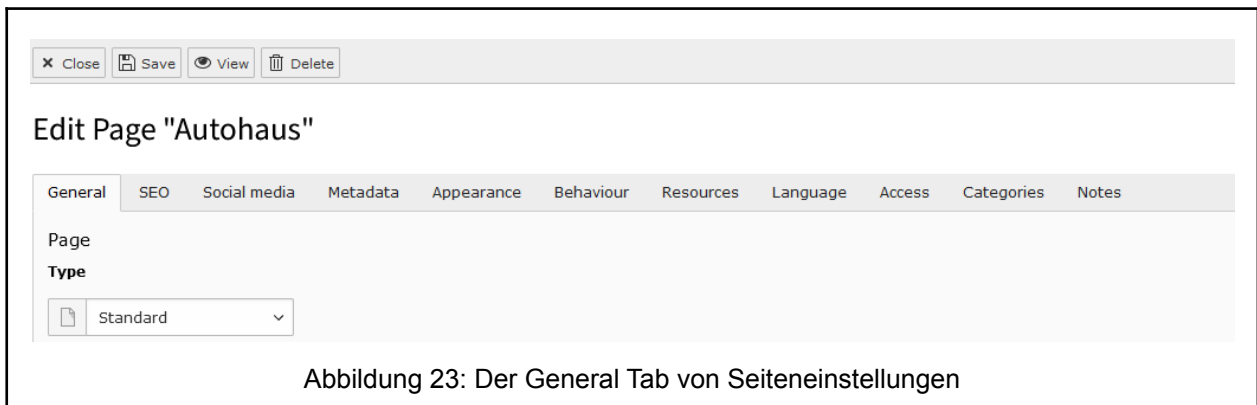


Abbildung 23: Der General Tab von Seiteneinstellungen

Für Themen, die nicht in diesem Abschnitt enthalten sind, ist die TYPO3 Backend Referenz von TYPO3 Kontributoren weiterzuempfehlen.⁴⁸

4 Projekt Realisierung

4.1 Setup

Um eine TYPO3-Instanz zu erstellen, wird ein Linux-Server mit vorinstalliertem MySQL in seinem Repository verwendet. Je nachdem, welcher Entwicklungspfad betrachtet wird, gibt es unterschiedliche Ansätze, um TYPO3 in dieser Umgebung einzusetzen. In der konkreten Situation stützt sich die TYPO3-Instanz auf Bibliotheken aus der TYPO3 Basis Distribution

⁴⁷ Vgl. TYPO3 Contributors: TYPO3 Tutorial for Editors / Content Elements / The rich text editor, in: typo3.org.

⁴⁸ Vgl. TYPO3 Contributors: Getting Started, in: typo3.org.

sowie TER-Bibliotheken, und darüber hinaus wird eine eigene Haupterweiterung aufgebaut, um davon einige der bereitgestellten Funktionen entsprechend den Anforderungen anzupassen oder zu erweitern. Daher wird die Verwendung von Composer zur Installation von TYPO3 und seinen Abhängigkeiten mehr empfohlen.⁴⁹ Composer bietet die Möglichkeit, alle Abhängigkeiten zu definieren und sie einfacher zu aktualisieren. Da der Composer alle Anfangskonfigurationen zwischen den Systemerweiterungen und der TYPO3-Umgebung für die anstehende Entwicklung vorbereitet, spart seine Verwendung während der Instanziierung auch Zeit.

Um mit der Installation zu beginnen, muss zuerst für das Projekt ein neues Verzeichnis auf dem Webserver erstellt werden. Um Apache-Server für Dateien zu signalisieren, die für die Domain bereitgestellt werden müssen, wird das Verzeichnis *htdocs* gebraucht. Damit Logs bei der Verarbeitung von Server-Anfragen übermittelt werden können, muss ein zweiter Ordner namens *logs* erstellt werden. Als Nächstes beginnt die Installation von Composer mit den folgenden Befehle im Terminal: *sudo apt-get update ; sudo apt-get install composer*. Danach ist der Server bereit, die TYPO3 Base Distribution über Composer herunterzuladen: *composer create-project typo3/cms-base-distribution:^10 www.typo3-project/htdocs*. Hier wird Version 10 installiert, weil es eine größere Vielfalt an TER Extensions gibt, die mit dieser Version kompatibel sind. Viele TER Extensions sind für die letzte TYPO3 version veraltet und es dauert normalerweise einige Zeit, bis sie von ihren Entwicklern aktualisiert werden. Mit erfolgreichem Download wird die folgende Ordnerstruktur innerhalb des TYPO3-Projekts angezeigt:

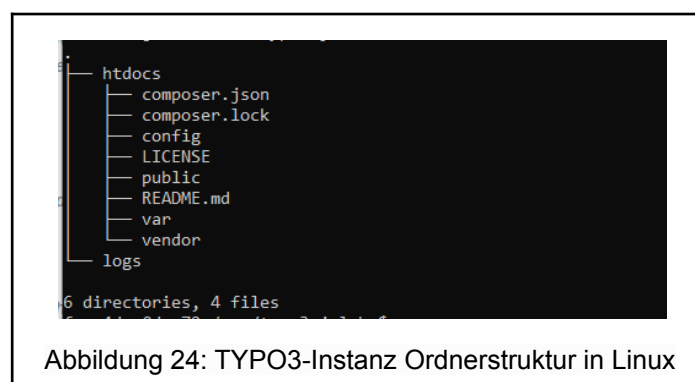


Abbildung 24: TYPO3-Instanz Ordnerstruktur in Linux

Die *composer.json* beinhaltet ein mehrdimensionales Array, das anfänglich eine Aufzeichnung aller Abhängigkeiten oder Systemerweiterungen und ihrer jeweiligen Versionen enthält, die in der Folder Struktur unter *public/typo3/sysext* zu finden sind. Jede neue, über Composer installierte Extension wird hier automatisch registriert. Ausnahme sind benutzerdefinierte Extensions, die manuell ergänzt werden müssen. *composer.lock* zeigt

⁴⁹ Vgl. TYPO3 Contributors: Installation and Upgrade Guide / Installing TYPO3 / Installing TYPO3 via composer, in: typo3.org.

den Verlauf aller installierten Abhängigkeiten in einer tieferen Informationsskala wie ihre eigenen Abhängigkeiten, ihre Code-Quellen-Links usw.

Der *config* Ordner enthält die Site-Konfigurationsdateien (*.yaml*) mit Informationen wie Basis-URL, Website-Titel und Sprache. Diese Konfigurationen können auch im Sites Modul über Backend geändert werden. Der *var* Ordner enthält temporäre Informationen wie Caches und Protokolle. Der *vendor* hält alle Bibliotheken, auf denen die TYPO3 Systemerweiterungen aufgebaut sind, wie z. B. Doctrine oder andere definierte Framework-Komponenten wie Symfony usw.

Zunächst muss für das CMS im MySQL-Server eine Datenbank angelegt werden, und es ist zu beachten, dass der DB Benutzer die Berechtigungen dazu bekommt. Folgende Befehle können dazu angegeben werden:

```
CREATE DATABASE autohaus-db;
GRANT ALL PRIVILEGES ON 'autohaus-user' TO 'autohaus-db'@'localhost' IDENTIFIED BY
'*****';
FLUSH PRIVILEGES;
```

Abbildung 25: Erstellen einer neuen Datenbank und neuen Benutzer in MySQL

Wenn der Server Benutzer nicht der Serveradministrator ist, muss dennoch sichergestellt werden, dass er die Berechtigung erhält, das Projekt zu bearbeiten und auszuführen. *chmod 775 -R www/typo3-project/htdocs* ist ein nützlicher Befehl. Jede der drei Benutzergruppen auf einem Server hat unterschiedliche Berechtigungen, einschließlich der Kompetenz, eine Anwendung zu lesen, zu schreiben und auszuführen.

Jede der im Befehl dargestellten Ziffern steht für eine Benutzergruppe. Die erste Ziffer steht für den „Eigentümer“, der die volle Berechtigung hat, dann kommt die „Gruppe“, zu der die normalen Nutzer gehören, der auch die volle Berechtigung benötigt, und die letzte Gruppe sind „andere Benutzer“. Zahl 5 steht für „only read and execute“. Hier nehmen andere Benutzer teil, die die Website nur im Browser aufrufen müssen, sowie Einblick in den Code haben, aber nicht über das Recht verfügen, ihn zu bearbeiten. -R steht für rekursiv, wobei alle im angegebenen Verzeichnis enthaltenen Dateien die Berechtigungsregeln bekommen.⁵⁰

Nach dem Erstellen einer leeren Datei mit dem Befehl *touch FIRST INSTALL* im *public* Verzeichnis, kann die Installation des CMS mit Hilfe des Browsers anfangen. Mit Beginn des Installationsvorgangs wird die Datei entfernt. Ausschließlich, der Installationsdateipfad kann

⁵⁰ Vgl. Computer Hope: Linux chmod command, in: Computer Hope, 2021.

im Browser aufgerufen werden: <https://typo3-project-myserver/typo3/public/install.php>. Bei den im Browser angezeigten Schritten, müssen die MySQL-Anmeldeinformationen für die angelegte Datenbank angegeben und ein neuer Backend-Benutzer eingerichtet werden. Inzwischen werden alle relevanten Tabellen für CMS-Kernfunktionalität von System Extensions bereitgestellt. Hier zählen beispielsweise Backend/Frontend Nutzer, Funktionalitäten für die Erstellung der initialen Inhaltselemente usw. Sie müssen zunächst in die DB eingetragen werden. Dafür soll eine Datenbankanalyse unter dem Maintenance Modul im Backend ausgeführt werden, um weiter die Tabellen in der Datenbank zu übernehmen.

4.2 Entwicklung

4.2.1 Erweiterung von Kernfunktionen

Wenn die TYPO3 Instanz bereit ist, kann die Entwicklungsphase beginnen. Die lokalen sowie alle anderen über Composer installierten TER Extensions werden im Verzeichnis `public/typo3conf/ext/` gepflegt. Das erste Paket in der Erweiterungskette, das mitgenommen wird, um Layouts, Navigationsleisten, Fußzeilen und verschiedene standardisierte Inhaltselemente für die Website zu erstellen, ist das Bootstrap-Package von Benjamin Kott.⁵¹ Beim Wechseln zum Stammpfad der Anwendung, wo sich die Root `composer.json` befindet, können die folgende Zeile ins Terminal eingegeben werden, um die letzte Version des Pakets zu installieren, welche mit der zuvor installierten TYPO3 Version kompatibel ist: `composer req bk2k/bootstrap-package 12`. Befehl `composer-dump autoload` würde zunächst dafür sorgen, nach erfolgreicher Installation die `composer.json` zu aktualisieren. Das Paket enthält durch TypoScript-Konfigurationen viele gestaltete Markup-Vorlagen. Um sie weiter verwenden zu können, müssen im Backend Templates für jede Seite erstellt und diese Konfigurationen eingeschlossen werden. Dies kann man mit folgenden Klicks erreichen:

Backend Modul Template -> select Info/Modify (in Docheader) -> Create new template -> Edit whole template record -> Includes Tab -> Bootstrap Package: Full Package (in Available Items) auswählen. TYPO3 gibt normalerweise der Setup-Konfiguration Vorrang, die innerhalb der CMS-Oberfläche vorgenommen wird, die hier unter dem General Tab zu finden ist. Diese anfängliche Einrichtung ist einfach ein auf der Webseite ausgegebenes Hallo Welt. Um die Konfiguration Quellen aus der neu installierten `bootstrap_package` Extension

⁵¹ Vgl. TYPO3: Bootstrap Package, in: extensions.typo3.org.

aufzunehmen, müssen dieses Standard-Setup gelöscht werden. Mit dem Klick auf die Schaltfläche „Save“ im Docheader, kann man das Template-Setup speichern.

Nach der Installation der Anfangspakete, kann die Entwicklung der Alpha-Extension (*autohaus*), mit der das Bootstrap-Paket gemäß den Anforderungen konfiguriert wird, beginnen. Dafür muss die in Abbildung 27 gezeigte anfängliche Ordnerstruktur aufgebaut werden, wobei die Namenskonventionen, wie auf Seite 18 erwähnt, einzuhalten sind.

Der erste Schritt ist, die *ext_emconf.php* zu erstellen, deren Funktion in Abschnitt 3.1.2.5 erklärt wurde.

```
<?php
$EM_CONF[$_EXTKEY] = [
    'title' => 'Autohaus',
    'description' => 'The main extension',
    'category' => 'alpha',
    'author' => 'Juljano Aliaj',
    'author_email' => '',
    'author_company' => '',
    'state' => 'stable',
    'createDirs' => '',
    'clearCacheOnLoad' => 0,
    'version' => '1.0.0',
    'constraints' => [
        'depends' => [
            'typo3' => '11.0.0-11.99.99',
            'Bootstrap_package' => '12.5.0'
        ],
        'conflicts' => [
        ],
        'suggests' => [
        ],
    ],
];
```

Abbildung 26: *ext_emconf.php* von Autohaus Extension ⁵²

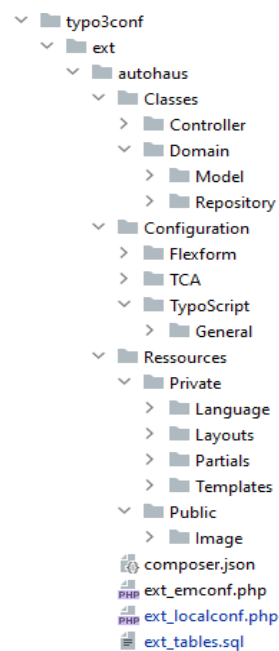


Abbildung 27: Ordnerstruktur für Autohaus Extension ⁵²

Zunächst wird eine *composer.json* benötigt (Abbildung 28). Es weist darauf hin, dass die Autohaus Extension von den TYPO3-Systemerweiterungen und auch vom gerade installierten *bootstrap_package* abhängt. Dies wird gemacht, um TYPO3 zu signalisieren, diese Abhängigkeiten vor der Autohaus-Extension, die darauf aufgebaut wird, zu laden.

⁵² Vgl. Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Creating a First extension / Create Folder Structure and Configuration Files, in: typo3.org

```
{
  "name": "vendorname/autohaus",
  "type": "typo3-cms-extension",
  "description": "An example extension",
  "license": "GPL-2.0-or-later",
  "require": {
    "typo3/cms-core": "^9.5 || ^10.4",
    "bk2k/bootstrap-package": "12.5.0"
  },
  "autoload": {
    "psr-4": {
      "VENDOR\\Autohaus\\": "Classes/"
    }
  },
  "extra": {
    "typo3/cms": {
      "extension-key": "autohaus"
    }
  }
}
```

Abbildung 28: Composer Konfiguration für Autohaus Extension⁵²

Solange die neue Extension nicht über Composer installiert wird, können Extension Klassen von der Anwendung nicht gefunden werden. Um dies zu lösen, müssen die folgenden Zeilen mit dem gemeinsamen Verzeichnis, indem Klassen für diese Extension aufgebaut sind, in den Root-Composer hinzugefügt werden:⁵³

```
"require": {
  "typo3/cms-core": "^9.5"
},
"autoload": {
  "psr-4": {
    "VENDOR\\Autohaus\\": "public/typo3conf/ext/autohaus/Classes/"
  }
},
```

Abbildung 29: Root Composer Konfiguration für das lokale Extension

Die beiden Extensions können jetzt im Extension Manager aktiviert werden. Autohaus TypoScript-Template muss jedoch in das Template-Modul eingeschlossen werden. Die Vorgehensweise ist auf Seite 32 am Beispiel des Bootstrap-Packages definiert.

Um einige Bootstrap Features zu verwenden, kann man zunächst das Homepage Layout unter *Pagesettings(Docheader)->Appearance->Layouts* definieren. Wenn bereits Unterseiten für die Website wie Shop, Kontakt etc. angelegt sind, können hier diese Unterseiten als Menüpunkt hinzugefügt werden, um sie in die Navigationsleiste der Website zu bekommen. Als Nächstes kann die Konfiguration des Bootstrap-Logos und der Fußzeile manuell oder

⁵³ Vgl. Clickstorm: Composer zur Installation/Verwaltung von TYPO3 verwenden, in: clickstorm.de, 05.01.2017.

über das Backend festgelegt werden. Um den Entwicklungsprozess zu beschleunigen, wird der zweite Weg gefolgt. Nach Auswahl der konstanten Kategorie im Template Modul, unter „Constants“ Abschnitt im Docheader, kann man für das Logo einen bestimmten Bildpfad eingeben. Normalerweise sind die Mediendateien im Verzeichnis *public/fileadmin* sowie im Backend unter dem Filelist Modul zu finden. Man kann den Bildpfad nach dem Hochladen kopieren und an die Konstante übergeben. Die Bildabmessungen und einige andere Bildattribute stehen auch als Option zur Verfügung. Die gleiche Vorgehensweise kann auch für die Fußzeile befolgt werden.

4.2.2 Fahrzeugpräsentationen

Nachdem die Grundlagen des TYPO3-Backends und dessen Fähigkeiten für das erste Kriterium, das Publishing Tool (Abschnitt 2.1.1) in dem Backend Aufbau Kapitel behandelt sind, wird nun eine Lösung für die zweite Anforderung, nämlich die Anzeige einer Liste von Autos zusammen mit Bild und Video-Präsentationen in der Alpha-Erweiterung vorgeschlagen. Hier kommt das MVC-Konzept zu Nutzen.

Für das Erstellen eines Auto Modells mit seinen Eigenschaften sowie entsprechenden Getter und Setter ist das Beispiel in Abbildung 5 empfohlen. Zu beachten ist, dass in der folgenden Demonstration keine vollständige Liste der Merkmale enthalten ist, die ein Unternehmen möglicherweise in seinen Daten verwendet, sondern nur einige wichtige Attribute zur Beschreibung der Methode.

Die Klasse muss in einer gleichnamigen PHP-Datei im Ordner Classes/Domain/Model gespeichert werden und muss TYPO3\CMS\Extbase\DomainObject\AbstractEntity Klasse erweitern.⁵⁴ Hier werden alle wesentlichen Merkmale aufgelistet, die ein Auto haben könnte, wie z. B. Marke, Modell, Preis, Farbe usw. Die folgenden Properties entsprechen zusätzlich der Anforderung an die Video- und Medienpräsentation:

```
/**
 * @var \TYPO3\CMS\Extbase\Persistence\ObjectStorage<\TYPO3\CMS\Extbase\Domain\Model\FileReference>
 */
protected $image ;

/**
 * @var \TYPO3\CMS\Extbase\Persistence\ObjectStorage<\TYPO3\CMS\Extbase\Domain\Model\FileReference>
 */
protected $videoPresentation ;
```

Abbildung 30: Definition von Media Properties in der Auto Klasse

⁵⁴ Vgl. Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Creating a First extension / Create The Domain Model, in: typo3.org.

ObjectStorage ist eine TYPO3-CMS-Klasse, die zum Instanzieren von Properties verwendet wird, deren Datentyp eine andere Klasse sein soll⁵⁵. In dieser Situation sind die Properties als Dateireferenzobjekte bezeichnet und dadurch Mediendateien können zugewiesen werden.⁵⁶

Der nächste Schritt besteht darin, eine Tabelle für das eben erstellte Modell in der Datenbank anzulegen, die dieselben Spalten wie die Eigenschaften der Auto-Klasse enthält. Ein Beispiel wurde in der Abbildung 19 gezeigt. Die SQL-Datentypen müssen logisch ausgewählt werden. Zu beachten ist, dass der Dateireferenzdatentyp hier einfach einem *varchar* Datentyp entspricht.

Zunächst zeigt die untere Abbildung eine mögliche Lösung für die TCA-Komponente der bereits angelegten Tabelle, wobei der Redakteur Autoartikel über das Backend pflegen kann:

```
<?php
defined ( 'TYPO3_MODE' ) or die ();
return [
    "ctrl" => [
        "label" => "model",
        "title" => "Autos",
    ],
    "columns" => [
        "model" => [
            "label" => "Model",
            "config" => [
                "type" => "input"
            ]
        ],
        "image" => [
            "label" => "Bild",
            "config" => \TYPO3\CMS\Core\Utility\ExtensionManagementUtility:
                getFileFieldTCAConfig('image')
        ],
        .....
    ],
    'types' => [
        '0' => ['showitem' => 'model, image, ...'],
    ],
];
```

Abbildung 31: Definition des TCA-Arrays für die Autoklasse ^{57 58}

⁵⁵ Vgl. Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Modelling the domain / Implementing the domain model, in: typo3.org.

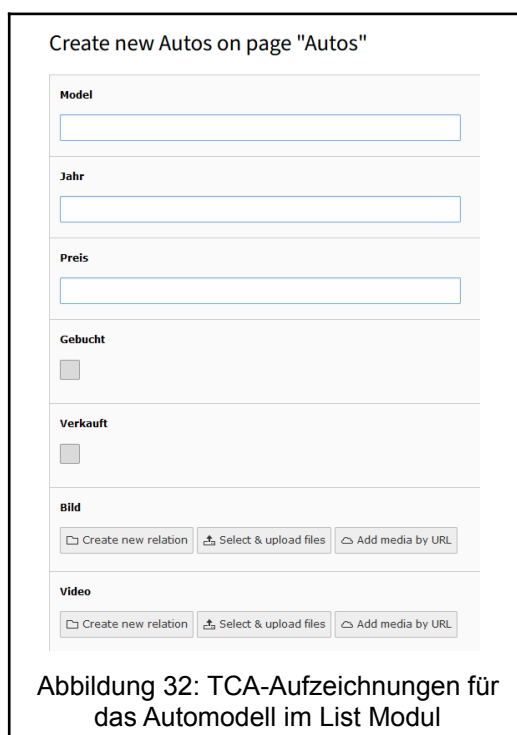
⁵⁶ Vgl. Codingmusikinsnetz: Extbase/Fluid, Add file fields using the FAL(File Abstraction Layer): reference multiple images(yourPictures), in: coding.musikinsnetz.de, o. D..

⁵⁷ Vgl. TYPO3 Contributors: TCA Reference / Field types (config > type) / Input, in: typo3.org

⁵⁸ Vgl. Codingmusikinsnetz, o. D..

Für Mediendateien, die in dem Modell deklariert wurden, erlaubt die *getFileFeildTCAConfig* Methode der ExtensionUtility unter *ctrl[columns][config]*, mehrere Dateien in der Tabelle eines Autos hinzuzufügen. Das letzte *types* Array bestimmt die Reihenfolge der TCA-Felder, die im Backend angezeigt werden.

Zunächst soll im Backend Seitenbaum ein Ordner erstellt werden, der als ein Backend-Speicher für die TCA-Daten dient. Zu seinem List Modul steht jetzt eine neue Record Möglichkeit zur Verfügung mit dem Titel, die in der Abbildung 31 unter *data[ctrl][title]* definiert wurde. Die untere Abbildung zeigt das Fenster für die Einträge:



Create new Autos on page "Autos"

Model

Jahr

Preis

Gebucht

☐

Verkauft

☐

Bild

Video

Abbildung 32: TCA-Aufzeichnungen für das Automodell im List Modul

Nachdem das Autoobjekt persistiert wurde, können seine Daten über die Controller-Schicht dem View zugewiesen werden. Um auf die Daten aus der Datenbank zugreifen zu können, muss zusätzlich eine PHP-Repository-Klasse unter *Classes/Domain/Repository* implementiert und hier injiziert werden. Diese Klasse ist dann eine Ergänzung zur CMS-Klasse Repository, die Methoden zur Verfügung stellt, die zum Erstellen von SQL-Abfragen für das Modell erforderlich sind. Die Klasse bietet auch allgemeine wie die *findAll* Methode an, die eine Liste aller Datensätze entsprechend dem Modell zurückgibt. Die Repository kann erstmal leer bleiben. Das folgende Snippet veranschaulicht eine vergleichbare Konstruierung der Repository Klasse im Controller:

```

/**
 * @var AutoRepository
 */
protected $autoRepository;

/**
 * @param AutoRepository $autoRepository
 */
public function injectAutoRepository(AutoRepository $autoRepository){
    $this->autoRepository = $autoRepository;
}

```

Abbildung 33: Definition und Injizieren der Auto Repository-Klasse in Controller ⁵⁹

Die Controller-Klasse erweitert die CMS-Klasse *ActionController* und wird im *Klassen/Controller* Ordner erstellt. Die Datenübergabe wird in der sogenannten Actions abgestimmt. Die erste Action Methode gibt eine Liste aller in der Datenbank gespeicherten Autos, und die zweite Methode stellt die Felder eines Autodatensatzes, die in dem Auto Modell definiert wurden, zur Verfügung (Abbildung 34).

```

public function listAction(){
    $autos = $this->autoRepository->findAll();
    $this->view->assign('autos', $autos);
}

public function detailAction(Auto $auto){
    $this->view->assign('auto', $auto);
}

```

Abbildung 34: Erstellen einer Action Methode für das Zuweisen von Auto Objekte ins View

Daten werden weiter an *Resources/Private/Template/Auto/List* übergeben. Folgendes Snippet zeigt eine vergleichbare Fluid Einstellung:

```

<f:for each="{autos}" as="auto">
    <p><f:link.action action="detail" arguments="{auto:auto.uid}">{auto.model}</f:link.action></p>
    <p>{auto.productionYear}</p>
    <p>{auto.price}</p>
    <f:for each="{videoPresentation}" each="{auto.videoPresentation}">
        <f:media file="{videoPresentation}" />
    </f:for>
    <f:for each="{image}" each="{auto.image}">
        <f:image image="{image}" />
    </f:for>
</f:for>

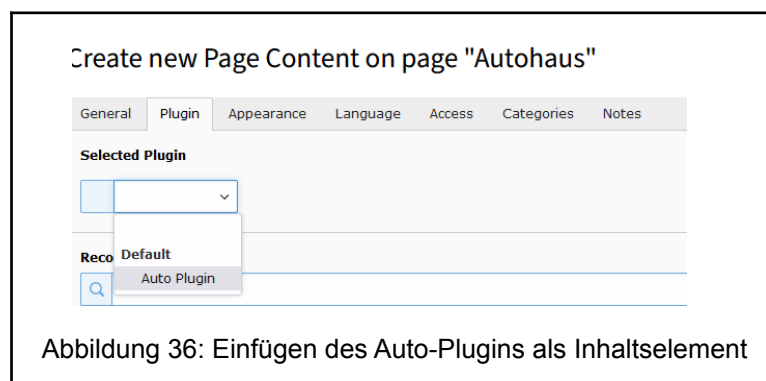
```

Abbildung 35: Markup-Template der listAction

⁵⁹ Vgl. TYPO3 Contributors: TYPO3 Explained / Dependency Injection, in: typo3.org.

Der Viewhelper *f:link.action* leitet den Nutzer mit der *uid* des Autoobjekts auf die in Abbildung 34 gezeigte *detailAction* weiter. Das spezifische Objekt mit dieser *uid* kann dann einem weiteren View zugeordnet werden, wo detaillierte Informationen über das Objekt aufgerufen und angezeigt werden können.

Sobald das MVC vollständig ist, kann die Registrierung des neuen Plugins beginnen, damit es auf Backend als verfügbares Inhaltselement erscheint. Die Vorgehensweise dafür ist auf Abbildungen 16 und 17 angewiesen. Durch Navigieren in das Page Modul im Backend kann ein neues Inhaltselement erstellt und das neue Plugin ausgewählt werden:

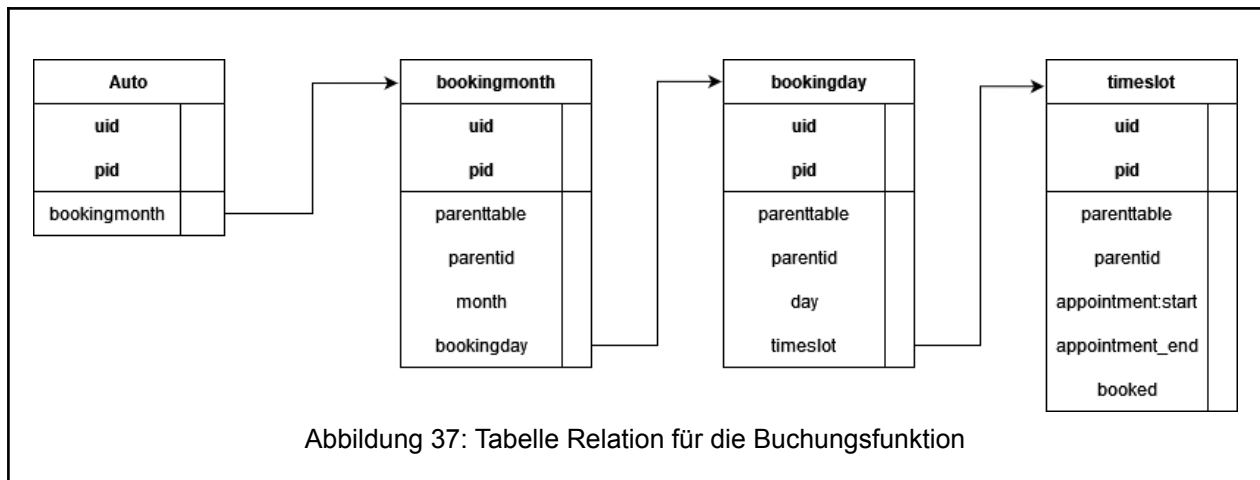


4.2.3 Online Anfragen

Für die dritte Anforderung wird zunächst ein Buchungssystem implementiert.

Als funktionale Anforderung sollte der Codesatz eine Liste von Buchungszeitfenstern für einen bestimmten Autoartikel rendern. Dem Zeitfenster wird ein bestimmter Tag, Monat und Fahrzeugartikel zugewiesen sowie ein klarer Terminbeginn und -ende in Stunden. Jedes Zeitfenster wird ein eindeutiges *uid* und ein Attribut namens *booked* haben, das umgeschaltet wird, wenn ein Website-Besucher eine Reservierung vornimmt. Außerdem muss, sobald der Termin ansteht, jedes noch nicht gebuchte Zeitfenster aus der Datenbank gelöscht werden. Weiterhin kann die Anzahl der Mitarbeiter, die in gleichen Zeitfenstern arbeiten, begrenzt sein, daher muss die Buchung je nach Mitarbeiterzahl auf gleichwertige Zeitfenster eingeschränkt werden.

Die folgende Abbildung zeigt die Relation der Tabellen, die für diese Funktion benötigt sind, wobei hier die Inline-Funktionalität von TCA zum Nutzen kommt:



Jede der neuen Tabellen sollte als Inline-Element der übergeordneten Tabelle behandelt werden. Beispielsweise ist *bookingmonth* ein Inline-Element und als Folge eine Kindtabelle der Tabelle *auto*. Für die zusätzlichen Tabellen braucht man noch die entsprechenden Modelle, Repositories und TCAs. Die bisherige im Kapitel 4.2.2 erläuterte Vorgehensweise dient weiter als Beispiel.

Danach soll dem Modell *Auto* eine neue Property namens *month*, von dem kürzlich eingerichteten *Month* Modell, hinzugefügt werden, das über die *ObjectStorage* zugänglich sein wird:

```
/**
 * @var \TYPO3\CMS\Extbase\Persistence\ObjectStorage<\Vendor\Autohaus\Domain\Model\Month>
 * $month
 */
protected $month;
```

Abbildung 38: Hinzufügen eines TCA Inline-Elements in der Autoklasse

Die neue Property muss weiterhin einem neuen TCA-Feld entsprechen:

```
'month' => [
    'exclude' => 1,
    'label' => 'Buchungsmonat',
    'config' => [
        'type' => 'inline',
        'foreign_table' => 'bookingmonth',
        'foreign_field' => 'parentid',
        'foreign_table_field' => 'parenttable',
        'maxitems' => 10,
        'appearance' => [
            'collapseAll' => 1,
            'expandSingle' => 1,
        ],
    ],
]
```

Abbildung 39: TCA Inline-Element in TCA-Array⁶⁰

⁶⁰ Vgl. Lobacher und Schams, 2015, S. 301

Um zwischen den einzelnen Inline-Elementen unterscheiden zu können, ist es wichtig, für jedes Element eine Elterntabelle und eine Eltern-ID anzugeben. Die anderen TCA-s und Modelle (*month*, *day*, *timeslot*) können ebenfalls mit der gleichen Technik aufgebaut werden.

Die *timeslot* TCA wird zusätzlich zwei weitere Felder für den Beginn und das Ende des Termins haben. Folgender Ausschnitt zeigt eine mögliche Lösung:

```
"ctrl" => [
    "label" => "appointment_start",
    "title" => "Buchung Stunden",
    'enablecolumns' => [
        'endtime' => 'appointment_start',
    ]
],
"columns" => [
    .....
    'appointment_start' => [
        'label' => 'Terminstart',
        'config' => [
            'type' => 'input',
            'renderType' => 'inputDateTime',
            'eval' => 'datetime,int',
            'default' => 0,
            'range' => [
                'upper' => mktime(0, 0, 0, 1, 1, 2038)
            ]
        ],
        'l10n_mode' => 'exclude',
        'l10n_display' => 'defaultAsReadOnly'
    ],
    .....
],
'types' => [
    '0' => ['showitem' => ' ..., appointment_start, ...'],
],
];
```

Abbildung 40: Definition von TCA-Array der Timeslot Klasse⁶¹

Durch *ctrl[enablecolumns][endtime]* wird definiert, dass der Beginn des Termins auch der Zeitpunkt ist, an dem der Datensatz gelöscht wird.

Nach Persistieren der Objekte und deren Eigenschaften bezieht sich der nächste Schritt auf die Controller-Schicht, mit deren Hilfe dem Redakteur ermöglicht wird, Zeitfenster für jedes Autoartikel zu generieren. Sobald die Repositories der entsprechenden Objekte konstruiert sind, wird eine Action innerhalb des neuen Controllers definiert, die ein Objekt namens

⁶¹ Vgl. TYPO3 Contributors: TCA Reference / Best practices / Common fields, in: typo3.org

TimeFromTo als Argument bekommt. Dieses Objekt hat Eigenschaften wie *month*, *dayFrom*, *dayTo*, *appointmentStart* und *appointmentEnd*. Diese Eigenschaften können mithilfe eines Fluid Forms⁶² durch ein definiertes Backend-Modul, dessen Konfiguration weiter unten beschrieben wird, ermittelt und zwischengespeichert werden.

```
$autos = $this->autoRepository->findAll();
foreach ($autos as $key => $value){
    if(!$this->autoRepository->count(
        'tx_autohaus_domain_model_month',
        'month',
        $timeFromTo->getMonth(), $value->getUid()),
    {
        $this->autoRepository->insertBookingMonth(
            $value->getUid(),
            $timeFromTo->getMonth());
    }
}
```

Abbildung 41: Hinzufügen des Buchungsmonats in tx_autohaus_domain_model_auto

Die Variable *\$autos* in der obigen Abbildung erhält eine Liste aller in der Datenbank enthaltenen Autoartikel. Die Foreach-Schleife durchschleift diese Objekte und prüft, ob in der jeweiligen Tabelle bereits ein Buchungsmonat existiert. Wenn nicht, wird der neue Monat hinzugefügt. Die Methoden *count* und *insertBooking* werden im Repository des Auto Objekts mithilfe von *QueryBuilder* definiert.⁶³ Dieselbe Methode kann für *days* und *timeslots* Actions verwendet werden. Weiterhin kann man, wenn der Editor einen Tagesbereich angibt, eine For-Schleife verwenden, um alle Tage innerhalb des Bereichs zu zählen und hinzuzufügen, wenn sie nicht vorhanden sind.

Um den Beginn und das Ende des Termins festzusetzen, kann man zwei Variablen definieren, die die vom Backend angegebenen Strings in Zeitstempel umwandeln, welche für die TCA-Felder *appointment_start* und *appointment_end* erforderlich sind:

```
$timestampFrom = strtotime($value->getDay()."-".$timeFromTo->getMonth()."-2022
".$timeFromTo->getTimeFrom());
$timestampTo = strtotime($value->getDay()."-".$timeFromTo->getMonth()."-2022
".$timeFromTo->getTimeTo());
```

Abbildung 42: String in Timestamp umwandeln

In speziellen Fällen, wenn ein Termin nicht mehr zur Verfügung stehen muss, kann der Redakteur im List Modul auf dem Backend die entsprechenden Zeitfenster für bestimmte Auto-Records deaktivieren. Dafür kann auf der timeslot Tabelle eine *tinyint* Spalte, die

⁶² Vgl. TYPO3 Contributors: Fluid Viewhelper Reference / typo3/fluid / form, in: typo3.org.

⁶³ Vgl. TYPO3 Contributors: TYPO3 Explained / Database (Doctrine DBAL) / Querybuilder, in: typo3.org.

einem vom Typ *check* TCA-Feld, sowie einer booleanen Property für die Timeslot Klasse entspricht, erstellt werden.

Für die Registrierung dieses Plugins ist die Konfiguration eines neuen zuvor erwähnten Backend-Moduls, mit Hilfe des Beispiels aus Kapitel 3.1.2.5 unter *ext_tables.php* zu befolgen. Um diesem Modul Fluid-Templates zuzuweisen, müssen auf *setup.typoscript* die für das View benötigten Template-Pfade übergeben werden:

```
module.tx_autohaus {
    view {
        templateRootPaths = EXT:autohaus/Ressources/Private/Templates/
        partialRootPaths = EXT:autohaus/Resources/Private/Partials/
        layoutRootPaths = EXT:autohaus/Resources/Private/Layouts/
    }
}
```

Abbildung 43: Definieren von Markup-Templates für das Backend-Modul in TypoScript ⁶⁴

Dies zeigt an, dass jedes in der Alpha-Extension definierte Modul die obigen Pfade für sein View verwendet.

Nachdem alle Backend-Anforderungen erfüllt sind, bezieht sich der nächste Schritt auf das View des *detailAction* im Auto-Plugin, das auf der Abbildung 34 dargestellt wurde. Hier kann man die Liste aller generierten Buchungszeitfenster getrennt nach den jeweiligen Tagen und Monaten rendern. Da die Autotabelle die übergeordnete aller anderen für das Buchungssystem verantwortlichen Tabellen ist, kann man damit die letzte Timeslot-Tabelle mit Hilfe von Viewhelper *f.for* erreichen. In der folgenden Abbildung wird eine Möglichkeit zur Anzeige von Zeitfenstern veranschaulicht.

```
f:if condition="!{timeslot.booked}">
    <f:then>
        <f:link.action action="form" arguments="{auto:auto, timeslot: timeslot}" >
            <div class="timeslot">
                <f:format.date format="H:i">{timeslot.appointmentStart}</f:format.date>
                -<f:format.date format="H:i">{timeslot.appointmentEnd}</f:format.date>
            </div>
        </f:link.action>
    </f:then>
    <f:else>
        <div class="timeslot-booked">
            <f:format.date format="H:i">{timeslot.appointmentStart}</f:format.date>
            -<f:format.date format="H:i">{timeslot.appointmentEnd}</f:format.date>
        </div>
    </f:else>
</f:if>
```

Abbildung 44: Markup für die Anzeige von Zeitfenstern

⁶⁴ Vgl. Kronovanet: Create a backend module, in: kronovanet, 2018.

Hier werden zwei Optionen auf dem Frontend gerendert. Die erste Option ist, wenn das Zeitfenster nicht gebucht ist bzw. wenn die *booked* Spalte der *timeslot* Tabelle 0 ist, wird eine *f.link.action* eingebunden, andernfalls ist das Zeitfenster nicht für die Buchung aktiv. Der Viewhelper *format.date* wandelt den Zeitstempel des Zeitfensters in Stunden und Minuten um.

Für die Modellierung des Templates sind die CSS Dateien unter Resources/Private/CSS zuständig, welche in dem Page Objekt in TypoScript zu deklarieren sind.⁶⁵ Die Ausgabe nach der Modellierung könnte wie folgt aussehen:



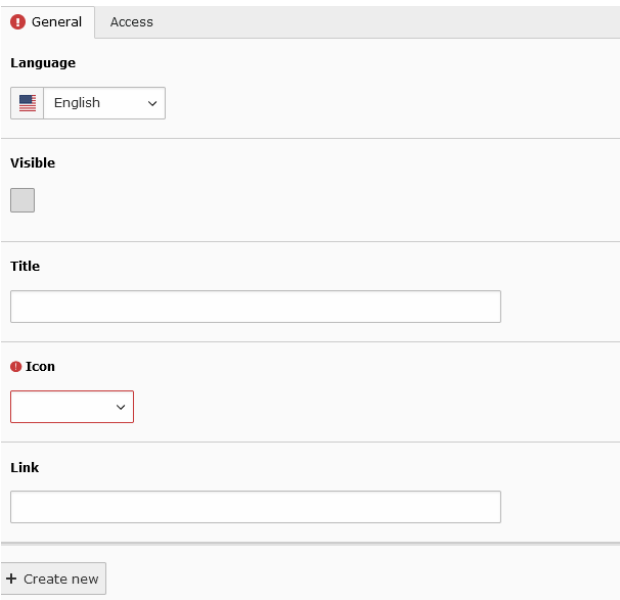
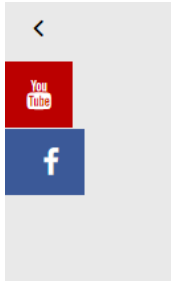
Die *link.action* des aktiven Zeitfensters leitet den Benutzer zu einer Action namens *form* (Abbildung 44), wo der Website-Besucher den Termin buchen und seine Buchungsdaten wie E-Mail, Telefonnummer usw. hinterlegen kann. Dazu kann ein neues Modell mit entsprechender Tabelle eingerichtet werden, die alle Buchungsinformationen bekommen wird. Mithilfe der bisher dargelegten Vorgehensweisen, um die Buchungsinformationen aller Interessenten dem Backend-Administrator anzuzeigen, können außerdem die Elemente der neuen Buchungsinformationen Tabelle in ein neues Backend Modul aufgelistet werden. Nachdem die Buchung abgeschlossen ist, kann mithilfe des *formAction* in *AutoController* das gebuchte Feld nach *timeslot uid* auf 1 umgeschaltet werden, um das Zeitfenster für andere Besucher zu deaktivieren. Hier kann man auch eine if-Anweisung verwenden, um alle anderen gleichen Zeitfenster zu buchen, wenn die Anzahl der Mitarbeiter, die in

⁶⁵ GRID Design: CSS Dateien einbinden, in: typo-script.de, o. D..

denselben Zeiten arbeiten, erreicht ist. Die Mitarbeiterzahl kann als TypoScript-Konstante definiert und vom Administrator über das Backend gesetzt werden.

4.2.4 Social Media Widgets

Das von Arun Chandran entwickelte Social Media Widget⁶⁶ wurde ausgewählt, um die vierte Anforderung zu erfüllen. Das Plugin bietet eine Auswahl an Social-Media-Widgets und kann über das Backend in den Seiteneinstellungen angepasst werden. Da der Autor keine Composer-Installation anbietet, wird dieses Paket manuell importiert. Die mit der bereits installierten TYPO3-Core kompatible Version steht auf TER als Zip-Ordner zur Verfügung. Da die TYPO3-Installation auf Composer basiert, muss jede andere manuell importierte Extension im Root-Composer hinzugefügt werden. Dazu dient die Einstellung aus der Abbildung 29 als Beispiel. Nach dem Aktualisieren des Root-Composers kann man sehen, dass die Extension in EM aktiviert ist. Weiterhin muss auf jeder Seite, auf der die Widget-Liste erscheinen soll, das statische TypoScript im Template-Modul eingeschlossen sein. Die Extension enthält auch eine Datenbanktabelle, die die URL und die Art des damit verbundenen Widgets registriert. Um die Tabelle anzuwenden, muss eine Datenbankanalyse in dem Wartungsmodul ausgeführt werden, um die ausgewählten Datenbankänderungen zu bekommen.

 <p>Abbildung 46: General Tab von Social Media Widget Plugin</p>	 <p>Abbildung 47: Anzeige von Social Media Widget im Frontend</p>
---	--

⁶⁶ Vgl. TYPO3: Social Media Widget, in: extensions.typo3.org.

Nach erfolgreicher Konfiguration erscheint im Backend auf den Seiteneinstellungen ein neuer Tab namens Social Media. Hier werden Social-Media-Widgets und entsprechende Links gepflegt (Abbildung 46). Die Widget-Liste wird dann auf der linken Seite der Website angezeigt (Abbildung 47).

4.2.5 Feedback und FAQ

Um diese Funktionalität zu erreichen, wird die von Jacco van der Post entwickelte Extension jpFAQ⁶⁷ ausgewählt. Nach dem Testen dieses Pakets wurde festgestellt, dass es einen TCA-Record zum Einfügen häufig gestellter Fragen und entsprechender Antworten durch den Editor in das List Modul bereitstellt. Kundenfragen können unterschiedlichen Kategorien sowie unterschiedlichen Kommentaren zugeordnet werden. Die Aufzeichnungen lassen sich dann mit Hilfe eines vordefinierten Plugins auf der Website ausgeben. Das Plugin kann per Konfiguration auch einige für das Feedback Eingabefelder anzeigen, damit die Webseitenbesucher verschiedene Fragen stellen, ihre Feedbacks sowie ihre Kontaktdaten, um später kontaktiert zu werden, hinterlassen können. Eine Schnittstelle zu diesen Rückmeldungen ist in der Extension für den Redakteur nicht vorgesehen, ihre Informationen werden jedoch in der Datenbank gespeichert. Um dem Redakteur die Rückmeldungen über eines Backend Moduls statisch anzuzeigen, muss ein neuer MVC-Kanal aufgebaut werden. Somit kann der Redakteur die relevanten Fragen, die für die Anzeige auf der Website geeignet wären, über das List Modul eintragen.

Nach den Anweisungen im Handbuch der Extension⁶⁸ kann man das Paket mit der folgenden Zeile im Terminal installieren: `composer req jvdv/jpfaq`, und das statische TypoScript für die Unterseite, wobei das Plugin verwendet wird, einschließen. Um mit dem Erstellen von Kategorien, Fragen und Antworten zu beginnen, muss ein Ordner im Seitenbaum eingerichtet und das jpFAQ Plugin eingebundet werden.

Mithilfe eines neuen Modells und einer Repository in der Alpha-Extension entsprechend der neuen Feedback-Tabelle können die gegebenen Feedbacks und Fragen persistiert werden. Die Zuordnung der Fremdtabelle für dieses Modell geschieht über PHP:

⁶⁷ Vgl. TYPO3: jpFAQ, in: extensions.typo3.org.

⁶⁸ Vgl. Jacco van der Post: Frequently Asked Questions, in: typo3.org.

```

return [
    \VENDOR\Autohaus\Domain\Model\Feeback::class => [
        'tableName' => 'tx_jpfaq_domain_model_categorycomment',
        'properties' => [
            'name' => [
                'fieldName' => 'name'
            ],
            'category' => [
                'fieldName' => 'category'
            ],
            'email' => [
                'fieldName' => 'email'
            ],
            'comment' => [
                'fieldName' => 'comment'
            ],
        ],
    ],
];

```

Abbildung 48: Zuordnung von Fremdtabelle in der Autohaus-Extension⁶⁹

Die Konfiguration erfolgt unter Configuration/Extbase/Persistence/Classes.php. Die Datei wird ein Array mit der Datenbanktabelle Zuordnung und den jeweiligen Feldern für die Klasse zurückgeben. Anschließend kann das Repository auf die Tabelle zugreifen. Man kann als Nächstes ein neues Backend-Modul nach dem Beispiel in Abbildung 18 und der in Abschnitt 4.2.3 gezeigten Methode bauen. Der *f:foreach* Viewhelper und HTML-Tabellen-Tags können verwendet werden, um alle Feedback-Informationen statisch anzuzeigen.

Für weitere Konfigurationen in der jpFAQ-Extension wird das Erweiterungshandbuch empfohlen.

⁶⁹ Vgl. Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Setting up the persistence layer / using foreign data sources, in: typo3.org.

4.2.6 TYPO3 Blog Extension

Für die sechste Anforderung ist die TYPO3 Blog Extension⁷⁰ eine geeignete Lösung. Der Blog wird von der TYPO3 GmbH angeboten und die letzte Version (11.0.2) ist kompatibel mit dem zuvor installierten TYPO3-Core. Die Extension Manual⁷¹ bietet ein ausführliches Setup für Composer Mode.

Laut Extension Manual kann in Page Tree eine Blog Seite mit weiteren Unterseiten wie Autoren, Tags, Kategorien usw. erstellt werden. Jede solche Aufteilung listet dann Blogbeiträge nach den Kategorien auf, indem das entsprechende Plugin auf jeder dieser Seiten als Inhaltselement eingefügt wird. Blogbeiträge können als eigenständige Seiten in einem Ordner erstellt werden, der als Datenspeicher für alle Blogs dient. Man kann jedem Blogbeitrag einen bestimmten Autor, Kategorie oder andere Aufteilung zuweisen, die den Blog auf den jeweiligen Unterseiten verfügbar macht. Dafür müssen diese Kategorisierungen im Ordner als TCA-Records über das List Modul eingetragen werden. Die Blog-Extension enthält verschiedene Layouts und verschiedene Inhaltselemente wie Texte und Medien. Hier sind die Funktionen von WYSIWYG, Bildbearbeitung usw. bereitgestellt. Damit Besucher Kommentare hinterlassen können, kann auf jeder Seite eines Blogeintrags ein in der Extension vorkonfiguriertes Plugin als Inhaltselement eingebunden werden.

Für weitere Informationen zur Extension wird die Verwendung des TYPO3 Blogs Extension Manuals⁷¹ empfohlen.

5 Fazit

5.1 Zusammenfassung

Diese Arbeit konzentrierte sich auf die Entwicklung einiger digitaler Kommunikationsstrategien für die Website eines Fahrzeughändlers. Die Kenntnis der Hauptaufgaben des Unternehmens und der Art seiner Kundenbeziehungen hat es ermöglicht, einige bemerkenswerte Konzepte für digitale Kundeninteraktion zu identifizieren und ihre Vorteile zu erläutern. Mit dem TYPO3 CMS wurden diese Kommunikationsprinzipien in nützliche Website Funktionen umgesetzt. Dafür wurden zuerst die wesentlichen Konzepte zum Verständnis von TYPO3 CMS betrachtet. Nach dem Aufbau einer Instanz im Composer-Modus ist das CMS an die definierten Anforderungen mit Hilfe

⁷⁰ Vgl. TYPO3: TYPO3 Blog Extension, in: extensions.typo3.org.

⁷¹ Vgl. TYPO3 GmbH Team: TYPO3 Blog Extension, in: typo3.org.

einer grundlegenden benutzerdefinierten Extension sowie einiger Bibliotheken aus dem TYPO3-Extension-Repository, angepasst worden. Nach der Konfiguration aller Extensions wurde durch Navigieren in dem Backend die Verwaltung von verschiedenen Inhaltselementen ermöglicht.

5.2 Zukünftige Arbeit

Das primäre Ziel war es, eine grundlegende Instanz zu erstellen und Verfahren zu definieren, um die Backendanforderungen zur Umsetzung der Kommunikationsprinzipien zu erfüllen. Durch eine Basisfunktionalität können sowohl das Backend als auch das Frontend der Anwendung weiterentwickelt und verbessert werden. Dadurch ist es möglich, Ajax-Aufrufe auf verschiedenen Templates zu implementieren, indem kleine Seitenobjekte in TypoScript erstellt und die erforderliche Javascript-Einstellung hinzugefügt werden, die es Benutzern ermöglicht, verschiedene Vorlagen durch Link-actions oder Fluid Formulare aufzurufen, ohne auf eine neue URL weitergeleitet zu werden. Bei der Terminbuchung zum Beispiel wäre dies eine geeignete Lösung. Das Seitenobjekt kann dann auch mit entsprechendem Javascript als Popup-Formular aufgerufen werden.

Das Buchungssystem kann weiterentwickelt werden. Ein Vorschlag für die zukünftige Entwicklung wäre die Definition einer Methode für das Unterteilen der vom Editor über Backend angegebenen Zeiträume in kleinere Zeitfenster.

Weiterhin, soweit grundlegende Klassen und Repositories definiert sind, können neue Datenbankfelder und ihre zugehörigen TCAs je nach Datenstruktur und Verfügbarkeit miteinbezogen werden.

TYPO3 bietet eine große Auswahl an Erweiterungen, daher könnte eine weitere potenzielle zukünftige Arbeit die Integration zusätzlicher Pakete wie Online-Zahlungen umfassen, in der Kunden Artikel online bestellen und geeignete Zahlungsmethoden auswähle; oder eine Facettensuche und Filter, bei denen Website-Besucher die Produktverfügbarkeit durchsuchen können.

6 Quellenverzeichnis

Carr, David/ Gray, Marcus: Beginning PHP: Master the latest features of PHP7 and fully embrace modern PHP development, Birmingham-Mumbai: Packt Publishing, Limited 2018, S. 1.

Clickstorm: Composer zur Installation/Verwaltung von TYPO3 verwenden, in: clickstrom.de, 05.01.2017, <https://www.clickstorm.de/blog/composer-installation-verwaltung-typo3/>, (abgerufen am 31.07.2021).

Codingmusikinsnetz: Extbase/Fluid, Add file fields using the FAL(File Abstraction Layer): reference multiple images (yourPictures), in: coding.musikinsnetz.de, o. D., <https://coding.musikinsnetz.de/typo3-extbase-fluid/general/add-file-fields-multiple/>, (abgerufen am 30.08.2021).

Coleman, Brian [Youtube]: Digitalization of the Car Industry:Following the Consumer, 2017, <https://www.youtube.com/watch?v=y0jxeFmg1w&t=217s> (05.06.2022), 17:00-17:25.

Computer Hope: Linux chmod command, in: Computer Hope, 06.2021, <https://www.computerhope.com/unix/uchmod.htm>, (abgerufen am 26.07,2022).

DAT Pressemitteilung, 2021, S. 5, online verfügbar unter: https://www.dat.de/fileadmin/user_upload/Grafiken_zur_Pressemitteilung_DAT_Report_2021.pdf.

Dragon2000: Why car dealers should be combining finance calculators with sales presentation videos, in: Dragon2000, 2020, <https://www.dragon2000.co.uk/why-car-dealers-should-be-combining-finance-calculators-with-sales-presentation-videos/>, (abgerufen am 18.06.2022).

Falkenberg, Karen: TYPO3 CMS 6.2 Grundlagen: für Redakteure und Integratoren, 1.Ausgabe, 1. Aktualisierung, Herdt, 2015, S. 8, 112.

GRID Design: CSS Dateien einbinden, in: typo-script.de, o. D, <https://www.typo-script.de/blog/typo3/css-dateien-einbinden/>,(abgerufen am 20.08.2022).

Jay Kang: Social Media Links & Your Website, in: SEOptimizer, 2021, <https://www.seoptimizer.com/blog/social-media-links/>, (abgerufen am 12.06.2022).

Jacco van der Post: Frequently Asked Questions, in: typo3.org, rendered on 02.2022, <https://docs.typo3.org/p/jvdp/jpfaq/11.5/en-us/Index.html>, (abgerufen am 07.08.2022).

Kronovanet: Create a backend module, in: kronovanet, 2018, <https://kronova.net/tutorials/typo3/extbase-fluid/create-a-backend-module.html>, (abgerufen am 01.08.2022).

Kurfürst, Sebastian/Rau, Jochen/ Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Creating a First extension / Configuring the Plugin, in: typo3.org, rendered on 05.2022,
<https://docs.typo3.org/m/typo3/book-extbasefluid/10.4/en-us/4-FirstExtension/7-configuring-the-plugin.html>, (abgerufen am 02.07.2022).

Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Creating a First extension / Create Folder Structure and Configuration Files, in: typo3.org, rendered on 05.2022,
<https://docs.typo3.org/m/typo3/book-extbasefluid/10.4/en-us/4-FirstExtension/7-configuring-the-plugin.html>, (abgerufen am 02.07.2022).

Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Creating a First extension / Create The Domain Model, in: typo3.org, rendered on 05.2022,
<https://docs.typo3.org/m/typo3/book-extbasefluid/10.4/en-us/4-FirstExtension/3-create-the-domain-model.html>, (abgerufen am 02.07.2022).

Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Model-view-controller (MVC) in Extbase, in: typo3.org, rendered on 05.2022,
<https://docs.typo3.org/m/typo3/book-extbasefluid/main/en-us/2-BasicPrinciples/1-Model-View-Controller-in-Extbase.html>, (abgerufen am 02.07.2022).

Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Modelling the domain / Implementing the domain model, in: typo3.org, rendered on 08.2022,
<https://docs.typo3.org/m/typo3/book-extbasefluid/main/en-us/5-Domain/2-implementing-the-domain-model.html>, (abgerufen am 04.08.2022).

Kurfürst, Sebastian/ Rau, Jochen/ & Contributors: Developing TYPO3 Extensions with Extbase and Fluid / Setting up the persistence layer / Using foreign data sources, in: typo3.org, rendered on 08.2022,
<https://docs.typo3.org/m/typo3/book-extbasefluid/main/en-us/6-Persistence/4-use-for-eign-data-sources.html>, (abgerufen am 17.08.2022).

Lammenett, Erwin:TYPO3 Online-Marketing-Guide: Affiliate- und E-Mail-Marketing, Keyword-Advertising,Suchmaschinen-Optimierung mit TYPO3, Wiesbaden, Deutschland:Gabler, 2007 S.13-14.

LDB Gruppe: So geht Kundenkommunikation 4.0 im Autohaus, in: LDB Gruppe, 2017,
<https://www.ldb.de/so-geht-kundenkommunikation-im-autohaus/>, (abgerufen am 11.06.2022).

LDB Gruppe: Vielen Dank für Ihren Anruf. Das Telefon im Autohaus , in: LDB Gruppe, 2020,
<https://www.ldb.de/vielen-dank-fuer-ihren-anruf-das-telefon-im-autohaus/>, (abgerufen am 05.06.2022).

Lobacher, Patrick/ Schams, Michael:TYPO3 Extbase, Modern Extension Development for TYPO3 CMS with Extbase & Fluid: Open Source Press GmbH, 2015, S. 34, 37-42, 301, 302.

McAdams, D./ Stone, R./ Wood, K. Functional Interdependence and Product Similarity Based on Customer Needs . *Res Eng Des* 11, 1–19 (1999), S. 2, <https://doi.org/10.1007/s001630050001>.

Mehta, Chintan/ Bhavsar, Antik/ Oza, Hetal/ Shah, Subhash:MySQL 8 Administrator's Guide:Effective Guide to Administering High-Performance MySQL 8 Solutions, Birmingham-Mumbai: Packt Publishing, Limited, 2018, S. 6.

SoftGuide: Content-Management-System, in: SoftGuide, o.D., <https://www.softguide.de/software-tipps/grundlagen-funktionen-von-cms>, (abgerufen am 21.06.2022).

Tyler Smith: How Your Current Traffic Impacts Your SEO, in: SmartBug.,2019 , <https://www.smartbugmedia.com/blog/current-traffic-impacts-seo>, (abgerufen am 14.06.2022).

TYPO3: Bootstrap Package, in: extensions.typo3.org, https://extensions.typo3.org/extension/bootstrap_package, (abgerufen am 28.07,2022).

TYPO3: jpFAQ, in: extensions.typo3.org, <https://extensions.typo3.org/extension/jpfaq>, (abgerufen am 07.08.2022).

TYPO3: Social Media Widget, in: extensions.typo3.org, https://extensions.typo3.org/extension/spt_socialmedia, (abgerufen am 28.07,2022).

TYPO3: TYPO3 Blog Extension, in: extensions.typo3.org, <https://extensions.typo3.org/extension/blog>, (abgerufen am 07.08.2022).

TYPO3 Contributors: Fluid Viewhelper Reference / typo3/fluid / form, in: [typo3.org](https://docs.typo3.org), rendered on 06.2022, <https://docs.typo3.org/other/typo3/view-helper-reference/9.5/en-us/typo3/fluid/latest/Form.html>, (abgerufen am 24.07.2022).

TYPO3 Contributors: Getting Started / General Principle / General Backend Structure, in: [typo3.org](https://docs.typo3.org), rendered on 07.2022, <https://docs.typo3.org/m/typo3/tutorial-getting-started/10.4/en-us/GeneralPrinciples/GeneralBackendStructure/Index.html>, (abgerufen am 24.07.2022).

TYPO3 Contributors: Installation and Upgrade Guide / Installing TYPO3 / Installing TYPO3 via composer, in: [typo3.org](https://docs.typo3.org), rendered on 03.2022, <https://docs.typo3.org/m/typo3/guide-installation/10.4/en-us/Index.html>, (abgerufen am 02.07.2022).

TYPO3 Contributors: TCA Reference / Introduction, in: typo3.org, rendered on 08.2022
<https://docs.typo3.org/m/typo3/reference-tca/main/en-us/Introduction/Index.html>,
(abgerufen am 16.08.2022).

TYPO3 Contributors: TCA Reference / Best practices / Common fields, in: typo3.org,
rendered on 08.2022
<https://docs.typo3.org/m/typo3/reference-tca/main/en-us/BestPractises/CommonFields.html#enablecolumns>, (abgerufen am 16.08.2022).

TYPO3 Contributors: TCA Reference / ['columns']['*']['config'], in: typo3.org, rendered on
08.2022,
<https://docs.typo3.org/m/typo3/reference-tca/10.4/en-us/ColumnsConfig/Index.html>,
(abgerufen am 16.08.2022).

TYPO3 Contributors: TCA Reference / Field types (config > type) / Input, in: typo3.org,
rendered on 08.2022,
<https://docs.typo3.org/m/typo3/reference-tca/main/en-us/ColumnsConfig/Type/Input/Index.html>, (abgerufen am 16.08.2022).

TYPO3 Contributors: TYPO3 Explained / Database (Doctrine DBAL) / Introduction, in:
typo3.org, rendered on 06.2022,
<https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/ApiOverview/Database/Introduction/Index.html>, (abgerufen am 10.07.2022).

TYPO3 Contributors: TYPO3 Explained / Dependency Injection, in: typo3.org, rendered on
08.2022 ,
<https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/ApiOverview/DependencyInjection/Index.html>, (abgerufen am 15.08.2022).

TYPO3 Contributors: TYPO3 Explained / Extension Development / Best practices and
conventions / Naming conventions, in: typo3.org, rendered on 08.2022,
<https://docs.typo3.org/m/typo3/reference-coreapi/10.4/en-us/ExtensionArchitecture/NamingConventions/Index.html>, (abgerufen am 15.08.2022).

TYPO3 Contributors: TYPO3 Explained / Extension Development / File Structure,
ext_emconf.php, in: typo3.org, rendered on 08.2022,
<https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/ExtensionArchitecture/FileStructure/ExtEmconf.html>, (abgerufen am 15.08.2022).

TYPO3 Contributors: TYPO3 Explained / Extension Development / File Structure /
ext_tables.php, in: typo3.org, rendered on 08.2022,
<https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/ExtensionArchitecture/FileStructure/ExtTablesSql.html>, (abgerufen am 15.08.2022).

TYPO3 Contributors: TYPO3 Explained / Extension Development / File Structure /
ext_tables.sql, in: typo3.org, rendered on 08.2022,

<https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/ExtensionArchitecture/FileStructure/ExtTables.html>, (abgerufen am 15.08.2022).

TYPO3 Contributors: TYPO3 Explained / Fluid, in: typo3.org, rendered on 07.2022, <https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/Index.html>, (abgerufen am 24.07.2022).

TYPO3 Contributors: TYPO3 Explained / Main classes and methods / High priority functions, in: typo3.org, rendered on 04.2022, <https://docs.typo3.org/m/typo3/reference-coreapi/7.6/en-us/ApiOverview/MainClasses/HighPriorityFunctions/Index.html>, (abgerufen am 15.08.2022).

TYPO3 Contributors: TYPO3 Explained / Namespaces, in: typo3.org, rendered on 08.2022, <https://docs.typo3.org/m/typo3/reference-coreapi/10.4/en-us/ApiOverview/Namespaces/Index.html>, (abgerufen am 15.08.2022).

TYPO3 Contributors: TYPO3 Tutorial for Editors / Content Elements / The rich text editor, in: typo3.org, rendered on 06.2022, <https://docs.typo3.org/m/typo3/tutorial-editors/10.4/en-us/ContentElements/RichTextEditor/Index.html>, (abgerufen am 23.07.2022).

TYPO3 Contributors: TypoScript in 45 Minutes / TypoScript - A quick overview / Why TypoScript, in: typo3.org, rendered on 04.2022, <https://docs.typo3.org/m/typo3/reference-typoscript/main/en-us/Introduction/Index.html>, (abgerufen am 22.07.2022).

TYPO3 Contributors: TypoScript in 45 Minutes / TypoScript - A quick overview / First steps, in: typo3.org, rendered on 04.2022, <https://docs.typo3.org/m/typo3/tutorial-typoscript-in-45-minutes/main/en-us/TypoScriptOverview/FirstSteps/Index.html>, (abgerufen am 24.07.2022).

TYPO3 Contributors: TypoScript in 45 Minutes / TypoScript - A quick overview / TypoScript ist just an array, in: typo3.org, rendered on 04.2022, <https://docs.typo3.org/m/typo3/tutorial-typoscript-in-45-minutes/main/en-us/TypoScriptOverview/TypoScriptAnArray/Index.html>, (abgerufen am 24.07.2022).

TYPO3 Contributors: TypoScript Reference, in: typo3.org, rendered on 07.2022, <https://docs.typo3.org/m/typo3/reference-typoscript/main/en-us/>, (abgerufen am 24.07.2022).

TYPO3 Contributors: TypoScript Reference / Using and setting TypoScript / Constants, in: typo3.org, rendered on 07.2022, <https://docs.typo3.org/m/typo3/reference-typoscript/main/en-us/Index.html>, (abgerufen am 24.07.2022).

TYPO3 GmbH Team: TYPO3 Blog Extension, in: typo3.org, rendered on 11.2021, <https://docs.typo3.org/p/t3g/blog/main/en-us/Index.html>, (abgerufen am 09.08.2022).

Valade, Janet: PHP and MySQL Für Dummies, US: John Wiley & Sons, Incorporated, 2017, S. , 34, 38, 40.

W3schools: MySQL Tutorial / MySQL RDBMS, in: W3schools, o.D.,
https://www.w3schools.com/mysql/mysql_rdbms.asp, (abgerufen am 20.06.2022).

7 Anhang

Parallel zur Vorbereitung dieser Arbeit wurde eine webbasierte Anwendung entwickelt, deren Quellcode sich auf der beiliegenden CD befindet.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe verfasst, und keine anderen als die angegebenen Quellen verwendet sowie die aus fremden Quellen direkt oder indirekt übernommenen Stellen/Gedanken als solche kenntlich gemacht habe. Diese Arbeit wurde noch keiner anderen Prüfungskommission in dieser oder einer ähnlichen Form vorgelegt. Sie wurde bisher auch nicht veröffentlicht. Hiermit stimme ich zu, dass die vorliegende Arbeit von der Prüferin/ dem Prüfer in elektronischer Form mit entsprechender Software auf Plagiate überprüft wird.

Wildau 25.08.2022

.....

Juljano Aliaj