Technische Hochschule
Brandenburg
University of
Applied Sciences

Einfluss der MACH-Architektur auf die Entwicklung traditioneller CMS und individuelle Betrachtung der Produktstrategie am Beispiel TYPO3

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science
des Fachbereichs Informatik und Medien der
Technischen Hochschule Brandenburg

vorgelegt von:
Stephan Kost

Betreuer: Prof. Dr. Michael Syrjakow
Zweitgutachter: M.Sc. Sebastian Kreideweiß

Berlin, den 01. Mai 2024

Kurzfassung

Die vorliegende Arbeit analysiert die traditionellen CMS WordPress, Drupal, TYPO3 und Joomla! anhand der von der MACH Alliance, als Merkmale moderner Webanwendungen vorgeschlagenen MACH-Kriterien und untersucht am Beispiel von TYPO3 welche Potenziale und Risiken die Kriterien noch für die zukünftige Entwicklung des CMS bereithalten.

Schlüsselwörter

CMS, Microservices, API-First, Cloud-native, SaaS, Headless, Decoupled, TYPO3, WordPress, Drupal, Joomla, MACH

Abstract

This paper analyzes the traditional CMS WordPress, Drupal, TYPO3 and Joomla! using the MACH criteria proposed by the MACH Alliance as characteristics of modern web applications and uses the example of TYPO3 to examine what potentials and risks the criteria still hold for the future development of the CMS.

Keywords

CMS, Microservices, API-First, Cloud-native, SaaS, Headless, Decoupled, TYPO3, WordPress, Drupal, Joomla, MACH

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Der weltweite Umsatz im Markt für öffentliche Cloud-Dienste wächst kontinuierlich. Die Anbieter investieren weiter in die Attraktivität der Infrastruktur, um in Zukunft noch mehr Kunden zu gewinnen. Den größten Teil ihres Gewinns (45%) konnten die Anbieter 2022 mit Software-As-A-Service Lösungen erwirtschaften und auch für die kommenden Jahre sei laut IDC-Research ein stabiles zweistelliges Wachstum in diesem Bereich zu erwarten. (Shirer, 2023). Auch im CMS-Markt entstehen neue SaaS-Lösungen, um von diesem Wachstum zu profitieren. Seit 2020 haben sich einige von Ihnen der MACH Alliance angeschlossen, um mit der MACH-Architektur einen Standard für moderne Webanwendungen zu definieren, der traditionellen Systemen überlegen sein soll. Die Alliance betreibt offensives Marketing gegenüber traditionellen Systemen, die als „legacy" und „monolithisch" im negativen Sinne charakterisiert werden (MACH Alliance, 2023b).

Traditionelle Open Source CMS, wie Drupal, TYPO3, WordPress oder Joomla! haben weltweit seit Anfang der 2000er Jahre eine große Verbreitung und Nutzerschaft erlangt (BuiltWith, o.J.) und es trotz bahnbrechender Innovationen seit ihrer Entstehung, wie der Etablierung von Smartphones und dem Internet of Things (siehe Abbildung 1) immer wieder geschafft sich am Markt zu behaupten. Zusammen stellen diese vier Systeme rund die Hälfte aller Webseiten im Internet bereit (TYPO3 Association, 2024b). Das Interesse am Erhalt und an der Weiterentwicklung der Systeme ist dementsprechend groß.

Diese Arbeit soll, die von der MACH Alliance beworbenen Vorteile der MACH-Architektur kritisch hinterfragen und gegebenenfalls relativieren. Andererseits soll am Beispiel von TYPO3 untersucht werden welche Chancen und Risiken sich für die traditionellen CMS aus den MACH-Kriterien noch für deren zukünftige Produktstrategie ergeben könnten.

1.2 Forschungsfragen

1) Inwiefern erfüllen traditionelle CMS bereits die MACH-Kriterien?

2) Wo liegen die Stärken und Schwächen traditioneller CMS im Vergleich zu MACH-basierten CMS?

3) Welche Chancen und Risiken bestehen für TYPO3 durch die Adaption weiterer Kriterien der MACH-Architektur?

4) Welche MACH-Kriterien sind Teil der Produktstrategie von TYPO3?

1.3 Zielstellung

Die Arbeit soll interessierten Entwicklern, Integratoren und Entscheidern die Anwendungsfälle der beiden Architekturansätze von Content-Management-Systemen aufzeigen und damit Investitions- und Entwurfsentscheidungen erleichtern. Die Befürworter der MACH-Architektur betreiben viel Marketing, um die positiven Aspekte ihrer Herangehensweise hervorzuheben, wodurch sich leicht ein verzerrtes Bild ergeben kann, welches System für welchen Anwendungsfall eingesetzt werden kann.

Durch die Analyse von Drupal, TYPO3, WordPress und Joomla! soll erkennbar werden, inwieweit die traditionellen CMS bereits die MACH-Kriterien berücksichtigen und für Projekte mit diesen Anforderungen eingesetzt werden können.

Die individuelle Betrachtung der Produktstrategie von TYPO3 soll allen, die mit TYPO3 professionell arbeiten beziehungsweise arbeiten wollen einen Einblick geben, in welche Richtung sich das System in Zukunft entwickeln wird (Kapitel 4). Die MACH-Kriterien sollen hierbei als Kriterienkatalog dienen, um Entwicklungspotenziale und -risiken abzuleiten. Im besten Fall können Initiativen in der Community angestoßen werden, um das System den gefundenen Anforderungen folgend weiterzuentwickeln und das Open Source Projekt zu stärken.

1.4 Aufbau

Nach der Einleitung werden in Kapitel zwei die grundlegenden Begrifflichkeiten dieser Arbeit definiert, wobei auch die grundlegende Architektur traditioneller CMS und MACH-basierter CMS gegenübergestellt wird. Weiterhin wird dem Ursprung der MACH-Kriterien nachgegangen, um in Kapitel 3 und 4 auf ein stabiles Grundverständnis der zahlreichen Begrifflichkeiten zurückgreifen zu können, die im Zusammenhang mit der MACH-Architektur Verwendung finden. In Kapitel 3 finden die MACH-Kriterien in der Analyse der vier traditionellen CMS TYPO3, WordPress, Drupal und Joomla! Anwendung. Im Vergleich zeigt sich, welches der Systeme sich bisher am meisten den MACH-Kriterien angenähert hat. Im vierten Kapitel wird die Produktstrategie von TYPO3 hinsichtlich der MACH-Kriterien untersucht. Sechs Experten aus der TYPO3-Community mit unterschiedlicher Spezialisierung geben dabei aktuelle Einblicke zur Entwicklung des CMS. Im letzten Kapitel folgt ein Fazit und ein Ausblick.

2 Grundlagen

2.1 Nicht-funktionale Anforderungen

Nicht-funktionale Anforderungen lassen sich am besten definieren, indem man vom Gegenteil ausgeht: „Nicht-funktionale Anforderungen sind alle Anforderungen, die nicht funktional sind" (Günther & Pflüger, 2014, S. 12). Sie beschreiben, wie eine Software funktionieren, und nicht was sie können, soll. Die nicht-funktionalen Anforderungen wirken sich zumeist auf die gesamte Softwarearchitektur aus. Nicht-funktionale Anforderungen beeinflussen einander gegenseitig. Ein höheres Maß an Wartbarkeit geht beispielsweise mit einer verringerten Funktionalität aber einer erhöhten Zuverlässigkeit einher (Balzert, 2011).

2.2 Softwarearchitektur

2.2.1 Definition

Nach Helmut Balzert beschreibt die Softwarearchitektur „die Strukturen eines Softwaresystems durch Architekturbausteine und ihre Beziehungen und Interaktionen untereinander sowie ihre physikalische Verteilung. Die extern sichtbaren Eigenschaften eines Architekturbausteins werden durch Schnittstellen spezifiziert" (Balzert, 2011, S. 23). Hansruedi Tremp beschreibt die Aufgabe der Softwarearchitektur in der Festlegung „grobgranularer Strukturen" und ergänzt, dass sich die Softwarearchitektur dabei auf das Anwenden von Prinzipien und Mustern stützt. Der Entwurf des Softwarearchitekten basiert in der Regel auf einer Referenzarchitektur (Tremp, 2021, S. 2). Im Softwarelebenszyklus ist die Festlegung der Softwarearchitektur die zentralen Aufgabe in der Entwurfsphase eines Softwareproduktes. Sie schließt direkt an die initiale Spezifikationsphase an. Die Architektur wird im Wesentlichen durch die, in der vorangegangenen Phase festgelegten Anforderungen beeinflusst und entscheidet über deren Implementierung. Sie beeinflusst alle nachfolgenden Phasen des Softwarelebens-zyklus (Balzert, 2011).

2.2.2 Ziele

Die Softwarearchitektur dient dazu die Anforderungen an Termine, Kosten, Funktionalität und Qualität im Rahmen eines Softwareprojektes unter den gesetzten Zielen und Randbedingungen in optimaler Weise in Einklang zu bringen (Tremp, 2021).

2.3 Content Management System (CMS)

2.3.1 Eingrenzung des Begriffs

Es gibt unterschiedliche Ausprägungen von CMS, die sich nur bedingt unter einer gemeinsamen Definition zusammenfassen lassen. Zumeist werden CMS anhand ihrer Funktionalität definiert. Allen CMS ist gemein, dass sie der Sammlung, Verwaltung und Bereitstellung von Inhalten dienen (Strekalova & Bouakkaz, 2022). In dieser Arbeit werden unter dem Begriff „CMS" Web Content Management Systeme (Web-CMS) verstanden, die Werkzeuge und Methoden zur Verwaltung von Inhalten bereitstellen, auf deren Basis Webseiten und andere Webanwendungen basieren (Spörrer, 2009).

2.3.2 Architektur früher CMS

Zu Anfang der 2000er Jahre entstanden zwei unterschiedliche Web-CMS-Architekturen. MovableType etablierte CMS vom Typ Static-Site-Generator. Deren Webseiten sind bei Abruf bereits statisch auf dem Webserver abgelegt und werden nur beim Ändern oder Erstellen von Inhalt neu generiert. Dieser Typ von CMS hat durch den Verzicht auf eine Datenbankabfrage den Vorteil, dass Webseiten schnell geladen werden können. Inhalte können getrennt vom Layout (Templates) in separaten Dateien gespeichert werden, wodurch sich leicht neue Seiten vom gleichen Erscheinungsbild erzeugen lassen (Neumegen, 2021).

Die Mehrzahl der CMS nutzte Anfang der 2000er jedoch relationale Datenbanken, um Templates (Vorlagen) serverseitig dynamisch mit Inhalten zu füllen und HTML-Dokumente zu generieren. Als systemunabhängiges Austauschformat für Inhalte etablierte sich anfangs XML. Einige Systeme setzten auch auf die Speicherung der Daten im XML-Format, um Inhalte mittels XSL-Stylesheets für

verschiedene Kanäle aufbereiten zu können, beispielsweise für die Ausgabe als PDF oder auf den damals verfügbaren WAP-Endgeräten (Jablonski & Meiler, 2002).

Die Entscheidung für den Einsatz relationaler Datenbanken fiel zum Zeitpunkt der Entstehung von dynamischen CMS, weil sie leistungsfähiger mit großen Datenmengen umgehen konnten als die damaligen objektorientierten Datenbanken. Im Gegenzug nahm man die Leistungsverluste in Kauf, die durch das Transformieren von XML in das relationale Schema beim Speichern, sowie bei der Rücktransformation in XML beim Abruf entstanden (Jablonski & Meiler, 2002).

2.4 Traditionelle CMS

2.4.1 Eingrenzung des Begriffs

Der Begriff „traditionelle Content-Management-Systeme" fasst im Rahmen dieser Arbeit die vier Web-CMS WordPress, Drupal, Joomla! und TYPO3 zusammen. Diese CMS stellen aktuell etwa 50% aller Webseiten im Internet bereit. Die vier CMS vereint, dass sie auf PHP basieren und schon seit vielen Jahren kostenfrei und Open Source am Markt verfügbar sind (TYPO3 Association, 2024b).

Das Adjektiv „traditionell" wurde gewählt, da es im Deutschen synonym die Bedeutungen „anerkannt", „erprobt", „gewohnt", „klassisch" und „verbreitet" in sich vereint (Dudenredaktion, o. J. c).

2.4.2 Entstehung und Verbreitung

Spörrer beschreibt, dass es zu Anfang der 2000er Jahre nach einer starken Wachstumsphase im Geschäft mit Content-Management-Systemen zu einer Konsolidierungsphase kam. Die Investitionsbereitschaft der Unternehmen war gering (Spörrer, 2009). Die durchschnittlichen Lizenzkosten für ein ausgereiftes CMS lagen laut Kontzer im Jahr 2001 bei etwa $500.000 Dollar (McKeever, 2003, zitiert nach Kontzer, 2001). Open Source CMS konnten unter diesen Startbedingungen zur starken Konkurrenz proprietärer Systeme heranwachsen, da sie von Anfang an mit ihrem professionellen Funktionsumfang eine

hochwertige und kostengünstige Alternative boten (Spörrer, 2009). In Abbildung 1 ist die Veröffentlichung der ersten stabilen Versionen traditioneller CMS eingetragen.

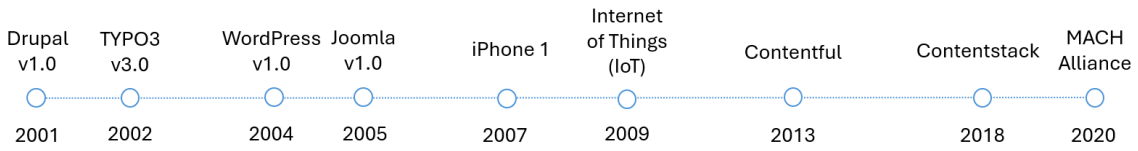


Abbildung 1: Zeitstrahl zur Veröffentlichung traditioneller und MACH-basierter CMS in Anlehnung an (Burgy, 2020)

Vergleicht man die Anzahl der Suchabfragen bei Google Trends weltweit zu den Begriffen „Drupal", „TYPO3", „Joomla!" und „WordPress" mittels Google Trends (siehe Abbildung 2), so ergeben sich zu jedem der Systeme deutliche Maxima. Im März 2007 erreichte zuerst TYPO3 sein Maximum. Das Interesse verringerte sich dann bis zu einer Konsolidierung bei etwa 6% des maximalen Interesses. Das Interesse für Joomla! kulminierte im März 2009 und nähert sich seither asymptotisch der Nulllinie an. Drupal erreichte im Januar 2011 sein Maxima. Das Interesse sank anschließend bis etwa zum Dezember 2020, wo es sich bei rund 12% des maximalen Interesses konsolidierte. WordPress hatte im Juli 2014 mit einem steilen Ausschlag nach oben seine größte Popularität erreicht, zwei Monate nachdem die native REST-API veröffentlicht wurde und stabilisiert sich seither bei etwa 25% des maximalen Wertes. Nur im Zeitraum zwischen November 2005 und Dezember 2009 belegte Joomla! Rang eins vor WordPress.

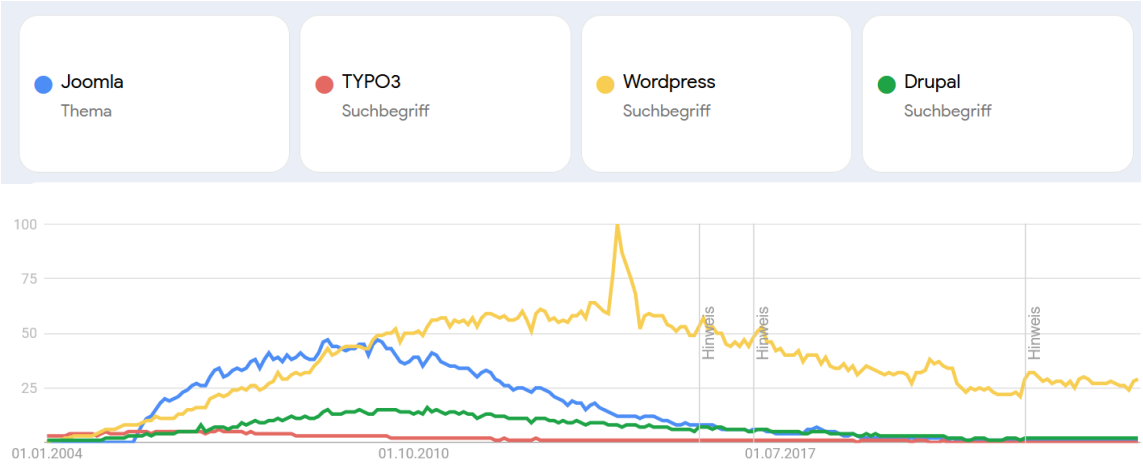


Abbildung 2: Weltweite Google-Suchanfragen normiert auf das Maximum im Zeitraum von 2004 bis heute zu den Begriffen „Joomla!", „TYPO3", „WordPress", „Drupal" (Google, 2024a)

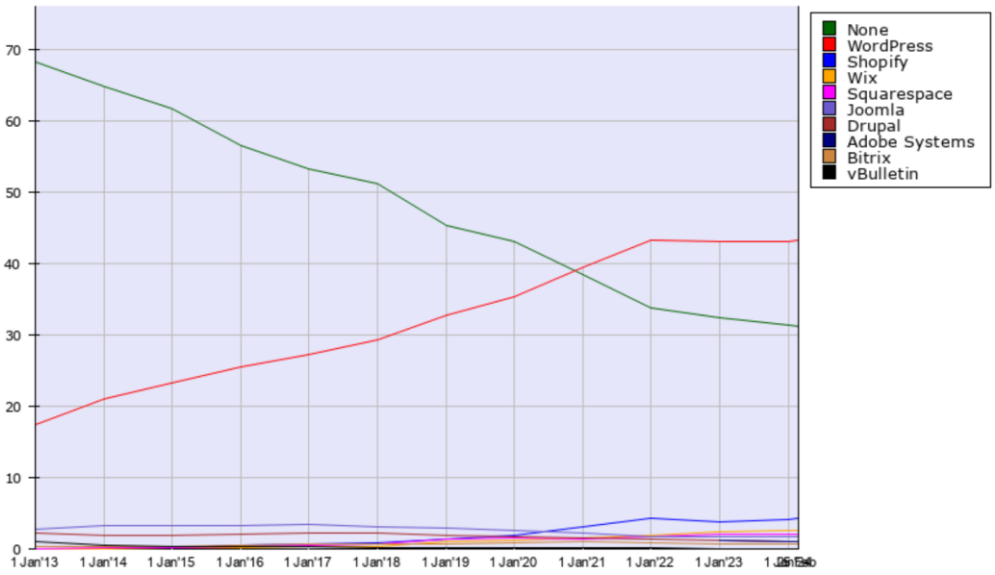


Abbildung 3: Jährliche Nutzung von Content Management Systemen in den Jahren von 2013-heute, Stand: 29.02.2024 (W3techs, o.J.)

Die stabil hohe Anzahl an Suchanfragen zum Begriff „WordPress" spiegelt sich auch in der aktuellen Statistik von w3techs.com wider (siehe Abbildung 3), in der 43,2% der Webseiten weltweit WordPress nutzen. Im historischen Vergleich, in den w3tech.com nur WordPress, Joomla! und Drupal in den Jahren seit 2013 einbezieht, ergibt sich ein Maximum für Joomla! mit 3,4% zum Stichtag, am 1. Januar 2017 und für Drupal mit 2,3% aller Webseiten am 1. Januar 2018. Seither nimmt der relative Anteil an Webseiten, der auf den beiden Systemen basiert stetig, wenn auch zunehmend langsamer ab. Erstaunlich ist wiederum, dass WordPress seinen Anteil bis zum Januar 2022 stetig mit Wachstumsraten um die 3-4% steigern konnte. Seitdem ist eine Konsolidierung des Anteil bei etwa 43,2% eingetreten. Die Anzahl der Webseiten, die ohne ein CMS betrieben wurden, beschrieb im gleichen Zeitraum einen gegenläufigen Trend und nahm seit dem Januar 2022 nur noch um rund 1% jährlich ab.

2.4.3 Architektur

2.4.3.1 Die Anfänge

Wie unter Punkt 2.2.1 bereits notiert, steht der Entwurf der Softwarearchitektur am Anfang einer jeden Anwendung und wirkt sich auf alle weiteren Phasen im Softwarelebenszyklus aus. Die Anfänge von WordPress und TYPO3 sollen hierfür exemplarisch sein, im Positiven, wie im Negativen. Die Systeme wurden zu Beginn

bottom-up statt *top-down* entwickelt. Der Entwickler von WordPress Michel Valdrighis war zunächst Anwender, dem ein effizientes Werkzeug für seine Arbeit als Blogger, fehlte. Er erkannte die Möglichkeiten des, dank dem Free Software Movement, lizenzkostenfrei und quelloffen verfügbaren Software-Stacks bestehend aus Linux, Apache, MySQL und PHP (LAMP) zur Bereitstellung von Webanwendungen und begann sein eigenes Problem parallel zum Selbststudium der Technologien zu lösen. Im Buch „Milestones: The Story of WordPress", welches sich anlässlich des 20-jährigen Jubiläums auf GitHub in der Entstehung befindet, wird beschrieben, dass sich Michels Code durch seine mangelnde Erfahrung in PHP und Softwaretechnik „organisch" entwickelte. Sein Fokus lag auf der Umsetzung neuer benutzerorientierter Features (funktionaler Anforderungen). Die Lösung des logischen Problems erfolgte, aufgrund fehlender Kenntnis theoretischer Konzepte, nach dem stream-of-consciousness-Ansatz (Wordpress, 2024, S. 26). Für Anwender ohne Entwicklererfahrung, die ihr Tool verbessern wollten, war der Code gut verständlich und lud dazu ein das System nach eigenen Bedürfnissen anzupassen, sodass die WordPress-Community schnell wuchs und beständig viele neue Funktionen hinzukamen (Wordpress, 2024).

Auch der Begründer von TYPO3 Kasper Skårhøj, der 1997 Firmenwebseiten gestaltete, erkannte das Potenzial des LAMP-Stacks für die Erleichterung seiner Arbeit und brachte sich PHP parallel zur Entstehung von TYPO3 selbst bei. Ihm genügte dafür die Online-Dokumentation. Er hatte zuvor Erfahrungen im maschinennahen Programmieren (Assembler, 80c81 CPUs) und in Pascal gesammelt. Er folgte beim Programmieren nach eigenen Angaben in einem Interview von 2003 mit Moritz Sauer im Wesentlichen seinem „inneren Kompass", um über das weitere Vorgehen während seiner Arbeit zu entscheiden. Hin und wieder holte er sich Inspiration bei anderen. Er glaubte seine Stärke läge in seiner visuellen Kreativität und hielt es daher nicht für möglich ein CMS zu programmieren, dass länger als einen Monat im Einsatz sein würde (Sauer, 2003).

Benni Mack, der heutige Leiter des Core Development Teams bescheinigt Kasper Skårhøj im TYPO3 Community Podcast „Application" vom 1. März 2021, dass sich viele seiner Konzepte als solide erwiesen haben, noch immer im Kern von TYPO3 wiederzufinden seien und für Stabilität und Konsistenz gesorgt hätten. Zwei Leitprinzipien, die TYPO3 immer noch verfolgt. Es gäbe erst Grund diese

Konzepte zu ändern, wenn man eine bessere Lösung dafür gefunden habe. TYPO3 sei zudem bereits objektorientiert gewesen, als Benni Mack 2005 seine Arbeit im Core Entwicklerteam begann. Es war chaotisch, aber auf gute Art und Weise (Mack, 2021). In besonderem Maße sieht sich Open Source Software auch der Schaffung und Einhaltung offener Standards verpflichtet (Fowler, 2014), auch Skårhøj war früh dieser Meinung, um qualitative und bezahlbare Alternativen zu kommerzieller Software anbieten zu können (Sauer, 2003).

Barker sieht traditionelle CMS seit Anbeginn mit dem „open source syndrome" (Barker, 2016, S. 21) konfrontiert. Open Source Software sei sehr entwickler-zentriert, da sie von Entwicklern für Entwickler geschrieben sei. Auch wenn die Systeme nicht final am Reißbrett entstanden seien, so wirke es sich positiv auf die Architektur der Systeme aus, dass die Entwickler den Anspruch hätten die Software regelmäßig hinsichtlich Eleganz, Effizienz und Wiederverwendbarkeit zu überarbeiten (Barker, 2016).

2.4.3.2 Schichten-Muster

Die Architektur traditioneller CMS-Systeme lässt sich gut am Beispiel von TYPO3 nachvollziehen (siehe Abbildung 4). Die traditionellen CMS folgen dem sogenannten Schichten-Muster (Balzert, 2011). Jede Schicht stellt ein isoliertes Subsystem dar, welches der darüberliegenden Schicht verschiedene Dienste anbietet. Das Muster zielt darauf ab zwischen den Schichten stabile standardisierte Schnittstellen zu verwenden, sodass Elemente der Architektur austauschbar sind und unabhängig voneinander entwickelt werden können. Der Umzug der Software auf eine neue Plattform gleichen Typs ist damit problemlos möglich. Das Muster fördert neben Portabilität, Wiederverwendbarkeit, Weiterentwickelbarkeit, Wartbarkeit, Testbarkeit und ermöglicht die physikalische Verteilbarkeit der Schichten, beispielsweise zwischen Client und Server (Balzert, 2011).

Grundvoraussetzung für den Betrieb traditioneller CMS ist eine Plattform bestehend aus einem Webserver mit PHP und einer Datenbank. Diese Dienste können orchestriert und skaliert werden. In der Anwendungsschicht befindet sich die Geschäftslogik. Der CMS-Kern stellt alle für das Content Management relevanten Funktionen über eine plattformeigene Programmierschnittstelle (PHP-API) bereit. Die Funktionalität lässt sich durch diese Schnittstelle modular

erweitern. Zudem verfügen die CMS über eine mitgelieferte Backend-Benutzeroberfläche für Content Management und Administration. Charakteristisch bei traditionellen CMS ist das plattformeigene Frontend-Framework zur Auslieferung der Inhalte, welches ebenfalls über plattformeigene Schnittstellen mit dem Backend kommuniziert (TYPO3 Association, o.J. b).

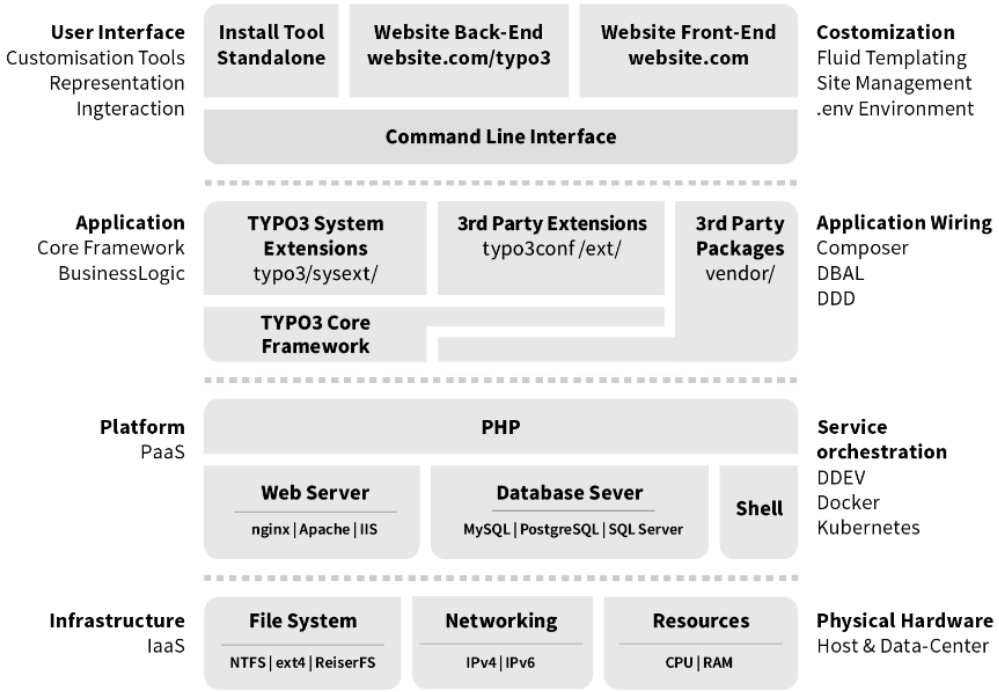


Abbildung 4: Schichten des TYPO3-Systems (TYPO3 Association, o.J. b)

Im Wesentlichen setzen traditionelle CMS auf die sogenannte Web-Architektur nach dem MVC-Muster (siehe Abbildung 5) um. Dabei wird die fertige HTML-Seite auf Anfrage serverseitig zusammengestellt und dann über eine kurzzeitige TCP-Verbindung an den Klienten ausgeliefert. Der Zustand der Anwendung wird vom Server verwaltet. Die Kommunikation findet sitzungsbasiert statt (Balzert, 2011).

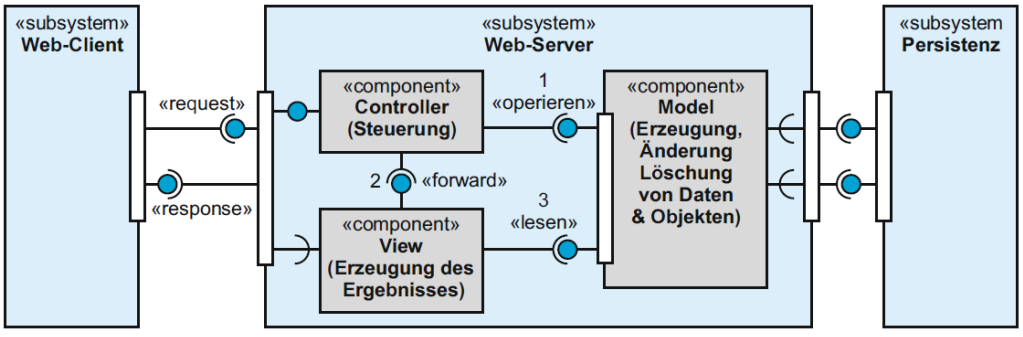


Abbildung 5: Web-Architektur nach dem MVC-Muster (Balzert, 2011, S. 197)

2.5 MACH-basierte CMS

2.5.1 Entstehung und Verbreitung

Die MACH Alliance wurde im Juni 2020 ohne Profitabsichten gegründet (siehe Abbildung 1). Unter den Gründungsmitgliedern befanden sich mit Amplience und Contentstack bereits zwei Anbieter, deren wichtigstes Produkt ein CMS ist. Nach drei Jahren zählte die MACH Alliance bereits 100 Mitglieder (MACH Alliance, 2023b). Contentful liegt aktuell mit 0,46% knapp hinter TYPO3 (0,5%) beim Anteil an den eine Million beliebtesten Webseiten, die builtwith.com regelmäßig auf die Nutzung von CMS prüft (BuiltWith, o.J.). Eine Kerneigenschaft MACH-basierter CMS ist *Headless*. Google Trends zufolge steigt das Interesse an „Headless CMS", seit 2015 kontinuierlich an, wobei es sich nach einem linearen Wachstum bis zum September 2020, aktuell volatil einem Maximum asymptotisch anzunähern scheint (siehe Abbildung 6).

Abbildung 6: Weltweite Google-Suchanfragen normiert auf das Maximum im Zeitraum von 2004 bis heute zu den zum Begriff „Headless CMS" (Google, 2024b)

2.5.2 Zertifizierung

Die Bewerber auf eine Mitgliedschaft in der MACH Alliance müssen ein dreistufiges Evaluationsverfahren durchlaufen, um Mitglieder zu werden. In der ersten Stufe werden die Informationen bewertet, die die Bewerber von sich in einem Formular preisgeben müssen. In der nächsten Stufe wird das Geschäft des Bewerbers auf technische Kriterien geprüft und schließlich werden im Aufnahmegremium die Ergebnisse diskutiert und eine Entscheidung getroffen. Das Aufnahmegremium besteht dabei aus mindestens 10 Freiwilligen von Unternehmen, die bereits Mitglied sind, wobei auf Diversität bei deren Auswahl hinsichtlich Herkunft, Fachgebiet und Geschäftsbereich der Firma geachtet wird. Zudem wird durch ein Non-Disclosure-Agreement sichergestellt, dass sensible Daten nicht weitergegeben werden. Die Auswahl der Freiwilligen verhindert, dass

sich Wettbewerber gegenseitig bewerten. Es gibt drei verschiedene Kategorien von Mitgliedern: System Integrators (SI), die sich am ehesten mit Agenturen vergleichen lassen, Softwareanbieter, die Independent Software Vendors (ISV) genannt werden, und Enablers. Bei Letzteren handelt es sich um große Anbieter von Cloud-Diensten. Die MACH Alliance versteht sich als Wertegemeinschaft, sodass Bewerber sich im Prozess zu den Grundwerten Kollaboration, Gemeinschaft und Diversität äußern müssen. Anbieter etablierter Software können sich auch für die Zertifizierung bewerben, allerdings müssen Sie ihr Geschäftsmodell komplett auf ein MACH-zertifiziertes Portfolio umstellen. Es dürfen keine neuen Lizenzen für *Legacy*-Produkte verkauft werden (MACH Alliance, 2023a).

2.5.3 MACH-Architektur

2.5.3.1 Vorläufer

2.5.3.1.1 Service-orientierte Architekturen (SOA)

Zu den Vorläufern der MACH-Architektur gehört SOA (service-orientierte Architekturen) und das dazugehörige SOAP-Protokoll, das Ende der 1990er Jahre entwickelt wurden, um eine standardisierte Schnittstelle für die Kommunikation unternehmensinterner Anwendungen zu realisieren. Häufig wurden so *Legacy*-Anwendungen in Dienste gekapselt werden, um untereinander und mit neueren Diensten zu kommunizieren (Balzert, 2011).

2.5.3.1.2 Twelve-Factor-App (Cloud-native)

Der Anbieter Heroku gehört zu den Pionieren im Hosting von Anwendungen in der öffentlichen Cloud. Mit dem Versprechen, dass man sich als Entwickler nicht mehr um die Infrastruktur für eine Anwendung kümmern müsse, gingen auch neue Prinzipien und Regeln für die Entwicklung von „Cloud-bereiten" Anwendungen einher, die Entwickler sich erst aneignen mussten. Es gab noch keine Referenzarchitektur. Bereits im Jahr 2012 entwickelte Heroku daher die „Twelve-Factors", um Entwicklern den Zugang zur eigenen Plattform zu erleichtern. Die „Twelve-Factors" entwickelten sich innerhalb weniger Jahre zum *Buzzword* (dt. „Modewort"), welches in Softwarearchitektur-Meetings zum Einsatz kam. Der Nachteil an Modewörtern, die schnell an Popularität gewinnen ist, dass das Wissen zu deren oft komplexer Bedeutung noch sehr unterschiedlich ist. Viele verwendeten den Begriff synonym zu einem weiteren Buzzword - „Cloud-native". Die Einhaltung der Twelve-Factors wurde zur Grundlage, um

Cloud-native Anwendungen zu entwickeln, die sich alle Vorteile öffentlicher Clouds zu Nutzen machen können (Hoffman, 2016). „Twelve-Factor-Apps" versprechen **schnell, skalierbar** und **sicher** zu sein (Stine, 2015).

2.5.3.2 MACH Alliance

Das namensgebende Akronym der Alliance setzt sich aus den Architekturprinzipien **M**icroservices, **A**PI-first, **C**loud-native und **H**eadless zusammen. Sie bilden die Grundlage der „MACH-Architektur" und werden durch funktionale und nicht-funktionale Softwareeigenschaften ergänzt. Zusammen ergeben sie die MACH-Kriterien, deren Einhaltung Voraussetzung für eine Mitgliedschaft in der MACH Alliance ist. Diese versteht sich als Hüter der Kriterien und möchte diese durch die Bereitstellung einer Plattform, von Materialien, Leitfäden, Zertifizierungen und Bildungsangeboten verbreiten und stärken (MACH Alliance, 2023c).

Die Alliance betreibt offensives Marketing gegenüber etablierter Software, die oft als „monolithisch" im negativen Sinne bezeichnet wird. Die Überlegenheit MACH-zertifizierter Lösungen wird betont. Unternehmen sollen durch Botschafter, deren Unternehmen sich bereits einer Transformation unterzogen haben, davon überzeugt werden den Schritt von alter weniger agiler Software hin zu zukunftssicherer qualitativer MACH-zertifizierter Software zu gehen, welche aus Bausteinen zusammengesetzt werden kann *(= Composable).* MACH-zertifizierte Lösungen sollen Unternehmen im sich rasch weiterentwickelnden Webanwendungsmarkt davor bewahren von der Plattform eines einzigen Anbieters abhängig zu sein, wenn es um die Auswahl und den Einsatz innovativer Lösungen geht (MACH Alliance, 2023c).

2.5.3.3 Microservices

Tremp definiert einen Microservice als „eigenständige fachlich klar fokussierte Teilfunktion einer Applikation" (Tremp, 2021, S. 64) und ergänzt, dass sie in einem eigenen Container ausgeführt werden, auf den über eine klar definierte Schnittstelle zugegriffen werden kann (Tremp, 2021). Der Begriff geht auf eine Konferenz in Venedig im Jahr 2011 zurück, bei der Softwarearchitekten eine Architektur diskutierten, der sie dann ein Jahr später den Namen „Microservices" gaben. Im Jahr 2014 erlangte der Begriff dann zum ersten Mal größere Aufmerksamkeit, als Adrian Cockcroft beim *Silicon Valley Microservices Meetup* über die Migration von Netflix zur Microservices Architektur referierte (Fowler, 2014).

MACH-basierte Anwendungen basieren auf Microservices. Sie werden aus lose gekoppelten austauschbaren Komponenten (Microservices) zusammengesetzt, die über eine sprachunabhängige Schnittstelle auf Netzwerkebene über das TCP/IP Protokoll miteinander kommunizieren und gemeinsam einen Geschäftsprozess abbilden. Jeder Microservice erfüllt eine klar abgegrenzte Teilaufgabe des Geschäftsprozess (=*single responsibility principle*) und kann unabhängig von der restlichen Anwendung in einem eigenen Prozess ausgeführt werden (Fowler, 2014).

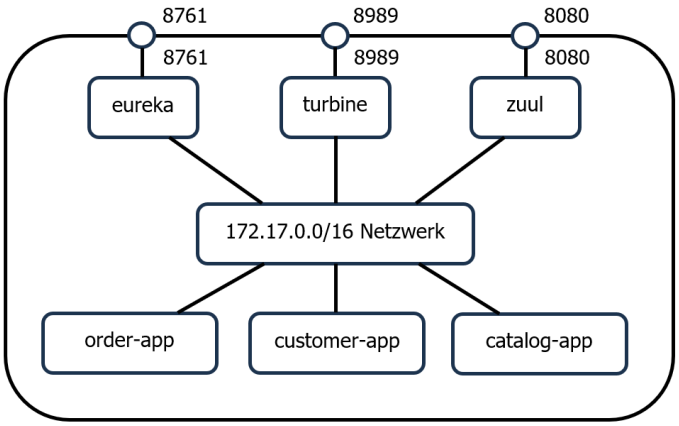


Abbildung 7: Beispielhafter Geschäftsprozess in Docker Compose (Wolff, 2018, S. 320)

Microservices werden zumeist mittels Docker zur Orchestrierung in separate Container verpackt und mit Orchestrierern wie Kubernetes oder Docker Compose zu einer Anwendung verknüpft (siehe Abbildung 7). Üblicherweise werden Hilfsdienste aus bekannten Bibliotheken für Standardfunktionalitäten eingesetzt, im Beispiel sind das zuul für das Routing externer Anfragen an die einzelnen Dienste, eureka für das Bekanntmachen der Dienste untereinander und turbine für das Sammeln der Monitoring-Daten der einzelnen Microservices. Alle drei gehören zum frei verfügbaren Spring Cloud Netflix-Stack für Java (Wolff, 2018).

MACH-basierte CMS realisieren im Prinzip den serverseitigen Teil von RIA-Web-Architekturen (siehe Abbildung 6). RIA steht dabei für (Rich Internet Application). Der Client erhält die serverseitig gespeicherten Daten in einem plattformunabhängigen Austauschformat (bspw. JSON) und rendert sich damit selbst sein Ausgabeformat. Der Zustand der Anwendung wird vom Klienten verwaltet, was die zustandslose Kommunikation zwischen Server und Klient ermöglicht (Balzert, 2011).

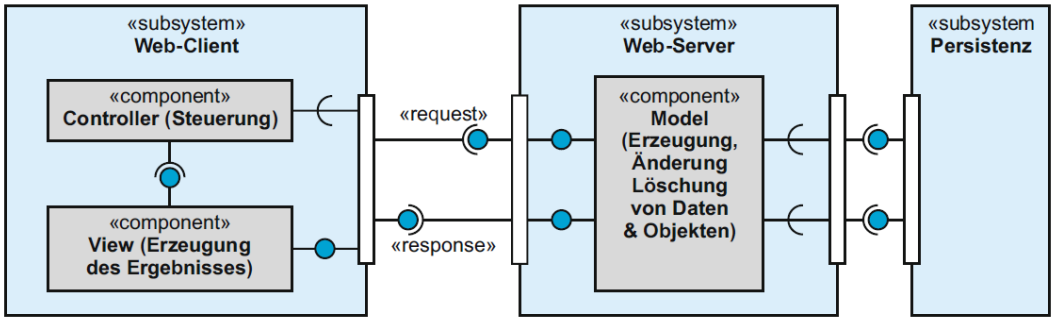


Abbildung 8: Beispiel für eine RIA-Web-Architektur (Balzert, 2011, S. 199)

2.5.3.3.1 Vorteile

Die Nutzung einer Microservice-basierten Architektur zielt auf die Erreichung von nicht-funktionalen Anforderungen im Rahmen von Softwareprojekten ab. Entwicklerteams können parallel und unabhängig voneinander an unterschiedlichen Komponenten der Software Änderungen vornehmen, diese testen und veröffentlichen (Änderbarkeit, Testbarkeit) (Tremp, 2021). Die Produktivität von Entwicklern und das Klima im Team kann gesteigert werden. Ihre Einsatzbereitschaft steigt, da sie mit ihrem Team für einen Teil der Anwendung die Verantwortung tragen und freier Entscheidungen treffen können (Wolff, 2018). Es besteht zudem weitestgehende Freiheit welche Technologie zur Lösung eines Problems genutzt wird, da die Schnittstellen plattformunabhängig sind (Interoperabilität). Fehlentwicklungen lassen sich leichter entfernen, da durch eindeutig definierte Schnittstellen keine Abhängigkeiten zum Rest der Anwendung bestehen (Austauschbarkeit). Im Betrieb führt der Ausfall einer Komponente durch Resilienzmaßnahmen nicht zum Ausfall einer anderen (Fehlertoleranz). Neue Versionen einer Anwendung können kontinuierlich und automatisiert bereitgestellt werden ohne die Notwendigkeit manueller Updates (Wartbarkeit) (Tremp, 2021). Zusätzlich können Legacy-Systeme durch die Integration von Microservices weiter genutzt und modernisiert werden (Kosteneffizienz) (Wolff, 2018).

Fowler beschreibt, dass Microservices auch insofern einen Vorteil haben, dass sie verhindern, dass mehr entwickelt wird, als wirklich gefordert war, da jeder Dienst vorab im Detail mit dem Kunden definiert wird (Kosteneffizienz). Die eindeutige Definition der Dienste ermöglicht es Lösungen für wiederkehrende Problemstellungen unter Entwicklern leichter zu teilen (Wiederverwendbarkeit). Die vielfältigen Automatisierungstechniken die große Cloud-Anbieter mittlerweile anbieten, ermöglichen automatisierte Tests (Testbarkeit) (Fowler, 2014) und eine feingranulare elastische Skalierung von Teilaufgaben der Anwendung (Skalierbarkeit) (Tremp, 2021).

2.5.3.3.2 Herausforderungen

Neben den Vorteilen existieren viele große Herausforderungen in der Nutzung Microservice-orientierter Architekturen. Die Reduzierung von Komplexität und Risiko auf Mikroebene geht mit größerer Komplexität und Risiko auf Ebene der Anwendung im Ganzen einher. Die Nutzung vieler Technologien erfordert für jeden Microservice eine hohe Spezialisierung der Mitarbeiter, wodurch diese schwer ersetzbar sind. Das Aufteilen eines Geschäftsprozesses in möglichst lose gekoppelte Microservices ist von entscheidender Bedeutung für deren späteren Mehrwert. Der Domain-Driven-Design Ansatz hilft dabei. Sollte ein Microservices nicht im richtigen Funktionsumfang definiert worden sein, gestaltet sich das Refactoring schwierig, sofern dabei Bestandteile von anderen Microservices herausgetrennt werden müssen. Schwierigkeiten verursacht auch die Latenz in der Kommunikation von Microservices über Netzwerkprotokolle. Die Verzögerungen summieren sich beim Einsatz vieler Microservices innerhalb einer komplexen Anwendung (Tremp, 2021).

Die Microservices-Architektur geht mit dem Einsatz mehrerer mitunter sogar unterschiedlicher Datenbanksysteme innerhalb einer Anwendung einher (siehe Abbildung 7). Jeder Microservice verwaltet idealerweise nur, die für ihn relevanten Daten. Sobald Microservice-übergreifende Daten gespeichert werden müssen, wie zum Beispiel Sitzungsdaten, kommt es entsprechend der Latenzen zu zeitweiligen Inkonsistenzen, da keine Transaktionen nach den ACID-Prinzipien, wie bei einer einzigen relationalen Datenbanken möglich sind (Fowler, 2014).

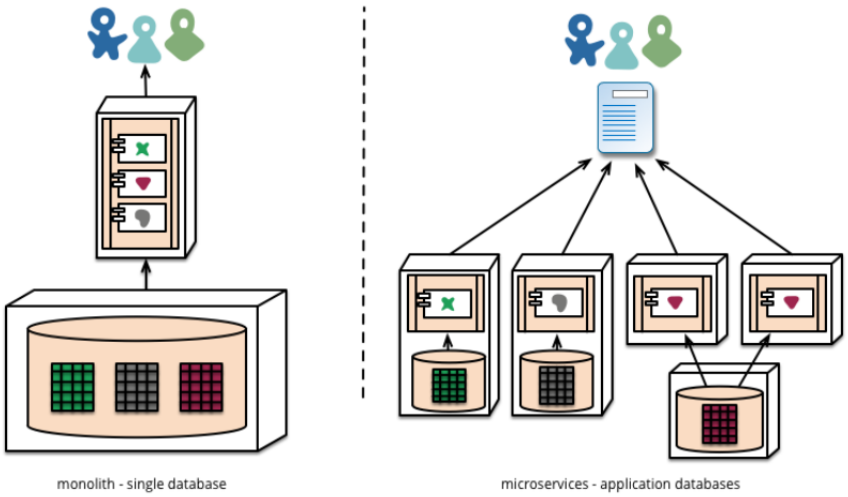


Abbildung 9: Vergleich Datenbanknutzung bei monolithischen Systemen gegenüber Microservice-basierten Systeme (Fowler, 2014).

Nach dem CAP-Theorem verbleibt durch die zunächst nicht erreichbare Konsistenz nur die Möglichkeit auf Partitionstoleranz und Verfügbarkeit bei Schreibvorgängen in der Datenbank zu setzen. Statt der ACID- werden die BASE-Prinzipien (**B**asically **A**vailable, **S**oft State und **E**ventual Consistency) angewendet, wobei die Daten auf die Datenbankknoten zu Gunsten der Verfügbarkeit repliziert werden und die Herstellung der Konsistenz nachträglich im Programmcode sichergestellt werden muss (Kratzke, 2023).

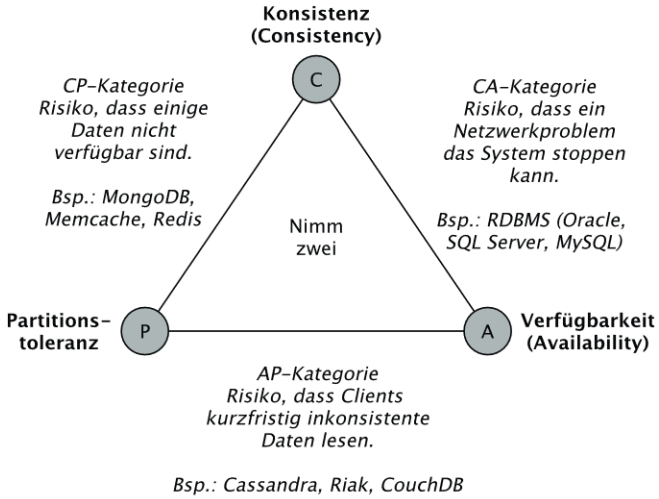


Abbildung 10: CAP-Theorem (Kratzke, 2023, S. 183)

Eine weitere Schwierigkeit birgt, dass Fehlertoleranz nicht automatisch beim Einsatz von Microservices erreicht wird. Das Zusammenspiel der Dienste muss in der Produktionsumgebung in Echtzeit analysiert werden. Dafür muss jeder Dienst in Monitoring- und Logging-Tools integriert werden. Wichtige Beobachtungsgrößen sind dabei die Latenz, der Durchsatz, Datenbankzugriffe, oder Zugriffe auf einen Dienst von außen (Fowler, 2014).

2.5.3.4 API-First

Die MACH Alliance fordert von unabhängigen Softwareanbietern (ISV), die sich auf eine Mitgliedschaft bewerben, dass die API vor der eigentlichen Anwendung entwickelt sein muss. Sie muss alle Funktionalität der Software zugänglich machen und REST oder GraphQL unterstützen. Im Bewertungssystem des Aufnahmeverfahrens macht das 15% der möglichen Gesamtbewertung aus (MACH Alliance, 2023a, S. 20).

Hoffman weist daraufhin, dass die Schnittstellendefinition vor der eigentlichen Entwicklung den entscheidenden Vorteil hat, dass sich Teams die parallel an weiteren Diensten arbeiten und von der Schnittstelle abhängen, darauf verlassen können, dass diese sich nicht im Projektverlauf verändert (Kompatibilität). Zudem lässt sich so zu Beginn sicherstellen, dass die Schnittstelle für alle Zielgeräte optimiert sein wird (Anpassungsfähigkeit). Die Schnittstellendefinition ist auch ein Artefakt, welches die Kommunikation und damit auch die Zusammenarbeit mit allen Projektbeteiligten erleichtert (Verständlichkeit). Zur automatisierten Dokumentation stehen mittlerweile umfangreiche Tools zur Verfügung (Hoffman, 2016).

API-first geht auf das *contract-first* Architekturmuster zurück. Es erlaubt kontinuierliche Integrationstests, indem die Schnittstellen der Anwendung mit der künftigen Umgebung regelmäßig getestet werden. Die Anwendung soll dadurch bei unvorhergesehenen Anforderungen organisch wachsen können, ohne dass der ursprüngliche Entwurf dem entgegen steht (Hoffman, 2016).

2.5.3.5 Cloud-native SaaS

Cloud-native Anwendungen wurden speziell für den Betrieb in der Cloud entwickelt. Sie können alle Vorzüge dieses Betriebsumfeldes nutzen (Fernando, 2023). Wie schon unter 2.5.3.1.2 beschrieben, wurde der Begriff „Cloud-native" nach 2012 von vielen synonym zum Konzept der „Twelve-Factors" verwendet, welches Netlify zu dieser Zeit veröffentlicht hatte. Zuvor war der Begriff erstmalig von Paul Fremantle dem CTO von WSO2 im Jahr 2010 verwendet worden, der einen Blog-Artikel über die Schwierigkeiten bei der Migration von Software in die Cloud damit überschrieb. Auf der Grundlage seiner Erfahrungen fand er Kerncharakteriska, die Cloud-native Anwendungen seiner Meinung nach ausmachen. Sie müssen *distributed* (dt. verteilt) sein, das heißt die Anwendung muss aus mehrere Knoten bestehen, die parallel zueinander ausführbar sind und die sich entweder gar nichts (shared-nothing-architecture) oder nur Konfiguration und Sitzungszustand teilen und zentral Protokollieren. Die Schnittstellen sollten sprach- und plattformunabhängig gestaltet sein, um externe Dienste bestmöglich einbinden zu können (Fremantle, 2010).

Cloud-native Anwendungen sind *elastic* (dt. elastisch). Je nach Last kann ein entsprechender Controller ihre Leistung horizontal, als auch vertikal skalieren. Sie sind *multi-tenant* (dt. Mehrmandantenfähigkeit), das heißt mehrere Mandanten

teilen sich die Hardwareressourcen, wobei kein Zugriff auf die Daten anderer Mandanten besteht. Das erlaubt elastische Mandanten innerhalb einer Anwendungsinstanz. Cloud-native *multi-tenant* Anwendungen erlauben *self-service*. Das heißt, dass Kunden können sich unabhängig von einem Administrator neue Mandanten einer Software erstellen, was wiederum die Bereitstellungszeit verringert (Fremantle, 2010).

Des Weiteren ist die feingranulare Messung und Abrechnung *(Granularly metered and billed)* eine wichtige Eigenschaft von Cloud-native Anwendungen. Von der ersten Minute an nutzen Sie nur so viele Ressourcen wie nötig. Und schließlich lassen sich Cloud-native Anwendungen inkrementell bereitstellen und testen, wodurch Versionsupdates Stück für Stück vorgenommen werden können, ohne den Produktionsbetrieb zu unterbrechen (Fremantle, 2010).

Stine, der CTO von Pivotal, erweitert 2015 die *Cloud-native* Definition von Fremantle in seinem Buch „Migrating to Cloud-native Application Architectures" um weitere Aspekte. Seine Definition umfasst die *Twelve-Factors* und ergänzt die Prinzipien „Microservices", „Self-service Agile Infrastructure", „API-Based collaboration" und „Anti-fragility" und hinsichtlich der Ziele die Unterstützung für mobile Geräte. Stine vertieft das *self-service* Charakteristikum von Fremantle. Jede *Cloud-native* App braucht nach seiner Definition auch ein eigenes Team, welches die eigene Plattform zum Betrieb entwickelt und betreibt. Der Kunde soll sich selbstständig aus Artefakten die passende Anwendungen nach den eigenen Wünschen zusammenstellen können und **schnell**, **sicher** und **skalierbar** arbeiten, ohne sich über die zugrundeliegende Installation und Konfiguration Gedanken machen zu müssen (Stine, 2015).

Zur Schnittstellenkommunikation von *Cloud-native* Anwendungen („API-Based Collaboration") ergänzt Stine, dass immer nur ein Dienst Zugriff auf einen Datenspeicher haben darf (Stine, 2015). Schließlich greift er mit „Antifragility" einen Begriff des Mathematikers Nassim Taleb als Eigenschaft für *Cloud-native* Software auf. Antifragile Software lernt mit Belastungen umzugehen, findet Schwachstellen, beseitigt diese und wird mit der Zeit dadurch stärker (Stine, 2015).

Im Jahr 2015 wurde die „Cloud Native Computing Foundation" (CNCF) gegründet, die das von Google entwickelte „Kubernetes" zur Orchestrierung von Containern

Open Source weiterentwickeln sollte. Die Organisation setzte mit „Kubernetes" und „Prometheus" de-facto Standards im Bereich *Cloud-native* und definierte 2018 den Begriff über die Verwendung der Kerntechnologien *Container*, *Microservices*, *DevOps*, *Declarative API*, *Immutable Infrastructure* und *Service mesh*. Diese Techniken ermöglichen die automatische Bereitstellung, Skalierbarkeit, Resilienz, Verwaltbarkeit und Überwachbarkeit von Systemen (Team, 2024).

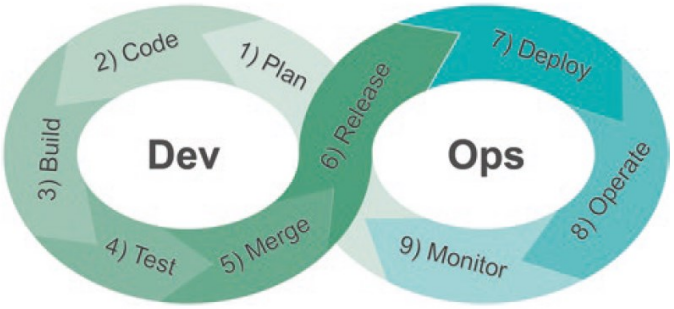


Abbildung 11: Schritte von DevOps (Tremp, 2021, S.63)

Das Prinzip *Software-as-a-Service* (SaaS) ist eng mit dem Begriff *DevOps* verbunden. Früher war der Nutzer von Software für den Betrieb verantwortlich. Bei *DevOps* ist ein Entwicklerteam für den gesamten Softwarelebenszyklus einer kleinen Komponente zuständig. Das Risiko, dass ein Fehler lange Zeit im Code ist wird geringer, da kontinuierlich lauffähige Versionen im Betrieb getestet werden und der Ursprung von Fehlern im kleinen Team schnell gefunden ist (Cloud Native Computing Foundation, o.J.). Der Nutzer kann sich also direkt auf die Abbildung seiner Geschäftslogik konzentrieren und muss sich nicht um Installation, Konfiguration und Wartung kümmern. Individuelle Anforderungen lassen sich in der Folge nur bedingt umsetzen, da die Konfigurationsmöglichkeiten der Systeme begrenzt sind (Barker, 2016).

2.5.3.6 Headless

Der Begriff *Headless* im Sinne der MACH-Architektur wird vor allem im Zusammenhang mit CMS verwendet. *Headless* CMS stellen ihre Inhalte ausschließlich über eine API bereit und verfügen über keine Komponente zur Darstellung. *Headless* CMS werden oft mit *Decoupled* CMS, wie den unter 2.3.3 erwähnten Static Site Generators verglichen, da beide Varianten die Inhalte von deren Darstellung trennen. Ein wesentlicher Unterschied besteht aber darin, dass Headless CMS auf Anfragen reaktiv Inhalte herausgeben. *Decoupled* CMS haben

aber bereits proaktiv alle Inhalte auf einem Webserver oder in einem Content Delivery Network (CDN) bereitgestellt. Sie verfügen zudem über ein Frontend-Vorlagenkonzept, welches Headless CMS nicht haben (Barker, 2017).
Laut Deane Barker gab es schon vor der Entstehung reiner Headless CMS, die Möglichkeit CMS Headless zu betreiben. Für WordPress wurde beispielsweise schon im Mai 2014 die „JSON REST API" Extension veröffentlicht (McCue, 2014). Die reinen Varianten die später entstanden, prägten dann den Begriff „Headless" für sich selbst, um sich von *Decoupled* CMS und CMS , deren REST-API nicht *API-First* entstanden war, abzugrenzen (Barker, 2017).

2.5.3.7 MACH-Kriterien

Die MACH Alliance definiert in ihrem Whitepaper (MACH Alliance, 2023c) die allgemeinen Kriterien für die Erfüllung der MACH-Architektur. Um eine detailliertere Bewertung der Hauptkriterien vorzunehmen, wurden den Hauptkriterien weitere Unterkriterien aus dem Admissions-Playback (MACH Alliance, 2023a) zugeordnet. Gleiches gilt für Kriterien, die nur an verschiedenen Stellen der Webseite der MACH Alliance zu finden sind und als Vorteil der MACH-Architektur gegenüber anderen Systemen hervorgehoben werden. Anhand dieses Kriterienkatalogs (Anhang 1 – Analyse-Tabellen) soll im anschließenden Kapitel eine Analyse der traditionellen Open Source CMS vorgenommen werden.

3 Analyse zum Einfluss der MACH-Architektur auf traditionelle Open Source CMS

3.1 Methodik: Quantitative Analyse

Anhand des im vorherigen Kapitel festgelegten Kriterienkatalogs soll untersucht werden in welchem Maße die traditionellen CMS bereits von den Prinzipien der MACH-Architektur beeinflusst werden.

Der Einfluss (E) eines Kriteriums wird in den folgenden Stufen angegeben:

- 0 = Kein Merkmal des CMS
- 1/2 = Initiative, Teillösung oder externe Lösung
- 1 = (geplantes) Kernmerkmal des CMS

Sofern ein Kriterium aus mehreren Unterkriterien besteht, ergibt die Summe der Einzelkriterien gleich eins. Die Bewertung eines Einzelkriteriums wird stichwortartig begründet und durch eine Quelle belegt. Bei insgesamt 10 Hauptkriterien ergibt sich eine maximal erreichbare Summe von 10 Punkten.

Alle Kriterien werden aus der Perspektive eines Zertifizierers betrachtet, der prüft, ob die MACH-Kriterien zu den Kernfeatures des CMS gehören und ob sich mit dem CMS eine MACH-basierte Plattform realisieren lässt.

Die detaillierten Analysetabellen finden sich in „Anhang 1 – Analyse-Tabellen" dieser Arbeit. Vergleich und Einschätzung der erreichten Bewertungen finden sich direkt im folgenden Abschnitt 3.2.

3.2 Vergleich erreichter Bewertungen

CMS	TYPO3	WordPress	Drupal	Joomla!
K1: Decoupled	0,50	0,50	0,50	0,50
K2: Cloud-Native	0,50	0,70	0,60	0,50
K3: Natively Headless and API Enabled	0,25	0,75	0,50	0,38
K4: Modular Implementation	0,88	0,75	0,75	0,75
K5: Agility and Extensibility	0,90	1,00	0,90	0,70
K6: Scalability	0,70	0,60	0,60	0,50
K7: Stability	0,50	0,50	0,50	0,50
K8: Strategic Alignment	1,00	1,00	1,00	1,00
K9: Documentation	0,75	0,75	0,75	0,25
K10: Ease-of-Use	0,50	0,83	0,83	0,33
Summe (max. 10)	**6,48**	**7,38**	**6,93**	**5,41**

3.3 Ergebnisse

Die Analyse der traditionellen CMS hat ergeben, dass diese bereits alle über 50% der MACH-Kriterien erfüllen.

Erstaunlicherweise erreicht WordPress von den untersuchten CMS die meisten Punkte, es folgen mit geringem Abstand Drupal und TYPO3 auf Platz zwei und drei. Joomla! landet etwas abgeschlagen, auf Platz vier.

WordPress sammelt vor allem durch Rückwärtskompatibilität, sowie automatische Updates und Upgrades Punkte. Weitere Stärken sind die gute Dokumentation, kostenfreie Trainings, Schulungsmaterialien und SDKs, die das CMS allen Benutzergruppen niedrigschwellig zugänglich machen. WordPress profitiert dabei von seiner sehr großen internationalen Community, die die Verfügbarkeit von vielen freiwilligen Unterstützern begünstigt.

Drupal erreicht Rang zwei in der Bewertung. Auch Drupal hat weltweit eine große Anzahl von Unterstützern und kann sich dadurch schnell an aktuelle Entwicklungen anpassen. Bereits 2016 wurden mit der API-First-Initiative gezielte Schritte in Richtung MACH-Kriterien unternommen. Das System punktet vor allem durch seine native „JSON:API", welche die JSON:API-Spezifikation (JSON:API, o.J.) umsetzt und eine standardisierte plattformunabhängige Schnittstelle für die meisten Anwendungsfälle bietet. Die kostenfreien Lernpfade

und öffentlichen Live-Trainings für alle Benutzergruppen ermöglichen es erste Eindrücke vom CMS zu gewinnen und es niederschwellig zu erlernen.

TYPO3 erreicht in den Kategorien „Modular Implementation", „Scalability" und „Documentation" die höchste Punktzahl unter den vier traditionellen CMS. Das liegt vor allem am Webhooks-Support, der 2022 in Version 12.1 zum Kernfeature wurde. Abschläge muss TYPO3 gegenüber Drupal und WordPress vor allem in der Kategorie „Natively Headless and API Enabled" hinnehmen, da es über keine native REST- oder GraphQL Schnittstelle verfügt. Der besonders lange Support-Zeitraum für eine Major-Version (3 Jahre LTS + 3 Jahre ELTS) wurde als native Upgrade-Freiheit bewertet, wodurch TYPO3 gegenüber seinen Konkurrenten in der Kategorie „Cloud-native" punkten kann, allerdings kommt es hier zu Abschlägen bei den Unterkategorien „PaaS oder SaaS" und „automatischen Upgrades". In der Kategorie „Documentation" erreicht TYPO3 so viele Punkte, wie die wesentlich weiter verbreiteteren Systeme Drupal und WordPress. Leider wurden nur kostenpflichtige Trainings und Lernpfade zu aktuellen Versionen gefunden, was zu Abzügen in der Kategorie „Ease-of-Use" führt.

Joomla! erreicht die wenigsten Punkte in der Analyse, obwohl seit Joomla 4 eine native REST-API zur Kernfunktionalität gehört. Besondere Anstrengungen zur Entkopplung des Kern wurden schon 2013 mit der Veröffentlichung des „Joomla! Frameworks" unternommen, allerdings greift das CMS bisher nur an wenigen Stellen auf das Framework zu. Es konnten keine detaillierten Trainings und Zertifizierungen für Entwickler gefunden werden, wodurch das System Abzüge in der Kategorie „Ease-of-Use" verbucht. Auch bei der Kategorie „Documentation" gab es weniger Punkte, da die Dokumentation an vielen Stellen kein einheitliches, aktuelles Erscheinungsbild hat, schlecht suchbar und unvollständig in Bezug auf aktuelle Versionen ist. Der Rückstand des CMS könnte sich durch die 9-jährige Release-Pause zwischen Version 3 und Version 4 erklären. Wie unter 4.1 erwähnt, fehlte in dieser Zeit eine Strategie und Roadmap in der Joomla!-Entwickler-Community, sodass wertvolle Zeit für aktuelle Entwicklungen verloren gegangen sein könnte. Teile der Community des CMS, das zeitweise populärer war als WordPress, wie unter 2.4.2 gezeigt, könnten sich in dieser Zeit dem CMS abgewandt haben, wodurch es zu Schwierigkeiten gekommen sein könnte, die Dokumentation und Schulungsmaterialien auf einem aktuellen Stand zu halten.

Als valider Benchmark für moderne CMS eignen sich die MACH-Kriterien nur bedingt, da sie unabhängig vom Anwendungsfall Architekturprinzipien vorgeben mit denen funktionale und nicht-funktionale Softwareeigenschaften erreicht werden sollen. Das widerspricht der Theorie zur Softwarearchitektur, die verlangt, dass in der Entwurfsphase auch der Anwendungsfall Berücksichtigung finden muss, wie unter 2.2.2 gezeigt. Alternative Umsetzungen, die eventuell effizienter und nachhaltiger sind, werden nicht honoriert. Beispielsweise soll generell auf Microservices gesetzt werden, auch wenn Modularität bereits auf Anwendungsebene erreicht ist und Microservices nur zu unnötigen Kosten durch Komplexität und Ressourcenverbrauch führen würden. Auch in der Kategorie „Stability" konnten alle vier Systeme nicht die volle Punktzahl erreichen, da das „Donut"-Geschäftsmodell der Open Source Systeme die Trennung von Entwicklung und Betrieb vorsieht. Das führt zu keinem entscheidenden Nachteil gegenüber MACH-basierten CMS, da auch die externen Hosting-Anbieter Service-Level-Agreements (SLAs) anbieten, die dem Kunden Sicherheit und Verfügbarkeit garantieren und die Wartung der Systeme inklusive Updateservice umfassen. Im Gegenteil wirkt sich die Trennung von Entwicklung Betrieb auch vorteilig für den Kunden aus. Im Wettbewerb der Anbieter kann er zum besten Angebot wechseln, ohne gebunden zu sein.

Die objektive Analyse wurde durch die stichwortartigen Definitionen der einzelnen MACH-Kriterien erschwert, welche Interpretationsspielraum bieten, da einige Begriffe in der Literatur, wie im Kapitel zwei gezeigt, unterschiedlich ausgelegt werden. Die Kriterien wurden den öffentlichen Materialien der MACH Alliance entnommen. Es wird angenommen, dass ein detaillierter interner Kriterienkatalog für die Zertifizierung der zukünftigen Mitglieder vorliegt, um objektive und wiederholbare Ergebnisse in der Zertifizierung zu erhalten. Eine weitere Schwäche der Analysemethode stellt auch die Tatsache dar, dass nicht ausgeschlossen werden kann, dass trotz intensiver Recherche im Internet Lösungen für einzelne Kriterien existieren, diese aber beispielsweise nur auf Anfrage von Agenturen angeboten werden.

WordPress, Drupal und Joomla! wurde SaaS als Kernfeature zugeordnet, da die Entscheidungsträger in der Kernentwicklung selbst eine SaaS-Plattform mit dem jeweiligen CMS betreiben und davon ausgegangen wird, dass dadurch Aspekte des SaaS-Betriebs bei der Kernentwicklung verstärkt Berücksichtigung finden.

Hinsichtlich der schnellen Erlernbarkeit der Systeme, die unter der Kategorie Ease-Of-Use zur Bewertung stand, sind empirische Vergleichsstudien nötig, um validere Aussagen treffen zu können. Die Tatsache, dass Drupal und WordPress nativ einen Drag&Drop Layout Editor anbieten wurde allerdings als deutlicher Vorteil gegenüber den anderen Systemen hinsichtlich der schnellen Erlernbarkeit durch die Nutzer bewertet.

Fraglich ist, ob die Mitglieder der MACH Alliance die Kriterien zu 100 Prozent erfüllen können. Hinsichtlich der Rückwärtskompatibilität zeigt das Beispiel Contentful, dass auch MACH-basierte Anwendungen auf Breaking Changes ihrer API angewiesen sind, um sich weiterentwickeln zu können. Bei Contentful finden diese Breaking Changes in unregelmäßigen Abständen und teilweise mehrfach innerhalb eines Jahres statt (Contentful, o.J.).

4 Individuelle Betrachtung der Produktstrategie am Beispiel TYPO3

4.1 Produktstrategie

Die Erarbeitung einer Produktstrategie gehört laut Becker zu den wichtigsten Entscheidungen des digitalen Produktmanagements. Sie definiert sich nach Chandler als Festlegung der grundlegenden und langfristigen Ziele für die Produktorganisation, sowie der benötigten Maßnahmen und Ressourcen zu deren Verwirklichung (Becker, 2023, zitiert nach Chandler, 1962).

Besondere Bedeutung misst Becker der Strategie in Folge des technologischen Wandels bei. Leicht verfügbare Standardlösungen zu komplexen Problemen verringern die Kosten und das technologische Risiko der Umsetzbarkeit. Andererseits steigen die Ansprüche der Kunden, hinsichtlich der Lösung ihrer Probleme, da sie einem wachsenden Angebot von Optionen gegenüberstehen, die sie optimal nutzen wollen. Anbieter von Software müssen angesichts von starkem Wettbewerb und knapper Ressourcen immer schneller entscheiden, welche Entwicklungsoptionen dem Produkt zum weiteren Erfolg verhelfen und gleichzeitig finanziell umsetzbar sind. Angesichts knapper Ressourcen ist es wichtig sich auf strategisch wichtige Bereiche zu fokussieren. Zur Festlegung von Zielen und den Schritten zur Erlangung dieser bedarf es zuallererst einer Analyse der Ausgangssituation, hierbei wird die aktuelle Marktposition evaluiert. Im ersten Schritt wird das „Produktspielfeld" (Becker, 2023, S. 52) abgesteckt, um die mögliche Lösungsmenge einzuschränken. Es leitet sich aus Vision und Mission der Unternehmung ab. Anschließend wir der Startpunkt, also die Ausgangssituation anhand von messbaren Key Performance Indikatoren (KPI) mit Bezug zum Wettbewerb bestimmt. Des Weiteren werden interne und externe Zukunftsfaktoren erfasst, bevor schließlich das Ziel mit klarem zeitlichen Bezug und der Weg dorthin festgelegt werden können. Das Ziel sollte dabei zu einem klaren Wettbewerbsvorteil führen (Becker, 2023).

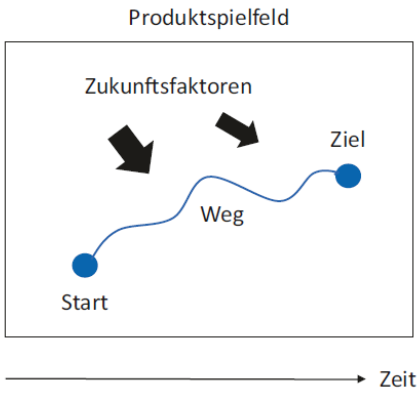


Abbildung 12: Elemente der Produktstrategie (Becker, 2023, S. 51)

Die Wichtigkeit einer Produktstrategie unterstreicht das Beispiel des traditionellen CMS Joomla! und der Entwicklung des Version 4. Der Freiwillige Contributor Luca Marzo berichtet im Podcast Joomcast von 2020, dass die Version pünktlich hätte released werden können, wenn es eine klar definierte Strategie und Roadmap gegeben hätte (Marzo, 2020).

4.2 Methodik: Experteninterview

4.2.1 Interviewdesign und Auswahl der Teilnehmer

Für die individuelle Betrachtung des Einflusses der MACH-Architektur auf die Produktstrategie von TYPO3 wurde die qualitative Forschungsmethode des mündlichen Experteninterviews per Online-Videokonferenz ausgewählt, um aktuelle Informationen aus Entwickler- und Anwenderkreisen des CMS zu erlangen. Das Experteninterview eignet sich zudem gut für das Thema, da es auf die Erlangung von „praxis- und erfahrungsbezogenem, sowie technischem Wissen" (Helfferich, 2019, S. 682) ausgelegt ist.

Es wurde nach Teilnehmern gesucht, die sich mit der Architektur von TYPO3 und im Idealfall auch mit der Architektur MACH-basierter Systeme auskennen, um deren Einschätzung zur Ausgangssituation und zu aktuellen Entwicklungen zu sammeln. Durch die gute Vernetzung meines Betreuers in die TYPO3 Community konnten Interviewpartner gewonnen werden, die direkt an der Weiterentwicklung des Systems beteiligt sind, die viele Erfahrungen mit TYPO3 in der Realisierung von Kundenprojekten erlangt haben oder die sich auf das Hosting von TYPO3 spezialisiert haben. Die Teilnehmer konnten wiederum den Kontakt zu weiteren

potenziell interessanten Interviewpartnern herstellen. Zusätzlich wurde nach passenden Teilnehmern in den Community-Kanälen des CMS gesucht. Durch die langjährige themenbezogene Erfahrung der Experten im beruflichen Kontext, kann nach Helfferich davon ausgegangen werden, dass deren Wissen „in besonderem Maß verallgemeinerbar und gültig ist" (Helfferich, 2019, S. 681).

Zur Vorbereitung auf das Interview erhielten die Experten einen Leitfaden (siehe Anhang 2 - Interviewleitfaden) mit den wichtigsten Fragen und Aspekten des Interviews. Durch die spezielle Thematik sollte so vorab auch die Möglichkeit gegeben werden, zu klären, welche der Fragen eventuell aus Vertraulichkeitsgründen nicht beantwortet werden können. Der Leitfaden fokussiert sich im Experteninterview auf Sachfragen und deren strukturierte Abfolge (Helfferich, 2019).

Das qualitative Interview ist ein nicht-standardisiertes wissenschaftliches Verfahren, welches keine objektiven Ergebnisse liefert. Die Antworten unterliegen immer der Subjektivität des Befragten. Erstes Gütekriterium ist es daher, dass die Subjektivität zusammen mit dem Kontext der Interviewsituation reflektiert werden muss (Helfferich, 2019). Das Mittel der Online-Videokonferenz hat hier positiven Einfluss, da die Interviews direkt am Arbeitsplatz und damit im natürlichen Umfeld des Experten durchgeführt werden können, sodass der Kontext der Interviewsituation weniger Einfluss hat (Lamnek, 2010). Ein weiteres Gütekriterium für die Gestaltung der Interviewsituation ist Offenheit, weshalb der überwiegende Teil der Fragen offen formuliert wurde, um die Interviewteilnehmer in ihren Antwortmöglichkeiten nicht einzuschränken. Ergänzend wurden Rückfragen gestellt, wenn die Antworten nicht eindeutig interpretiert werden konnten oder ein wichtiger Aspekt der Fragestellung noch nicht betrachtet wurde (Helfferich, 2019). Nach Lamnek ist auch Flexibilität ein wichtiges Gütekriterium. So wurde im Laufe eines Interviews, statt auf die strenge Reihenfolge zu achten, situationsbezogen entschieden, welche der Fragen und Aspekte sich gut in den Gesprächsverlauf einfügen und das Expertenwissen des Interviewpartners in besonderem Maße adressieren, mit dem Ziel Breite und Tiefe der Antworten zu verbessern (Lamnek, 2010).

4.2.2 Interviewleitfaden

Im ersten Abschnitt wurden die Rolle und relevante Erfahrungen des Experten zum Thema ermittelt, um dessen Standpunkt bei der späteren Analyse besser einordnen zu können. Der weitere Leitfaden (siehe Anhang 2 - Interviewleitfaden) folgt der Methodik beim Festlegen einer Produktstrategie nach Becker, wie unter 4.1 beschrieben.

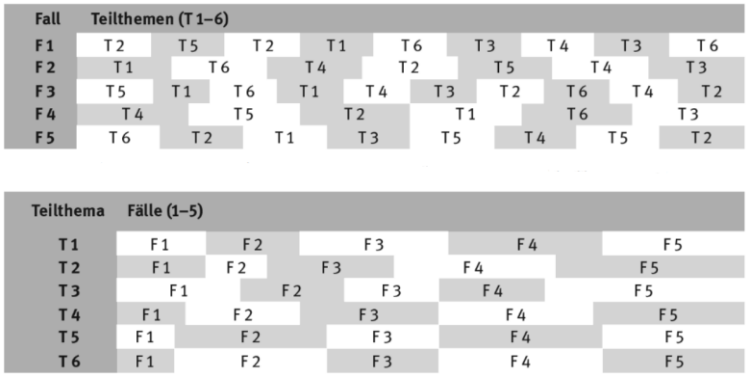
Im zweiten Teil des Interviews wird daher, das Produktspielfeld und der Startpunkt für TYPO3 am CMS-Markt, untersucht. Um das Produktspielfeld, also mögliche Entwicklungsoptionen zu begrenzen, wurden Fragen zu Stärken und Schwächen von TYPO3 und zur Produktvision gestellt. Mithilfe der PESTEL-Analyse, sollten anschließend externe und interne Zukunftsfaktoren identifiziert werden, welche für Innovationsdruck bei TYPO3 sorgen. Bezüglich dieser Einflüsse wurde wiederum erfragt, welche notwendigen Entwicklungen sich daraus für das CMS ableiten. Mit Blick auf die künftige Produktstrategie wurden nach möglichen Zielstellungen für die Weiterentwicklung gefragt. Im vierten Teil wurden die Kriterien der MACH-Architektur zur Diskussion gestellt, um deren Potenzial und Risiken als Weg für die Erreichung der Ziele zu erörtern. Abschließend sollten die Teilnehmern noch einem Ausblick auf künftige Entwicklungen geben und konnten fehlende Aspekte ergänzen.

4.2.3 Methodik zur qualitativen Datenanalyse

Die Interviews wurden mit dem Online-Videokonferenzsystem BigBlueButton aufgezeichnet. Das Audio wurde anschließend mit der Transkriptionsfunktion von Microsoft Word automatisch in Textform gebracht und anschließend durch den Vergleich von Aufnahme und generiertem Text korrigiert. Da der Fokus beim Experteninterview auf der Erhebung von Fakten- und Erfahrungswissen liegt (Helfferich, 2019), wurde Verbales und Non-Verbales transkribiert, sofern es für Verständnis und Thema relevant war. Sätze, die dem Nachdenken geschuldet ungeordnet oder grammatikalisch falsch artikuliert wurden, sind zugunsten der Verständlichkeit korrigiert worden.

Für die Auswertung des Interviews wurde die von Misoch vorgeschlagene typologisierende Analyse angewendet. Nach der Transkription folgt dabei die inhaltsgetreue Paraphrase relevanter Textpassagen und im nächsten Schritt

deren Codierung, wobei jede relevante Textpassage einen Code erhält und dadurch einem Thema zugeordnet wird. (siehe Abbildung 13: Tabelle oben: Identifizierung von Themen in Interviews; Tabelle unten Strukturierung nach Themen .



Fall	Teilthemen (T 1–6)								
F 1	T 2	T 5	T 2	T 1	T 6	T 3	T 4	T 3	T 6
F 2	T 1	T 6	T 4	T 2	T 5	T 4	T 3		
F 3	T 5	T 1	T 6	T 1	T 4	T 3	T 2	T 6	T 4	T 2
F 4	T 4	T 5	T 2	T 1	T 6	T 3			
F 5	T 6	T 2	T 1	T 3	T 5	T 4	T 5	T 2	

Teilthema	Fälle (1–5)				
T 1	F 1	F 2	F 3	F 4	F 5
T 2	F 1	F 2	F 3	F 4	F 5
T 3	F 1	F 2	F 3	F 4	F 5
T 4	F 1	F 2	F 3	F 4	F 5
T 5	F 1	F 2	F 3	F 4	F 5
T 6	F 1	F 2	F 3	F 4	F 5

Abbildung 13: Tabelle oben: Identifizierung von Themen in Interviews; Tabelle unten Strukturierung nach Themen (Misoch, 2019, S. 124)

Die übergeordneten Themen entsprechen den Schritten zur Festlegung einer Produktstrategie nach Becker, wie unter 4.1 beschrieben, nach denen sich auch der Interviewleitfaden richtet (Misoch, 2019). Die gewonnenen Informationen zur Produktstrategie von TYPO3 werden unter Punkt 4.3 gesammelt, interpretiert und reflektiert. Beispielhaft wurde für den Verweis auf die Interviews in der Analyse, statt der Langform: (Anhang 3, Interview A), die Kurzform (A) Verwendet.

4.2.4 Expertise der Interviewpartner

Alle Teilnehmer haben langjährige Erfahrung in der Umsetzung von Projekten mit TYPO3 gesammelt und sind in hohem Maß in der Community aktiv. Den Teilnehmern wurde aufgrund ihrer Erfahrungen und aktuellen Tätigkeit eine Spezialisierung zugeordnet.

Spezialisierung	Kernentwicklung	Agentur	Hosting
Teilnehmer	A, B	C, D	E, F

4.2.5 Produktspielfeld

Die Vision beschreibt allgemein das Ziel eines Unternehmen und die Mission den Weg dorthin. Vision und Mission sind ideeller Natur und liefern einen ersten

Anhaltspunkt für das Produktspielfeld (Becker, 2023, S. 53). TYPO3 möchte mit seinem Angebot ermöglichen, dass jeder Mensch in seiner Einzigartigkeit sein Potenzial erreichen kann. Die Mission ist nachhaltige, erstklassige Open Source Content Management Software anzubieten, die das ermöglicht. Die Software muss daher Wissensaustausch, Zusammenarbeit, Freiheit und Vernetzung fördern, bezahlbar, zugänglich und einfach zu nutzen sein und über ein Community verfügen, die offen ist und zum Beitragen einlädt (TYPO3 Association, o.J. c).

Nach Becker stellt sich für das Produktspielfeld die Frage: „Für welche Zielgruppe(n) und in welchem Kontext lösen wir mit welcher Priorität welche Probleme, um für die Zielgruppe(n) welche Ziele zu erreichen?" (Becker, 2023, S. 54).

TYPO3 ist seit langer Zeit ein kostenfreies Open Source CMS, welches von drei Säulen getragen wird. Es wird von der Community für die Community entwickelt. Die TYPO3 Association ist ein nicht-gewinnorientierter Verein, welcher die Entwicklung koordiniert und finanziell unterstützt und von den Beiträgen seiner Mitglieder getragen wird. Die TYPO3 GmbH ist ein gewinnorientiertes Tochterunternehmen der TYPO3 Association, welche Support und verschiedene Dienstleistungen für das CMS anbietet und ebenfalls die Entwicklung des CMS vorantreibt. Aktuell arbeiten alle drei Partner (Community, Association und GmbH) an einer gemeinsamen Produktstrategie (A). Um langfristig gesteckte Entwicklungsziele erreichen zu können, hat die TYPO3 Association begonnen Mittel für die Finanzierung über einen Release-Cycle hinaus zu akquirieren (C).

Von den Interviewpartnern, wurden einstimmig mittelgroße bis große Unternehmen, sowie Behörden und Institutionen als oberste Zielgruppe für das Produkt TYPO3 genannt (1A-1F). Das spiegelt sich im Entwicklungsmodell von TYPO3 wider, welches Priorität auf die Entwicklung moderner und sicherer Software zur Bereitstellung und Verwaltung komplexer Webseiten legt, in dem möglichst aktuelle Versionen externer Komponenten und Bibliotheken verwendet werden (B, D, F). Der Vision von TYPO3 folgend, nannte ein Kernentwickler (B) und ein Hosting-Anbieter (E) auch kleine Instanzen, beziehungsweise alle Menschen, einen wichtigen Bestandteil der Zielgruppe, da sie der Community Nachwuchs brächten (B) und die Verbreitung des Systems fördern (E).

Grundlegender Bestandteil des Produktspielfeldes wird auch in Zukunft die On-Premise Version sein (A, B, C, F), die es den Kunden ermöglicht im Vollbesitz ihrer Daten (A, C) und unabhängig von externen Cloud-Anbietern zu sein (A). Das deckt sich mit der Mission Freiheit zu fördern. Ein Teilnehmer wies auch darauf hin, dass Unternehmen mit strengen Compliance-Anforderungen, zum Beispiel aus der Finanz- oder Medizinbranche, sogar auf On-Premise Versionen angewiesen seien (F). Eine offizielle Cloud-native SaaS Lösung sei aktuell keine Zielstellung, die verfolgt werde seitens der Kernentwickler, die sich als Teil der Community verstehen (A, B).

4.2.6 Startpunkt

4.2.6.1 Marktposition

TYPO3 steht vor allem in Konkurrenz zu anderen PHP-basierten Open Source CMS (D), wie man auch in Abbildung 14 erkennen kann, wechseln ursprünglich TYPO3-basierte Webseiten mit großer Mehrheit auf WordPress, aber auch auf Contao, Drupal oder Joomla!. Ein Hosting-Experte verweist darauf, dass TYPO3 in der Vergangenheit noch vermehrt für kleine Webseiten eingesetzt wurde und viele dieser Seiten mit der Zeit auf WordPress oder NoCode-Baukastenlösungen, wie von WIX umgestellt würden, womit sich auch die Zahlen in Abbildung 14 erklären ließen (F). Der zweite Hosting-Experte sieht das Marketing als Enterprise Content Management System dafür verantwortlich, dass Nutzer im Bereich kleiner und mittlerer Unternehmen verloren gingen, obwohl TYPO3, seiner Meinung nach Potenzial hätte, in diesem Bereich erfolgreich zu sein (E). Ein anderer Experte befindet hingegen, dass TYPO3 von Beratern nicht für die Umsetzung kleiner Webseiten empfohlen werden sollte (D).

PHP-basierte Open Source CMS werden trotz der Konkurrenzsituation überwiegend als Partner wahrgenommen, die sich in ihrer gegenseitigen Entwicklung unterstützen (A, B, D). Ein Zeugnis dessen ist auch die kürzlich gegründete Open Website Alliance der Systeme WordPress, Drupal, TYPO3 und Joomla!, die sich für den Einsatz von Open Source Systemen statt proprietärer Systeme einsetzen will (TYPO3 Association, 2024b). Der Markt sei ohnehin so groß, dass es kaum zu echten Konkurrenzsituationen unter den Mitglieder der Allianz komme (B, C), auch weil die Mitglieder der Allianz alle ihr eigenes Entwicklungsmodell haben (B). WordPress zum Beispiel verfolge nicht das Ziel technologisch immer auf dem neuesten Stand zu sein, wie das bei TYPO3 der Fall

sei (B). TYPO3 profitiere im deutschen Markt von seiner historisch gewachsenen Stärke. Dank des großen qualitativen Angebotes an Dienstleistern sehe man sich auch einer stabil hohen Nachfrage gegenüber. (C).

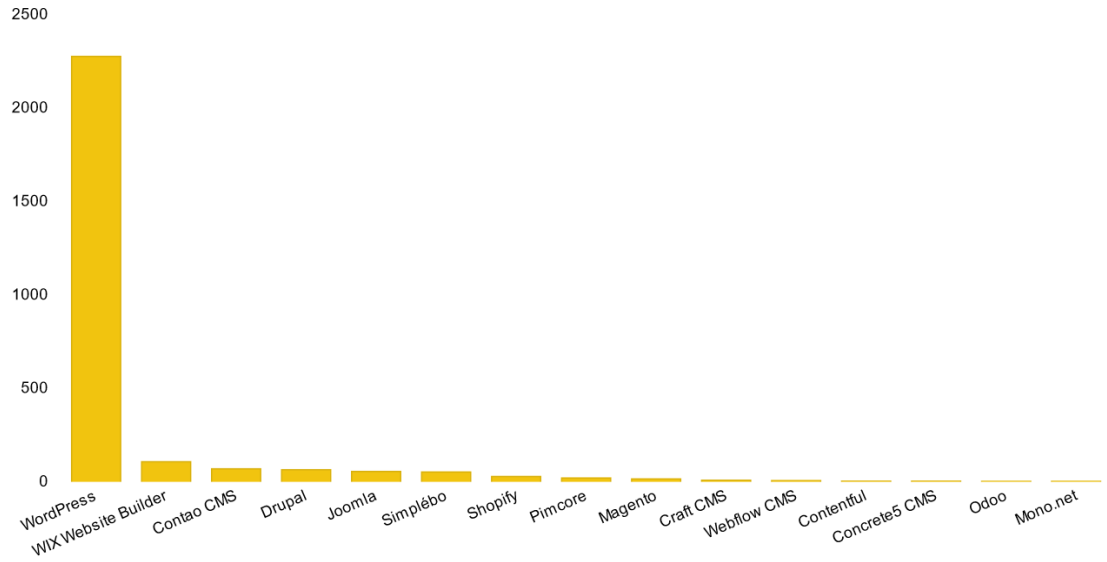


Abbildung 14: Zu welchen bekannten CMS sind TYPO3-Webseiten zwischen dem 10.09.2023 und 08.01.2024 gewechselt (Hansen, 2024)

Die MACH-basierten CMS werden bisher überwiegend nicht als Konkurrenz wahrgenommen, was verschiedene Gründe hat. Zweimal wurde genannt, dass die Kunden im Besitz ihrer eigenen Daten bleiben wollen, was nur mit einer On-Premise Lösung sichergestellt werden kann (A, C). MACH-basierte Systeme würden eher Kunden ansprechen, die ihre Infrastruktur ohnehin schon vorwiegend in der Cloud betreiben würden (F), wobei es da mittlerweile auch schon negative Erfahrungen gebe, sodass Kunden zu On-Premise-Produkten zurückkehrten (B). Uneins ist man sich hinsichtlich des passenden Anwendungsfalls für MACH-basierte Systeme. Für kleine bis mittlere Auftritte ist MACH zu kompliziert (C, E). Für große, komplexe Systeme ergibt sich eine Schnittmenge mit der Zielgruppe von TYPO3, vor allem in technisch komplexen Projekten, wo auch Headless (C, F) und die Bespielung mehrerer Kanäle nachgefragt werden (F), allerdings lassen die Lizenzmodelle bei MACH-basierten Systemen dann schnell die Kosten explodieren, weil nicht nur nach Traffic, sondern auch nach Anzahl der verwendeten Content Types abgerechnet werde (A). Ein Teilnehmer ist der Meinung, dass MACH seine Stärken lediglich beim Einsatz in Großunternehmen hat. (E).

Für die Zukunft ist man sich unter den Interviewteilnehmern mehrheitlich sicher, dass MACH-basierte CMS und traditionelle CMS nebeneinander existieren werden, da sie sich auf bestimmte Anwendungsfälle spezialisieren werden (C, F). Beide Arten von Systemen können voneinander lernen und werden sich einander annähern (C). Mittlerweile haben zum Beispiel viele MACH-basierte CMS eine Preview-Funktion implementiert, die es zuvor nur bei traditionellen CMS gab (A).

4.2.6.2 Bisherige Relevanz MACH für Interviewpartner

Die Erfahrungen der Teilnehmer in Bezug auf das Thema MACH sind sehr unterschiedlich. Für einen Interviewteilnehmer, der sich auf CMS-Projekte mit TYPO3 spezialisiert hat, hatte MACH bisher keine Relevanz bei seiner Arbeit (D). Ein Entwickler verband mit dem Thema verschiedene Tools, die API-First basiert sind und in der Kernentwicklung zum Einsatz kommen, wie zum Beispiel GitLab (B). Der Geschäftsführer einer Agentur bietet neben TYPO3 mittlerweile auch StoryBlok, ein CMS, welches Mitglied der MACH Alliance ist, an, um sich neue Geschäftsfelder zu erschließen. Er vermutet, dass die Entwicklung in Richtung API-First und Cloud-native bisher keinen großen Stellenwert bei TYPO3 annehmen konnte, da es nicht zentral, sondern von der Community entwickelt wird (C). Der zweite Kernentwickler verweist darauf, dass REST- und graphQL-APIs schon vor vielen Jahren mit TYPO3 umgesetzt wurden, um Content als JSON auszugeben. Cloud-basiertes Hosting und Headless würden auf Kundenwunsch schon seit Jahren immer mal wieder umgesetzt (A, B). Auch einige Kunden eines Hosting-Expertem hätten sich schon vor Jahren für den API-First Ansatz entschieden. Er habe zudem verschiedene Lösungen zum Thema im Portfolio seiner Firma, zum Beispiel TYPO3 SaaS, TYPO3 Headless und für letzteres passende Templates für die Frontend-Entwicklung mit VueJS (E). Der zweite Hosting-Experte hat die Entwicklung seines Unternehmens vom traditionellen Host-Anbieter hin zur Cloud-Plattform mit vorangetrieben. API-First bzw. Headless-CMS seien schon öfter von seinen Kunden angefragt worden, da mittlerweile häufiger die Anforderung besteht verschiedene Kanäle mit den Inhalten zu bespielen (F).

4.2.6.3 Stärken und Schwächen gegenüber MACH-basierten Systemen

Zu den großen Stärken von TYPO3 zählt, dass es seit über 25 Jahren erfolgreich auf dem Markt ist (C) und dadurch zu einem ausgereiften Content Management System heranwachsen konnte, welches die meisten Kundenanforderungen nativ erfüllt. Der mitgelieferte Funktionsumfang ist größer als bei vergleichbaren CMS, sodass oft unnötiger Entwicklungsaufwand vermieden werden kann (A, C, E, F).

Bei den folgenden Features hat TYPO3 einen Wettbewerbsvorteil gegenüber MACH-basierten CMS: Multi-Domain-Hosting (A, B), Mehrsprachigkeit (D), ausgefeilte Rechteverwaltung (A, B, D), Workspaces (B, C) und Versionierung (C). Zudem verfügt das System über eine starke Community, die das CMS unter der Koordination der TYPO3 Association kontinuierlich renoviert, an Kundenanforderungen aus eigenen Projekten anpasst und neue Lösungen teilt (A, B, E). Die Erweiterbarkeit gehört dabei seit langem zur Stärke des Systems. Die Schnittstelle zur mächtigen Sprache PHP ermöglicht alle denkbaren Anpassungen. MACH-basierte Systeme haben keine solche direkte Schnittstelle zu einer Hochsprache (A, B, C).

Auch die geringen Kosten für eine Webseite auf Basis von TYPO3 wurden als Stärke genannt. Sechs bis acht Jahre bedarf es keiner Hauptversionswechsel und damit keiner teuren Wartungsarbeiten, was eine gute Planbarkeit mit sich bringt (C). Das Preismodell von MACH-basierten CMS werde hingegen regelmäßig angepasst (A). Generelle Vorteile von Open Source On-Premise Lösungen sind in dem Sinne auch die Unabhängigkeit von einem bestimmten Betreiber. Man bleibt im Besitz seiner Daten und kann ohne Probleme zu einem neuen Anbieter wechseln kann, wenn der bisherige sein Geschäft einstellt oder seine Preise erhöht (A, C, F). Auch die Plattformtechnologien sind kostenfrei und Open Source verfügbar, sodass es eine Vielzahl von Anbietern gibt, alternativ kann man das CMS auch auf dem eigenen Server betreiben (F). Ein Alleinstellungsmerkmal sieht ein Kernentwickler darin, dass das System durch seinen Fokus auf die Verwendung aktueller Technologie von Version zu Version schneller werde und damit weniger Ressourcen verbrauche. Bei MACH-basierten Systemen würden Architekturbedingt generell mehr Ressourcen verbraucht als bei TYPO3 (B). Im Unterschied zu MACH-basierten CMS beginnt man beim Content Design mit TYPO3 nicht bei null. Ein Kernentwickler sieht dabei die Gefahr, dass Strukturen von unerfahrenen Entwicklern angelegt und schlecht dokumentiert werden, sodass es im Vergleich zu TYPO3 schwierig sei, diese Systeme zu warten, was er zuletzt bei einem Kunden erlebt habe (A).

Zu den Schwächen von TYPO3 gegenüber MACH-basierten Systemen gehört, dass es ein Legacy-System ist, dass über viele Jahre gewachsen ist (B) und gerade in der Anfangszeit durch die Handschriften vieler unterschiedlicher Entwickler geprägt wurde, als der Entwicklungsprozess noch weniger zentral koordiniert wurde, als es heute der Fall ist (A). In der Folge gibt es noch an

einigen Stellen heterogenen Code und keine strikten Schnittstellen im Kern, was die Änderbarkeit verlangsamt und mit langfristigen Refactoring-Maßnahmen verbessert werden muss (A, B). Zusätzlich sind an einigen Stellen im Kern historisch bedingt noch eigene Lösungen implementiert, die schrittweise durch PHP-weite Standardbibliotheken ersetzt werden sollen, um den Wartungsaufwand für den Kern zu reduzieren (D). Auch der Prozess des semantischen Content Designs ist gegenüber MACH-basierten Systemen noch zu kompliziert und muss vereinfacht werden, da er wichtig für die Wiederverwendbarkeit von Inhalten ist, eine zunehmend wichtiger werdende Eigenschaft von CMS deren Bedeutung man auch bei TYPO3 erkannt hat (A).

4.2.7 Zukunftsfaktoren (PESTEL-Analyse)

Zukunftsfaktoren müssen nach Becker anhand ihrer Eintrittswahrscheinlichkeit in der Produktstrategie priorisiert werden (Becker, 2023). Im Interview wurden die Zukunftsfaktoren differenziert nach den Kategorien der PESTEL-Analyse erfragt, die bei Nagel näher beschrieben sind (Nagel, Mieke, & Teuber, 2020). Es wurde speziell danach gefragt, welche Zukunftsfaktoren für Innovationsdruck bei TYPO3 sorgen und damit für die Produktstrategie relevant sein könnten.

4.2.7.1.1 Politisch

Die Teilnehmer konnten keine politischen Zukunftsfaktoren nennen, was auf eine stabile politische Lage hindeutet.

4.2.7.1.2 Wirtschaftlich

Wirtschaftlich gesehen, konnten die Teilnehmer auch keine spezifischen Zukunftsfaktoren nennen, was wiederum auf eine stabile wirtschaftliche Lage und Nachfrage bei TYPO3 hindeutet. Der Markt sei groß genug (B). Bisher habe man sich immer den Bedürfnissen des Marktes angepasst (A).

4.2.7.1.3 Soziokulturell

Ein Hosting-Experte hält eine neue Generation von Entscheidern für möglich, die anspruchsvoller in Bezug auf CMS-Projekte werden. Diese würden öfter nach frei zugänglichen Schnittstellen, nach Microservice-Deployment und nach Headless fragen, um ihre Inhalte auf verschiedenen Geräten abrufen zu können (F).

Ein Kernentwickler beobachtet einen anderen Trend, dass sich Entscheider immer öfter aufgrund privater Kontakte für ein bestimmtes CMS entscheiden würden,

anstatt auf die Meinung eines Experten zu hören (A). Ebenfalls beobachtet er, dass Berufsanfänger oft nur JavaScript-Kenntnisse mitbringen, was das Interesse an CMS, die mit JavaScript Frontend-Frameworks harmonieren steigern könnte. Generell sei es wahrscheinlich, dass Berufsanfänger sich vorrangig für neue Technologien und Lösungsansätze interessieren, statt für bestehende (C).

Ein Agentur-Experte hält es für möglich, dass das negative Image, was die Konkurrenz von TYPO3 (teuer, monolithisch, kompliziert) kreiere sich negativ auf den Entwicklernachwuchs und die Kundschaft auswirken könnte (C). Der Agentur-Experte geht auch davon aus, dass mangelnde Expertise und Bereitschaft in der Umsetzung von Digitalisierungsprojekten dazu führen wird, dass trotz technischer Möglichkeiten keine vollständigen Lösungen umgesetzt werden (C).

4.2.7.1.4 Ökologisch-geografisch

Ein Kernentwickler nannte, dass es ihm ein wichtiges Anliegen ist, dass TYPO3 mit jeder neuen Version weniger Ressourcen pro Instanz verbrauche (B). Auch ein Agentur-Chef findet, dass das Thema Nachhaltigkeit an Bedeutung auf dem CMS-Markt gewinnen könnte (C).

4.2.7.1.5 Technologisch

Mit hoher Wahrscheinlichkeit wird die schnelle Anpassungsfähigkeit von TYPO3 in Zukunft eine noch größere Rolle spielen, um dem eigenen Anspruch gerecht werden zu können, ein modernes, schnelles und sicheres CMS zu sein, welches technologische Neuerungen schnell integriert. TYPO3 verwendet Standardbibliotheken, die regelmäßig in ihrer aktuellen Version integriert werden müssen (B, D, E). ,

Die Mehrheit der Interviewteilnehmer hält eine wachsenden Nachfrage nach Headless-fähigen CMS für wahrscheinlich (A, B, C, E, F). Zudem werde die Integration mit vorhandenen Systemen wichtiger, da mittlerweile, anders als vor 20 Jahren, bereits meist eine gewachsene IT-Infrastruktur in Unternehmen vorhanden sei, in die man sich einfügen müsse (A, C). Auch die Bedeutung der Wiederverwendbarkeit von Content durch semantische Strukturierung und Ausgabe über plattformunabhängige Schnittstellen werde zunehmen (A, C). Eventuell werde künftig kein Content mehr in TYPO3 produziert, da schon jetzt Inhalte häufig auf anderen Plattformen, wie z.B. Google Docs kollaborativ entstünden und dann nur noch in das CMS eingefügt werden (C).

Die Hosting-Experten sehen auch eine zunehmende Bereitschaft von Kunden ihre IT-Dienste in die Cloud zu verlagern, wodurch Cloud-kompatible Lösungen mit nativer Unterstützung Cloud-spezifischer Funktionalität, wie beispielweise elastischer Skalierung, an Bedeutung gewinnen könnten (E, F). Ein Kernentwickler hingegen hält eine Rückkehr von Kunden zu On-Premise Lösungen für möglich, da er bereits erste Kunden habe, die schlechte Erfahrungen mit der Verlagerung ihrer IT-Dienste in die Cloud gemacht haben (B).

4.2.7.1.6 Rechtlich

Als rechtlicher Zukunftsfaktor wurde die gesetzliche Verpflichtung zur Barrierefreiheit genannt, die ab 2025 auch für Nicht-Regierungsorganisationen gilt und bereits einen umfassenden Entwicklungsprozess im Usability-Bereich von TYPO3 angestoßen hat, der weiter von Bedeutung sein wird (D). Der Cyber Resilience Act oder das Online-Zugangs-Gesetz wurden entgegen den Erwartungen nicht genannt.

4.2.8 Mögliche Ziele

Aus Produktspielfeld, Startpunkt, Zukunftsfaktoren und den Antworten der Teilnehmer lassen sich Entwicklungsziele für TYPO3 ableiten. Sie bleiben teilweise hypothetisch, da die offizielle Produktstrategie noch nicht veröffentlicht wurde. Ziele sollten Ursache und Wirkung klarstellen und messbar sein. Gute Ziele sollten einen nachhaltigen Wettbewerbsvorteil anstreben (Becker, 2023).

Die kurzfristigen Ziele sind bei TYPO3 im 18-monatigen Entwicklungszyklus von Hauptversion zu Hauptversion angesiedelt, und umfassen in der Regel die Integration aktueller Versionen von PHP und der extern verwendeten Bibliotheken, bspw. Doctrine DBAL oder Symfony (B, D). Zudem fokussiert man sich auf ein bestimmtes Leitthema bei der Priorisierung neuer Features. Die Erreichung dieser Ziele soll TYPO3 seinen Wettbewerbsvorteil, als technologisch modernes CMS sichern, das sicher ist und von Version zu Version immer schneller wird (B), dabei soll der Spagat zwischen einfachem und mächtigem System erhalten bleiben (A).

Ein langfristiges Ziel von Entwicklerseite, welches über einen Zeitraum von etwa fünf Jahren erreicht werden soll, sei, laut den Kernentwicklern und einem

Agentur-Experten, die weitere Entkopplung, Modularisierung und sinnvolle Kapselung im Kern des CMS, was für eine verbesserte Änderbarkeit und Wartbarkeit sorgen und den Integrationsaufwand neuer Versionen von Drittanbieter-Bibliotheken verringern wird (A, B, D). Die Zeitersparnis könnte in neue Features investiert werden und so für einen Wettbewerbsvorteil sorgen.

Zusätzlich hat die TYPO3 Association zuletzt im Rahmen einer Analyse festgestellt, dass der Produktentwicklungsprozess stärker mit Blick auf die Bedürfnisse von Anwendern und Erwartungen von Entscheidern gesteuert werden muss (C). Ein langfristiges Entwicklungsziel diesbezüglich ist die Verbesserung der Usability, die Anwendern und Integratoren den Einstieg und die Arbeit mit TYPO3 erleichtern soll (B). Im Hinblick auf rechtliche Vorgaben und aus innerer Überzeugung, gemäß der Vision, ist es schon seit längerem das Ziel TYPO3 barrierefrei zu gestalten, wobei hier schon wichtige Meilensteine erreicht wurden (D). Das semantische Content Design soll zudem vereinfach werden (A).

Ein anderes Ziel mit Blick auf die Entscheider ist die Steigerung der nativen Interoperabilität von TYPO3, beispielsweise durch die Bereitstellung weiterer nativer Schnittstellen und Ausgabeformate (A, C, E). Ein Hosting-Experte ist der Meinung, dass diese Qualität in Zukunft wichtiger ist, als die Fähigkeit eines Systems alles zu können (E). Skalierbarkeit gehört ebenfalls zu den Anforderungen in anspruchsvollen Projekten mit TYPO3 (A, F), daher gibt es Bestrebung diese zu erleichtern (A).

Ein Hosting-Experte hielte es für ein gutes Ziel die Anzahl der aktiven TYPO3 Instanzen durch geeignete Maßnahmen zu steigern. Zum Beispiel durch die Stärkung weiterer Anwendungsfälle (Flexibilität). Da mehr Instanzen auch eine stärkere Community zur Folge hätten (E).

Ein Hosting-Experte würde auch Lösungen zur vereinfachten Bereitstellung und Wartung von TYPO3 begrüßen, um den zeitlichen Aufwand diesbezüglich zu verringern (F).

4.2.9 Weg: Potenzial und Risiken der MACH-Architektur hinsichtlich der Ziele

4.2.9.1 Microservices

Die Zerlegung des TYPO3-Kerns in Microservices, die auf Netzwerkebene miteinander kommunizieren, bietet aktuell kein Potenzial, um Entwicklungsziele zu erreichen und wird daher auch nicht verfolgt (B, D). Aus Sicht des Kunden ergibt sich kein direkter Mehrwert, da er die gleiche Funktionalität erwartet, die das System bereits zuvor geliefert hat. Es ergeben sich lediglich Vorteile aus Entwicklungs- und Betriebssicht, wie das unabhängige Entwickeln und Skalieren einzelner Komponenten. Um den Funktionsreichtum von TYPO3 zu erhalten, wäre dafür aber ein langwieriger und kostspieliger Transformationsprozess notwendig, um das vorhandene System nachzubilden (A, B, F), der mit neuer Komplexität im System erkauft werden müssten. Dazu gehören das Beherrschbarmachen der durch die Netzwerkkommunikation entstehenden erhöhten Latenzen und das notwendige Entwickeln von Resilienzstrategien für den Ausfall einzelner Dienste. Der Transformationsprozess ist anspruchsvoll und in der Theorie nur für Individualsoftware beschrieben, die nicht millionenfach in unterschiedlichen Installationen existiert, wie das bei TYPO3 der Fall ist, daher wäre es mit weniger Risiko verbunden, ein solches Projekt bei null zu beginnen, als das bestehende System zu überführen (F).

Die Zerlegung in Microservices ist in der Theorie auch oft organisatorisch motiviert, wobei man eine komplexe Fachdomäne in abgrenzbare Bereiche aufteilen möchte (F). Die Strukturkomponenten von TYPO3 entsprechen aber bereits den typischen Expertisen (Backend, Frontend, Inhalte) und Strukturen (IT-Administration, Anwendungsentwicklung, Marketing) beim Kunden (C).

Auch im Hinblick auf das Ziel ein nachhaltiges CMS anzubieten, dass von Version zu Version weniger Ressourcen verbraucht, vermutet ein Kernentwickler, dass Microservices der falsche Weg sind. Die Bedingung, dass alle Dienste über das Netzwerk kommunizieren müssen erzeuge, allein durch das zusätzliche Verpacken und Entpacken der Informationen einen Mehrverbrauch.

Bezüglich einer vereinfachten Bereitstellung von TYPO3 in der Cloud inklusive nativer Skalierbarkeit würde die Transformation des Kerns hin zur Microservice-Architektur keine bedeutenden Vorteile gegenüber der vorhandenen Architektur

und nativen Konfigurationsmöglichkeiten bringen, die TYPO3 bereits zu einem sehr leistungsfähigem CMS machen, welches Millionen paralleler Anfragen bearbeiten kann. Die Verwendung von CDN und Caching seien viel effizientere Mittel zur Skalierung bei zumeist ohnehin statisch Inhalten (A, B, C). Bereits jetzt lässt sich TYPO3 durch die Trennung in Datenbankserver, Webserver, Backend und Frontend gut in kleinere Stücke zerlegen und in separaten Container bereitstellen (B, D) Typische Lösungen umfassen mehrere Frontend-Nodes und die Verwendung skalierbarer Datenbanken. Zur Realisierung zustandsloser Prozesse gemäß der Twelve-Factors und der read-only Policy in Containern wurden bereits Pfade im Core offengelegt, um das Caching entweder auf Node-Ebene oder zentral konfigurieren zu können (B).

Aus Hosting-Sicht wäre eine weitere Zerlegung in Microservices nicht nötig, da Skalierbarkeit für viele Kunden nicht relevant sei (D, E, F), wenn doch, sei es oft zweckmäßig vertikal zu skalieren (F). Lösungen zur horizontalen Skalierung seien ebenfalls vorhanden, wobei man mehrere Instanzen des Kern in Containern verpackt parallel betreibe. Allerdings sei der Konfigurationsaufwand hierfür hoch, vor allem für die Koordination von Sitzungsdaten und Cachedateien. Ein gemeinsamer Standard für das Deployment von TYPO3, beispielsweise in Form eines offiziellen Container-Images und eines Helm-Charts für Kubernetes würden die Bereitstellung in der Cloud erleichtern (F).

4.2.9.2 API-First

Der API-First Ansatz birgt großes Potenzial die Änderbarkeit und Wartbarkeit von TYPO3 zu verbessern, beispielsweise wäre es dadurch möglich die Backendoberfläche komponentenbasiert zu gestalten (A) oder alternative Backends anzubieten (C), was wiederum auch die Usability hinsichtlich unterschiedlicher Nutzergruppen verbessern könnte.
Die Kernentwicklung verfolgt daher das langfristige Ziel alle Backend-Funktionen über eine einheitliche plattformübergreifende API, verfügbar zu machen, sodass Frontend und Backend auch auf getrennter Hardware betrieben werden können. Datenbankoperationen sollen zukünftig also nicht nur über HTML-Templates möglich sein (A). Allerdings birgt dieser Ansatz die Schwierigkeit, dass Alleinstellungsmerkmale von TYPO3, wie die komplexe Authentifizierungs- und Berechtigungsschicht 1:1 aufwendig und langwierig nachgebildet werden müssen (A,B), während die PHP-API ebenfalls gepflegt werden muss, da sie weiterhin die Grundlage von TYPO3 bleiben soll (A). Auch für die PHP-API besteht die

Zielstellung, sie im kontinuierlichen Refactoring-Prozess restriktiver und klarer, im Sinne des API-First Ansatzes zu definieren. Auch das wird wesentlich zur Verbesserung der Änderbarkeit und Wartbarkeit von TYPO3 beitragen (A,B).

4.2.9.3 Cloud-native SaaS

Beim Potenzial des Cloud-native SaaS Ansatz für die Erreichung der Ziele von TYPO3 ist man sich uneins unter den Interviewpartnern. Die Kernentwickler sehen wenig Potenzial (A, B), da die On-Premise-Lösung weiterhin die Basis von TYPO3 bleiben soll und nicht parallel das Ziel einer Cloud-native-SaaS-Version verfolgt werden könne, zumindest nicht von Kernentwicklerseite. Die Erweiterbarkeit über die PHP-API sei ein essenzielles Feature (A). Sie steht im Einklang mit der Mission von TYPO3 und würde bei einer offiziellen SaaS-Lösung wegfallen oder nur noch eingeschränkt zur Verfügung stehen (A). Eine Cloud-Lösung sei auch deshalb keine Zielstellung der Kernentwicklung, weil das Hosting in den Verantwortungsbereich der Agenturen falle (D). Ziel sei es eine On-Premise-Basisversion bereitzustellen, aus der von Agenturseite bei Bedarf robuste SaaS-Lösungen mit speziellen Features entwickelt werden können. (B) Ein SaaS-Geschäftsmodell funktioniere nur, wenn es sich um proprietären Code handle, da sonst kein Wettbewerbsvorteil entstehe (D). Zudem gebe es schon eine Reihe von Anbietern, die TYPO3 SaaS in der Cloud anbieten würden, inklusive vorkonfigurierter Templates zum erleichterten Einstieg (C). Mit Blick auf das Geschäftsmodell (Donut-Modell) anderer Open Source CMS wie WordPress und Drupal, deren wichtigste Kernentwickler erfolgreich Cloud-Hosting anbieten, ist es nicht ausgeschlossen, dass die, sich in Arbeit befindliche, Produktstrategie für TYPO3 auch Entwicklungen in diese Richtung vorsehen wird (C). Diese werden wahrscheinlich in der Verantwortung der TYPO3 GmbH liegen (A). Hierzu ist aber bisher nichts näheres bekannt (C).

Cloud-Hosting widerspräche im Moment der Mission von TYPO3 eine bezahlbare Content-Management-Lösung bereitzustellen, da das Hosting in der Cloud erfahrungsgemäß etwa zehn Mal so teuer sei, wie On-Premise (B). Man sei zudem den Preisänderungen der Anbieter unterworfen (A). Dazu kommt, dass auch der zur Verfügung gestellte Ressourcenüberschuss und der notwendige Netzwerkverkehr nicht zur Verbesserung der Nachhaltigkeit von TYPO3 beitragen würden (B).

Ein Hosting-Experte sieht jedoch Potenzial in einer niederschwelligen Cloud-native SaaS-Lösung, da von ihr Zugkraft und Vertrauenswürdigkeit ausginge, die sich positiv auf die Verbreitung von TYPO3 und die Stärkung der Community im Sinne der Vision auswirken könne (E). Von Hosting-Seite betrachtet bietet der Cloud-native Ansatz weitere Vorteile. Es gebe noch kleine wiederkehrende Schritte bei der Bereitstellung von TYPO3 in Containern, die gut automatisiert werden könnten, wenn die *Twelve Factors* konsequent berücksichtigt würden. Dazu gehört beispielsweise die Nutzung von Umgebungsvariablen zum Verweis auf Konfigurationsdaten, statt Dateien, die aufwendig vorinitialisiert werden müssten (F).

Cloud-native Anwendungen sollten, wie unter 2.5.3.5 erwähnt, Mehrmandanten-fähig sein, inklusive *self-service*-Prinzip, der strikten Trennung von Mandantendaten und dem Teilen von Hardwareressourcen. Diese Fähigkeit würde dem Nachhaltigkeitsziel von TYPO3 zugutekommen, da man als Betreiber weniger Ressourcen für die gleiche Menge an Kunden benötigen würde (F). Ein Hosting-Experte experimentiert in diese Richtung, da sich auch der Wartungsaufwand vermindern würde (E). Dieser Weg gehört nicht zur Strategie von TYPO3, da vermutlich sehr viel Bestandscode dafür geändert werden müsste (C), beziehungsweise eine Microservice-basierte Architektur dafür nötig wäre (B).

4.2.9.4 Headless

Der Headless-Ansatz steht für die Bereitstellung nativer plattformunabhängiger Standard-Schnittstellen, wie REST und GraphQL, zur Ausgabe und zur Verwaltung von Content und bietet damit Potenzial zielgemäß die Interoperabilität und Flexibilität von TYPO3 zu steigern (A, C, D). Tatsächlich wird in der Kernentwicklung daran gearbeitet, dass weitere Ausgabeformate vereinfacht nativ bereitgestellt werden können (A). Dem nativen Aspekt wird von den Hosting-Experten besondere Bedeutung beigemessen, da Kunden mit dieser Anforderung erst dadurch auf ein entsprechendes CMS aufmerksam würden. Ein Hosting-Experte vermutet, dass er vornehmlich Headless-Anfragen zu WordPress bekomme, da der Ansatz dort schon seit längerem nativ implementiert sei und aktiv beworben werde (F). Die Entscheider würden bei einer nativen Lösung weniger Umsetzungsrisiko fürchten, da deren langfristige Wartung gesichert sei (E, F).

Headless-Lösungen für TYPO3 werden von einigen Agenturen seit Jahren in Eigenentwicklung für die Kunden angeboten (A, B, D). Seit 2020 gibt es auch die Headless-Extension, die parallel zur Kernentwicklung von der offiziellen PWA-Initiative vorangetrieben wird (TYPO3 Association, o.J. d). Es sei aktuell unklar, ob die Extension in ihrer aktuellen Form in den TYPO3-Kern integriert würde. Die Lösung müsse, wie es bei allen Extensions der Fall sei, einen Stand erreicht haben, dass sie sich über mehrere Jahre hinweg im Kern gut warten lasse. Parallel zum Kern könne sich eine Extension schneller entwickeln, da sie nicht den strengen Regeln des Kerns unterworfen sei. Bereits vorab träfe man auf Anfrage von Extension-Entwicklern, Anpassungen im Kern, um die avisierte Integration zu erleichtern (B). Ein Hosting-Experte spricht sich dafür aus, dass die Headless-Lösung natives, aber auch optionales Kernfeature wird, da sie nicht für jeden Anwendungsfall gebraucht werde (E). Zwei Experten geben zu bedenken, dass das volle Potenzial von Headless hinsichtlich Interoperabilität und Austauschbarkeit erst erreicht werden könne, wenn ein offener Standard für CMS entwickelt würde. Auch die MACH-basierten Systeme hätten hier noch keine Initiative ergriffen, obwohl die MACH-Architektur ein hohes Maß an Interoperabilität verspreche (A, C).

4.2.9.5 Modular Implementation

Schon früh und kontinuierlich verfolgt TYPO3 das Kriterium der Modularität mit der Entkopplung von Frontend und Backend und der Einführung von Extensions. Im Kern findet ein kontinuierlicher Refactoring-Prozess mit dem Ziel loserer Kopplung und sinnvollerer Kapselung statt, sodass Bestandteile leichter entfernt, hinzugefügt, dokumentiert, gewartet oder ausgetauscht werden können (A, B, C). Ein Beispiel hierfür ist die von der Datahandler & Persistence Initiative angestrebte Integration der Extbase-Extension in den CMS Kern, was die Vereinheitlichung der bisher festverdrahteten unterschiedlichen Schnittstellen für Datenbankoperationen von Frontend, Backend und der Extbase-Extension ermöglichen würde, siehe Abbildung 15 (A).

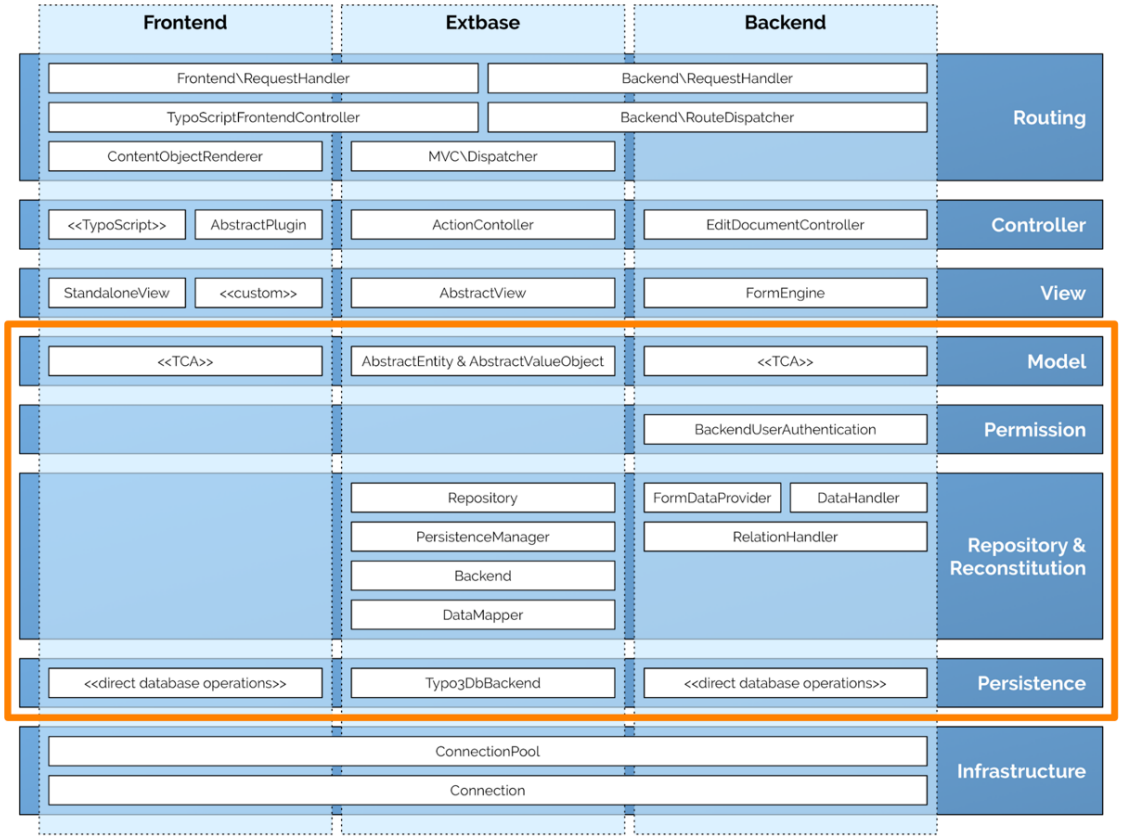


Abbildung 15: Verschiedene Schnittstellen für Datenbankoperationen bei TYPO3 (TYPO3 Association, o.J. e)

Ab Version 13 wird es auch optionale Kernfeatures geben (B, C). Composer ermöglicht es dann Minimalversionen des Kerns zu installieren, die nach dem Baukastenprinzip erweitert werden können. Die Modularität ist jedoch, anders als bei MACH-basierten Systemen, auf Anwendungsebene realisiert und es wird keine Modularität auf Netzwerkebene angestrebt, dadurch bleibt Kunden laut einem Agentur-Experten viel Lehrgeld erspart, um die zusätzliche Komplexität bei MACH, vor allem durch die Verwendung von Microservices (siehe Punkt 2.5.3.3.2 und 4.2.9.1) zu durchdringen, bevor sie die Vorteile der gewonnenen Modularität auf Netzwerkebene nutzen können (C).

4.2.9.6 Agility and Extensibility

Hinsichtlich der MACH-Kriterien zu Agilität und Erweiterbarkeit verfolgt TYPO3 keine konkreten Entwicklungsziele, da die Kriterien bereits nativ umgesetzt werden, wie unter 3.3 dargelegt. Lediglich, was die Ausweitung der Rückwärtskompatibilität betrifft, bestünde hier noch Potenzial (siehe 3.2). Allerdings würde das die Geschwindigkeit und die Weiterentwicklung von TYPO3 ausbremsen und nicht dem Produktspielfeld entsprechen, wie unter 4.2.5 dargelegt (B, D, F). Ein ausgereifter *Deprecation*-Prozess sorgt bereits für

Transparenz und Regelmäßigkeit (B, F). Betreiber können über 6-8 Jahre hinweg stabile Schnittstellen nutzen (C), was dem Ziel von TYPO3 nachhaltig und erschwinglich zu sein zugutekommt. Lediglich aus Sicherheitsgründen kann es in diesem Zeitraum (innerhalb einer LTS-Version) zu Breaking Changes der PHP-API kommen. Änderungen werden dabei sorgfältig unter allen Beteiligten abgestimmt (B). Bessere Nutzungs-Statistiken, wie sie SaaS-Anbieter zur Verfügung hätten, könnten die Entscheidungen erleichtern (A).

4.2.9.7 Scalability

TYPO3 ist bereits sehr gut über die nativen Caching-Optionen skalierbar. Um elastische Skalierbarkeit zu vereinfachen, gibt es langfristige Bestrebungen in der Kernentwicklung, dass man die Anzahl der Frontend-Nodes, die die Seite bereitstellen vereinfacht verändern kann, zusätzlich soll es durch die plattformunabhängige API leichter werden das Backend auf einer separaten Infrastruktur zu betreiben (A). Eine weitere Zerlegung des Kerns in Microservices ist nicht vorgesehen, wie unter 4.2.9.1 bereits dargelegt. Ein offizielles Container-Image sei kein Ziel der Kernentwicklung, es gebe aber Bestrebungen in der Community sich auf einen Standard für ein solches in Zukunft zu einigen (A).

4.2.9.8 Stability

Eine Entwicklung hinsichtlich der Stability-MACH-Kriterien könnte TYPO3 nur verfolgen, wenn auch der Betrieb im Rahmen einer offiziellen Cloud-native SaaS Variante verantwortet werden würde, daher sind diese Kriterien aktuell nicht relevant für die Produktstrategie von TYPO3.

4.2.9.9 Strategic Alignment

Die TYPO3 Association wird durch Mitgliedsbeiträge finanziert. Das Geschäftsmodell der Mitglieder, meist Agenturen oder selbständige Webentwickler, basiert teilweise oder ganz auf dem lizenzkostenfreien Einsatz von TYPO3 (TYPO3 Association, 2024b). Die Mitglieder haben also großes Interesse an der Pflege und Weiterentwicklung von TYPO3, um die Anforderungen in ihren Projekten effizient umsetzen zu können. In der Entwickler-Community werden Erfahrungen und Lösungen aus Kundenprojekten uneigennützig geteilt (A). Damit sich TYPO3 in Zukunft stärker an den Bedürfnissen von Entscheidern und Anwendern orientiert, hat die TYPO3 Association begonnen den Entwicklungsprozess stärker zu koordinieren. Aktuell wird, als Bestandteil dieser Initiative, auch eine offizielle Produktstrategie entwickelt. (C).

4.2.9.10 Documentation

Die Dokumentation von TYPO3 wird von Jahr zu Jahr besser (E). Jede Änderung am System wird dokumentiert und sofern es sich um Breaking Changes handelt, mit einem Migration Guide versehen. Deprecation Errors helfen eine Hauptversion lang nicht mehr unterstützte Codezeilen zu finden und zu aktualisieren (F). Die Dokumentation bietet offizielle Tutorials, z.B. für Installation und Deployment (E). Ein Experte sieht hinsichtlich der Dokumentation noch Potenzial für weitere Tutorials inklusive mehr Beispielcode, zum Beispiel für die Bereitstellung von TYPO3 in der Cloud (A). Zudem würden einheitlichere und restriktivere Schnittstellen im Kern zur Vereinfachung der Dokumentation beitragen (A).

4.2.9.11 Ease-of-Use

Eine vermehrte Kundenzentrierung ist erklärtes Ziel der TYPO3 Association. Es besteht Potenzial TYPO3 leichter erlernbar, zugänglich und nutzbar zu machen (A, E). Die Vereinheitlichung der Schnittstellen im Kern werde dazu beitragen, dass das System von Entwicklern leichter benutzbar werde (A). Die Stärke von TYPO3 komplexe Anforderungen umsetzen zu können, müsse erhalten bleiben, sodass das tiefgründige Erlernen von TYPO3 nicht wesentlich vereinfacht werden kann (D). Ein Hosting-Anbieter, der auch eine TYPO3-Agentur im nicht-deutschsprachigen Ausland betreibt, wünscht sich detaillierte problemlösungs-orientierte Videokurse mit realen Beispielen, wie die von Wolfgang Wagner, auch in englischer Sprache, um die Einarbeitung neuer Mitarbeiter zu beschleunigen. Diesen sei es mit den vorhandenen Materialien in der Regel nicht möglich TYPO3 autodidaktisch ausreichend zu erlernen, um damit eigenständig Probleme zu lösen (E).

Auch auf Anwenderseite nannten die Experten mögliche Verbesserungen hinsichtlich vereinfachter Nutzbarkeit. Das semantische Content Design soll durch die Integration der Content Blocks Extension in den Kern vereinfacht werden, sodass sich Integratoren viele Schritte beim Erstellen eigener Content Elemente (Content Types) sparen können (B). Auch alternative Backendoberflächen, die durch den API-First-Ansatz möglich würden, könnten die Erlernbarkeit und Nutzbarkeit je nach Erfahrung des Anwenders verbessern (E).

4.3 Auswertung

4.3.1 Validität

Die Interviews waren mit einer Dauer von 45 bis 60 Minuten verabredet. In diesem zeitlichen Rahmen ließen sich nicht jedem Teilnehmer alle Fragen des Interviewleitfadens stellen, da sich die Gespräche je nach Verlauf und Expertise des Experten auf bestimmte Aspekte konzentrierten. Ein Experte konnte zu MACH-bezogenen Themen keine Antworten geben. Zwei von sechs Teilnehmern (A, C) hatten bereits Erfahrung in der Umsetzung von Projekten mit MACH-basierten CMS gesammelt. Die Aussagen der restlichen Teilnehmer zu MACH-basierten CMS waren von weniger Expertise geprägt, sodass die Güte dieser Informationen geringer bewertet werden muss. Sie müsste durch weitere Interviews mit Experten, die beide Systeme sehr gut kennen verbessert werden. Nichtsdestotrotz hat sich gezeigt, dass die Teilnehmer mit den wesentlichen Konzepten der MACH-Architektur sehr gut vertraut sind, da die MACH-Kriterien schon viele Jahre vor der MACH Alliance Einfluss auf traditionelle CMS genommen haben und sich darin auch Projekte mit diesen Anforderungen umsetzen lassen. Folglich ist davon auszugehen, dass die Aussagen zu MACH-basierten Systemen relevante Expertise darstellen, auch wenn durch den großen Erfahrungsschatz der Teilnehmer im CMS TYPO3, der es ermöglicht CMS-Projekte unterschiedlichster Art und Schwierigkeit umzusetzen, davon auszugehen ist, dass die Vergleiche zwischen MACH-basierten und traditionellen CMS von einer gewissen Subjektivität geprägt sind.

Bei der geringen Teilnehmerzahl (6 Personen), lassen sich keine Aussagen ableiten, welche der Antworten von der Mehrheit der TYPO3 Community geteilt werden. Die Aussagen zur Produktstrategie entsprechen nicht der offiziellen Produktstrategie von TYPO3, die sich gerade in der Entwicklung befindet, sondern stellen die individuelle Sichtweise einzelner Experten aus der Community dar. Durch deren langjährige Erfahrung mit TYPO3, ist davon auszugehen, dass die gegebenen Antworten, besonders hinsichtlich ihrer Spezialisierung von Relevanz sind und eventuell auch in der offiziellen Produktstrategie von TYPO3 wiederzufinden sein werden. Als besondere Schwierigkeit hat sich, wie unter Kapitel drei herausgestellt, dass einige der MACH-Kriterien Interpretationsspielraum bieten.

4.3.2 Ergebnisse

Die Analyse der Interviews hat ergeben, dass die Experten vor allem in der nativen über 25 Jahre gewachsenen Funktionsvielfalt von TYPO3 einen Wettbewerbsvorteil gegenüber MACH-basierten CMS sehen. Sie stellt ein stabiles Fundament für die individuelle und schnelle Anpassung des Systems an unterschiedliche Anforderungen bereit, ein wichtiges Beispiel ist die ausgefeilte Benutzerrechteverwaltung (weitere siehe 4.2.6.3). Weitere Stärken sind die flexible Erweiterbarkeit des CMS, die Kosteneffizienz in CMS-Projekten, die Verwendung aktueller Technologien und die Befähigung des Kunden im Vollbesitz seiner eigenen Daten zu bleiben. Schwächen gegenüber MACH-basierten CMS hat TYPO3 vor allem durch heterogenen Code historischen Ursprungs im Kern, der die Wartbarkeit und Änderbarkeit des Systems erschwert. Bisher gibt es keine native plattformunabhängige Schnittstelle.

Die Analyse der Interviews hat auch ergeben, dass die MACH-Kriterien sowohl Chancen als auch Risiken für eine mögliche Produktstrategie von TYPO3 bereithalten.

Die Kriterien Strategic Alignment, Documentation und Ease-of-Use sind vorrangig mit Chancen hinsichtlich der Ziele verbunden. Hier besteht das Potenzial die Anzahl der Nutzer zu erweitern und das eigene Image attraktiv zu halten und zu verbessern. Die Risiken sind hierbei eher gering, da die Stärken des Systems bei Entwicklungen in diese Richtung erhalten bleiben können.

Das Kriterium Modular Implementation gehört zu den Grundfesten von TYPO3. Eine Stärke ist das Extension-Prinzip, welches die flexible Erweiterbarkeit gewährleistet. Für die Produktstrategie bietet sich hierbei das Potenzial, die API des Kerns einheitlicher und restriktiver, ähnlich einer REST-API zu gestalten, sowie den Kern hinsichtlich der Prinzipien lose Kopplung und sinnvolle Kapselung zu refaktorisieren, andernfalls bestünde das Risiko, dass sich die Wartbarkeit und Änderbarkeit der Software verschlechtert.

Bei allen anderen Kriterien sind die Chancen mit erheblichen Risiken verbunden, da umfassende und langwierige Änderungen am CMS-Kern notwendig wären,

sodass nicht absehbar ist, ob sich Anstrengungen und Investitionen in Zukunft auszahlen werden.

Bei den Kriterien Cloud-native SaaS und Stability überwiegt das Risiko für den TYPO3-Kern, da sich das System dafür zu einer reinen SaaS-Lösung entwickeln müsste. Die Mehrzahl der Teilnehmer sieht aber in der On-Premise-Lösung eine wesentliche Stärke von TYPO3. Sie sei der Grund, dass sich TYPO3 in keiner wesentlichen Konkurrenzsituation zu MACH-basierten Systemen befindet, da viele Kunden großen Wert auf den Besitz ihrer eigenen Daten lägen oder sogar darauf angewiesen seien. Eine SaaS-Lösung könne nur ein modifizierter Ableger des Kerns sein.

Aussichtsreich erscheinen die Kriterien Headless und API-First, da die Schnittmenge der Anwendungsfälle zwischen MACH-basierten CMS und TYPO3 bei anspruchsvollen Enterprise Projekten liegt, bei denen diese Anforderungen zunehmend eine Rolle spielen. Hier haben MACH-basierte Systeme einen Wettbewerbsvorteil, da sie diese Anforderungen nativ erfüllen können. Das Risiko besteht hier aber darin, dass es sehr aufwendig ist, den Funktionsreichtum der PHP-API in einer plattformunabhängigen API abzubilden und diese parallel zu warten. Noch dazu wird die Nachfrage nach TYPO3 aktuell nicht durch die wachsende Zahl von MACH-basierten beziehungsweise von Headless CMS gedämpft, sodass diese Entwicklung keine Dringlichkeit besitzt.

Ähnlich ist es beim Kriterium (elastic) Scalability, dass MACH-basierte Systeme zwar nativ anbieten, was aber in TYPO3-Projekten selten nachgefragt wird. Der Kern von TYPO3 ist bereits gut vertikal als auch horizontal skalierbar und spezialisierte Agenturen verfügen über das nötige Wissen dafür, dass Sie uneigennützig in der Community teilen, um diese Anforderung zu erfüllen. Ein aufwendiger Umbau des Kerns hinsichtlich dieses Kriteriums würde also wenig Gewinn versprechen.

Auch Microservices haben kein Potenzial hinsichtlich der Produktstrategie. Sie bringen eine erhöhte Komplexität mit sich, die für die Mehrzahl der Anwendungsfälle, für die TYPO3 nahezu schlüsselfertige Lösungen mit geringem Anpassungsaufwand anbieten kann, keine Vorteile mit sich bringt. Skalierbarkeit und Modularität löst TYPO3 bereits auf andere Art und Weise. Auch in Bezug auf die Nachhaltigkeit der Software würden sich Microservices nachteilig auswirken,

da sie mit einem höheren Energieverbrauch einhergehen. Bisher haben sich Microservices nur bei sehr großen Webanwendungen, wie Netflix und Amazon bewährt. TYPO3 i

Beim Kriterium Agility und Extensibility ergibt sich kein Entwicklungspotenzial, da eine klar definierte und nachhaltige *Deprecation*-Policy diesen Aspekt sehr gut regelt.

5 Letztes Kapitel

5.1 Fazit

Die Analyse im dritten Abschnitt dieser Arbeit hat gezeigt, dass die Kriterien der MACH-Architektur bereits Einfluss auf die Entwicklung von TYPO3 ausgeübt haben, auch schon bevor es zur Gründung der MACH Alliance kam. Einige der Kriterien werden bereits nativ erfüllt, andere zumindest teilweise oder durch die Lösungen von Drittanbietern.

Unter den MACH-Kriterien finden sich auch relevante Ansatzpunkte für die zukünftige Produktstrategie von TYPO3, wie im vierten Teil dieser Arbeit festgestellt werden konnte. Die Experten verwiesen auf Chancen und Risiken, wobei sich herausgestellt hat, dass die native Umsetzung vieler MACH-Kriterien mehr Risiken als Chancen für die Entwicklung von TYPO3 birgt. Andere MACH-Kriterien bergen zudem kein Potenzial, da sie keine Vorteile gegenüber dem Ist-Zustand mit sich bringen.

Eine wichtige Erkenntnis ist auch, dass sich die MACH-Kriterien nicht als allgemeingültiger Benchmark für moderne Webanwendungen eignen, da sie Eigenschaften vorgeben, ohne den konkreten Anwendungsfall zu berücksichtigen. Softwarearchitektur sollte aber, wie unter 2.2.2 mit Verweis auf die Theorie beschrieben, Termine, Kosten, Funktionalität und Qualität in Einklang zu bringen. Die Organisationsstruktur des Kunden scheint dabei ein wichtiges Kriterium, ob die MACH-Architektur gewinnbringend eingesetzt werden kann. Die MACH-Kriterien liefern dennoch viele valide Anhaltspunkte zur Bewertung moderner Webanwendungen.

Eine generelle oder deutliche Überlegenheit der MACH-basierten Systeme gegenüber den traditionellen Systemen hinsichtlich der MACH-Kriterien, so wie es die Marketing-Materialien der MACH Alliance suggerieren, konnte nicht festgestellt werden. Großes Potenzial zeigt sich aber bei TYPO3 vor allem im Bereich Dokumentation und der erleichterten Benutzung für alle Nutzergruppen, was durch den Ausbau von kostenfreien und interaktiven Lern- und Schulungsmaterialien, sowie dem beständigen Ausbau der Dokumentation inklusive problemlösungsorientierter Tutorials erreicht werden kann.

CMS jeglicher Art profitieren von semantischer Content-Modellierung, unabhängig von der Art des CMS hängt die Qualität des Content-Modells von der Erfahrung der Entwickler und Benutzer ab. Bei MACH-basierten CMS ist das in besonderem Maße der Fall, da die Modellierung bei null beginnt. Traditionelle CMS bieten hingegen nativ eine grundlegende, erweiterbare Struktur, sodass der Prozess für unerfahrene Entwickler zwar weniger fehleranfällig ist, die Content-Modellierung dadurch aber auch leicht an Priorität verlieren kann.

5.2 Ausblick

Es bleibt abzuwarten, ob die MACH-basierten CMS den traditionellen CMS zur starken Konkurrenz erwachsen und die traditionellen CMS Nutzer an die MACH-basierten CMS verlieren. Bisher, nach über 10 Jahren der Koexistenz hat die Mehrheit der TYPO3-Experten keine Bedenken diesbezüglich, da der CMS-Markt ausreichend groß für alle sei.

Die Arbeit hat gezeigt, dass traditionelle CMS und MACH-basierte CMS voneinander lernen. Es bleibt zu hoffen, dass das Versprechen von Interoperabilität und Austauschbarkeit der MACH-Architektur gestärkt wird, indem man sich mit den traditionellen CMS auf einen gemeinsamen Standard für Headless-CMS-APIs einigt. Drupal hat hier bereits einen wichtigen Schritt mit der Adaption des JSON:API-Standards unternommen.

Die MACH-Kriterien schlagen vor, die Erlernbarkeit und die Qualität der Dokumentation an realen Problemstellungen zu testen. Für weitere vergleichende Betrachtungen von MACH-basierten und traditionellen CMS wäre es interessant im Rahmen einer quantitativen Studie zu ermitteln, wie lange Programmieranfänger ohne Erfahrung in beiden Systemen brauchen, um typische Webprojekte umzusetzen. Auch eine Analyse der Kosten für die Umsetzung in Systemen beider Art würde Entscheidern und Entwicklern eine wichtige Handreichung liefern, das richtige CMS für ihren Anwendungsfall auszuwählen.

Spannend wird auch, ob MACH-basierte CMS sich weiter entwickeln können, ohne dass einzelne MACH-Kriterien, wie zum Beispiel die Rückwärtskompatibilität, dem Fortschritt entgegenstehen.

Literaturverzeichnis

Balzert, H. (2011). *Lehrbuch der Softwaretechnik: Entwurf, Implementierung, Installation und Betrieb* (3. Ausg.). Heidelberg: Spektrum Akademischer Verlag.

Barker, D. (2016). *Web content management: Systems, features, and best practices.* Sebastopol, CA: O'Reilly Media.

Barker, D. (08. 02. 2017). *The State of the Headless CMS Market*. Abgerufen am 29. 02. 2024 von https://deanebarker.net/tech/blog/state-of-the-headless-cms-market/

Becker, C. (2023). Produktstrategie – Das Fundament des Produktmanagements. In S. Hoffmann (Hrsg.), *Digitales Produktmanagement* (2. Ausg., S. 49-64). Wiesbaden: Springer Gabler.

BuiltWith. (o.J.). *CMS Usage Distribution in the Top 1 Million Sites*. Abgerufen am 29. 02. 2024 von https://trends.builtwith.com/cms/

Burgy, P. (17. 07. 2020). *A brief history of the Content Management System*. Abgerufen am 03. 04. 2024 von https://opensource.com/article/20/7/history-content-management-system

Cloud Native Computing Foundation. (o.J.). *Cloud Native Glossary*. Abgerufen am 29. 02. 2024 von https://glossary.cncf.io/

Contentful. (o.J.). *API Changes*. Abgerufen am 29. 03. 2024 von https://www.contentful.com/developers/api-changes/

Dudenredaktion. (o. J. c). *traditionell*. Abgerufen am 29. 02. 2024 von Duden Online: https://www.duden.de/suchen/synonyme/traditionell

Fernando, C. (2023). Designing Next-Gen Enterprise Software Systems with Cloud-Native Architecture. In *Solution Architecture Patterns for Enterprise: A Guide to Building Enterprise Software Systems* (S. 269–311). Berkeley, CA: Apress. doi:10.1007/978-1-4842-8948-8_8

Fowler, M. (25. 03. 2014). *Microservices*. Abgerufen am 29. 02. 2024 von https://martinfowler.com/articles/microservices.html

Fremantle, P. (28. 05. 2010). *Cloud Native*. Abgerufen am 29. 02. 2024 von http://pzf.fremantle.org/2010/05/cloud-native.html

Google. (2024a). *Google Trends: Joomla TYPO3 Wordpress Drupal*. Abgerufen am 29. 02. 2024 von

https://trends.google.com/trends/explore?date=all&q=Joomla,TYPO3,Wordpress,Drupal&hl=de

Google. (2024b). *Google Trends: Headless CMS*. Von https://trends.google.com/trends/explore?date=all&q=headless%20CMS&hl=de abgerufen

Günther, A., & Pflüger, A. (2014). Nicht-funktionale Anforderungen - die heimlichen Stars. In C. Rupp, & SOPHISTEN (Hrsg.), *Requirements-Engineering und -Management* (6. Ausg., S. 267-298). München: Carl Hanser Verlag GmbH & Co. KG. doi:10.3139/9783446443136.012

Hansen, T. (09. 01. 2024). *TYPO3 Version Check*. Abgerufen am 25. 03. 2024 von https://www.t3versions.com/statistics-detail/22

Helfferich, C. (2019). Leitfaden- und Experteninterviews. In N. Baur, & J. Blasius (Hrsg.), *Handbuch Methoden der empirischen Sozialforschung* (S. 669–686). Wiesbaden: Springer Fachmedien Wiesbaden. doi:10.1007/978-3-658-21308-4_44

Hoffman, K. (2016). *Beyond the Twelve-Factor App.* Sebastopol, CA: O'Reilly Media.

Jablonski, S., & Meiler, C. (April 2002). Web-Content-Managementsysteme. *Informatik-Spektrum, 25*(2), S. 101-119. doi:20.500.12116/10294

JSON:API. (o.J.). Abgerufen am 20. 04. 2024 von https://jsonapi.org/

Kratzke, N. (2023). *Cloud-native Computing* (2. Ausg.). München: Carl Hanser Verlag GmbH & Co. KG. doi:10.3139/9783446479258

Lamnek, S. (2010). *Qualitative Sozialforschung.* Weinheim: Beltz Verlagsgruppe.

MACH Alliance. (Oktober 2023a). *Admissions Playbook.* Abgerufen am 29. 02. 2024 von https://register.machalliance.org/hubfs/Admissions%20Playbook.pdf

MACH Alliance. (13. 12. 2023b). *MACH Alliance Surpasses 100 Members as it Closes Out Milestone 2023*. Abgerufen am 29. 02. 2024 von https://machalliance.org/newsroom/mach-alliance-surpasses-100-members-as-it-closes-out-milestone-2023

MACH Alliance. (11. 07. 2023c). *MACH Maturity Assessment.* Abgerufen am 29. 02. 2024 von https://machalliance.org/content-hub/mach-maturity-assessment-whitepaper

Mack, B. (01. 03. 2021). Meet Benni Mack, TYPO3 Core Project Lead, Germany. *Application Podcast* . (J. A. McGuire, Interviewer) Abgerufen am 29. 02. 2024 von https://typo3.org/article/meet-benni-mack-typo3-core-project-lead-germany-application-podcast-s1e6

Marzo, L. (03. 2020). JoomCast. *016 - Chat with Luca Marzo, Secretary at Open Source Matters*. (C. Medaan, & T. Davis, Interviewer) Abgerufen am 20. 03. 2024 von https://open.spotify.com/episode/64xdDbC7H9yEQbYZB0YAwO

McCue, R. (25. 05. 2014). *JSON REST API: Version 1.0*. Abgerufen am 29. 02. 2024 von https://make.wordpress.org/core/2014/05/25/json-rest-api-version-1-0/

McKeever, S. (2003). Understanding Web content management systems: evolution, lifecycle and market. *Industrial Management and Data Systems, 103*(8/9), 686-692. doi:10.1108/02635570310506106

Misoch, S. (2019). *Qualitative Einzelinterviews* (2. Ausg.). Berlin Boston: Walter de Gruyter GmbH.

Nagel, M., Mieke, C., & Teuber, S. (2020). *Methodenhandbuch der Betriebswirtschaft.* Stuttgart: utb GmbH.

Neumegen, M. (20. 08. 2021). *Jamstack CMS: The Past, The Present and The Future*. Abgerufen am 29. 02. 2024 von https://www.smashingmagazine.com/2021/08/history-future-jamstack-cms/

Sauer, M. (29. 04. 2003). *Interview Portrait – Kasper Skårhøj – Was Open Source und Christentum gemeinsam haben*. Abgerufen am 29. 02. 2024 von https://phlow.de/moritz-mo-sauer/schreibt/2003-04-29-typo-3-interview-kaspar/

Shirer, M. (06. 07. 2023). *Worldwide Public Cloud Services Revenues Surpass $500 Billion in 2022, Growing 22.9% Year Over Year, According to IDC Tracker*. Abgerufen am 29. 02. 2024 von IDC: https://www.idc.com/getdoc.jsp?containerId=prUS51009523

Spörrer, S. (2009). Grundlagen und Abgrenzungen. In *Content Management Systeme: Begriffsstruktur und Praxisbeispiel* (S. 5–57). Wiesbaden: Springer Fachmedien Wiesbaden. doi:10.1007/978-3-658-24351-7_2

Stine, M. (2015). *Migrating to cloud-native application architectures* (2. Ausg.). Sebastopol, CA: O'Reilly Media.

Strekalova, Y. A., & Bouakkaz, M. (2022). Content Management System (CMS). In L. A. Schintler, & C. L. McNeely (Hrsg.), *Encyclopedia of Big Data* (S. 208–211). Cham: Springer International Publishing. doi:10.1007/978-3-319-32010-6_43

Team, F. B.-U. (2024). Microservices in the Cloud Native Era. In *Cloud-Native Application Architecture: Microservice Development Best Practice* (S. 1–

25). Singapore: Springer Nature Singapore. doi:10.1007/978-981-19-9782-2_1

Tremp, H. (2021). Einführung in die Softwarearchitektur. In *Architekturen Verteilter Softwaresysteme: SOA & Microservices - Mehrschichtenarchitekturen - Anwendungsintegration* (S. 1–18). Wiesbaden: Springer Fachmedien Wiesbaden. doi:10.1007/978-3-658-33179-5_1

TYPO3 Association. (05. 02. 2024b). *TYPO3 Association Co-Founds the Open Website Alliance*. Abgerufen am 29. 02 2024 von https://typo3.org/project/press/press-releases/open-website-alliance/english

TYPO3 Association. (o.J. a). *TYPO3 Release Notes*. Abgerufen am 05. 03. 2024 von https://get.typo3.org/release-notes

TYPO3 Association. (o.J. b). *TYPO3 Explained*. Abgerufen am 29. 02. 2024 von https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/Introduction/Index.html#overview

TYPO3 Association. (o.J. c). *The TYPO3 Project's Vision, Mission, and Purpose*. Abgerufen am 24. 03. 2024 von https://typo3.org/community/values/vision-mission-purpose/project

TYPO3 Association. (o.J. d). *Progressive Web Apps Initiative*. Abgerufen am 29. 03. 2024 von https://typo3.org/community/teams/typo3-development/initiatives/pwa

TYPO3 Association. (o.J. e). *Datahandler & Persistence Initiative*. Abgerufen am 29. 03. 2024 von https://typo3.org/community/teams/typo3-development/initiatives/persistence/

W3techs. (o.J.). *Historical yearly trends in the usage statistics of content management systems*. Abgerufen am 29. 02. 2024 von https://w3techs.com/technologies/history_overview/content_management/all/y

Wiggins, A. (2017). *The Twelve-Factor App*. Abgerufen am 29. 02. 2024 von https://12factor.net/

Wolff, E. (2018). *Microservices: Grundlagen flexibler Softwarearchitekturen* (2. Ausg.). Heidelberg: dpunkt. verlag.

Wordpress. (2024). *Milestones: The Story of WordPress.* Wordpress. Abgerufen am 29. 02. 2024 von https://wordpress.org/book/

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
CI/CD	Continuous Integration/Continuous Development
CDN	Content Delivery Network
ELTS	Extended Long-Term Support
KI	Künstliche Intelligenz
LAMP	Linux-Apache-MySQL-PHP
LTS	Long Term Support
SaaS	Software-As-A-Service
SLA	Service Level Agreement

Anhang 1 – Analyse-Tabellen

1.1 TYPO3

Tabelle 1: Quantitative Analyse zum Einfluss der MACH-Kriterien auf TYPO3

Kriterium	E	G	B	Begründung
K1: Decoupled				
K1a: Microservices	1/2	1	1/2	Multi-Core-Ansatz möglich: Mehrere Kerne kommunizieren über Webservices-API, Optionale Backend-Module ermöglichen Kern-Konfiguration hinsichtlich bestimmter Aufgaben[1]
K2: Cloud-native				
K2a: PaaS oder SaaS	1/2	1/5	1/10	PaaS und SaaS Angebote ext. Hosting-Anbieter[2]
K2b: Upgradefrei	1	1/5	1/5	Lange Zeit kein Upgrade notwendig durch: LTS-Version = 3 Jahre, keine Breaking Changes + kostenfreier Support + Optional kostenpflichtiger ELTS für zusätzliche 3 Jahre verfügbar[3]
K2c: automatische Updates	1/2	1/5	1/10	Halb-automatisierte Updates über Composer oder Backend[4] + Drittanbieter-Update Service[5]
K2d: automatische Upgrades	1/2	1/5	1/10	Tools wie das TYPO3 Rector Plugin[6] oder die Extension TYPO3 Console[7] bieten Unterstützung bei Automatisierung, Agenturen bieten Upgrade-Service[8] = Managed Service Model

[1] TYPO3 Association. (o.J.). TYPO3 Deprecation Policy. Abgerufen am 17.04.2024 von https://typo3.org/article/typo3-v130-the-oceans-calling
[2] T3-Planet. (o.J.). *TYPO3 SaaS*. Abgerufen am 05.03.2024 von https://t3planet.de/typo3-saas
[3] TYPO3 Association. (o.J.). *TYPO3 Deprecation Policy.* Abgerufen am 05.03.2024 von https://typo3.org/article/typo3-deprecation-policy
[4] TYPO3 Association (o.J.). TYPO3 Upgrade Guide. Abgerufen am 16.03.2024 von https https://docs.typo3.org/m/typo3/guide-installation/main/en-us/Minor/Index.html#minor
[5] Toujou (o.J.) Abgerufen am 16.03.2024 von https://www.toujou.de/websites/typo3-relaunch-und-updates/
[6] TYPO3-Rector. (o.J.) Abgerufen am 16.03.2024 von https://www.typo3-rector.com/
[7] Hummel, H. (26.06.2017) *TYPO3 Upgrades from command line.* Abgerufen am 19.03.2024 von https://insight.helhum.io/post/162277469210/typo3-upgrades-from-command-line
[8] B13. (o.J.). TYPO3 Upgrade Services. Abgerufen am 19.03.2024 von https://b13.com/de/loesungen/typo3-upgrades

K2e: Mehrmandanten-fähigkeit inklusive *self-service*, getrennter Datenbanken und Ressourcensharing	0	1/5	0	Multi-Site fähig, Keine getrennte Datenhaltung[9]
K3: Natively Headless and API Enabled				
K3a: REST	1/2	1/4	1/8	Teil der Headless Extension[10]
K3b: GraphQL	0	1/4	0	Extension nicht produktionsreif[10]
K3c: vollständige Backend-Logik über plattformunabhängige API verfügbar	0	1/4	0	reine Content-API[10]
K3d: Verfügbare SDKs für native Apps (Desktop-App, Mobile)	1/2	1/4	1/8	Vue.js Template und Manifest for PWA-Development[10]
K4: Modular Implementation				
K4a: Composable modular architecture	1	1/4	1/4	Erweiterbarkeit durch Extension-Prinzip, Trennung von Plattform, Backend und Frontend[11]
K4b: Einfache Integration mit externen Diensten	1/2	1/4	1/8	Extension Repository mit vielen von der Community geteilten Lösungen für Anbindung an populäre ext. Dienste (bspw. Bezahldienstleister)[12] und Möglichkeit der Anbindung durch eigene Extensions
K4c: Webhooks	1	1/4	1/4	Built-In Webhooks-Support[13]
K4d: Unterstützung inkrementeller, unabhängiger Entwicklung und Implementierung	1	1/4	1/4	stabile Schnittstellen innerhalb einer LTS und Trennung von Backend und Frontend erlauben unabhängiges Entwickeln und Test neuer Feature[3]
K5: Agility and Extensibility				
K5a: Transparente Updates, Upgrades	1	1/5	1/5	Alle Änderungen öffentlich in Release Notes[14]
K5b: Regelmäßige Updates	1	1/5	1/5	Regelmäßige Wartungs- und Sicherheitsupdates[3]
K5c: Klare Schnittstellen-versionierung	1	1/5	1/5	Semantische Versionierung nach [Major].[Minor].[Patch]-Muster[3]

[9] TYPO3 Association. (o.J.). *Massive Multisite and Multilingual.* Abgerufen am 05.03.2024 von https://typo3.org/cms/features/massively-multisite-multilingual
[10] (TYPO3 Association, o.J. e)
[11] (TYPO3 Association, o.J. b)
[12] TYPO3 Association. (o.J.). *TYPO3 Extension Repository.* Abgerufen am 05.03.2024 von https://extensions.typo3.org
[13] TYPO3 Association. (o.J.). *API A-Z.* Abgerufen am 05.03.2024 von https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/Introduction/Index.html#overview
[14] (TYPO3 Association, o.J. a)

K5d: Rückwärts-kompatibilität	1/2	1/5	1/10	Rückwärtskompatibilität bis einschließlich letzter Major-Version[3]
K5e: Unterstützung CI/CD Ansatz	1/2	1/5	1/5	Verschiedene ext. How-to-Guides[15] Schnelle CI/CD-Pipeline für agile Core-Entwicklung[16]
K6: Scalability				
K6a: automatisierte elastische Skalierung	1/2	1/5	1/10	Inoffizielle Skalierungsansätze für die Cloud, auch horizontal durch Duplizierung des Kerns[17]
K6b: historische Leistungsstatistiken	1/2	1/5	1/10	durch externe Analysetools[18] oder vom Plattformanbieter bereitgestellt[19]
K6c: Caching Doc	1	1/5	1/5	TYPO3-Docs[20]
K6d: Webhooks Doc	1	1/5	1/5	TYPO3-Docs[20]
K6e: Containerization Doc	1/2	1/5	1/10	Inoffizielle Container-Setup-Dokus[21]
K7: Stability				
K7a: Garantierte Verfügbarkeit & Leistung	1/2	1/3	1/6	Service-Level-Agreements (SLA) bei Hosting-Anbietern verfügbar[2]
K7b: Garantierte Sicherheit	1/2	1/3	1/6	Halbautomatisierte-Updates in Eigenverantwortung des Betreibers, kostenfreie Sicherheitsupdates für 3 Jahre (LTS) + 3 Jahre kostenpflichtig (ELTS)[3], zusätzlich Service-Level-Agreements (SLA) bei Hosting-Anbietern verfügbar[2]
K7c: Customer-Support	1/2	1/3	1/6	Kostenpflichtiger SLA für Geschäftskunden Kernfunktionalität[22] + Kostenpflichtiger Support durch Agenturen[1] und kostenfreier Support durch Community[23]

[15] Chauhan, D. (12.01.2022) *TYPO3 Gitlab CI/CD Auto Deployment.* Abgerufen am 05.03.2024 von https://t3planet.de/blog/typo3-gitlab-deployment/
[16] B13. (o.J.). Abgerufen am 16.03.2024 von https://b13.com/core-insights/typo3-core-testing-infrastructure-part-1
[17] Grau, S. (14.11.2021) *TYPO3 on AWS - Scalability and Performance* Abgerufen am 05.03.2024 von https://typo3.com/blog/typo3-on-aws-scalability-and-performance
[18] Moog, Susanne. (o.J.). *Empowering TYPO3 Development with Sentry.* https://susi.dev/blog/sentry-typo3/
[19] Platform.sh. (o.J.). *Infrastructure metrics.* Abgerufen am 05.03.2024 von https://docs.plattform.sh/increase-observability/metrics.html
[20] TYPO3 Association. (o.J.). *TYPO3 Documentation.* Abgerufen am 05.03.2024 von https://docs.typo3.org
[21] Helmich, M. (o.J.). *A docker image for running TYPO3*. Abgerufen am 23.03.2024 von https://GitHub.com/martin-helmich/docker-typo3
[22] TYPO3 Association. (o.J.). *Protect The Value of Your Website With a TYPO3 SLA.* Abgerufen am 05.03.2024 von https://typo3.com/blog/protect-the-value-of-your-website-with-a-typo3-sla
[23] TYPO3 Association. (o.J.). *TYPO3 Support.* Abgerufen am 05.03.2024 von https://typo3.org/help

K8: Strategic Alignment				
K8a: Zusammenarbeit mit Kunden zur Weiterentwicklung	1	1	1	Core Entwickler sind selbst in Agenturen und Community aktiv, wodurch aktuelle Kundenanforderungen in Entwicklung einfließen[24] + Initiativen holen Community-Feedback ein[25], Analysen und Endnutzerbefragungen durch Product Strategy Group[26]
K9: Documentation				
K9a: aktuelle, selbst-erklärende, suchbare, einfach benutzbare Dokumentation inklusive Tutorials	1	1/2	1/2	Documentation-Team aktualisiert und verbessert zusammen mit Community beständig die Doku[27]
K9b: Qualität der Doku wird durch Einsatz an realen Projekten getestet	1/2	1/2	1/4	Doku wird durch Freiwillige aus Community an eigenen Projekten getestet und es besteht die Möglichkeit Verbesserungen vorschlagen[28]
K10: Ease-of-Use				
K10a: schnelle Erlernbarkeit Benutzer	1/2	1/3	1/6	Aktive Initiativen zur Vereinfachung, z.B. UX-Initiative[28]
K10b: schnelle Erlernbarkeit Entwickler	1/2	1/3	1/6	Kontinuierliches Refactoring der über 25 Jahre gewachsenen Software, hin zu loserer Kopplung und mehr Einheitlichkeit[29]
K10c: Live oder On-Demand Training für alle Nutzergruppen verfügbar	1/2	1/3	1/6	Kostenpflichtige Trainings- und Zertifizierungskurse[30]
Ergebnis TYPO3			**6,48**	von 10

[24] TYPO3 Association. (o.J.). *TYPO3 CMS Development Roadmap.* Abgerufen am 16.04.2024 von https://typo3.org/cms/roadmap
[25] TYPO3 Association. (o.J.). https://typo3.org/article/initial-phase-of-the-acl-enhancement-initiative
[26] TYPO3 Association. (o.J.). *The TYPO3 CMS Product Strategy Group.* Abgerufen am 05.03.2024 von https://typo3.org/article/the-typo3-cms-product-strategy-group
[27] TYPO3 Association. (o.J.). *The TYPO3 Documentation Team* Abgerufen am 16.04.2024 von https://typo3.org/community/teams/ documentation
[28] TYPO3 Association. (o.J.). Abgerufen am 16.03.2024 von https://typo3.org/community/teams/user-experience-ux
[29] TYPO3 Association. (o.J.). Abgerufen am 16.03.2024 von https://typo3.org/community/teams/typo3-development/initiatives/persistence
[30] TYPO3 Association. (o.J.). *Official training courses led by TYPO3 professionals.* Abgerufen am 05.03.2024 von https://typo3.com/services/official-typo3-training-courses

1.2 WordPress

Tabelle 2: Quantitative Analyse zum Einfluss der MACH-Kriterien auf WordPress

Kriterium	E	G	B	Begründung
K1: Decoupled				
K1a: Microservices	1/2	1	1/2	Integrierte REST-API auf Netzwerkebene erlaubt Container-Multi-Core-Ansatz bei dem Kerne je nach Konfiguration verschiedene Aufgaben bereitstellen[31]
K2: Cloud-native				
K2a: PaaS oder SaaS	1	1/5	1/5	Wichtigster Core-Contributor und Firmengründer betreibt mit WordPress.com – SaaS Angebot[32]
K2b: Upgradefrei	0	1/5	0	Nur die letzte Nebenversionsnummer wird offiziell gewartet (Bsp. 6.5) = ein halbes Jahr kein Upgrade notwendig[33]
K2c: automatische Updates	1	1/5	1/5	Automatisierte Updates für Minor-Releases (Bsp. 6.5.1->6.5.2) seit Version 3.7[34]
K2d: automatische Upgrades	1	1/5	1/5	Automatisierte Upgrades für Major-Releases (Bsp. 6.5.->6.6) aktivierbar[34]
K2e: Mehrmandanten-fähigkeit inklusive *self-service*, getrennter Datenbanken und Ressourcensharing	1/2	1/5	1/10	Mehrmandantenfähiges WordPress inklusive separater Datenbanken und *self-service* bei externen Hosting Anbietern[35]
K3: Natively Headless and API Enabled				
K3a: REST	1	1/4	1/4	Kernfeature, Grundlage für WordPress Block Editor[36]
K3b: GraphQL	1/2	1/4	1/8	GraphQL-Plugin[37]

[31] Pollock, J. (21.09.2017). *Embracing the Microservice Architecture Pattern With WordPress.* Abgerufen am 10.04.2024 von https://torquemag.io/2017/09/embracing-the-microservice-architecture-pattern-with-WordPress/

[32] WordPress.com. (o.J.). *Wordpress-Hosting*. Abgerufen am 10.04.2024 von https://wordpress.com/de/hosting/

[33] WordPress.org. (o.J.). *Supported Versions.* Abgerufen am 10.04.2024 von https://wordpress.org/documentation/article/supported-versions/

[34] WordPress.org. (o.J.). *Updating WordPress.* Abgerufen am 10.04.2024 von https://WordPress.org/documentation/article/updating-WordPress/

[35] Wildcloud.com. (o.J.). *Solve all multisite limitations with multi-tenant WordPress.* Abgerufen am 10.04.2024 von https://wildcloud.com/multisite-versus-wildcloud/

[36] WordPress.org. (o.J.). *REST API Handbook.* Abgerufen am 10.04.2024 von https://developer.WordPress.org/rest-api/

[37] WordPress.org. (o.J.). *WPGraphQL.* Abgerufen am 11.04.2024 von https://WordPress.org/plugins/wp-graphql/

K3c: vollständige Backend-Logik über plattformunabhängige API verfügbar	1/2	1/4	1/8	Weniger Funktionsumfang als PHP-API[38] + Grundlage des WordPress Block Editor (Gutenberg Projekt), daher umfangreich und beständige Weiterentwicklung[36]
K3d: Verfügbare SDKs für native Apps (Desktop-App, Mobile)	1	1/4	1/4	Client Libraries für PHP, JavaScript, Ruby, C#/.NET, Dart/Flutter[39]
K4: Modular Implementation				
K4a: Composable modular architecture	1	1/4	1/4	Erweiterbarkeit durch Plug-In Prinzip, Trennung von Plattform, Backend und Frontend[40]
K4b: Einfache Integration mit externen Diensten	1/2	1/4	1/8	Plug-In Repository mit vielen von der Community geteilten Lösungen für Anbindung an populäre ext. Dienste (bspw. Bezahldienstleister)[41] und Möglichkeit der Anbindung durch eigene Plug-Ins
K4c: Webhooks	1/2	1/4	1/8	Webhooks-Plugin[42]
K4d: Unterstützung inkrementeller, unabhängiger Entwicklung und Implementierung	1	1/4	1/4	Ziel dauerhaft stabile Schnittstellen, Rückwärtskompatiblität und Trennung von Backend und Frontend erlauben unabhängiges Entwickeln und Test neuer Features[43]
K5: Agility and Extensibility				
K5a: Transparente Updates, Upgrades	1	1/5	1/5	Alle Änderungen öffentlich in Release Notes[44]
K5b: Regelmäßige Updates	1	1/5	1/5	Regelmäßige Wartungs- und Sicherheitsupdates[45]
K5c: Klare Schnittstellen-versionierung	1	1/5	1/5	Major-Versions (Bsp. 6.5) für neue Features and APIs Minor-Versions für Sicherheitsupdates und Bug-Fixes[43]
K5d: Rückwärts-kompatibilität	1	1/5	1/5	Rückwärtskompatibilität hat Priorität, in seltenen Fällen gebrochen[43]

[38] Gatsby. (o.J.). *What is Headless WordPress?* https://www.gatsbyjs.com/docs/glossary/Headless-WordPress/

[39] WordPress.org. (o.J.). *Client libraries.* Abgerufen am 11.04.2024 von https://developer.wordpress.org/rest-api/using-the-rest-api/client-libraries/

[40] WPShout.com. (29.05.2023). *WordPress Template Hierarchy Explained.* Abgerufen am 11.04.2024 von https://wpshout.com/WordPress-template-hierarchy/

[41] WordPress.org. (o.J.). *Plugins.* Abgerufen am 11.04.2024 von https://WordPress.org/plugins/

[42] WordPress.org. (o.J.). *WP Webhooks.* Abgerufen am 11.04.2024 von https://de.WordPress.org/plugins/wp-webhooks/

[43] WordPress.org. (o.J.). *Version Numbering* . Abgerufen am 11.04.2024 von https://make.wordpress.org/core/handbook/about/release-cycle/version-numbering/

[44] WordPress.org. (o.J.). *Releases.* Abgerufen am 11.04.2024 von https://WordPress.org/news/category/releases/

[45] WordPress.org. (o.J.). *Security.* Abgerufen am 11.04.2024 von https://WordPress.org/about/security/

K5e: Unterstützung CI/CD Ansatz	1	1/5	1/5	Nativ möglich da Entwickler im Besitz des Source Codes + Verschiedene ext. How-to-Guides[46]
K6: Scalability				
K6a: automatisierte elastische Skalierung	1/2	1/5	1/10	Inoffizielle Skalierungsansätze für die Cloud, auch horizontal durch Duplizierung des Kerns[47]
K6b: historische Leistungsstatistiken	1/2	1/5	1/10	durch Plugins[41]
K6c: Caching Doc	1	1/5	1/5	WordPress Developer Resources[48]
K6d: Webhooks Doc	1/2	1/5	1/10	Plug-In Doc[41]
K6e: Containerization Doc	1/2	1/5	1/10	Inoffizielle Container-Setup-Dokus[49]
K7: Stability				
K7a: Garantierte Verfügbarkeit & Leistung	1/2	1/3	1/6	Service-Level-Agreements (SLA) bei Hosting-Anbietern[50]
K7b: Garantierte Sicherheit	1/2	1/3	1/6	Automatische Sicherheitsupdates und Service-Level-Agreements (SLA) bei Hostinganbietern[32]
K7c: Customer-Support	1/2	1/3	1/6	24/7 Support bei Hosting-Anbietern[32]
K8: Strategic Alignment				
K8a: Zusammenarbeit mit Kunden zur Weiterentwicklung	1	1	1	Unternehmen, die WordPress intensiv nutzen sollten WordPress über die „Five for the Future"-Initiative unterstützen, wobei 5% der Mitarbeiter für Core-Contribution-Aufgaben eingesetzt werden. So lässt sich die Weiterentwicklung entsprechend der Kundenanforderungen beeinflussen[51]

[46] Bewersdorff, S. (14.11.2022). *Continuous Deployments for WordPress Using GitHub Actions*. Abgerufen am 11.04.2024 von https://css-tricks.com/continuous-deployments-for-WordPress-using-GitHub-actions/

[47] Gadsdon, M. (21.07.2022). *Creating a Scalable WordPress Deployment on AWS*. Abgerufen am 11.04.2024 von https://www.logicata.com/blog/creating-scalable-WordPress-on-aws/

[48] WordPress.org. (o.J.). *Cache.* Abgerufen am 11.04.2024 von https://developer.WordPress.org/advanced-administration/performance/cache

[49] DigitalOcean (o.J.). *How To Install WordPress With Docker Compose*. Abgerufen am 11.04.2024 von https://www.digitalocean.com/community/tutorials/how-to-install-WordPress-with-docker-compose

[50] WordPress.com. (o.J.). *WordPress-Hosting.* Abgerufen am 11.04.2024 von https://WordPress.com/de/hosting/

[51] WordPress.com. (o.J.). *Five for the Future.* Abgerufen am 16.04.2024 von *https://WordPress.org/five-for-the-future/*

K9: Documentation				
K9a: aktuelle, selbst-erklärende, suchbare, einfach benutzbare Dokumentation inklusive Tutorials	1	1/2	1/2	Documentation-Team aktualisiert und verbessert zusammen mit Community beständig die Dokumentation und Freiwillige schreiben Tutorials zu aktuellen Themen[52]
K9b: Qualität der Doku wird durch Einsatz an realen Projekten getestet	1/2	1/2	1/4	Doku wird durch Freiwillige aus Community an eigenen Projekten getestet und es besteht die Möglichkeit Verbesserungen vorzuschlagen[52]
K10: Ease-of-Use				
K10a: schnelle Erlernbarkeit Benutzer	1	1/3	1/3	Bereits seit langem Zielstellung in offizieller Roadmap mit bereits erreichten Meilensteinen, Kernmodul Drag&Drop Gutenberg Block Editor[53]
K10b: schnelle Erlernbarkeit Entwickler	1/2	1/3	1/6	Verschiedene Initiativen zur Verbesserung[54]
K10c: Live oder On-Demand Training für alle Nutzergruppen verfügbar	1	1/3	1/3	Kostenfreie Live-Workshops für alle Benutzergruppen[52]
Ergebnis WordPress			**7,38**	von 10

[52] WordPress.com. (o.J.). Make WordPress Documentation. Abgerufen am 16.04.2024 von https://make.WordPress.org/docs/

[53] WordPress.com. (o.J.). *Roadmap*. Abgerufen am 16.04.2024 von https://WordPress.org/about/roadmap/

[54] WordPress.com. (o.J.). *Feature Projects Overview*. Abgerufen am 16.04.2024 von https://make.WordPress.org/core/features/

1.3 Drupal

Tabelle 3: Quantitative Analyse zum Einfluss der MACH-Kriterien auf Drupal

Kriterium	E	G	B	Begründung
K1: Decoupled				
K1a: Microservices	1/2	1	1/2	Multi-Core-Ansatz möglich: Mehrere Kerne kommunizieren über Webservices-API, Bestrebungen Standard-Kern kleiner zu machen durch Entkopplung von Modulen[55]
K2: Cloud-native				
K2a: PaaS oder SaaS	1	1/5	1/5	Wichtigster Core-Contributor Acquia bietet Drupal-SaaS an[56]
K2b: Upgradefrei	1/2	1/5	1/10	Bis zu 2 Jahre kein Upgrade notwendig durch: LTS-Version = 2 Jahre, keine Breaking Changes + kostenfreier Support[57]
K2c: automatische Updates	1/2	1/5	1/10	Halb-automatisierte Updates für Patch-Releases (Bsp. 9.1.0->9.1.1)[58]
K2d: automatische Upgrades	0	1/5	0	Nicht möglich, da Breaking Changes an Public APIs[57]
K2e: Mehrmandanten-fähigkeit inklusive *self-service*, getrennter Datenbanken und Ressourcensharing	1/2	1/5	1/10	Mehrmandantenfähiges „Hybrid"-Drupal, gemeinsame Codebasis mit verschiedenen Datenbanken, Ressourcensharing eines LAMP-Stacks[59]
K3: Natively Headless and API Enabled				
K3a: REST	1	1/4	1/4	Kernfeature seit Drupal 8.5, vereinfacht durch „JSON:API" für 90% der Anwendungsfälle[60]
K3b: GraphQL	1/2	1/4	1/8	Erweiterungsmodul[61]

[55] Buytaert, Dries. (09.05.2022). *A plan for Drupal 11*. Abgerufen am 18.04.2024 von https://dri.es/a-plan-for-drupal-11

[56] Acquia. (o.J.). *Drupal Hosting*. Abgerufen am 18.04.2024 von https://www.acquia.com/solutions/drupal-hosting

[57] Drupal.org. (20.12.2023). *Release process overview*. Abgerufen am 18.04.2024 von https://www.drupal.org/about/core/policies/core-release-cycles/release-process-overview

[58] Drupal.org. (24.10.2023). *Automatic Updates Initiative overview and roadmap*. Abgerufen am 18.04.2024 von https://www.drupal.org/project/ideas/issues/2940731

[59] Acquia. (o.J.). *The Drupal Powered Enterprise*. Abgerufen am 18.04.2024 von https://www.acquia.com/resources/white-paper/drupal-powered-enterprise

[60] Drupal.org. (19.10.2021). *API-first initiative*. Abgerufen am 18.04.2024 von https://www.drupal.org/project/ideas/issues/2757967

[61] Drupal.org. (o.J.). *GraphQL*. Abgerufen am 18.04.2024 von https://www.drupal.org/project/graphql

K3c: vollständige Backend-Logik über plattformunabhängige API verfügbar	1/2	1/4	1/8	API-First-Initiative: alle Module sollten in Zukunft Web Service API bereitstellen, teilweise umgesetzt, mittlerweile inaktiv[62]
K3d: Verfügbare SDKs für native Apps (Desktop-App, Mobile)	1/2	1/4	1/8	Externe SDKs für native Android und iOS Apps + Progressive Web App Module[63]
K4: Modular Implementation				
K4a: Composable modular architecture	1	1/4	1/4	Erweiterbarkeit durch Modul-Prinzip, Trennung von Plattform, Backend und Frontend[64]
K4b: Einfache Integration mit externen Diensten	1/2	1/4	1/8	Module Repository mit vielen von der Community geteilten Lösungen für Anbindung an populäre ext. Dienste (bspw. Bezahldienstleister) und Möglichkeit der Anbindung durch eigene Plug-Ins[64]
K4c: Webhooks	1/2	1/4	1/8	Webhooks-Modul[65]
K4d: Unterstützung inkrementeller, unabhängiger Entwicklung und Implementierung	1	1/4	1/4	4 Jahre stabile Schnittstellen erlauben unabhängiges Entwickeln und Test neuer Features[57]
K5: Agility and Extensibility				
K5a: Transparente Updates, Upgrades	1	1/5	1/5	Alle Änderungen öffentlich in Release Notes[66]
K5b: Regelmäßige Updates	1	1/5	1/5	Regelmäßige Wartungs- und Sicherheitsupdates[66]
K5c: Klare Schnittstellen-versionierung	1	1/5	1/5	Semantische Versionierung nach [Major].[Minor].[Patch]-Muster[67]
K5d: Rückwärts-kompatibilität	1/2	1/5	1/10	Rückwärtskompatibilität bis einschließlich letzter Major-Version[67]

[62] Buytaert, D. (19.07.2018). *How Drupal continues to evolve towards an API-first platform*. Abgerufen am 18.04.2024 von https://www.drupal.org/blog/how-drupal-continues-to-evolve-towards-an-api-first-plattform

[63] Drupal.org. (o.J.). *Native mobile application development*. Abgerufen am 18.04.2024 von https://www.drupal.org/docs/mobile-drupal-sites/native-mobile-application-development

[64] Drupal.org. (o.J.). *Download & Extend*. Abgerufen am 18.04.2024 von https://www.drupal.org/project/project_module

[65] Drupal.org. (o.J.). *Webhooks*. Abgerufen am 18.04.2024 von https://www.drupal.org/project/webhooks

[66] Drupal.org. (o.J.). *Releases for Drupal core.* Abgerufen am 18.04.2024 von https://www.drupal.org/project/drupal/releases

[67] Drupal.org. (18.03.2024). *Allowed changes during Drupal core release cycles.* Abgerufen am 18.04.2024 von https://www.drupal.org/about/core/policies/core-change-policies/allowed-changes/

K5e: Unterstützung CI/CD Ansatz	1/2	1/5	1/5	Hosting-Anbieter bietet vorkonfigurierte Drupal Pipelines[68] + externe How-To-Guides[69]
K6: Scalability				
K6a: automatisierte elastische Skalierung	1/2	1/5	1/10	Inoffizielle externe How-To-Guides[70]
K6b: historische Leistungsstatistiken	1/2	1/5	1/10	Monitoring-Modul[71]
K6c: Caching Doc	1	1/5	1/5	Drupal Doc[72]
K6d: Webhooks Doc	1/2	1/5	1/10	Webhooks-Module Doc[65]
K6e: Containerization Doc	1/2	1/5	1/10	Inoffizielle Container Setup Dokus[73]
K7: Stability				
K7a: Garantierte Verfügbarkeit & Leistung	1/2	1/3	1/6	Service-Level-Agreements (SLA) bei Hosting-Anbietern[56]
K7b: Garantierte Sicherheit	1/2	1/3	1/6	Halbautomatisierte-Updates in Eigenverantwortung des Betreibers, kostenfreie Sicherheitsupdates für 2 Jahre[54], Service-Level-Agreements (SLA) bei Hosting-Anbietern[58]
K7c: Customer-Support	1/2	1/3	1/6	24/7 Support bei Hosting-Anbietern[74]
K8: Strategic Alignment				
K8a: Zusammenarbeit mit Kunden zur Weiterentwicklung	1	1	1	Strategie-Initiativen für die Weiterentwicklung des Kerns richten sich nach Vision, Markt, Diskussion in der Community und erhobenen Daten (Usability Studien und Umfragen)[75]

[68] Acquia. (o.J.). *CI/CD & Deployment Automation*. Abgerufen am 18.04.2024 von https://www.acquia.com/products/drupal-cloud/cloud-plattform/deployment-automation

[69] Wodby Blog. (13.10.2016). *Drupal 8 CI/CD with Docker via Jenkins. Part 1: Integration.* Abgerufen am 18.04.2024 von https://blog.wodby.com/drupal-8-ci-cd-with-jenkins-part-1-integration-eabd0f5c4f75

[70] AWS. (o.J.). *Deploying a high-availability Drupal website with an external Amazon RDS database to Elastic Beanstalk*. Abgerufen am 18.04.2024 von https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/PHP-hadrupal-tutorial.html

[71] Drupal.org. (o.J.). *Monitoring.* Abgerufen am 18.04.2024 von https://www.drupal.org/project/monitoring

[72] Drupal.org. (22.09.2022). *Caching Overview.* Abgerufen am 18.04.2024 von https://www.drupal.org/docs/7/ managing-site-performance-and-scalability/caching-to-improve-performance/caching-overview

[73] Jain, A. (25.04.2020). *How To Install Drupal with Docker Compose.* Abgerufen am 18.04.2024 von https://www.digitalocean.com/community/tutorials/ how-to-install-drupal-with-docker-compose

[74] Acquia. (o.J.). *Services to Support Your Success.* Abgerufen am 18.04.2024 von https://www.acquia.com/products/acquia-cloud-plattform/support

[75] Drupal.org. (o.J.). *Strategic Initiatives.* Abgerufen am 18.04.2024 von https://www.drupal.org/about/core/strategic-initiatives

K9: Documentation				
K9a: aktuelle, selbst-erklärende, suchbare, einfach benutzbare Dokumentation inklusive Tutorials	1	1/2	1	Documentation-Working-Group aktualisiert und verbessert zusammen mit Community beständig die Dokumentation und Freiwillige schreiben Tutorials zu aktuellen Themen[76]
K9b: Qualität der Doku wird durch Einsatz an realen Projekten getestet	1/2	1/2	1/4	Doku wird durch Freiwillige aus Community an eigenen Projekten getestet und es besteht die Möglichkeit Verbesserungen vorzuschlagen[76]
K10: Ease-of-Use				
K10a: schnelle Erlernbarkeit Benutzer	1	1/3	1/3	Verschiedene Initiativen zur Verbesserung[66], Kernmodul Drag&Drop Drupal Layout Builder[77]
K10b: schnelle Erlernbarkeit Entwickler	1/2	1/3	1/6	Verschiedene Initiativen zur Verbesserung[66]
K10c: Live oder On-Demand Training für alle Nutzergruppen verfügbar	1	1/3	1/3	Kostenfreie Lernpfade und öffentliches Live-Training für alle Benutzergruppen[78]
Ergebnis Drupal			**6,93**	von 10

[76] Drupal.org. (o.J.). *Documentation Working Group Overview.* Abgerufen am 18.04.2024 von https://www.drupal.org/docs/working-group/documentation-working-group-overview

[77] Drupal.org. (o.J.). *Layout Builder module.* Abgerufen am 24.04.2024 von https://www.drupal.org/docs/8/core/modules/layout-builder

[78] Acquia. (o.J.). *Welcome to Acquia Learning Services*. Abgerufen am 18.04.2024 von https://training.acquia.com/

1.4 Joomla!

Tabelle 4: Quantitative Analyse zum Einfluss der MACH-Kriterien auf Joomla!

Kriterium	E	G	B	Begründung
K1: Decoupled				
K1a: Microservices	1/2	1	1/2	Joomla!-Framework wurde vom CMS entkoppelt, enthält unabhängige PHP-Module, mit denen sich Microservices erstellen lassen. CMS soll künftig darauf aufbauen, nutzt bisher nur einen kleinen Teil davon[79]
K2: Cloud-native				
K2a: PaaS oder SaaS	1	1/5	1/5	Open Source Matters bietet mit cloudaccess.net Joomla!-SaaS an, Erlöse finanzieren auch das Open Source Projekt[80]
K2b: Upgradefrei	1	1/5	1/5	Bis zu 4 Jahre kein Upgrade notwendig, Major-Version mindestens 4 Jahre ohne Breaking Changes unterstützt[81]
K2c: automatische Updates	1/2	1/5	1/10	Halb-automatisierte Updates für Patch-Releases (Bsp. 9.1.0->9.1.1)[82]
K2d: automatische Upgrades	0	1/5	0	Migration notwendig, Breaking Changes zwischen Major-Versionen[83]
K2e: Mehrmandanten-fähigkeit inklusive *self-service*, getrennter Datenbanken und Ressourcensharing	0	1/5	0	keine vorhandenen oder geplanten Lösungen gefunden
K3: Natively Headless and API Enabled				
K3a: REST	1	1/4	1/4	Kernfeature seit Joomla! 4[84]
K3b: GraphQL	0	1/4	0	keine vorhandenen oder geplanten Lösungen gefunden

[79] Joomla.org. (o.J.). *Joomla! Framework.* Abgerufen am 20.04.2024 von https://framework.Joomla.org/

[80] OpenSourceMatters. (o.J.). *Frequently Asked Questions.* Abgerufen am 20.04.2024 von https://www.opensourcematters.org/organisation/faq.html#3-where-does-open-source-matters-get-its-money/

[81] Joomla.org. (o.J.). *Release and support cycle.* Abgerufen am 20.04.2024 von https://docs.Joomla.org/Release_and_support_cycle

[82] Joomla.org. (o.J.). *Joomla! CMS Version.* Abgerufen am 20.04.2024 von https://docs.Joomla.org/Joomla!_CMS_versions/de

[83] Joomla.org. (o.J.). *Joomla! Project Roadmap .* Abgerufen am 20.04.2024 von https://developer.Joomla.org/roadmap.html

[84] Joomla.org. (o.J.). *Joomla! Core APIs.* Abgerufen am 20.04.2024 von https://docs.Joomla.org/J4.x:Joomla!_Core_APIs

K3c: vollständige Backend-Logik über plattformunabhängige API verfügbar	0	1/4	0	keine vorhandenen oder geplanten Lösungen gefunden
K3d: Verfügbare SDKs für native Apps (Desktop-App, Mobile)	1/2	1/4	1/8	Native Apps aus bestehenden Joomla!-Webseiten erzeugen mittels externer Anbieter[85]
K4: Modular Implementation				
K4a: Composable modular architecture	1	1/4	1/4	Erweiterbarkeit durch Extension-Prinzip[86], Trennung von Plattform, Backend und Frontend, Joomla! Framework steht separat als PHP-Framework zur Anwendungsentwicklung bereit[79]
K4b: Einfache Integration mit externen Diensten	1/2	1/4	1/8	Extension Repository mit vielen von der Community geteilten Lösungen für Anbindung an populäre ext. Dienste (bspw. Bezahldienstleister) und Möglichkeit der Anbindung durch eigene Plug-Ins[86]
K4c: Webhooks	1/2	1/4	1/8	Webhooks-Extension[86]
K4d: Unterstützung inkrementeller, unabhängiger Entwicklung und Implementierung	1	1/4	1/4	4 Jahre stabile Schnittstellen erlauben unabhängiges Entwickeln und Test neuer Features[81]
K5: Agility and Extensibility				
K5a: Transparente Updates, Upgrades	1/2	1/5	1/10	Alle Updates öffentlich in Release Notes[82]
K5b: Regelmäßige Updates	1	1/5	1/5	Regelmäßige Wartungsupdates und unmittelbare Sicherheitsupdates, sobald Sicherheitslücke entdeckt[87]
K5c: Klare Schnittstellen-versionierung	1	1/5	1/5	Semantische Versionierung nach [Major].[Minor].[Patch]-Muster[81]
K5d: Rückwärts-kompatibilität	1/2	1/5	1/10	Rückwärtskompatibilität innerhalb einer Major-Version[81]
K5e: Unterstützung CI/CD Ansatz	1/2	1/5	1/10	Verschiedene externe How-To-Guides[88]

[85] appypie. (o.J.). *Joomla! App Builder.* Abgerufen am 20.04.2024 von https://www.appypie.com/de/konvertieren-sie-die-Joomla-website-in-app

[86] Joomla.org. (o.J.). *Joomla! Extensions Directory™.* Abgerufen am 20.04.2024 von https://extensions.Joomla.org/

[87] Joomla.org. (15.02.2023). *Joomla! Security Strike Team.* Abgerufen am 20.04.2024 von https://developer.Joomla.org/security.html

[88] Joomla.org. (o.J.). *JAB17 - Your Joomla! dev-ops, ci & cd toolbox.* Abgerufen am 20.04.2024 von https://www.Joomla.de/wissen/Joomla!-event-videos/jandbeyond/jab17-your-Joomla-dev-ops-ci-cd-toolbox

K6: Scalability				
K6a: automatisierte elastische Skalierung	1/2	1/5	1/10	Inoffizielle How-To-Guides[89]
K6b: historische Leistungsstatistiken	1/2	1/5	1/10	Monitoring-Extension[90]
K6c: Caching Doc	1	1/5	1/5	Joomla! Doc[91]
K6d: Webhooks Doc	0	1/5	0	Keine Doku für Extension verfügbar
K6e: Containerization Doc	1/2	1/5	1/10	Inoffizielle Container Setup Dokus[92]
K7: Stability				
K7a: Garantierte Verfügbarkeit & Leistung	1/2	1/3	1/6	Service-Level-Agreements (SLA) bei Hosting-Anbietern[93]
K7b: Garantierte Sicherheit	1/2	1/3	1/6	Halbautomatisierte-Updates in Eigenverantwortung des Betreibers[82], oder Service-Level-Agreements (SLA) bei Hosting-Anbietern[94]
K7c: Customer-Support	1/2	1/3	1/6	24/7 Support bei Hosting-Anbietern[95]
K8: Strategic Alignment				
K8a: Zusammenarbeit mit Kunden zur Weiterentwicklung	1	1	1	Core Entwickler sind selbst in Agenturen und Community aktiv, wodurch aktuelle Kundenanforderungen in Entwicklung einfließen[96]
K9: Documentation				
K9a: aktuelle, selbst-erklärende, suchbare, einfach benutzbare Dokumentation inklusive Tutorials	0	1/2	0	Doku-Suche auf Google-Basis mit viel Werbung und wenig zielführenden Ergebnissen, Doku-Layout unübersichtlich, Viele Tutorials basieren auf Joomla! v3[97]

[89] Joomla.de. (o.J.). *JAB17 - Your Joomla! dev-ops, ci & cd toolbox.* Abgerufen am 20.04.2024 von https://www.Joomla.de/wissen/Joomla!-event-videos/jandbeyond/jab18-Joomla!-at-scale

[90] Joomla!.org. (o.J.). *Super Monitoring.* Abgerufen am 20.04.2024 von https://extensions.Joomla.org/extension/super-monitoring/

[91] Joomla.org. (o.J.). *Cache.* Abgerufen am 20.04.2024 von https://docs.Joomla.org/Cache

[92] goNeuland.de. (29.05.2023). *Joomla! – mit Docker Compose und Traefik installieren.* Abgerufen am 20.04.2024 von https://goneuland.de/Joomla-mit-docker-compose-und-traefik-installieren/

[93] cloudaccess.net. (o.J.). *Uptime Guarantee.* Abgerufen am 20.04.2024 von https://www.cloudaccess.net/guarantees.html

[94] Joomla.org. (o.J.). *Joomla! Update-Service.* Abgerufen am 20.04.2024 von https://hosting-pilot.com/Joomla-update-service/

[95] cloudaccess.net. (o.J.). *Support Overview.* Abgerufen am 20.04.2024 von https://www.cloudaccess.net/Joomla-WordPress-support.html

[96] Walton, P. (20.10.2023). *Embracing the Future: Exploring the Features of Joomla! 5.* Abgerufen am 20.04.2024 von https://magazine.Joomla.org/all-issues/october/embracing-the-future-exploring-the-features-of-Joomla-5

[97] Joomla.org. (o.J.). *Joomla! ® Documentation.* Abgerufen am 20.04.2024 von https://docs.Joomla.org/Main_Page

K9b: Qualität der Doku wird durch Einsatz an realen Projekten getestet	1/2	1/2	1/4	Doku wird durch Freiwillige aus Community an eigenen Projekten getestet und es besteht die Möglichkeit Verbesserungen vorzuschlagen[98]
K10: Ease-of-Use				
K10a: schnelle Erlernbarkeit Benutzer	1/2	1/3	1/6	Verschiedene Initiativen zur Verbesserung[99]
K10b: schnelle Erlernbarkeit Entwickler	1/2	1/3	1/6	Verschiedene Initiativen zur Verbesserung[99]
K10c: Live oder On-Demand Training für alle Nutzergruppen verfügbar	0	1/3	0	Keine detaillierten Trainings und Zertifizierungen für Entwickler gefunden
Ergebnis Joomla!			**5,41**	von 10

[98] Joomla.org. (o.J.). *How to Contribute to Joomla! Documentation.* Abgerufen am 20.04.2024 von https://docs.Joomla.org/JDOC:How_to_Contribute_to_Joomla_Documentation

[99] Joomla.org. (o.J.). *Joomla Issue Tracker - CMS.* Abgerufen am 20.04.2024 von https://issues.Joomla.org/

Anhang 2 - Interviewleitfaden

Dieser Interviewleitfaden ist Teil der Bachelorarbeit mit dem Thema „Einfluss der MACH-Architektur auf die Entwicklung traditioneller CMS und individuelle Betrachtung der Produktstrategie am Beispiel TYPO3“. Zum Leitfadeninterview sind verschiedene Experten eingeladen, die sich intensiv mit TYPO3 auf professioneller Ebene auseinandersetzen.

Das Interview wird anhand des folgenden Leitfadens geführt und im Anschluss mittels qualitativer Methoden ausgewertet. Die Ergebnisse sollen interessierten Entwicklern, Entscheidern und Integratoren einen Einblick geben, welche Produktstrategie TYPO3 angesichts der zunehmenden Zahl MACH-basierter CMS verfolgt. Die MACH-Kriterien sollen hierbei Diskussionsanreize für mögliche Entwicklungsziele liefern. Im besten Fall kann das Open Source Projekt durch die Arbeit gestärkt werden.

Es wird darum gebeten der angehängten Einverständniserklärung zuzustimmen, damit das Interview im Rahmen der Bachelor-Arbeit ausgewertet werden kann.

1 Einleitende Fragen: Rolle des Interviewpartner, Relevante Erfahrungen

1. Seit wie vielen Jahren beschäftigst du dich mit Content Management Systemen?

2. Was hat dich dazu bewegt in diesem Bereich zu arbeiten?

3. Welches war dein erstes CMS?

4. In welcher Rolle arbeitest du aktuell und worauf hast du dich spezialisiert?

5. Mit welchen CMS hast du Erfahrungen gesammelt?

6. Wann und in welchem Kontext sind dir die folgenden Stichwörter (MACH-Architektur) zum ersten Mal begegnet?

    a. Microservices

    b. API-first

    c. Cloud-native SaaS

    d. Headless

7. Inwiefern waren MACH-basierte Lösungen bereits von Bedeutung bei deiner Arbeit (bspw. Anforderung in einem Kundenprojekt, Ausschreibung)?

2 Produktstrategie TYPO3

2.1 Analyse der Ausgangssituation

1) Auf welchen Markt (welche Zielgruppe, welche Anwendungsfelder) fokussiert sich TYPO3 aktuell?

2) Welche externen Faktoren sorgen aktuell für Innovationsdruck bei TYPO3?
    - Politisch, Ökonomisch, Soziokulturell, Technologisch, Ökologisch, Rechtlich

3) Welche konkreten Anforderungen leiten sich daraus für die Weiterentwicklung von TYPO3 ab?

4) Wie gut lassen sich diese Anforderungen mit der aktuellen Architektur von TYPO3 lösen?

5) Steht TYPO3 in Konkurrenz zu MACH-basierten CMS?

6) Wo liegen deiner Meinung nach die wichtigste Stärke bzw. Schwäche von TYPO3 gegenüber MACH-basierten CMS?

    a) Stärken?

    b) Schwächen?

2.2 Zielsetzung

1) Welche Ziele sollte TYPO3 verfolgen, um seine Position am Markt gegenüber MACH-basierten CMS zu behaupten und zu stärken?

2.3 Ermittlung und Analyse möglicher Optionen

1) Welche der folgenden MACH Kriterien hat TYPO3 bereits berücksichtigt? Bei welchen besteht noch Potenzial? Welche bergen mehr Risiken als Chancen?

    a) Microservices
        i) CMS besteht aus voneinander unabhängigen Komponenten
        ii) Unabhängiges Entwickeln, Bereitstellen und Testen von Komponenten

    b) API Enabled
        i) Alle Funktionen des Backends über API erreichbar

    c) Headless/Decoupled
        i) Nativ vom Frontend getrennte Backend-Logik
        ii)

d) Cloud-native SaaS
   i) für die Nutzung in der Cloud optimiert

e) Modular Implementation
   i) Fähigkeit der einfachen Integration mit anderen Technologien
   ii) Unterstützung inkrementeller Implementierung/ Erweiterung

f) Agility and Extensibility
   i) Automatische Updates
   ii) Rückwärtskompatibilität
   iii) API-Versionierungsmethodik

g) Scalability
   i) Unterstützung bei Optimierung von Kosten und Performanz in Cloudumgebung
   ii) Performancestatistiken

h) Stability
   i) Detaillierte Statusmonitoring-Tools

i) Strategic Alignment
   i) Unterstützung der Kunden bei Weiterentwicklung des Systems gemäß ihrer Bedürfnisse

j) Documentation
   i) Verständlichkeit, Aktualität, Durchsuchbarkeit inklusive Tutorials

k) Easy-to-Use
   i) Schnelle Erlernbarkeit
   ii) Verfügbare Trainings

2) Welche Entwicklungen sind möglich?
   a) Microservices
   b) API-first
   c) Cloud-native SaaS
   d) Headless

   e) Welche Grenzen sind dem gesetzt?
   f) Welche Trade-Offs müsste man dabei in Kauf nehmen?

3 Abschließende Fragen

1) Was wünschst du dir zukünftig für die Arbeit mit TYPO3?

2) Werden MACH-basierte CMS in Zukunft traditionelle CMS verdrängen?

3) Gibt es noch einen Aspekt zum Thema, den du ergänzen möchtest?

Anhang 3 - Interviewtranskripte

2.1 Interview A

00:00:13 Forscher

Seit wie vielen Jahren beschäftigst du dich mit Content Management Systemen?

00:00:24 Kernentwickler 1

25 Jahre.

00:00:27 Forscher

25 Jahre, Oh, das ist eine lange Zeit.

00:00:34 Kernentwickler 1

Ganz kurz, ich hab irgendwann mal meine eigene Website gemacht, dann PHP und MySQL und vorher statisches HTML gemacht, um Filme auf DVD zu testen. Ganz primitiv wäre das mein eigenes Content Management System gewesen damals.

00:00:51 Kernentwickler 1

Dann habe ich während meiner Studienzeit bei einer Agentur gearbeitet, die ein eigenes Content Management System hatte. Also ein proprietäres System, damals gab es eben proprietäre und dann kamen so ein bisschen die Open Source Systeme. Das heißt, ich hab eigentlich erst ein Java-basiertes proprietäres System eingesetzt, als Redakteur, und dann wollten die Kunden immer mehr TYPO3 bei der Agentur, in der ich als Werkstudent gearbeitet habe.

00:01:34 Forscher

Wie hieß das System damals?

00:01:38 Kernentwickler 1

Jede Agentur hat ein eigenes gehabt. Ich weiß nicht mehr, wie das hieß, also relativ unwichtig.

00:01:46 Forscher

OK in welcher Rolle arbeitest du jetzt aktuell, worauf hast du dich spezialisiert?

00:01:54 Kernentwickler 1

Also, ich habe 2 Hüte auf. Ich bin zum einen Geschäftsführer bei einer Agentur, das mache ich seit 17 Jahren. Wir arbeiten da mit TYPO3, WordPress und Magento auch als Shopsystem genutzt, und haben irgendwann vor 8 Jahren gesagt, wir machen nur noch TYPO3 und dementsprechend ist das auch meine Hauptaufgabe. Ich bin technischer Leiter bei uns in der Firma mit gut 20 Leuten und wir sind technische Dienstleister für größere Unternehmen im internationalen Bereich, kurz gesagt. Auch ein paar Non-Profits, ein paar im öffentlichen Sektor, aber das sind so die wesentlichen. Der Fokus liegt bei uns

auf Mehrsprachigkeit, Integration mit Shopsystemen oder anderen Systemen natürlich auch. CRMs zum Beispiel und Publishing schon auch. Das ist so der eine Teil.

00:03:09 Kernentwickler 1

Und warum wurdest du an mich verwiesen? Ich bin auch seit 2007 mit als Entwickler bei TYPO3 dabei. Das heißt, ich hab in vielen Content Management Systemen Erfahrungen gesammelt, da können wir auch später gleich noch mal darauf eingehen. Im Core Team war ich seit 2007 Release Manager für TYPO3 und dann wurde ich vor 10 Jahren Leiter der Core-Entwicklung.

00:03:49 Forscher

Sind dir da schon mal die Begriffe Microservices, API-First, Cloud-Native und Headless begegnet?

00:03:59 Kernentwickler 1

Also die begegnen mir die ganze Zeit. Wenn du jetzt diese 2 Hüte immer so hast, dann gibt es im TYPO3 Kontext natürlich auch viele Leute, die auf irgendwelchen Camps fragen, wie funktioniert das mit Headless? Wie binde ich eine API an? Was kann da TYPO3 machen und TYPO3 war da damals, also Headless kam (...) Also ich bin auch im TYPO3 Kontext viel unterwegs. Ich war zum Beispiel Anfang des Jahres in den USA auf einer CMS-Konferenz, auf der es 20 verschiedene Content Management System Hersteller gab. Wir saßen an einem runden Tisch und wir waren das einzige Open Source Projekt. Wir waren das einzige kostenlose. Die anderen waren so zweigeteilt in: „Wir machen komplett MACH, sind auch in der MACH Alliance“ und den Rest, dementsprechend sagen mir die Begriffe alle was, logischerweise nicht erst seit Januar, sondern weil sich das so die letzten Jahre auch so ein bisschen etabliert hat.

00:05:10 Kernentwickler 1

Und es kommt ja auch aus dem Hintergrund, dass finde ich, da nehme ich wahrscheinlich jetzt ein paar Sachen vorweg, aber ich mache das jetzt einfach mal, dass in dem Moment, das war 2014, ‘15 kam eben dieses Buzzword Headless, und das heißt du hast viele Systeme, die kein Frontend haben, wie die Klassiker, die kennen wir heutzutage ja auch, die SaaS-basiert sind. Also wir hätten sowas wie Contentful oder StoryBloks und die reden mittlerweile gar nicht mehr von Headless und sondern von „Composable CMS“. Weil sie festgestellt haben: Naja, wir haben ja ganz viele Sachen, die ein normales, ein traditionelles CMS hat, gar nicht mehr, also ein Standard-Preview/Vorschau.

00:05:57 Kernentwickler 1

Und das haben Sie dann nachgerüstet. Da haben wir festgestellt, dieses: „Ich designe mir meine Content Typen, die ich brauche, das ist natürlich Gold wert. So kurz nochmal zum Kontext, in dem mir die Stichwörter begegnet sind. Das Thema Microservices betrifft ja nicht nur CMS oder halt meine TYPO3 Bubble, sondern Microservices kann auch sein: Ich habe eine Applikation und möchte einen anderen Service anbinden oder ein anderes Team entwickelt eine Schnittstelle und das wird als Microservice gebaut.

00:06:33 Kernentwickler 1

Microservice heißt für mich einfach verschiedene Applikationen, die an irgendeiner Stelle zusammenlaufen, die können unabhängig voneinander gepublished und released werden, dementsprechend ist auch API-First.

00:06:49 Kernentwickler 1

Wo begegnet mir das? Bestimmt vor 5 Jahren auf irgendwelchen TYPO3 Camps, oder 8 Jahren. Man liest online über die Sachen. Manche Kunden haben Services, die man anbinden möchte, das heißt es sind viele Machine-to-Machine-Systeme, wo du einfach APIs brauchst. Das heißt ich habe vor vielen Jahren schon irgendwelche REST-Sachen implementiert. Davor gab‘s SOAP, was gängig war und GraphQL machen wir auch verschiedene Integrationen.

00:07:21 Kernentwickler 1

Und Cloud-native ist, aus meiner Sicht, eher eine Business Entscheidung an erster Stelle, wo Kunden sich entscheiden: Wir wollen das gar nicht selbst hosten oder es ist viel besser, wir wollen keine Ressourcen dafür, wir kaufen uns das irgendwo ein, wir müssen uns nicht um die Wartung und das Hosting kümmern, und Headless ist bei der Integration von verschiedenen Systemen immer relevant, nicht erst seit gestern.

00:07:57 Kernentwickler 1

Wir hatten vor Corona schon irgendwelche Systeme, die sich mit Salesforce integrieren sollen oder Sales Force Shop und TYPO3 musste einfach nur den Content als JSON ausliefern. Das habe ich gemacht, bevor TYPO3 überhaupt irgendwelche Headless Extensions hatte.

00:08:16 Forscher

Ja, klingt gut. Also ist das schon lange ein Thema bei TYPO3. Dementsprechend wollte ich fragen. Gibt es denn da eine Konkurrenz zu den MACH-basierten Systemen oder ist man eigentlich in einem ganz anderen Bereich unterwegs?

00:08:42 Kernentwickler 1

Also die Frage ist, finde ich. MACH deckelt oder kombiniert verschiedene Aspekte. Also die 4 Hauptaspekte natürlich und ich schmeiße mal in den Raum, das macht vielleicht nicht immer alles zusammen Sinn.

00:09:09 Kernentwickler 1

Ich geh mal kurz auf die Schwachstellen von dem MACH-Thema ein. Weil, wenn man es durchliest und so einen reißerischen Artikel irgendwo auf der MACH Alliance Seite oder irgendeinen Blogpost liest, klingt das super: Die Entwicklung von Microservices, du kannst verschiedene Teams haben, die verschiedene Stärken haben, du brauchst kein Frontend für deine APIs oder du brauchst nur JavaScript-Leute, die können dann einen Microservice designen, cool.

00:09:40 Kernentwickler 1

Ich finde das Microservices meistens mit API-First gekoppelt sind. Ich hab zufällig diese Woche erlebt, mal wieder, dass es ziemlich schmerzhaft ist, wenn man das überabstrahiert. Also das heißt: Man setzt komplett auf Microservices und man denkt ja eigentlich, man ist entkoppelt voneinander und nachher gibt es keinen, der das Ding wartet. Das finde ich, ist die größte Herausforderung.

00:10:08 Forscher

Ja es sorgt für eine größere Komplexität, sagt man auch, die alle miteinander zu verknüpfen und sicherlich auch zu warten.

00:10:17 Kernentwickler 1

Ja, deswegen finde ich den Begriff „Composable“ auch treffender muss ich ehrlich sagen, weil Headless, um auf den Punkt Headless zu kommen, sehr entwicklerspezifisch ist. Das heißt, es ist natürlich schon praktisch, man entkoppelt die Darstellung bei dem Content Management System, also die Ausgabe von der Pflege. Das hat TYPO3 effektiv schon immer so gemacht.

00:10:45 Kernentwickler 1

Aber Headless hat das halt jetzt im Namen und ist halt standardmäßig JSON oder GraphQL. Also bei einem CMS muss ich das auch immer ein bisschen hinterfragen, denn es ist zwar cool, wenn es Headless ist, aber das, was ich in der Praxis erlebe ist, dass die Entwickler, die ein Frontend machen, relativ wenig Dokumentation hinterlegen. Also, dass es nicht gestreamlined ist, um mit den Teams zu arbeiten.

00:11:19 Kernentwickler 1

Und dann ist das Thema nochmal mit dem Cloud Native. Cloud-native ist für mich eigentlich der schwammigste Begriff, weil es für mich so ein bisschen nach Cloud-first klingt. Es gibt viele Systeme, die sagen, OK, wir jetzt auch eine Cloud-basierte Variante neben der On-Premise, also self-hosted Variante.

00:11:36 Kernentwickler 1

Da finde ich, dass Cloud-native heißt, ich muss mich um nichts kümmern. Ist für mich eine Riesengefahr, habe ich jetzt vor ein paar Wochen noch mal so mitbekommen. Dann entscheidet sich der Hersteller (...) Also das Coole ist natürlich, man hatte mit dem API-First Gedanken ja immer Zugriff auf alle seine Daten, jetzt kommt Cloud-native, ich weiß nicht, ob du dir Contentful schon angekuckt hast, find ich ja schon krass, dass du pro Datensatz bezahlen musst. Also je mehr Content ich hab, desto mehr muss ich bezahlen. Das finde ich weird, weil eigentlich ging es immer darum, je mehr Besucher ich hab, desto mehr muss ich bezahlen. Das finde ich besser.

00:12:20 Forscher

Ja das passt nicht richtig zusammen.

00:12:23 Kernentwickler 1

Und dann gab es eben immer wieder und das hör ich immer wieder, die Mitteilung: „Ja, wir erhöhen nächsten Monat unsere Preise. So, das heißt, das finde ich super intransparent,

schlecht planbar, was die Kosten betrifft und du hast schon einen krassen Vendor-Lockin. Das heißt, ich finde das am kritischsten, gekoppelt mit dem Thema: „Ich möchte meine Daten besitzen, die Daten sind das höchste Gut an dem Ganzen, nicht die Präsentation, nicht die Ausgabe, sondern die strukturierten Daten."

00:13:05 Kernentwickler 1

Das können traditionelle Content Management Systeme machen, aber dann kommt man vielleicht nicht so leicht an die Daten. Wenn man API-First denkt, kann man auch sagen, dass SQL eine API ist, also nicht, dass ich das cool finde, aber es wird in dem Kontext auch nicht so gesehen.

00:13:24 Kernentwickler 1

Genau. Und der Cloud-native Ansatz eben heißt: Eigentlich gehören einem die Daten nicht. Ich bin da begrenzt und das finde ich eigentlich am kritischsten.

00:13:40 Kernentwickler 1

Deswegen zurück zu deiner Frage: Gibt es da eine Konkurrenz? Ich würde sagen, alle, die das nicht so machen, die sind die Konkurrenz. Aber ich sehe wenig Vorteile für ein Content Management System, dass das Cloud-native laufen soll, wenig. Es gibt sie, aber die überwiegen für mich nicht. Vielleicht bin ich da auch ein bisschen zu traditionell, weil ich das schon so lange mache. Und ja, TYPO3 ist eben nicht auf diesen API-First oder Headless-Zug aufgesprungen und jetzt kommt dann wieder: „Ah, Preview wäre doch gut", und so ein bisschen: „Wir machen alles mit JavaScript und dann braucht man doch wieder Server-side Rendering und dann kommt es irgendwie wieder zusammen. Flexibles Denken ist da mit dabei.

00:14:27 Forscher

Ja, gibt es denn auch externe Faktoren, die die Entwicklung so ein bisschen in Richtung MACH drängen, also dass man sich dem gar nicht entziehen kann? Von politischer Seite zum Beispiel?

00:14:48 Kernentwickler 1

Ich würde eher sagen, dass man sich früher ein Content Management System gewählt hat, weil der IT-Kollege das empfohlen hat. Heute muss man wesentlich mehr Faktoren berücksichtigen. Häufig werden da auch Entscheidungen getroffen, die von vom Geschäftsführer oder von den Geschäftsführern oder der Geschäftsführer-Riege kommen, aus der Begründung heraus: „Ich war mit dem Golfspielen und deswegen habe ich jetzt Salesforce." und das kann anscheinend alles.

00:15:17 Kernentwickler 1

Die Faktoren gibt es trotzdem von politischer Riege. Ich vermute in welche Richtung du möchtest, aber da würde ich auch sagen: Nein. Sondern es ist schon eher ganzheitlicher gesehen, was der Markt braucht. Das heißt? Den Microservice-Gedanken haben wir jetzt bei TYPO3 nicht so verankert, aber die anderen Themen sind ja trotzdem machbar. Also Cloud-native oder halt Cloud-basiertes TYPO3 funktioniert mit TYPO3 mittlerweile und da kamen eher Industriepartner, die sagen: „Hey kuck mal, wir haben ein neues System, wär doch cool TYPO3 läuft darauf, weil es so einen großen Marktanteil hat."

00:16:08 Kernentwickler 1

Headless kommt bei größeren Firmen immer wieder zum Einsatz und das wollte ich jetzt vorab sagen, weil die Wahl des Content Management Systems mittlerweile eher darauf fußt, wie es sich in das bestehende Ökosystem einfügt. Vorher hattest du nichts und hast dann ein CMS gehabt.

00:16:31 Kernentwickler 1

Jetzt hast du von mir aus 5 andere Systeme ein PIM, ein DAM, ein CRM und das wird von mir aus auch gleichzeitig gebaut und da würde ich jetzt nicht von Microservices reden, sondern tatsächlich von Integration. Spricht mit den APIs, damit da Sachen zusammenlaufen und Headless ist für mich in dem Sinne wirklich nur die Ausgabe. Headless ist für mich auch am schwierigsten zu definieren, weil eine Website Headless zu machen, sodass sie JSON ausgibt ist cool, aber ich finde dabei ganz viele Dinge nicht berücksichtigt. Also es ist jetzt nicht so, als würde ich nur TYPO3 machen, ich habe auch Contentful-Projekte gemacht, Angular und ähnliches und da kommst du an so einen Punkt, dass du alles mit JavaScript machen kannst, und dann musst du eine Mail verschicken bei deinem Formular und dann fängst du halt wieder von vorne an. Die traditionellen Content Management Systeme haben diese Probleme alle gelöst und es steckt wesentlich mehr Erfahrung darin. Also Daseins-, Lebenszeit und das macht vieles einfacher. Das heißt, es kommt meistens schon von den Kundenanforderungen, was da gewünscht ist. Cloud ist immer eine Frage. Und dann ist das ein Beratungsthema.

00:18:06 Forscher

Was ist personaltechnisch aufwendiger, ein TYPO3 Projekt zu machen oder ein MACH-basiertes Projekt?

00:18:21 Kernentwickler 1

Hm, also wir haben uns auf TYPO3 spezialisiert, dementsprechend ist es für uns mit TYPO3 einfacher, hängt aber davon ab, welche Leute du hast. Wenn du halt PHP-Entwickler hast oder Erfahrung-mit-TYPO3-Entwickler, ist das kein Problem. Wenn du Leute hast, die vom Studium kommen und einfach nur JavaScript Projekte mit VueJS gemacht haben, ist wahrscheinlich eine MACH-basierte Lösung einfacher zum Starten.

00:18:46 Forscher

Würdest du sagen, dass mehr unterschiedliche Expertise bei MACH-basierten Systemen benötigt wird?

00:18:56 Kernentwickler 1

Nein, das würde ich nicht so sagen, das ist wie Äpfel und Birnen. Also hart gesagt. Also, ich weiß nicht wie viele andere MACH-basierte Systeme du dir angeschaut hast. Ich weiß jetzt gar nicht, wer in der MACH Alliance zum Beispiel so drin ist. Ich habe aber natürlich verschiedene Kunden oder verschiedene andere Content Management Systeme gesehen, die zum Beispiel Cloud-native sind und Headless und dann in die Cloud gehen, also zum Beispiel Magento ist Cloud-first, also diese Magento Commerce Plattform, und daran hängt ja auch das Adobe Experience Manager System und dort kannst du gar keine PHP-Sachen schreiben, logischerweise oder Java, sondern das läuft alles durch deren

Schnittstellenportal. Das heißt du kannst eigentlich nur noch dein Styling ändern. Du bist da viel eingeschränkter als bei dem System, mit dem ich arbeite.

00:20:22 Kernentwickler 1

Das heißt (..) und das find ich wiederum, glaub ich ganz gut. Bei MACH-basierten Lösungen geht es erstmal (..) natürlich geht es auch um die Technik, aber du musst dir viel mehr Gedanken machen über deine Content Architektur und das wünsche ich mir eigentlich für jedes Content Management System.

00:20:43 Kernentwickler 1

Als Hintergrund, ich hatte vorher gesagt, TYPO3 ist auch ein bisschen ein Composable CMS. Composable heißt: Im Vergleich zu den früheren Content Management Systemen, wo du ein Feld hast, in das du den Content für die ganze Seite einträgst, kannst du den Content wiederverwenden, indem du elementbasiert arbeitest. Also ich setze nicht irgendwo ein Bild rein, sondern ich habe wirklich ein Bild, was ich mehrfach verwenden kann, das Problem hatte TYPO3 vor 10 Jahren gelöst. Und ich kann ein Schlagwort hinterlegen und das kommt dann überall raus oder einen Alternativtitel.

00:21:20 Kernentwickler 1

Wo ich bei TYPO3 eine Richtung sehe, die wichtig ist, wäre Content Design. Bei MACH-basierten Lösungen startest du ja bei 0 und bei TYPO3 hast du sowas wie Text, Header, Text-mit-Bild oder Text-und-Medien, doch eigentlich musst du dem Content Bedeutung zuordnen. Dann wird jedes Content Management System besser. Das heißt, ich sage, ich habe hier keinen Datensatz Text-mit-Bild, sondern ich habe hier einen Personendatensatz, wo der Text die Beschreibung ist, die Headline der Name und das Bild der Ansprechpartner oder sowas. Also in der Hinsicht zu denken, das kann TYPO3 ja auch, denn es kommt out-of-the-box kommt mit generischen Content Blocks/Content Types. Bei einer MACH-Anwendung andererseits startest du bei 0 und musst das selbst designen.

00:22:12 Kernentwickler 1

Semantischer Content ist der wertvolle Teil und dann machen alle Sachen, die bei der MACH-Architektur drin sind auch Sinn. Wenn du anfängst ein MACH-system bei null zu benutzen und fängst an mit Text-mit-Bild, dann hast du nichts gewonnen.

00:23:52 Forscher

Wie wird der Nutzer in MACH-basierten Systemen bei der Orchestrierung und elastischen Skalierung unterstützt. Ich vermute das läuft ohne Zutun im Hintergrund. Wie funktioniert das bei TYPO3?

00:24:04 Kernentwickler 1

Egal welches System du nutzt, du kannst jedes superlangsam machen. Das ist so mein Tagesgeschäft, dass ich kucken muss, warum TYPO3 Systeme langsam sind. Du kannst TYPO3 superschnell machen.

00:24:22 Kernentwickler 1

Und wenn du skalieren möchtest, dann skalieren wir meistens, weil wir weltweit irgendwelche Kunden haben, also in China zum Beispiel. Dann bringt mir die Skalierung

mit Kubernetes-Clustern oder mit Cloud-Lösungen nichts, weil du dann andere sinnvollere Lösungen brauchst. Wir arbeiten viel mit CDNs, das heißt tatsächlich laufen die Seiten dann ziemlich schnell und dann ist es Cloud-native und hyperskalierbar.

00:24:55 Kernentwickler 1

Wenn du Seiten hast, die einfach viel Traffic haben, und zwar Spitzen haben. Also mein erstes Beispiel ist ein namhafter Bierhersteller, der während eines Fußballspiels im Fernsehen eine Gewinnspiel-Kampagne startet und dann gehen alle auf die Website. Das geht mit TYPO3 auch.

00:25:17 Kernentwickler 1

Mein zweites Bespiel wäre ein namhafter Konzertkartenhändler, der eine Kampagne startet, um Tickets für einen Superstar zu verkaufen und woraufhin die TYPO3-Seite down ging und seitdem wir das machen, passiert das nicht mehr. Da reden wir schon so von 200 bis vierhunderttausend Besuchern pro Sekunde und da skalieren wir nicht die Systeme im Hintergrund sondern, sondern wir haben einfach ein optimiertes Caching davor und da ist es auch völlig egal ob du MACH hast oder nicht. Beim Superbowl wird eine Werbung geschaltet und die verweist auf eine Drupal-Seite, also ich bin auch mit Drupal in Verbindung, und die Seite hält das natürlich auch aus, dass dann was weiß ich wie viele Millionen auf diese Webseite gehen.

00:26:02 Kernentwickler 1

Da kann man zwar die Bestandteile des Backends alle skalieren. Es ist aber meistens gar nicht das Thema, sondern einfach, dass du diese Anzahl von Besuchern abfangen kannst. Da wäre meist ein statischer Static Site Generator ausreichend, den es auch für die Headless-Anwendungen oder für die MACH-Anwendungen gibt. Bei wenigen dynamischen Inhalten, wie zum Beispiel einem Formular, reicht es ein System zu haben, welches statische HTML-Seiten ausgibt. Aber es gibt für alle Varianten, MACH- oder nicht Lösungen, um mit diesem Thema umzugehen.

00:26:46 Forscher

OK, das heißt für TYPO3 gibt es eigentlich kein Potenzial, wenn man an die Zerlegung in einzelne Microservices denkt?

00:26:56 Kernentwickler 1

Doch, da gibt es eigentlich nur eine Bestrebung und die gibt es schon länger. TYPO3 besteht aus zwei Komponenten, dem Frontend und dem Backend. Also Frontend ist die Webseite und Backend ist die Adminoberfläche und der erste Meilenstein in der Hinsicht ist ein nicht ausgesprochenes Ziel, denn es ist einfach ein sehr langer Weg über mindestens 5 Jahre bis dahin. Der Core von TYPO3 ist erweiterbar durch Extensions und der Core von TYPO3 selbst besteht aus ca. 30 Extensions. Eine davon ist das Frontend, eine das Backend und wir wollen das so entzerren, dass du theoretisch 10 Frontend-Nodes, also 10 Sites hochfahren kannst, die das Frontend handeln und ein Backend, was auf einer komplett anderen Infrastruktur läuft, um die Eingabe zu machen. Mit Blick auf das Frontend wollen wir standardmäßig weitere Ausgabeformate unterstützen, es gibt zwar die Headless-Erweiterung, mit der man JSON ausgeben kann oder seit 15 Jahren RSS-Feeds, aber wir wollen es noch einfacher machen verschiedene Ausgabeformate

auszugeben. Es gibt für Headless keinen einheitlichen Standard, was Content Management Systeme angeht, das ist ein bisschen schade.

00:28:53 Kernentwickler 1

Beim Punkt API-First würde ich dann eher sehen, dass dieses Backend, also die Adminoberfläche bei TYPO3 sehr strikt und fix ist. Wir arbeiten auch daran, dass das mehr Komponenten-basiert passiert, funktioniert und dafür bräuchtest du den API-First-Ansatz. Also das Lesen und Schreiben soll nicht nur mit HTML-Templates funktionieren. Das ist ein größerer Aufwand, aber das sind so für die 2 Bereiche, die 2 Ecken, wo wir intern auch eine Perspektive haben oder ich.

00:29:34 Forscher

Spannend. Und könnte man eigentlich alle Funktionen des Backends über solche APIs bereitstellen oder sind manche einfach zu komplex?

00:29:50 Kernentwickler 1

Nein, das geht. Prinzipiell ist das der Weg, aber aktuell haben wir es noch nicht. Es gibt ganz viele Faktoren, die da mit reinhängen. Das eine Thema ist, dass out-of-the-box, also sprich was dieser Core mitliefert, eine Authentifizierung nur auf Session-basis für den Client vorhanden ist, aber es ist keine Authentifizierung über einen Bearer-Token vorgesehen. Man kann sich reinhängen und das dazu bauen und das machen einige. aber zum Beispiel diese Authentifizierungsschicht oder auch die Berechtigungsschicht die TYPO3 als Stärke, habe ich noch in keinem anderen System gesehen, was MACH oder Nicht-MACH ist, um das einzugrenzen.

00:30:51 Kernentwickler 1

Beispiel Multi-Site, also Multi-Tenant-Support in TYPO3. Da habe ich beispielsweise 10 Projekte in einer Instanz und ein Redakteur darf nur die „Über uns“-Seite bearbeiten. Diese ganzen Commissions müsste man auch in die API-First-Variante abbilden. Das ist natürlich ein bisschen schwieriger, das nachher in der GUI zu machen. Aber an sich hat TYPO3 viel und der Prozess das zu migrieren in eine flexiblere Variante, der zieht ist halt einfach ein Monster-Projekt, aber wir gehen Schritt für Schritt in diese Richtung.

00:31:12 Kernentwickler 1

Früher, vielleicht noch als kleiner Hintergrund, wurde TYPO3 ganz allein von Kasper Skårhøj entwickelt und konzeptioniert. Später hat er das geöffnet und war dann so eine Art Gatekeeper, das heißt man konnte ihm Patches schicken und er hat sich um die Zusammenführung mit dem vorhandenen Code gekümmert. Danach gab es ein Team und die haben sich ein bisschen untereinander abgestimmt, dann ist Kasper weg und es gab einen neuen Team-Lead. Vor mir gab es zwei Team-Leads.

00:32:04 Kernentwickler 1

Vor mir war das so ein bisschen Patchwork, also wirklich im Sinne von Patchwork. Da kam einer um die Ecke, der sagte: „Ich hab hier ein ganz großes neues Feature, das ist fertig“, und dann haben sich das gewisse Leute angeschaut und gesagt: „Ja, okay passt so.“ Und du musst dir einfach vorstellen, Patchwork heißt, die unterschiedlichen Handschriften der Leute stecken in diesem Projekt und die Hauptaufgabe ist, dass wir diese Sachen

konsolidieren, also zum Beispiel keine fünf verschiedenen JavaScript-Frameworks im Backend zu haben und es nachher wartbar zu machen, damit wir schneller in neue Richtungen gehen können. Und das ist eigentlich ein Prozess, den wir seit ja 7, 8 Jahren fahren. Immer neben der Feature-Entwicklung, dass wir schauen, dass dieses Grundsetup, das Grundsystem solide ist.

00:32:56 Kernentwickler 1

Also vielleicht hast du schon mal diesen Begriff Extbase gehört? Das sind also Plugins für Extension-Autoren, die können ein Extbase-Plugin schreiben. Die Schwierigkeit besteht darin, dass Extbase bisher auch eine Extension ist, die mit dem Core kompatibel gehalten werden muss. Diese Extension muss mit dem Core zusammengeführt werden, damit wird Dokumentationen, Pflege und Wartung vereinfacht und man kann sich schneller in neue Innovationsrichtungen bewegen.

00:33:37 Kernentwickler 1

Eine Stärke von TYPO3 ist zudem die Rückwärtskompatibilität. Sie hindert vieles, aber die Vorteile überwiegen die Nachteile. Statt einfach ein komplettes Feature abzuschalten und die Kunden sich selbst damit zu überlassen, prüfen wir immer, wird das verwendet und wie wird das verwendet? Das ist bei einem System, das, zum Beispiel, Cloud-First oder Cloud-Only ist, wesentlich einfacher, weil man mehr Statistiken hat und das haben wir eher wenig.

00:34:08 Forscher

OK, ja das sind viele spannende Einblicke.

00:34:14 Kernentwickler 1

Ich muss noch eine Sache zu API-First sagen. Also das Schöne ist ja, wenn du API-First hast, bedeutet dass, das die Frontend-Entwickler oder JavaScript-Entwickler, eine API bzw. eine Doku haben mit der sie arbeiten können und auf der anderen Seite haben sie eine GUI, um ihr Headless System zu managen, also Content zu managen. Das muss gar nicht zwingend über diese GUI sein, denn du kommst ja auch komplett API-Only arbeiten, aber da kenne ich eigentlich kein System.

00:34:42 Kernentwickler 1

Und API-First heißt, dass diese Adminoberfläche auch komplett über diese API läuft und das ist alles cool. Eines der Hauptvorteile ist, dass immer, wenn du was an der API änderst, dass du, dass Du das einmal definiert hast, was denn möglich ist und TYPO3 kommt von dem Hintergrund, alles ist möglich, immer, und das, weil jede Extension kann ja wirklich Schnee regnen lassen auf deiner Webseite und keine Ahnung, dass so viel möglich war, dass wir das immer ein bisschen besser einschränken müssen, um diese API, die auf PHP-Seite da ist, zu schärfen und zu definieren. Und das macht es schon schwierig, ja, oder deswegen ist es ein bisschen langwierig.

00:35:38 Forscher

Ja, kann ich mir vorstellen, dass man an vielen Stellen die Schnittstellen noch klarer definieren muss, wenn das so über die Jahre gewachsen ist. Fällt dir noch was ein, wie man TYPO3 noch Cloud-freundlicher machen kann? Wir haben schon einiges angesprochen.

00:36:10 Kernentwickler 1

Das, das sind hauptsächlich Dokumentationsthemen. Also es sollte eine Anleitung geben und Boilerplate-Beispiele geben, um so ein TYPO3 in der Cloud einfach aufzusetzen. Ich sag jetzt mal innerhalb von ein paar Minuten. Diese Bestrebungen gibt es und auch verschiedene Vorstöße aus der Community. Es gilt die halt eben auch zusammenzukehren und zusammenzufassen. DA bin ich ein bisschen involviert. Aktuell ist es da auch so, dass jeder, der die Anforderungen hat, durch Kundenprojekte, das auch selbst macht und es geht darum da einen gemeinsamen Standard zu finden. Ich bin da nur halb involviert, weil das System TYPO3, also dieser Core das eigentlich schon kann.

00:37:20 Kernentwickler 1

Ein Beispiel ist die Stadt Quebec, deren Webseite läuft mit TYPO3. Die machen auch alles auf AWS und Co. Und dann kam diese ganze Covid-Welle und dann hatten die 10 mal so viel Traffic auf ihrer Seite, und zwar dauerhaft und die haben das irgendwie alles skaliert und ich kann dir mindestens 15 andere Beispiele von Firmen nennen, die das Einsetzen.

00:37:47 Kernentwickler 1

Und dann bin ich der Meinung: Ok, wir müssen uns alle an einen Tisch setzen und ihr schickt mir eine Liste von Sachen, die noch nicht Cloud-freundlich sind, die mit dem Core System zu tun haben und dann kann ich konkret an diesen Aufgaben arbeiten. Stand heute habe ich einige Projekte im Einsatz, die einfach in der Cloud laufen und das funktioniert. Das heißt was funktioniert denn nicht, wäre meine konkrete Frage, außer Dokumentation und einheitliche Standards, weil es jeder anders macht.

00:38:14 Forscher

Ja, also es bedarf einfach nur, dass man sich ne individuelle Lösung baut, aber man bekommt es nicht so out of the Box, dass man direkt hergehen kann, vielleicht einen Container hat, den man dann in der Cloud deployed und dann funktioniert alles im Wesentlichen.

00:38:38 Kernentwickler 1

Das ist so der Weg. Das ist der Weg dahin, den wir jetzt gerade gehen. Es ist nicht mein Hauptthema in der TYPO3 Core Entwicklung, Container zur Verfügung zu stellen, die alles können oder die dieses Basis-Setup mitbringen, dafür gibt es andere in der Community.

00:39:19 Forscher

Dieser ganze Prozess, bei dem man schaut, dass die PHP-APIs über andere APIs verfügbar gemacht werden, sind dem irgendwelche Grenzen gesetzt?

00:39:41 Kernentwickler 1

Ich weiß nicht genau, was Du mit der Frage hören möchtest. Aber die Grenze ist schon, es gibt gewisse Sachen, bei denen ich als Lead der Entwicklung sage, daran rütteln wir nicht. Es klingt so dramatisch, aber letztendlich heißt es ja, wir werden auch aufgrund der Historie oder der letzten 10, 15 Jahre zum Beispiel nicht entscheiden: Ab morgen, gibt es eine neue Version und die funktioniert nicht mehr mit PHP, sondern die läuft nur noch mit Node.js und JavaScript.

00:40:25 Kernentwickler 1

Also das das werden wir nicht machen und das geht vielleicht in deine Richtung, was die Cloud betrifft. Wir werden eigentlich immer auch ein Basissystem anbieten, was heißt immer, ich weiß nicht, was in Zukunft ist, aber mein Bestreben ist es auch, die dreihunderttausend Instanzen, die wir haben, auch weiterhin benutzen zu können.

00:41:18 Kernentwickler 1

Ja, also es gibt viele, viele Dinge, die wir aus meiner Sicht wahrscheinlich nicht in eine Version, reinpacken können, die wir jedem User, jedem Betreiber einer Website von TYPO3 zur Verfügung stellen können, sondern das hängt immer ein bisschen von der Infrastruktur ab und den Systemanforderungen. Wenn es dann eine SaaS-Lösung gibt, dann kann man da natürlich wesentlich mehr Funktionen einbauen, die robuster sind. Wir haben zum Beispiel MySQL als Datenbasis, das kannst du jetzt nicht einfach wechseln.

00:41:58 Kernentwickler 1

Oder wir können nicht sagen, du brauchst jetzt immer Solr oder Elasticsearch als Basis.

00:42:07 Kernentwickler 1

Und das heißt, dass wir im Grund Core diese APIs verbessern können, aber trotzdem diese APIs anbieten müssten, dass man das sehr gut und besser definiert erweitern kann und andere Systeme unter der Haube dazu bauen kann. Und daraus könnten auch eine oder mehrere SaaS-Lösungen entstehen, die Features haben, die das Basisprodukt nicht hat.

00:42:40 Forscher

Ok wir haben ziemlich viele meiner Punkte, meiner Fragen angesprochen. Ich bin sehr zufrieden.

00:42:55 Kernentwickler 1

Ich würde noch eine Sache nur zur Klärung mit reinbringen. Du musst dir vorstellen, es gibt die TYPO3 Association, da gibt es das Association Board, also einen Vorstand. Daran hängen die Teams. Es gibt circa 12 Teams und du kannst ein Team Leiter sein, ohne dass Du Association Mitglied bist. Die Association ist so der Überbegriff und ist überall drin. Die Association ist ein Verein und hat eine hundertprozentige Tochter, die TYPO3 GmbH, mit der ich nichts zu tun habe. Ich arbeite da nicht, also ich habe schon mit der zu tun, also ich bin da im regelmäßigen Austausch, aber ich sehe mich immer mehr als dritten Teil, nämlich neben der Association. Die Firma ist darauf aus in irgendeiner Form Gewinn zu machen und das Projekt und das Produkt voranzubringen.

00:44:05 Kernentwickler 1

Und dann gibt es den dritten Teil der Community und den möchte ich auch abdecken. Ich hab viele Kundenprojekte und ich möchte die Erfahrungen teilen und den Austausch mit der Community und dort auch ein offenes Ohr haben und gehöre damit zum dritten Teil, zur Community.

00:44:26 Kernentwickler 1

Dementsprechend sind das meine Ziele, die wir hier besprochen haben, was ich natürlich mit der Association und der GmbH abstimme. Das ist jetzt aber nicht die Cloud-Lösung, weil das auch noch eines deiner Fragen waren.

00:44:46 Kernentwickler 1

Eine Produktstrategie ist aktuell in der Entwicklung mit der TYPO3 Association, der TYPO3 GmbH und den Vertretern der Community, wie mir logischerweise. Und wenn es eine Cloud Lösung gibt, die man einfach klicken kann und kaufen kann, dann ziemlich sicher nicht von mir, weil ich die Community bin und das müsste dann eher bei der GmbH verankert sein.

00:45:17 Kernentwickler 1

Und dementsprechend bin ich da auch dafür, aber wie das nachher im Detail dann aussehen könnte, sowas als offizielle Lösung zu machen, da kann ich wenig dazu sagen, was verlässlich ist, um es mal so zu sagen.

00:45:39 Forscher

Noch eine abschließende Frage. Was wünscht du dir in Zukunft für die Arbeit mit TYPO3?

00:45:52 Kernentwickler 1

Ich wünsche mir, dass es einfacher und verständlicher ist ein TYPO3 Projekt aufzusetzen, was ready-to-go ist, also einfach direkt funktionieren kann, sowohl was die Ausgabe der Website oder andere Formate betrifft.

00:46:12 Kernentwickler 1

Und dass es auch für die Redakteure, Spaß macht dieses System zu benutzen, weil es so einfach ist und trotzdem so mächtig. Und diesen Spagat, den finde ich auch am spannendsten an TYPO3 auch, dass die Leute nicht sagen: „Oh, das ist viel zu kompliziert WordPress ist einfacher.“ Natürlich ist das auch Gewohnheitssache, aber ich denke, da kann man sich auch viel von Lösungen, Content Management Systemen, die sich die letzten 7 Jahre so entwickelt haben, abschauen, wie die Konzepte heutzutage lösen. Also sowohl von der API, aber auch natürlich von der GUI. Da ist auch Teil meines Jobs, dass ich da auch immer ein bisschen über den Tellerrand schaue.

2.2 Interview B

00:00:03 Forscher

Seit wie vielen Jahren beschäftigst du dich mit Content Management Systemen?

00:00:10 Kernentwickler 2

Über 20 fürchte ich.

00:00:12 Forscher

OK, was hat dich dazu bewegt in dem Bereich zu arbeiten?

00:00:18 Kernentwickler 2

Tatsächlich wollte ich mich in ein größeres Open Source System einbringen und bin dann im Verlauf des Studiums bei meinem ersten Arbeitgeber gelandet, eine klassische Werbeagentur und dort hat man irgendwann erkannt, dass sie nur überleben kann, wenn sie anfängt, Webseiten etwas professioneller zu bauen, und da bin ich dann zügig auf Content-Management-Systeme gestoßen und habe in TYPO3 erkannt, dass das eventuell das Projekt sein könnte, in das ich mich tatsächlich auch irgendwie sinnvoll einbringen kann.

00:01:09 Forscher

OK, es war also auch dein erstes CMS.

00:01:11 Kernentwickler 2

Damit war TYPO3 tatsächlich mein erstes CMS. Es gab zu dem Zeitpunkt eine Palette anderer, die aber aufgrund der ersten Kundenanforderungen frühzeitig ausschieden, weil sie einfach nicht leistungsfähig genug waren und dann wurde es schon relativ zügig, relativ dünn und TYPO3 ist dann übriggeblieben, unter anderem weil es bereits zu dem Zeitpunkt bereits Extension-fähig war, das war damals relativ neu, dass man so ein Software-Produkt großflächig erweitern kann.

00:01:55 Kernentwickler 2

Das ist auch immer noch eine der Stärken des Systems, das man an vieles drankommt und vieles sinnvoll erweitern kann.

00:02:07 Forscher

Hast du dann im Verlauf auch mit anderen CMS Erfahrungen gemacht?

00:02:13 Kernentwickler 2

Rudimentär. Gelegentlich kuckt man so ein bisschen über den Tellerrand. Ich habe mit kommerziellen Systemen wenig bis gar nichts zu tun gehabt. Gelegentlich ist man mal in WordPress unterwegs und schüttelt sich bei der Codequalität. Das ist halt so ein bisschen schwierig.

00:02:31 Kernentwickler 2

Von Drupal habe ich mal ein paar Konzepte gesehen, NEOS so ein bisschen, aber auch am Ende nur Konzepte, wie es denn dann gelöst wird, das heißt, ich bin schon relativ stark auf TYPO3 fokussiert und bekomme auch in meiner Rolle nicht unglaublich viel von anderen Produkten mit.

00:02:59 Forscher

OK, wie kann man deine Rolle beschreiben, worauf hast du dich spezialisiert?

00:03:09 Kernentwickler 2

Ich bin Mitarbeiter einer Agentur, wir machen ausschließlich TYPO3 Projekte auf einem ziemlich hohen Niveau und man hat mich eingestellt, um Vollzeit TYPO3-Entwicklung zu machen. Das heißt ich mache wenige bis gar keine Projekte. Ich werde gelegentlich mal hinzugezogen, wenn es Core-Probleme gibt, die zu fixen sind, die in einem Projekt auffallen und löse dann im Prinzip die Core Issues oder berate bei Dingen, die irgendwie sinnvoll zu lösen sind mit TYPO3.

00:03:52 Kernentwickler 2

Aber meine Kernaufgabe ist Core Entwicklung. Da bin ich hauptsächlich im Refactoring des Systems unterwegs. Das heißt, ich modernisiere die Komponenten, stelle sie flexibler auf und löse Issues, die oft schon seit vielen Jahren im System schlummern. Ich wühle ganz unten.

00:04:48 Kernentwickler 2

Also ich bohre so die dickeren Bretter, alles das, was nicht so nebenbei machbar ist, wo du also tatsächlich jemanden brauchst, der in der Lage ist, sich Wochen oder Monate mit einzelnen Dingen herumzuschlagen, um die substanziell besser aufzustellen.

00:05:11 Forscher

Wird das dann durch Service-Level-Agreements finanziert, also dass die Kunden zu euch kommen, wenn sie Probleme mit dem Core haben? Oder wird das durch die TYPO3 GmbH finanziert?

00:05:30 Kernentwickler 2

Beides, also ja, es gibt Issues, bei denen Kunden, unsere Agentur kontaktieren und Anfragen, ob wir bestimmte Dinge im Core lösen können. Das passiert. Ich werde aber auch über die gelösten Issues im Core, also auf dem Bug-Tracker Issue-System von der TYPO3 GmbH querfinanziert.

00:06:04 Forscher

OK, ja, spannend. Dann kommen wir zu den Bestandteilen der MACH-Architektur, den Microservices, API-Fist, Cloud-Native SaaS und Headless. In welchen Kontext sind dir die Stichwörter schon mal begegnet?

00:06:33 Kernentwickler 2

Ich überspringe Microservices, weil das ist, glaube ich so momentan, für mich in PHP das Schwammigste.

00:06:42 Kernentwickler 2

API-First, hab ich bei verschiedenen Produkten einfach schon gesehen. Unser Review System Gerrit ist API-First. Das heißt, die die Oberfläche da drauf macht nichts anderes als API-Calls. Und man kann alles auch irgendwie zu Fuß mit einem Curl-Request ziehen.

00:07:00 Kernentwickler 2

Ebenfalls API-First ist GitLab, wo ich auch schon bis zur gewissen Tiefe eingedrungen bin, einfach deshalb, weil ich die TYPO3 Core CI, also die TYPO3 Continuous Integration Struktur pflege, warte und aufbaue und in dem Zuge die Runner betreibe, also Core Runner und was noch und daher kenne ich so ein bisschen die API von GitLab und die machen das ähnlich die Oberfläche sind API-Calls. Auch wenn man da in den Issue-Trackern sucht, dann sieht man, dass das entsprechend entwickelt wird.

00:07:56 Kernentwickler 2

Cloud-native Software-as-a-Service habe ich wahrscheinlich das erste Mal in Action gesehen mit Travis auch ein CI-Dienst, der heutzutage nicht mehr stark verwendet wird, weil die sich ihr eigenes Modell zerstört haben, aber das war für uns im Core so der erste Software-As-a-Service-Dienst, den wir regelmäßig genutzt haben, um halt CI zu machen. Ich denke in dem Kontext ist mir das das erste Mal begegnet.

00:08:37 Kernentwickler 2

Headless immer mal wieder, auch schon seit vielen Jahren, dass man mal einen Kunden hat, der seine Daten Headless beziehen möchte, über API-Calls, im simpelsten Fall, dass Daten als JSON ausgeliefert werden sollen. Und das kommt schon seit längerer Zeit immer mal wieder in den Projekten vor.

00:09:21 Forscher

Du meintest Microservices sind im Moment kein Thema bei PHP. Inwiefern?

00:09:30 Kernentwickler 2

Nee, ich würde das anders formulieren. Ich würde nicht sagen, dass das kein Thema ist, aber ich glaube, dass das bei vielen PHP-Projekten ein kleineres Problem ist, als man denkt, was schlicht daran liegt, dass PHP im Kern Request-orientiert ist. Das heißt, ich habe einen HTTP-Request, der kommt an meinem PHP an, wird verarbeitet, dann kommt ein Response zurück und dann ist es vorbei und dann kommt der nächste Request.

00:10:07 Kernentwickler 2

Das heißt, ich hab außer der Datenhaltung unten drunter, wenig übergreifenden State. Ich habe jetzt also nicht ein Java, das mir jede Menge Objekte im Speicher hält, einfach deshalb, weil das beim Request einmal aufgebaut und danach wieder weggeschmissen wird. Das heißt PHP by Nature skaliert gut flach erstmal.

00:10:39 Kernentwickler 2

Damit ist so ein bisschen die Frage, wie man Microservice an der Stelle versteht. Man könnte also, wenn man dieses Thema verfolgt über Leute, die einen Artikel darüberschreiben und dann auf Mastodon oder Twitter posten oder so. Dann gibt es da Leute, die halt postulieren, dass PHP in dem Sinne erstmal das Problem nicht hat.

00:11:11 Kernentwickler 2

Ja, zusammen, kombiniert mit TYPO3 ergibt sich halt, dass man den Monolithen TYPO3 gut in kleinere Stücke hacken kann. Ich kann also zum Beispiel eine Cloud-Infrastruktur aufbauen, bei der ich viele Frontend-Prozesse habe, die das Backend nicht können oder den Zugriff darauf verweigern. Das würde aus Sicherheitsgründen gerne mal gemacht.

00:11:51 Kernentwickler 2

Und ich kann auch, zum Beispiel, die Sachen, die per CLI aufgerufen werden, zentral herausfiltern und da einen eigenen Node für machen und auch das Skalieren, wenn das denn dann nötig ist. Das heißt ich habe als Single Point of Failure in dem Sinne die Datenbank, die man auch skalieren kann, wenn man die passenden Datenbankprodukte dafür hernimmt.

00:12:19 Kernentwickler 2

Damit kann man TYPO3 schon gut verteilen und skalierbar machen. Und das liegt unter anderem daran, dass das Halt einfach ein PHP-Produkt ist. Das heißt, das wäre so ein bisschen meine Ansicht dazu. In dem Sinne ist das Produkt nicht so monolithisch wie man vielleicht am Anfang glauben könnte.

00:12:56 Forscher

Die MACH-basierten Systeme vermitteln, dass die monolithischen Systeme nicht horizontal und elastisch im Cloud-Umfeld skalieren können, weil sie eben aus einem untrennbaren Ganzen bestehen, aber wenn du sagst, dass sich TYPO3 gut zerlegen lässt, dann wäre das ja widerlegt.

00:13:30 Kernentwickler 2

Ja, es gibt da verschiedene Leute, die da tatsächlich gerade zu PHP und Microservices auch Artikel im Netz haben und so argumentieren, die also sagen PHP lässt sich relativ gut zerlegen, weil das halt Request-orientiert arbeitet und wenn ich Microservices strikt nehme, dann würde das bedeuten, ich habe ganz viele kleine Dienste verteilt, die alle miteinander sprechen können. Dafür brauche ich dann die passenden Schnittstellen zwischen diesen einzelnen Diensten, damit die miteinander sprechen können und austauschbarer werden.

00:14:04 Kernentwickler 2

Das ist dann bei TYPO3 nicht mehr so hart der Fall. Also ich kann bestimmte Komponenten trennen, aber ich habe oftmals keine starke API zwischen diesen Komponenten. Dann wird es schwierig.

00:14:26 Forscher

Und der Aspekt der Zustandslosigkeit wäre ja auch ein bisschen schwierig zu realisieren bei TYPO3, weil es diese Stärke hat, dass es über eine ausgefeilte Rechtverwaltung verfügt, die auf Session-Basis funktioniert. Ist das richtig?

00:14:52 Kernentwickler 2

Also, um im Backend administrieren zu können, muss ich eingeloggt sein und brauche damit eine Session. Die Session API von TYPO3 ist aber so, dass ich die im Notfall auch verteilen kann. Wenn also mein Single-Point-of-Failure nicht mehr funktioniert.

00:15:20 Kernentwickler 2

Vielleicht ist die interessantere Frage an der Stelle sowas wie Caching. Da ist dann eher die Frage, auf welchem Level spielt man. Wenn ich viele Cloud-basierte Nodes habe, die möglicherweise auch automatisch rauf und runter skalieren in der Anzahl, dann gibt es bestimmte Komponenten, die man halt auf den einzelnen Nodes cashed und es gibt andere, die man globaler cashen möchte. Und bei all diesen Varianten hat TYPO3 eine relativ starke Möglichkeit, ordentlich einzugreifen und das zu realisieren.

00:16:00 Kernentwickler 2

Also wenn dir zum Beispiel der bei Default datenbankbasierte Page-Cash zu groß wird, dann kann ich da ausweichen und ich könnte auch pro Node eigene Cashes daran hängen.

00:16:13 Kernentwickler 2

Ich kann noch zentral fahren und die dateibasierten Caches, die würde ich üblicherweise auf den einzelnen Notes fahren, dann sind die unabhängig voneinander. Das geht auch. Das ist auch in den letzten Jahren etabliert worden, weil zum Beispiel, ich weiß nicht, ob das heute noch so ist, die Google Cloud Infrastruktur nicht erlaubt, dass man in die PHP-Container Dateien schreibt. Das heißt der ganze PHP-Container ist read-only.

00:16:47 Kernentwickler 2

Und wenn man denn dann doch auf so einem Rechen-Node, auf so einem Frontend-Node dynamische Dateien generiert, wie zum Beispiel, verschiedene Caches, die in Bootstrap gebaut werden, wie Dependency Injection etwa, dann müssen die in ein eigenes Overlay rein, das schreibbar ist. Und diese Situation hat uns dazu gezwungen, im System bereits hinzugehen und diese Arten von Directories voneinander sinnvoll zu trennen.

00:17:27 Kernentwickler 2

Dass es also nicht großartig gemischt wird und ich in der Lage bin, zum Beispiel ein read-only PHP-Container zu deployen. Und das sind dann Cashes, die lokal liegen. Also es gibt unterschiedliche Aspekte, die erlauben Dinge zentral zu machen, oder zumindest teilweise auch verteilt vorzubehalten.

00:17:59 Forscher

Okay, jetzt sind wir schon zu den Details in Sachen Microservices vorgedrungen aus dem späteren Teil meines Leitfadens, aber es bot sich gerade so an.

00:18:16 Kernentwickler 2

Ich hoffe, ich bringe das jetzt nicht komplett durcheinander, dass ich da jetzt schon ein bisschen tiefer reingegriffen habe bei den Microservices. Das Problem ist auch, das das so ein blödes Buzzword ist, bei dem wahrscheinlich 5 Leute 8 verschiedene Verständnisse davon haben.

00:19:17 Kernentwickler 2

Die anderen 3 Punkte sind klarer. API-First ist klar. Headless überschneidet sich so ein bisschen mit API-First, denn das kommt dann quasi mit. Und Cloud-native SaaS, da ist halt spannend, dass das auch relativ häufig misused wird. Also zu sagen: Wir sind Cloud-native SaaS und On-Premise geht nicht. Das ist ein Ziel, das wir nicht verfolgen.

00:20:33 Forscher

Die Frage wäre mehr, ob der Betrieb von TYPO3 in der Cloud optimiert werden könnte. Denkst du es besteht Potenzial bei TYPO3 darin, dass man mehrere Nutzer in eine Instanz bringt und diese sich dann die Hardwareressourcen teilen können?

00:22:15 Kernentwickler 2

Das würde man vermutlich eher vermeiden, also spätestens dann, wenn einer dieser Kunden API-Zugriff bekommt, sieht er alle Instanzen. Aber Multi-Domain-Hosting geht in TYPO3 seid immer. Wir kennen Instanzen bei denen 500 Webseiten in einer Instanz hängen. Und da sind die Zugriffe entsprechend über Backend-Benutzerrollen voneinander getrennt, sodass der Editor der einen Instanz, die andere gar nicht sieht.

00:23:05 Kernentwickler 2

Eine Rieseninstanz zu machen, in der ich ganz viele unterschiedliche Kunden habe, weiß ich nicht, ob das ein sinnvolles Geschäftsmodell an der Stelle ist. Ich würde annehmen, dass ein toujou hingeht und getrennte Instanzen pro Kunde fährt.

00:23:28 Forscher

Ok dann würde das keine Vorteile bringen deiner Meinung nach, weil das ist so ein Punkt in der Cloud-Native Philosophie, dass sich mehrere Klienten in einer Instanz befinden und die sich dann elastisch skaliert, je nachdem, welcher Kunde gerade Bedarf hat und vielleicht spart das ein bisschen Ressourcen, könnte ich mir vorstellen.

00:24:10 Kernentwickler 2

Ja das passt alles in diese Cloud-native Strategie rein, wenn ich alles über Microservices entkoppele, weil ich im Prinzip pro Kunde nur die Sachen anbinde, die dieser braucht. Das wird bei einem, in dem Sinne dann wieder, Monolithen, wie TYPO3 schwieriger, weil ich eine Instanz habe, die dann alles können müsste. So das will ich dann wahrscheinlich auch nicht tun.

00:24:37 Forscher

Okay, kommen wir noch mal zurück zum zweiten Teil meines Leitfadens. Bevor wir später wieder zum Technischen zurückkehren. Auf welchen Markt fokussiert sich TYPO3 aktuell?

00:24:56 Kernentwickler 2

Ich würde sagen Mittelstand, also kleine bis mittelgroße Auftritte, Behörden und Institutionen. Da ist TYPO3 klassisch vertreten. Das heißt nicht, dass wir kleine Instanzen aus dem Blick verlieren. Die sind für das Open Source System wichtig ist, weil da so ein bisschen junges Blut herkommt.

00:25:37 Kernentwickler 2

Aber die Hauptanwendungsfelder sind denke ich, der etwas anspruchsvollere Mittelstand, sowie größere Auftritte von Behörden und Institutionen.

00:25:54 Forscher

OK, und was sorgt dafür Innovationsdruck (Politisch, Ökonomisch, Soziokulturell, Ökologisch-geografisch, Technologisch, Rechtlich)?

00:26:06 Kernentwickler 2

Genau, ich habe die Punkte gelesen und habe mir schon ein paar Gedanken dazu gemacht.

00:26:13 Forscher

Ökonomisch.

00:26:19 Kernentwickler 2

Dass wir das System vereinfachen und leichter zugänglich machen müssen, insgesamt, sowohl in der Entwicklung als in der Bedienung. Das ist zum jetzigen Zeitpunkt ein starkes Topic in der gesamten Entwicklung.

00:26:40 Kernentwickler 2

Weil wir am Ende des Tages tatsächlich einfach ein Legacy-System sind. Die Software ist nicht die allerjüngste. Sie ist nicht vor 2 Jahren from-Scratch geschrieben worden, sondern wir tragen 20 Jahre Software durch die Gegend und da sind wir ständig am Modernisieren und am Verbessern. Und das müssen wir auch tun. Wir können das nicht einfach liegen lassen, so wie es ist für immer. Das heißt, wir haben technologischen Modernisierungsdruck.

00:27:16 Kernentwickler 2

Und aus meiner Sicht auch ökologisch, denn ich würde mal steil behaupten, dass das Softwareprodukt TYPO3 eines der wenigen Produkte ist, das systematisch von Version zu Version schneller wird und damit weniger Ressourcen verbraucht und man sich immer mehr Instanzen auf immer weniger Hardware kleben kann. Und das ist auch für mich persönlich ein wichtiger Aspekt. Die großen Bremsen aus dem System auszubauen und zu verbessern, damit wir noch schneller antworten können und noch weniger verbrauchen, um Instanzen zu hosten.

00:28:07 Forscher

Und gibt es da Unterschiede zur Konkurrenz? Entwickeln die sie sich in eine andere Richtung?

00:28:18 Kernentwickler 2

Die Frage ist jetzt, wer ist die Konkurrenz? Also da ich die kommerziellen Produkte nicht gut kenne, kann ich das nicht beurteilen. Ich kann ein bisschen was zu WordPress sagen an der Stelle, das als CMS weltweit extreme Verbreitung gefunden hat und WordPress hat ein anderes Entwicklungs-Modell, technologisch. WordPress entwickelt sich nicht weiter.

00:28:51 Kernentwickler 2

Da wird nicht strukturell umgebaut. Das sieht man zügig im Code, wenn man da anfängt reinzuschauen und das ist ein valides Entwicklungsmodell, denn man kann argumentieren: „Wir bauen keine größeren Sachen um, weil dann würden wir ja irgendwas kaputt machen, was vielleicht Erweiterungen benutzen, damit bleibt ein WordPress, technologisch auf dem technischen Stand, den es vor 15 oder 20 Jahren schon hatte. Und erfindet sich in dem Sinne nicht neu.

00:29:27 Kernentwickler 2

Und da grenzt sich TYPO3 relativ stark ab, weil wir das Gegenteil davon machen. Vielleicht nicht das direkte Gegenteil, aber wir haben da ein anderes Entwicklungsmodell und sagen: Wir wollen mit aktueller Architektur mitgehen und renovieren uns regelmäßig durch und nehmen dabei in Kauf, dass zum Beispiel Erweiterungen sich dann auch regelmäßig anpassen müssen bei neuen Major-Releases und dazu haben wir auch die passenden Strategien entwickelt, um das zu begleiten. Also Änderungen werden ordentlich dokumentiert und man gibt den Entwicklern und den Editoren Informationen an die Hand, inwieweit sie sich selbst anpassen müssen oder wie Arbeiten oder Workflows jetzt laufen, nachdem wir das geändert haben.

00:30:37 Kernentwickler 2

Richtung „Technologisch“ habe ich stärker nachgedacht, weil gerade dieses Tupel aus WordPress, Drupal und TYPO3 drei verschiedene Entwicklungsmodelle fährt und die kann man an diesen drei Produkten auch ganz gut festmachen und alle 3 davon sind valide.

00:31:03 Forscher

Ja, da wir jetzt schon den ökologischen Aspekt angesprochen haben, würdest du sagen das lässt sich auf jeden Fall besser mit der Architektur von TYPO3 lösen als mit einer MACH-basierten?

00:31:25 Kernentwickler 2

Weiß ich nicht. Ich habe den Eindruck, dass Cloud-basiertes Hosting oft mehr Overhead mitbringt, als es reduziert und dass der Vorteil viele unterschiedliche Dinge auf eine Harfe zu packen am Ende des Tages wahrscheinlich gar nicht ordentlich ausgespielt wird.

00:32:00 Kernentwickler 2

Ja, also es gibt einen Grund, dass wenn ich meine Applikationen in die Cloud packe, ich üblicherweise ans Hosting eine 0 hänge. Das ist so der Klassiker. Ich gehe von einem On-Premise oder von einem Standard-spezialisierten Hosting mit meiner Applikation in die Cloud und das kostet mich das Hosting das Zehnfache und das hat Gründe. Der Grund ist nicht nur, dass der Anbieter daran sehr viel Geld verdient, sondern auch, dass da üblicherweise mehr Hardware verwendet wird als weniger. Und wenn ich das tatsächlich

zu Ende spiele, eine Microservice-Architektur, dann bekomme ich per Definition mehr Overhead rein, einfach weil ich viel mehr Netzwerkkommunikation erzeuge, zum Beispiel, und viel mehr Daten verpackt, über das Netzwerk geschoben und wieder entpackt werden müssen. So, das heißt, man hat definitiv dadurch einen höheren Overhead.

00:33:07 Kernentwickler 2

Ob das dann am Ende tatsächlich besser funktioniert und ob das ökologisch die sinnvollere Lösung ist, das trotzdem zu machen, kann ich nicht wirklich beurteilen. Ich bin da ein bisschen kritisch.

00:33:22 Forscher

Ja, das klingt erstmal plausibel, wie du das erklärt hast. Denkst du, dass TYPO3 in Konkurrenz zu MACH-basierten CMS steht?

00:33:45 Kernentwickler 2

Erstmal stehen wir in Konkurrenz zu allen anderen CMS, wobei die Open Source CMS gerade die PHP-basierten mehr und mehr zusammenarbeiten. Insgesamt die PHP-Projekte ohnehin. Ich weiß nicht, ob du das mitbekommen hast. Das TYPO3 Documentation Team stellt gerade das Documentation Rendering auf neue Beine und das Ganze ist REST-basiert, reStructuredText.

00:34:29 Kernentwickler 2

Das bisherige Rendering ist eine Python Lösung mit Sphinx darunter und das war so ein bisschen unbefriedigend, und zwar für mehrere PHP-Projekte, die da Probleme hatten. Dazu gehören namentlich Symphony und Doctrine DBAL, wenn ich das richtig im Kopf habe und diese beiden Projekte haben sich zusammengetan, haben gesagt, wir lösen das und zum jetzigen Zeitpunkt haben wir ein Rendering der Dokumentation, das im Prinzip von diesen 3 Projekten gestützt und gebaut wurde, aktiv. Das heißt, wir haben durchaus gerade im Open Source Bereich ein gutes Miteinander und bieten im Prinzip eine Palette von verschiedenen Produkten, auch teilweise im selben Bereich an.

00:35:31 Kernentwickler 2

Es gibt auch viele Agenturen, die einfach mehrere CMS gleichzeitig anbieten. Das heißt ja, irgendwie sind Sie Konkurrenz, aber irgendwie arbeiten sie auch zusammen und aus TYPO3-Sicht. Es gibt Leute, die formulieren, der Markt ist so groß, dass es gerade zu den Open Source Systemen keine harte Konkurrenz gibt oder sich gegenseitiges Zerfleischen, sondern eher im Gegenteil.

00:36:13 Forscher

Wie gut modular, lässt sich denn TYPO3 entwickeln und implementieren? Können Entwicklungsteams unabhängig voneinander an verschiedenen Bereichen der Software arbeiten oder gibt es da Einschränkungen?

00:36:42 Kernentwickler 2

Also wir haben an verschiedenen Stellen eine gute Trennung. Wir haben zum einen die Extension-Fähigkeit des Systems. Das heißt, ich kann 2 verschiedene Teams haben, die 2 verschiedene Erweiterungen entwickeln, in unterschiedlichen Bereichen. Das ist in der

Praxis aber häufig gar nicht so wichtig. In der Praxis hat man häufig eher eine Trennung zwischen den PHP-Entwicklern, die die serverseitige Architektur zusammenbauen und den Frontend‘lern, die die Oberfläche dazu bauen. Und da liefert TYPO3 über das Templating-System eine gute Schnittstelle bei der die Backend-Entwickler, die Daten zur Verfügung stellen und die Frontend’ler, diese dann zu einer ordentlichen schicken Webseite zusammenbauen. So, das heißt wir haben mehrere orthogonale Schnittstellen, an denen dann auch parallel entwickelt werden kann. Das heißt, ja ich würde sagen, wir sind modular.

00:38:14 Kernentwickler 2

Dazu kommt, dass das Kernsystem immer stärker darauf hinarbeitet, immer weniger Abhängigkeiten in sich zu haben. Also die neue Version 13 liefert die Möglichkeit, 2 zentrale Extensions nicht mehr zu deployen oder zu shippen, wenn die auf dem jeweiligen System nicht mehr benötigt werden. Und damit ist die ist der Kern reduziert auf in Anführungsstrichen „nur“ 5 Extensions, die eigentlich immer da sind, und davon können vermutlich noch weitere wegfallen.

00:38:47 Kernentwickler 2

Und das ist eines der Ziele im Refactoring-Prozess Abhängigkeiten sauber voneinander zu trennen und den Kern immer schlanker zu machen. Beziehungsweise immer mehr Abhängigkeiten so herauszutrennen, dass die im Zweifel nicht mehr nötig sind und auch stärker getrennt voneinander entwickelt werden können. Das heißt, wir versuchen im Prinzip aktiv immer modularer zu werden.

00:39:14 Forscher

Das heißt, wenn man ein neues Feature für den Kern entwickelt, wie verhält sich das dann beim Testing? Gibt es da einen Flaschenhals, weil da gerade noch Tests von einer anderen Entwicklung laufen? Also ich will darauf hinaus, was bei Microservices immer als Vorteil gepriesen wird, und zwar, dass schneller lauffähige Versionen veröffentlich werden können, dadurch dass unabhängige Tests gemacht werden können. Gibt es da noch Potenzial bei TYPO3 oder funktioniert das alles schon ganz gut?

00:40:36 Kernentwickler 2

Naja, also wenn ich mir eine klassische Instanz anschaue, dann habe ich da üblicherweise die verschiedenen TYPO3-Core-Komponenten drin und je nach Anforderung des Kunden werden die um verschiedene Extensions erweitert, die manchmal aufeinander aufbauen, manchmal aber auch nicht unabhängig voneinander laufen, und die Komponenten, die unabhängig voneinander sind, können auch unabhängig voneinander entwickelt werden. So die hängen dann auch nicht voneinander ab. Und die kann ich auch austauschen.

00:41:22 Kernentwickler 2

Also, ich kann sagen: „Ich möchte das News-System nicht und nehme ein anderes und dann muss ich mich um die Komponente News kümmern und um alle Stellen, an denen das in der Zeit drin ist.

00:41:39 Kernentwickler 2

Parallel dazu kann ich halt sagen. Mein Mini-Shop, den ich nicht in der Seite implementiert habe, den ich entwickle unabhängig davon.

00:42:11 Forscher

Und wie funktioniert das in der Kernentwicklung, wird sich dann vor Änderungen auf die künftige Schnittstelle geeinigt, falls es Abhängigkeiten zwischen verschiedenen Kernbereichen gibt? Wie kann man sich das vorstellen?

00:43:20 Kernentwickler 2

Ja, das kommt drauf an. Wir haben ein recht gutes Regelkorsett, das bestimmt, wie der Prozess zu verlaufen hat, wenn API geändert wird. Das läuft im Normalfall darauf hinaus, dass wir einfach nicht irgendwelche Sachen wegschmeißen können, sondern wir müssen einen Deprecation-Prozess einhalten, für Dinge, die als API oder Schnittstelle klassifiziert sind.

00:44:05 Kernentwickler 2

Da weichen wir auch im Regelfalle nicht von ab. Das heißt, da ist drin, dass wir API-Änderungen dokumentieren und dass wir die auch nicht einfach zwischen Major Versionen löschen dürfen, sondern wir haben einen Deprecation-Prozess dazwischen.

00:44:30 Kernentwickler 2

Zwischen Minor-Versionen geht das sowieso nicht, da dürfen wir das einfach nicht tun und auf Patch-Level auch nicht. Es gibt auch nur sehr selten Ausnahmen zu, die häufig dann mit Security-relevanten Fragen zu tun haben. Also es gibt den Fall, dass wir ein Security-Problem lösen müssen, das uns dazu zwingt in der Konfiguration oder in der API eine Änderung vorzunehmen und das sind so ziemlich die einzigen Ausnahmen, an denen wir an unserem Prozess vorbeikommen. Und das ist auch immer nicht schön, wenn wir das tun müssen.

00:45:18 Kernentwickler 2

Weil wir damit nämlich unser Versprechen brechen, dass wir nicht in Minor- oder Batch-Level-Versionen brechen. Das tritt auch nur selten auf. Das machen wir jetzt nicht jede Woche, sondern das sind Ausnahmeerscheinungen, die alle paar Jahre einmal auftreten.

00:45:37 Kernentwickler 2

Ansonsten haben wir einen formulierten Prozess dazu, an dem wir uns auch halten. Der zweite Teil der Frage bezieht sich jetzt darauf, wie wir diese Entscheidung treffen, dass wir API oder Schnittstellen ändern müssen oder wollen.

00:45:59 Kernentwickler 2

Das ist dann halt wieder so eine Geschichte, die hängt so ein bisschen von der Natur des Open Source Systems ab, weil wir A nicht hundertprozentig wissen, wer wann wo warum daran arbeitet und ob das Thema dann auch tatsächlich immer zu Ende verfolgt wird und deshalb sind wir bei solchen Sachen relativ kritisch und diskutieren die dann mit den Beteiligten. Da sind üblicherweise auch Core-Teammitglieder mit dabei, die dann sagen:

„Ja, das können wir so machen oder nicht, oder lass mal diese Komponente noch umbauen.“

00:46:44 Kernentwickler 2

Ich gebe ein Beispiel, wir haben in der Version 12, 2 neue Extensions aufgenommen, die sich lustigerweise auf diesen MACH-Bereich in dem Sinne mehr oder weniger direkt beziehen, nämlich Reactions und Webhooks.

00:46:57 Kernentwickler 2

Beide Erweiterungen sind erst mal standalone entwickelt worden und wurden dann als Pull-Request für den TYPO3-Core gestellt und zu dem Zeitpunkt haben wir uns das im Prinzip angekuckt und festgestellt, dass da ein 2 Komponenten enthalten sind, die allgemeiner gelöst werden könnten. Eine von den Komponenten hat zum Beispiel eine UUID eingefügt innerhalb der Reactions und da war klar, dass ist ein Teil, den sollte TYPO3 einfach können im Core und damit haben wir diesen Teil getrennt herausgezogen und den als Einzelpatch, als Core-Funktionalität implementiert und die Erweiterung, die das nutzt, ist dann in dem Sinne nur eine standalone Erweiterung.

00:47:58 Kernentwickler 2

So, das sind also Architekturfragen, die müssen im Einzelfall bewertet und beantwortet werden. Und die trifft kein Entwickler allein, sondern das findet dann im Review-Prozess statt und da sind unterschiedlich viele Leute mit eingebunden, je nachdem, was es für ein Topic ist.

00:48:23 Forscher

Ja, ok, spannend.

00:48:25 Kernentwickler 2

Beantwortet das ungefähr die Frage oder bin ich daneben gewesen?

00:48:32 Forscher

Ja, es geht so in die Richtung, auf die ich hinauswollte, weil API-First heißt ja man entscheidet ganz am Anfang, welche Schnittstellen brauche ich in der neuen Version. Und erst danach fängt man an zu entwickeln, wenn ich das richtig verstanden habe?

00:49:08 Kernentwickler 2

Ja, nicht hundertprozentig, einfach dadurch, dass wir Core-intern diese Art von Schnittstellen so nicht haben. Ja, also wenn du unter Schnittstellen verstehst, dass wir mit JSON oder HTTP-Requests miteinander sprechen, dann haben wir das so in der Core-Entwicklung an vielen Stellen einfach gar nicht.

00:49:33 Kernentwickler 2

Da geht es dann eher um die PHP-Programmier-API.

00:49:37 Forscher

Genau, ich denke man kann auch im erweiterten Sinne von API-First sprechen, wenn man eine PHP-Schnittstelle hat und diese zuerst festgelegt.

00:49:49 Kernentwickler 2

Ja, das klappt nach meiner Erfahrung in der Praxis nicht. Das ist so ein bisschen das Problem, was Wasserfallentwicklung hat, dass ich also vorher Pflichten- und Lastenheft mache und ich damit alles definiere, was der Kunde nachher bekommt und haben will. Das geht auch häufig schief und zu sagen: „Ich designe die API komplett vorher." Das ist häufig allein deshalb schwierig, weil man in einem komplexen System gerne mal Details übersieht.

00:50:37 Kernentwickler 2

Und dann stellt man nachher fest, Oh warte mal, meine API muss ja doch noch bisschen anders sein. Das heißt, was da eher passiert, ist, dass die involvierten Entwicklerinnen in Kontakt stehen und sich dann im Laufe des Prozesses abstimmen und die eigentliche API dann im Laufe des Prozesses entsteht, obwohl man vorher, sagen wir mal, zumindest eine nebulöse Vorstellung davon hatte, wo man hinmöchte. Vielleicht auch eine konkretere nebulöse Vorstellung, aber meistens ist die doch nicht so ganz final. Einfach weil das in der Praxis schwierig ist. Nach meiner Erfahrung.

00:51:24 Forscher

Ja, ich kenne das ja auch aus meinen, wenn man am Anfang etwas entwirft, dann entwickelt sich das erst mit der Zeit bis dann alles funktioniert. Also ein Wasserfallmodell wird nicht bis zum finalen Produkt funktionieren, es sei denn der Weg dorthin ist bereits sehr klar. Deswegen ist API-First, vielleicht auch ein etwas leeres Versprechen, weil es besonders schwierig ist Microservices später zu ändern, also zum Beispiel, aus dem einen Microservice etwas herauszunehmen und in den anderen Microservice hineinzupacken. Das ist dann wesentlich komplexer, als wenn man innerhalb von PHP etwas refactorn muss.

00:52:21 Kernentwickler 2

Ja, aber ich könnte ein Beispiel nennen oder versuchen, ein Beispiel zu konstruieren, in dem das deutlicher wird, diese Argumentation. Nehmen wir an, wir haben eine Schnittstelle, die liefert mir alle Content-Elemente einer Seite. Da kann ich mich leicht hinstellen und dafür eine API-Definieren, die sagt: „Du gibst mir bitte einen GET-Request, in dem du die Page-ID mit angibst und dann liefere ich dir alle Content-Elemente als JSON und die einzelnen Elemente haben, zum Beispiel, den Titel und vielleicht noch Detailinformationen.

00:53:08 Kernentwickler 2

So, und dann stell ich das als Service in mein Netz und dann stell ich plötzlich fest, Ah, warte mal, wenn ich jetzt aber ein eingeloggter Benutzer bin, wenn ich Backend-User bin und ich möchte ein Preview machen, dann möchte ich ein anderes Set von Content-Elementen bekommen, nämlich auch welche, die, zum Beispiel, erst in der Zukunft live geschaltet werden. Oder die noch hidden sind, weil ein Editor die auf hidden gestellt hat.

00:53:36 Kernentwickler 2

Und dann bin ich schon da dran, meine API zu erweitern, um da sagen zu können: „Du musst mir aber bitte auch diese Hidden-Records geben, weil ich nämlich ein User bin, der ein Preview machen darf und von dieser Art Fragen gibt es eine ganze Reihe und das liegt

so ein bisschen daran, dass wir halt in TYPO3 mindestens 3 streng orthogonale Dimensionen beim Content haben. Ich habe also die Page mit den Content-Elementen und ich hab dazu orthogonal hidden und startTime- und endTime-Records. Ich habe die User-Schichten dazu und ich habe auch Workspaces da dran.

00:54:26 Kernentwickler 2

So, das heißt also die Frage, welche Content-Elemente gebe ich dir jetzt, wenn du sagst, gib mir alle Content-Elemente, ist gar nicht so einfach zu beantworten und da was zu übersehen ist ziemlich simpel. Das heißt a-priori eine API zu designen, die das alles kann, dafür muss ich das System schon sehr gut kennen, damit das nachher stimmt. Das ist nicht so einfach, dass es auf Anhieb gelingt.

00:54:51 Forscher

Ja, deswegen ist der Prozess sicherlich auch ein langwieriger, den jetzt diese Headless-Extension geht, die dann sicherlich in nächster Zeit immer mehr Features auch realisieren will. Denkst du, dass es eine Option ist, dass man diese später auch mit in den Kern nimmt oder wäre das keine gute Idee?

00:55:19 Kernentwickler 2

Das kann ich zum jetzigen Zeitpunkt nicht beurteilen. Ich würde mal andersrum argumentieren, dass wir versuchen, im Core Dinge nicht zu verbauen, sodass wir einer Headless Extension das Leben nicht künstlich schwerer machen. Da ist bereits Feedback von verschiedenen Headless-Entwicklern gekommen, die gesagt haben: „Hör mal können wir im Core die eine oder andere Ecke umstellen, damit wir das leichter haben."

00:56:12 Kernentwickler 2

Ob die Headless-Extension selbst in der Lösung so nativ in den Core kommt, kann ich zum jetzigen Zeitpunkt nicht sagen, da müssen wir im Prinzip abwarten, wie sich das entwickelt. Vermutlich ist es zum jetzigen Zeitpunkt günstiger zu schauen in welchen Bereichen die jetzige Lösung große Schwierigkeiten hat, weil der Core da nicht ordentlich mitspielt. Und diese Sachen im Core zu vereinfachen oder so aufzustellen, dass sie flexibler werden und sinnvoller damit umgehen, damit die Extension weniger Stress hat.

00:56:48 Kernentwickler 2

Vermutlich ist das zunächst ein Weg, den erst mal eine Weile verfolgt und dann kann man immer noch sehen, ob man die in den Core nimmt. Der Vorteil so etwas noch nicht in den Core zu nehmen, ist natürlich auch, dass die Entwicklung auf Extension-Seite damit viel schneller sein kann, weil die Extension sich dann nicht zwingend den Core-Regeln unterwerfen muss.

00:57:16 Kernentwickler 2

Das ist ein Prozess, den wir schon mehrfach gesehen haben, auch bei anderen Lösungen, dass die gut funktionieren, wenn sie erstmal parallel zum Core entwickelt werden und dann später möglicherweise in kleineren Schritten den Weg in den Core finden.

00:57:32 Kernentwickler 2

Ein gutes Beispiel ist die Konsolen-Implementierung von Helmut Hummel, der das bewusst genauso getan hat. Er hat gesagt: Nein, ich will nicht den großen Schritt machen, um das Ganze in den Core zu bringen, sondern ich muss erstmal kucken, wie das läuft und dann in kleinen Schritten sehen, wie man das häppchenweise in den Core verfrachtet, um dann im Core auch eine ordentliche Lösung zu bekommen.

00:58:03 Forscher

Ja, das heißt ja auch, dass man dann vom Extension-Entwickler zum Core-Entwickler werden muss, ist das richtig? Also es muss sich jemand um diese Komponente kümmern.

00:58:15 Kernentwickler 2

Ja, genau. Das kommt aus Core-Sicht dann noch dazu, dass wir jede Lösung, die wir in den Core nehmen, langfristig maintainen.

00:58:25 Kernentwickler 2

Das heißt, das muss dann mindestens so aufgestellt sein, dass wir sicher sind, dass das mindestens für eine Weile läuft und dass das auch einen Stand erreicht hat, mit dem das halt maintainbar ist. Am Ende haben wir das Teil dann am Hals. Wenn die Hauptentwickler weg sind, dann muss der Core das mindestens maintainen und im Idealfall auch weiterentwickeln.

00:58:59 Kernentwickler 2

So, das heißt, das ist nochmal eine weitere Hürde dazu. Wir machen das gerade mit der Content-Blocks-Extension. Content-Blocks ist ein Ansatz, um Integratoren zu befähigen sehr viel einfacher neue Content-Elemente zu bauen.

00:59:19 Kernentwickler 2

Zum jetzigen Zeitpunkt ist das ein halbes Dutzend Schnittstellen, die angepasst werden und benutzt werden müssen, um ein neues Content-Element zu registrieren. Das ist viel Arbeit. Integratoren, die das geübt sind, brauchen also circa eine halbe Stunde und nicht geübte länger, um ein neues Custom-Content-Element ins System einzubringen. Die Content-Blocks-Extension macht das sehr viel einfacher, vielleicht sogar so, dass man da später irgendwann klicken kann, aber zum jetzigen Zeitpunkt reicht es im Prinzip, eine Datei aufzubauen und dann passiert die ganze Magie unten drunter, um das Content-Element ins System zu bringen.

00:59:59 Kernentwickler 2

Diese Extension hätte theoretisch schon in der 12 in den Core gekonnt und das haben wir zu dem Zeitpunkt noch nicht gemacht, weil die da einfach noch nicht weit genug war. Das heißt die Entwickler der Extension haben jetzt noch mal Zeit da weiterzugehen und dann kucken wir, ob wir eine Integration in die 13 reinbekommen davon, denn an der Stelle sind sich im Prinzip alle Kernentwickler oder alle Leute, die irgendwie am Core schrauben oder beitragen, einig, dass man dieses Problem lösen will. Also, dass man an der Stelle den Integratoren das Leben leichter machen will an der Stelle, um schneller entwickeln zu können in Agenturen.

01:00:53 Forschers

Ja, das ist ja auch so ein Konzept, was die MACH-basierten CMS sehr intensiv verfolgen, dass man eigene Content-Types anlegen kann und das ist sicherlich auch für TYPO3 vorteilhaft, wenn es da integriert ist, kann ich mir vorstellen.

01:01:14 Kernentwickler 2

Ja so geht das schon seit langem. Das ist nur zum jetzigen Zeitpunkt noch so ein bisschen holprig. Es wäre also schön, wenn da eine neue API dazu kommt, mit der das einfacher wird, um so ein bisschen die Hürden unten drunter zu streichen, damit da Integratoren nicht mehr weiter darüber nachdenken müssen.

01:01:41 Forscher

Kommen wir zu den abschließenden Fragen. Was wünschst du dir zukünftig für die Arbeit mit TYPO3?

01:02:28 Kernentwickler 2

Ich hoffe, dass wir weiter den Weg verfolgen ein System mit moderner Software-Architektur zu etablieren und das weiter zu verbessern, weil dadurch so viele andere Vorteile abfallen: Wir werden schneller, leichter verständlich, die Hürden werden kleiner und wir werden insgesamt flexibler. Ich hoffe also, dass wir weiter entkoppeln, State offensichtlicher machen, sinnvoller Kapseln, und wir uns damit weiter modernisieren. So und ich hoffe, dass ich ein Teil davon sein kann.

01:03:25 Forscher

Ja, sicherlich. Gut die nächste Frage brauche ich vermutlich nicht stellen, ob MACH-basierte CMS in Zukunft traditionelle CMS verdrängen werden. Ich denke mal dazu hast du eine klare Meinung, dass es nicht so sein wird, liege ich da richtig?

01:03:46 Kernentwickler 2

Naja, also wenn MACH-basiert bedeutet, es gibt kein On-Premise mehr, ich weiß nicht, dann vermutlich eher nicht. Ich könnte mir sogar schon fast vorstellen, dass es zunehmend Unternehmen gibt, die versucht haben, alles Cloud-basiert zu machen und auch inzwischen die möglichen Nachteile dabei fühlen und möglicherweise auch stärker in der Zukunft hingehen und das gegeneinander abwägen, die Vor- und Nachteile, und sich dann auch weiterhin auf nicht-Cloud-basierte Systeme einlassen, wenn es große Vorteile hat. Von daher nehme ich an, dass MACH-basierte CMS so in der Form traditionelle CMS nicht vollständig verdrängen werden.

01:05:20 Forscher

Was hältst du von KI-Integration in TYPO3?

01:05:26 Kernentwickler 2

Haben wir schon. Es gibt verschiedene Extensions, die zum Beispiel beim Übersetzen von Elementen eine Übersetzung vorschlagen. Da ist wieder dieselbe Frage, ob das dann irgendwann in den Core kommt oder nicht. Das werden wir sehen. Aber ja, es gibt irgendwie mindestens 2 oder 3 Ansätze, die verschiedene Teile des Systems mit KI unterstützen.

Notizen zu relevanten Gesprächsinhalten nach Beendigung der Aufnahme:

Kernentwickler 2:

- TYPO3 Core CI/CD Pipeline braucht lediglich 6 min zum Test des kompletten Kerns
  ➔ Kein Flaschenhals bei Integration neuer Features
- 1200 Tests in 1h auf 4 On-Premise Hardware Maschinen
- Hoch parallelisiert, containerbasiert
- eigene Server stabiler, schneller und kostengünstiger als Testing in Cloud

2.3 Interview C

00:00:19 Forscher

Seit wie vielen Jahren beschäftigst du dich mit Content Management Systemen?

00:00:24 Agentur-Experte 1

Oh, das ist schon recht lange. Ich denke mal so Ende der 90er Jahre, also seit ‘98 ’99 ist das ein Thema. Also ich habe die Agentur, die mir gehört 1996 gegründet. Dort haben wir noch mit der Hand Webseiten gestrickt und irgendwann mal haben wir News-Module entwickelt mit PHP respektive auch anderen serverseitigen Sprachen. Bei meiner Marktbetrachtung sind wir über Content Management Systeme gestolpert, die es damals gab. Da gab es nicht nur Open Source Systeme, sondern da gab es sehr viele Content Management Systeme, die mit einer Lizenz ausgestattet kamen und ich glaube, seitdem beschäftige ich mich mit dem Thema Content Management. Primär auch, um Websites zu machen, die Kunden kamen ja damals noch mit der Anforderung, „mach mir doch ne Website“ und gar nicht so von wegen „bitte installier mir mein Content Management System“ und konfiguriere das für mich so, dass ich damit arbeiten kann. Also es war eher so, man den Kunden ein Content Management System installiert, sodass man auf manuelle Arbeiten selbst verzichten konnte.

00:01:53 Forscher

Dann hast du den ganzen Wandel miterlebt.

00:01:57 Agentur-Experte 1

Ja, am eigenen Leibe. Also Leute sagen ja, dass Content Management eine gelöste Aufgabe sei. Es ist schon sehr vielschichtig, was angefordert wird und auf welche Art und Weise. Also die Systeme stehen stellvertretend dafür, wie Unternehmen sich selbst organisieren. Also darin sehe ich große Parallelen.

00:02:40 Forscher

Welches war dein erstes CMS?

00:02:47 Agentur-Experte 1

Das ist eine gute Frage. Also ich glaube, wir haben uns verschiedene angeschaut. Ich glaube mit TYPO3 haben wir wirklich die meisten Erfahrungen gemacht. Auch die ersten Erfahrungen glaube ich. Und sonst gab es noch natürlich andere Systeme wie redDot oder Interwoven oder was es da halt alles gab. Also es gab teilweise exotische Systeme, die so XML basiert konfiguriert wurden. Ich sag mal, das ist ein bisschen so die Frage, mit was die Kunden kamen, ich glaub. Aber ich glaub TYPO3 ist das, was wir als Allererstes dann auch tatsächlich so in Besitz genommen haben.

00:03:30 Forscher

OK, und es ist dann immer bei den traditionellen CMS geblieben. Oder hast du auch mal Erfahrungen mit Headless CMS bzw. MACH-basierten CMS gemacht?

00:03:48 Agentur-Experte 1

Ja, ja, also sagen wir mal, vor vielen Jahren bin ich über ein Content Management System gestoßen, was mir recht revolutionär erschien, das nennt sich Prismic.

00:04:05 Agentur-Experte 1

Das ist ein Content Management System, das schon damals auf einem Server lief und nicht mit fertigen Content Modellen ausgeliefert wurde, sondern die konnte man sich selbst erstellen. Das fand ich interessant. Aber dann war der Markt noch nicht so weit in Deutschland, dass die Leute gesagt hätten, ach ja, wir gehen jetzt auf so ein amerikanisches System, was auf einer Cloud läuft. Wo man ganz, ganz viel Arbeit machen muss und so weiter. Also da war das noch nicht so weit. Sagen wir mal vor knapp 10 Jahren bin ich über Prismic gestoßen und habe das immer im Auge gehabt, immer parallel dazu noch mit anderen Content Management Systemen. So wie Alfresco. Das ist ein DMS, also ein System, was stark mit einem Dokument Management System einherkam, weil das noch eine Aufgabe war, die viele hatten. Also es gab auch den Begriff des „Enterprise Content Managements“ und dabei eher das Thema der Organisation von Dokumenten und was darum herum passierte. Da habe ich immer ein Auge immer draufgehabt, ja.

00:05:29 Agentur-Experte 1

Aber heutzutage, ist die Begriffsklärung eine andere dadurch, dass das Thema Cloud eine große Akzeptanz erlangt hat, nicht überall, aber schon recht große Akzeptanz und das Thema der Trennung, also von Content und Darstellung wieder eine große Rolle spielt. Also das, was Content Management Systeme eigentlich versprechen. Das Thema ist zum einen schon gelöst worden. Nichtsdestotrotz, glaube ich, wenn man sich heute anschaut, wie Menschen Content produzieren, dann passiert das in vielen Fällen außerhalb des Content Management Systems. Das heißt, da wird Content online kollaborativ in Google- und Word-Dokumenten geschrieben und dann von dort aus dann in das Content Management System übertragen.

00:06:53 Forscher

Okay. Sind dir die Begriffe Microservices, API-First, Cloud Native und Headless bei deiner Arbeit schon einmal begegnet?

00:07:02 Agentur-Experte 1

Ja. Wir, in meinem Unternehmen suchen natürlich nach neuen Betätigungsfeldern rund um Content Management und da ist uns natürlich das Thema MACH und die MACH Alliance über den Weg gelaufen. Wir selbst haben eine Spielwiese gebaut, das ist eine StoryBlok Anwendung oder StoryBlok Content Management System, was auf Cloudflare Workern ausgeführt wird und wo wir auch Content, zum Beispiel aus unserem Shop, draufspielen und dann nutzen wir auch von Shopware die API-Ausgaben für die Produkte, zum Beispiel. Das ist mir schon bekannt.

00:08:04 Agentur-Experte 1

Ich glaube, dass das Thema MACH, so eine Reaktion darauf ist auf, dass Content Management bei den Menschen in der Branche so ein bisschen abgenudelt ist und parallel halt dieser JamStack sich aufgebaut hat. Also, wenn man so schaut in der Anwendungsentwicklung oder in die Welten, in denen sich die Unternehmen bewegen, sind entweder Java, .NET-basierte oder PHP-basierte Welten, dann gab es irgendwann mal diesen Schritt in Richtung der schnell ausführbaren JavaScript-Engines mit v8, sage ich mal. Dann hat sich im entsprechenden Applikationsserver eine Reihe an Möglichkeiten ergeben und ich glaube, dass die Entwickler per se immer neugierig sind und auch ganz froh sind, wenn sie neue Dinge tun können. Und dann ist eine logische Konsequenz, dass man dahin geht, wo die Interessen der Entwickler sind.

00:09:29 Agentur-Experte 1

Das sind so Reaktionen auf: Das ist schon ein gelöstes Problem, aber lass uns das Problem vielleicht mal mit einer anderen Technologie mal lösen.

00:09:41 Forscher

OK. Ergibt sich daraus eine Art Innovationsdruck für TYPO3 oder ist man eigentlich in einem ganz anderen Bereich unterwegs mit TYPO3?

00:09:55 Agentur-Experte 1

Naja, ich sag mal. Da gibt es unterschiedliche Dimensionen. Ich sag mal jemand, der seit vielen Jahren erfolgreich mit TYPO3 unterwegs ist. Kennt die Vorteile der Plattform, kennt natürlich auch die entsprechenden Investitionszyklen, die doch recht lang sind. Also man kann da mit TYPO3 schon 6 bis 8 Jahre seine Webseite betreiben, ohne große Technologiewechsel. Also sprich man installiert ein TYPO3 und geht dann sag ich mal über den normalen Support und dann den ELTS Support. Hat man 6 Jahre das Projekt stabil gehalten. Das ist das eine.

00:10:45 Agentur-Experte 1

Sicherlich im Wettbewerb und dort, wo es wirklich auf den Use Case ankommt, Content möglichst oft standardisiert wiederverwenden zu können, glaube ich, gibt es da definitiv einen Innovationszwang sogar. Also ich glaub das ist aber jetzt dem Umstand geschuldet, dass TYPO3 nicht zentral entwickelt wird, also da gibt es nicht einen, der quasi über die Entwicklung bestimmt, sondern da ist halt die Community, die über die Dinge, die sie tun will und wo sie ihre Aufmerksamkeit reinstecken will und insofern ist das Thema API-first oder Cloud noch nicht so ganz auf der Agenda.

00:11:46 Forscher

Ja, gibt es seitens der Politik Einflüsse wo?

00:11:52

Ah ja. Ich sag mal TYPO3 ist seit 25 Jahren recht erfolgreich im deutschen Markt unterwegs. Also hauptsächlich im deutschsprachigen Markt, das ist auch da, wo TYPO3 wirklich die große Landschaft an Anbietern hat und auch große Nachfrage. Und wenn man sich so die typischen Kunden anschaut, sag ich mal von den Mittelstandskunden oder den Organisationen, die hauptsächlich informieren wollen beziehungsweise so eine Brücke schlagen wollen in andere Systeme, dann glaube ich, gibt es erstmal keinen großen Druck, vom Markt aus, quasi von den Nachfragenden, aber ich sage mal das Problem stellt sich halt eher auf der anderen Seite, dass die Entwickler nicht zuhauf sind und natürlich die Thematik TYPO3. TYPO3 ist teuer. TYPO3 ist kompliziert, TYPO3 ist nicht sexy, das trägt schon dazu bei, dass man reagieren muss.

00:13:06 Agentur-Experte 1

Als Reaktion darauf hat die TYPO3 Association, die versucht das Projekt weiterzuentwickeln, eine große Analyse durchgeführt und festgestellt, dass dieser Produktentwicklungsprozess stärker gesteuert werden müsste und dieser Produktentwicklungsprozess muss aufbauen, auf die tatsächlichen Bedürfnisse der User, also sprich A) Wer sind im Redaktionsalltag die Menschen, was haben Sie für Aufgaben, wie müssen Sie die lösen? Bis hin auch dazu, welche Erwartungen haben Einkäufer oder haben Entscheider zu dem Produkt gerade speziell zu der Integration und wie die Integration auch stattfindet, ob sie eher durch standardisierte, im Tool schon bereits verfügbare Schnittstellen passiert oder eben durch, wie es halt jahrelang passiert ist, durch Agenturen im Einzelprojekt umgesetzte Maßnahmen.

00:14:19 Agentur-Experte 1

Es gibt aktuell eine große Initiative, die versucht längere und größere Themen an den Produktentwicklungsprozess anzuhängen, der ja doch je Release 18 Monate dauert, sodass nicht nur das angegangen wird, was man in 6 Monaten Entwicklungszeit schaffen könnte. Und die Initiative beschäftigt sich auch damit, mit welchen Geldern Innovation geschaffen werden kann. Das ist das, was ich dir berichten kann, quasi aus dem, was ich in meiner Rolle auch natürlich als Präsident der TYPO3 Association, dir auch sagen kann.

00:15:04 Agentur-Experte 1

Eine Produktvision ist in der Entstehung und die ist schon intern gefestigt. Da bedarf es aber auch, dass man jetzt in die Stakeholder Kommunikation geht und das passiert jetzt und wird in diesem Jahr sicherlich ein bisschen konkreter passieren als das, was ich dir jetzt sagen kann.

00:15:36 Forscher

Ja, das klingt spannend.

00:15:38 Agentur-Experte 1

Ist es auch.

00:15:41 Forscher

Da bin ich mal gespannt, was da passiert.

00:15:47 Agentur-Experte 1

Wenn man sich natürlich anschaut, was andere Wettbewerber oder andere Content Management Systeme machen. Also da gibt es ja nicht nur uns. Also sagen wir mal, wenn man in der Open Source Welt schaut, dann gibt es natürlich Anbieter, wo die Gründer Firmen gegründet haben, also wo ein Mad Mullenberg oder ein Dries Buytaert entschieden haben, WordPress oder Drupal, drumherum um die Produktentwicklung baue ich, gemäß dem Open Source Donut-Modell, also das Produkt ist das Loch und drumherum ist der Donut und das ist, was ich anbieten kann, also Hosting oder Services oder Add-Ons und so weiter und sofort. Da kann ich dir sagen, dass es auch zum Teil in diese Richtung gehen wird. Wie das dann am Ende konkret passieren wird, ob man das selbst anbietet oder ob man mit Partnern etwas anbietet. Ähm, wenn du dir natürlich Content Management Systeme anschaust, wie WordPress, sie leben auch von einem Ökosystem eines eingebauten Shops bzw. eines eingebauten Erweiterungsmarktplatzes.

00:17:08 Agentur-Experte 1

Ja, also da gibt es ganz, ganz viele Dinge. Wenn man sich ähnliche Projekte anschaut wie Shopify, die auch begonnen haben mit lokal ausführbaren Entwicklungen. Und mittlerweile ist es eine brutale Cloud Plattform mit allem Möglichen geworden, dann ist die Frage, ob man sich mit der Produktrichtung in Richtung dieser Angebote bewegt.

00:17:46 Forscher

Eine Voraussetzung für ein Cloud-native Angebot ist, dass man sowohl die Entwicklung als auch den Betrieb der Plattform übernimmt. Im Prinzip ist das bei TYPO3 immer noch getrennt, richtig?

00:18:02 Agentur-Experte 1

Ja, wie gesagt, wobei es ja da auch teilweise Unterschiede gibt. Ich mein, wenn du dir sagen wir mal Cloud-native Anwendungen, die ja häufig nach dieser Idee der Twelve-Factor-App gestrickt sind… ich weiß nicht, ob du darüber gestolpert bist?

00:18:19 Forscher

Ja, ist mir bekannt.

00:18:22 Agentur-Experte 1

Da sind per se Anwendungen, die erstmal viele Dinge voneinander trennen und TYPO3 kann das natürlich nicht, weil die Umstände unter denen TYPO3 entstanden ist, waren eben die: Menschen hatten noch als Vergleich die Software, die sie installiert haben. Also sie haben eine CD bekommen und haben die auf ihre Premise installiert und haben da quasi erwartet die Software löst alle meine Probleme auf meiner Premise. So ist das das totale Gegenteil zu denen die sagen: „Ich habe eine Cloud und ich hole mir 5 verschiedene Anwendungen, die in der Cloud laufen und ich bringe die einfach über eventbasierte Queues oder über irgendwelche API-Gateways, zusammen.

00:19:14 Agentur-Experte 1

Ich glaube, dass es dennoch in der Zukunft einen Markt geben wird für On-Premise-Lösungen, die aber intelligent mit einer Plattform verknüpft werden. Seit einem Jahr gibt es etwas wie ChatGPT und OpenAI und da gibt es ChatGPT als Webanwendung und ich kann aber von OpenAI auch die ganzen Funktionen über eine API nutzen, dann bedient das letztendlich 2 verschiedene Märkte. Das bedient den Markt der Endanwender, also Leute die im Chat, also in einem Browser mit künstlicher Intelligenz interagieren wollen, aber der API-Markt bedient Value Added Provider, also Leute, die mit künstlicher Intelligenz irgendeinen Dienst anbieten wollen, also vielleicht ihre eigenen Ideen entsprechend noch erweitern wollen und die dann auch Dinge zusammen orchestrieren. Und ich glaube, ob es dann eine standalone Anwendung ist oder eine mobile Anwendung oder sogar eine iPhone oder Android Anwendung ist dann die Sache des Anbieters dieses Service. Und ich glaube ähnliches wird es natürlich auch für Content Management geben. Man sagt dann: „Ich habe meine Content Management Anwendung. Ich brauche aber nicht für eine bestimmten Use Case eigene Infrastruktur vorzuhalten. Sondern, wenn ich zum Beispiel Bilderkennung brauche, integriere ich die Anbieter, die mir Bilderkennung geben, gleiches gilt für Bildgenerierung oder für Textgenerierung, SEO, etc.“ und ich glaube, dass das ein Thema sein wird, bei dem die Menschen sagen werden: „Ja, ich will trotzdem einen gewissen On-Prem und wenn es am Ende nur Cloud-Prem ist. Die Leute wollen es irgendwie besitzen.

00:21:24 Agentur-Experte 1

Wie gesagt, wenn du schaust, was Cloud bedeutet, was Cloud ist, da gibt es ja Abwandlungen noch und nöcher, sag ich mal von den riesigen Cloud-Anbietern hin zu den spezialisierten Cloud-Anbietern, wie eben sage ich mal DigitalOcean oder Heroku oder FlyNet oder wie sie alle heißen wo ich dann noch zusätzlich zur Anwendung die ganzen Operations dazubekomme, im Sinne von aus meinem Code wird eine Anwendung und so weiter und ich glaube da passiert in vielen Dimensionen unheimlich viel gerade und das hat auch alles seine Berechtigung.

00:22:06 Forscher

Ja, wie könnte man TYPO3 noch Cloud-freundlicher machen? Also zum Beispiel für die Bereitstellung in der Cloud?

00:22:17 Agentur-Experte 1

Ja, also eine große Geschichte ist eine Initiative, die ich versuche seit Jahren (...) Also es gab ja verschiedene Wege von TYPO3 in die Cloud. 2016 haben wir mit plattform.sh und damals unter der Vermittlung von Microsoft und Azure haben wir eine TYPO3 Composer-basiertes Template entwickelt. Also die Leute konnten nach plattform.sh deployen und ihre ganzen Git-Branches waren dann entsprechende Cloudumgebungen. Da hat dann die plattform.sh-Umgebung, das Aufsetzen der Infrastruktur übernommen, also das Aufsetzen des Web-Servers, der Datenbank als Service und so weiter. Also ich denke mal, das ist gar nicht das Problem.

00:23:26 Agentur-Experte 1

Da gibt es ein Thema mit der Akzeptanz, beziehungsweise ich glaube das ist auch so eine witzige Diskussion. Also wenn man von Cloud redet meinen alle Leute gleich: Ja, ich muss

da jetzt ein Kubernetes Cluster machen und das muss ich irgendwie selbst betreiben können und ich muss einfach unheimlich viel Ownership übernehmen und das ist so ein bisschen Cloud falsch verstanden. Also Cloud bedeutet eigentlich: Du nimmst etwas, was dir jemand zur Verfügung stellt und bringst nur deine eigene Anwendung dahin. Und ich glaube, wenn wir von TYPO3 reden, dann müsste TYPO3 sich an manchen stellen sich von dieser Grund- DNA lösen: „Ja ich mache alles selbst. Ich laufe in einem Linux oder in einem LAMP-System also ja idealerweise mit der Datenbank MySQL auf dem gleichen Server und PHP und so weiter und sofort“ und da müsste man wahrscheinlich ran und müsste beginnen zum Beispiel Frontend wie Backend voneinander zu trennen.

00:24:39 Agentur-Experte 1

Es wäre zum Beispiel auch jetzt überlegen, ob zum Beispiel das Frontend-Rendering tatsächlich der richtige Weg ist, also ob man nicht zum Beispiel sagt: Beim Speichern der Seite im Backend produziere ich auch die Seite und nicht erst beim Abrufen der Seite. Also ich habe halt in TYPO3 ein Vorgehensmodell, was wirklich stark noch aus der Zeit geprägt ist: „Ja, die Software im Frontend holt sich die ganzen Sachen, die ganzen Assets im Moment des Abrufs, damit ein Frontend-User im Zweifelsfall 3 Content Elemente für sich privat sehen könnte, aufgrund einer privaten Zugriffs-Berechtigung und man könnte hingehen und sagen: Man macht einen Frontend-Worker der nur Content zusammensucht, der durch das Backend schon final abgespeichert worden ist. Das würde aber bedeuten, dass man da, wie gesagt, am Charakteristikum von TYPO3 arbeitet.

00:25:52 Agentur-Experte 1

Ich glaub das du findest dieses Thema unter dem Begriff „Tiered Application“, also das man wirklich die Applikation unterteilt in deren große Use Case Blöcke und sagt: „ich trenne die Content-Erstellung von dem tatsächlichen Publishing und wenn ich Content erstelle, dann weiß ich aber eben auch beim Speichern, dass das gepublished wird und für welchen Kanal das gepublished wird und im Grunde genommen der Frontend-User, der heut noch die Seitengenerierung auslöst, vorgefertigte Inhalte zugestellt bekommt. Das ist eine Dimension, in die man denken müsste.

00:26:48 Forscher

Das wäre also ein möglicher Schnitt, um das Ganze in kleinere Baustein zu zerlegen, die dann eventuell auch skaliert werden können?

00:26:57 Agentur-Experte 1

Ja, also was du jetzt nicht ändern werden kannst, ist, ich nenn das mal die Administrationsoberfläche oder das Backend. Klar, jetzt kannst du hingehen und sagen: Ach, das sieht das altbacken aus, aber wenn du dir mal Backends anschaust, die sehen alle gleich aus und ob die jetzt in weiß oder schwarz sind oder ob die Icons farbig sind oder selbsterklärend sind oder das ist glaub ich Geschmackssache, also da könnte man jetzt auch hingehen und sagen, wir machen einfach einen Backend-Overhaul oder man setzt wie auch andere Anbieter, also Pimcor zum Beispiel, schon sehr früh auf API, also die Anwendung API-orientiert zu gestalten. Dort ist man dabei, das Backend auszutauschen und zu sagen: es gibt alternative Backends dafür. So weit sind wir in TYPO3 nicht, weil wir da einfach noch an viel zu vielen Stellen im Backend Dinge tun, die nicht Rest-API oder

Graph-QL API basiert sind, sondern die APIs liegen halt in der PHP-Anwendung und sind nicht exponierte standardisierte Schnittstellen.

00:28:19 Agentur-Experte 1

Ich glaub das ist aber nicht das Thema, wenn man auf die Frage zurückkommen würde: Wie bekommen wir TYPO3 Cloud-basierter? Dann müssen wir noch mal zurückkehren zu den 12 Faktoren. Welche haben wir, welche können wir in TYPO3 nachbilden, welche gehören zusammen oder an welcher Stelle ist das auch auflösbar?

00:28:51 Agentur-Experte 1

Ich bin da nur auf einer Ebene unterwegs, wie gesagt, ich umgebe mich ja gern mit gutdenkenden Menschen und lasse mir das immer auch gerne erklären, aber Ich glaube das ist ein Dialog, den man ganz behutsam gehen muss, also nicht nur einfach nur des Business wegen, aber auch nicht nur der Kunst wegen. Also man muss dann da schauen, dass man auch erklärt, warum und weswegen man Dinge tut und welche Vorteile das hat und auf welche Use Cases man sich vielleicht zum Beispiel vorbereiten müsste. Denn es kann ja irgendwann mal sein, dass mein System keinen Content mehr erzeugen wird, sondern nur noch speichern wird. Die Erzeugung selbst, kann dann durch andere Dinge passieren, also sprich durch eventuelle digitale Agenten oder durch andere Agglomerationen oder Aggregationen und, und, und.
Das, glaube ich, ist etwas, wo man, wenn man über die Zukunft von Content Management redet, muss man sich die Frage stellen, für wen produziert man dann am Ende den Content?

00:30:07 Forscher

Noch ein wichtiger Vorteil der MACH-Architektur, der oft angeführt wird, ist die Skalierbarkeit. Da könnte ich mir noch vorstellen, dass man bei TYPO3 vielleicht schaut, dass man mehr Mandanten in ein System reinbekommt**. Das wär vielleicht ein Vorteil für die Agenturen.**

00:30:33 Agentur-Experte 1

Genau, auch vor dem Hintergrund der Nachhaltigkeit macht es ja gar keinen Sinn, dass jeder sich da ein Stück Blech hinstellt, um für 2 Tage im Jahr, wo man viel Traffic hat, gerüstet zu sein. Klar gibt es viele Strategien, wo man ganz viel über Frontend und Cloudfront und Auslieferung lösen kann. Wenn es aber tatsächlich um Applikationsthemen geht, und da ist TYPO3 auch sehr stark diese innerhalb des Content Management Systems mit abzubilden, dann geht es auch ein bisschen darum, dass man dann sagt: „OK, das macht gar keinen Sinn, dass 20 Leute 20 große Server haben. Könnten diese 20 Menschen vielleicht auf einem großen Server sein und den so verwenden, dass der weniger Energie verbraucht als die 20 einzelnen Server?“ Und ich glaub, dass das halt auch ein Nachhaltigkeitsaspekt ist, wenn man zum Beispiel auch Anwendungen schlafen legen kann, die gar keine Besucher haben. Also man muss jetzt nicht unbedingt einen Server bevorraten. Insofern, glaube ich, ist das ein Vorteil.

00:32:01 Agentur-Experte 1

Wenn man skaliert, also jetzt nicht unbedingt in der Vertikalen, sondern eher in der Horizontalen und jetzt nicht unbedingt auf Seiten der Besucher, sondern auf Seiten der

Editoren oder auf Seiten der Websites, die man auf so einem System hosten würde. Also die Mandantenfähigkeit ist definitiv etwas, wo man schauen müsste. Ich glaub das TYPO3 halt per se in einer einzelnen Datenbank denkt. Dadurch, dass die Datenbanktabellen halt auch alle an vielen Ecken miteinander, logisch verknüpft sind. Durch die Page-Tabelle und die Content-Tabelle hast du halt ganz vieles, was verknüpft ist. Es müsste möglich sein, diverse Page-Tabellen durch einen Mandanten oder ein Sharding zu realisieren

00:33:05 Agentur-Experte 1

Es bedeutet da halt eben auch, dass man dann die Anwendung entsprechend umschreiben müsste. Also da haben wir zu viel Code-Basis, die das wahrscheinlich erst mal nicht ermöglichen würde.

Ja, da wäre auch zu überlegen, in welchen Bereichen finden Websites statt? Wenn man sich so eine durchschnittliche Website anschaut von einem Mittelstandskunden, der vielleicht pro Monat 20000 Besucher auf der Seite hat, dann reden wir eben über nicht so ganz viel Traffic. Und wo vielleicht einer 2 Millionen oder 10 Millionen oder hundert Millionen Besucher auf der Webseite hat, dann macht es da halt keinen Sinn so eine Installation auf so einer SaaS-Plattform zu betreiben.

00:33:58 Agentur-Experte 1

Denn die Preismodelle müssen da auch passen. Also, wenn du überall die SaaS-Angebote anschaust, gibt es immer so „Pre“, „Team“, keine Ahnung, wie Sie alle heißen mögen, „Company“ und dann gibt es Enterprise und bei Enterprise gibt es nie einen Preis, da steht dann immer kontaktieren Sie den Sales und ich glaube das ist ein Thema, da muss man drüber nachdenken.

00:34:28 Forscher

Ja. OK. Noch eine andere Frage. Wie modular lässt sich TYPO3 bereits weiterentwickeln und bereitstellen?

00:34:47 Agentur-Experte 1

TYPO3 hat von der PHP-Renaissance profitiert. Die PHP-Renaissance ist getrieben worden, sag ich mal von 2 Aspekten, einmal von der Selbsterkenntnis der PHP-Community, dass man die Sprache verständlicher machen muss, kohärenter im Sinne von, dass man sich mal einigen sollte, wie man Funktionen benennt und dass die alle irgendwie auch verständlich sind und nicht nur einem Kreis von Erleuchteten klar zugänglich ist. Also das ist eine Geschichte. Sprich der Weg von PHP 5 nach PHP 7 und 8. Da ist viel passiert. Parallel dazu hat das Abhängigkeitsmanagement mit Composer dazu geführt, dass man TYPO3 jetzt auch auflösen können. Man muss nicht mehr ein vollständiges TYPO3 vom ZIP-File oder TAR-File irgendwo mit allen Erweiterungen installieren, sondern man kann sich mit Composer eine TYPO3 Installation holen, die minimal groß ist und man kann dann mit Composer TYPO3 um Funktionen erweitern. Also das heißt Funktionen, die aus dem Kern heraus passieren, so wie Workspaces oder Versionierung, aber auch Erweiterungen, wie eine Enterprise Suche oder irgendeine gute SEO-Lösung et cetera, kann man alle mittlerweile mit einem „composer install“ hinzufügen. Und ich glaube, insofern ist das eine gute Art und Weise, TYPO3 in kleinere Module zu zerlegen.

00:36:44 Agentur-Experte 1

TYPO3 selbst hat auch dann immer mehr begonnen Funktionen, die es vorher selbst zur Verfügung gestellt hat, die Teil des Cores waren, vom symfony-Framework zu holen. Zum Beispiel bestimmte Symphonie-Funktionalitäten, also HTTP-Requests mit Guzzle zu machen anstelle eigene fopen() zu schreiben oder so Sachen. Also da ist viel Code glaub ich auch weggeschmissen worden, zugunsten der PHP-Funktionen, die es da jetzt gibt. Und auch das Thema Autoloading und was man da halt alles als PHP-Entwickler braucht hat schon dazu beigetragen, dass es per se viel modularer geworden ist als vielleicht vor 10 Jahren war.

00:37:38 Agentur-Experte 1

Kannst du das nachvollziehen?

00:37:40 Forscher

Ja. Es ist auch eine große Anforderung der MACH-Architektur oder der MACH Alliance, dass Software *Composable* ist. Ist das TYPO3 im Prinzip schon?

00:37:52 Agentur-Experte 1

Ja, aber die Frage ist, an welcher Stelle die Komposition stattfindet. Und ich glaube, wenn man da die MACH Alliance richtig versteht, bezieht sich die Komponierbarkeit immer auf die API-Ebene, ist also immer auf der REST- oder graphQL-Ebene zu verorten

00:38:10 Agentur-Experte 1

Also natürlich, die Systeme, die ich dann habe, wenn die Cloud-basiert sind, dass die unterschiedlichster Güte sind, also wenn ich zum Beispiel vergleiche, StoryBlok mit Contentful oder sagen wir, ich vergleiche StoryBlok mit einem Shopsystem wie CommerceTools. Dann habe ich bei CommerceTools eine viel, viel größere Anzahl an APIs, die ich dann als Entwickler verwenden kann und das ist natürlich eine ganz andere Form von Komponenten, die ich verwenden kann. Die Frage ist halt, auf welcher Höhe du diese Komponenten vergleichst. Was sie zur Verfügung stellen, ob die das intern zur Verfügung stellen, wie zum Beispiel in der PHP-Welt, wo ich sage ich installiere mir halt Funktionen und kann dadurch meine Applikationen im Kern erweitern oder ich habe ein Bündel an API-Schnittstellen, die ich irgendwie miteinander zusammenfüge. Das ist so der Unterschied, wie ich den wahrnehme.

00:39:17 Agentur-Experte 1

Eine große Kritik, die ich sehe, für das, was von der MACH-Allianz gemacht wird, ist, dass es an Standardisierung fehlt. Ja, also man versteckt sich so ein bisschen hinter den Begriffen wie „Cloud“ und „Composable“ und „REST“ und „API“ und hat jetzt gar nicht Interoperabilität im Sinne. Also dass man sagt: „Hey, ich kann jetzt zum Beispiel zwei Shops parallel nutzen, weil der eine Macht das besonders gut und der andere macht das besonders gut. Ich kann auch nicht *Drop-in-Replacements* machen, das heißt also, die Vorteile, die durch Standardisierung dazukommen würden oder durch zumindest ein *Beige Book* oder *Orange Book* oder wie auch immer man sowas nennen würde. Also wenn wir sagen die MACH-Allianz steht für eine gewisse Form von Interoperabilität, dann ist das halt schwierig und wenn man sich das genauer anschaut, wenn man so entlang der verschiedenen Use Cases denkt, also sprich Business to Consumer oder Business to

Business, wo dann vielleicht auch die Unterschiede anders sind. Also wenn es dann darum geht, die Integration mit Marketing-Automation oder mit CRM-Systemen zu realisieren, dann glaube ich, bist du auch stark am Basteln.

00:40:49 Agentur-Experte 1

Also das Versprechen ist da und ich meine Dries Buytaert, ich weiß nicht, ob du den Artikel gelesen hattest, in dem er über die MACH-Fatigue gesprochen hat. Er hat gesagt, das Problem ist, dass die Leute etwas versprochen bekommen und in der Praxis ist es unheimlich anstrengend, die Dinge umzusetzen. Dinge, die vielleicht auch in den Monolithen, also sprich in den alten Systemen aus der Box sprangen und da sind, muss man sich in einem Composable-System auf Anwendungsebene zusammenstückeln, wobei man unheimlich viel Lehrgeld dafür zahlt, um Dinge nachzubauen, die man vielleicht vorher hatte. Und auf einmal ist der ganze Vorteil letztendlich weg, wo man sich gesagt hatte: „Oh, ich komme schneller, ich kann schneller liefern, ich kann schnell auf Änderungen reagieren." Das ist sicherlich das Versprechen von MACH. Auf der anderen Seite habe ich halt das Thema, dass ich ganz, ganz viel Groundwork machen muss und ich an manchen Stellen halt dann bei Adam und Eva anfange.

00:41:57 Forscher

Ja.

00:42:00 Agentur-Experte 1

Die Welt wird nicht einfacher, nur weil man sie einfacher haben will. Also das ist so ein bisschen meine Realität.

00:42:11 Forscher

Ja, das ist schon sehr viel. Auch andere Expertise gefragt als bei der bisherigen Entwicklung mit einem Monolithen. Jetzt muss man sich Gedanken machen, wie kommunizieren die einzelnen Dienste miteinander.

00:42:27 Agentur-Experte 1

Genau, genau. Wo landen die Daten auch am Ende? Was ist die *Single-Source-of-Truth*. Wie gehe ich mit Daten um, die sich verändern? Also welche Dimensionen habe ich da an Daten, welcher Natur, in welchen Systemen wandern sie und wer kann mit den Daten dann Dinge tun? Also wenn du zum Beispiel sagst: „Naja ich habe so einen *Enterprise Use Case,* wo der Content zum einen nach innen wirkt, also sprich ich muss gleichzeitig sowohl meine Webseite und das Intranet damit beschicken, sowie aber auch eben den interaktiven Messestand und so weiter, dann setzt sowas schnell organisatorische Begebenheiten voraus, die du erstmal schaffen musst. Also halt, dass auch abteilungsübergreifend gedacht wird. Also Content Management ist auch immer eine Form von meine Burg, deine Burg. Da gibt es Leute, die dann sagen: „Ja, unsere Website, unsere Website, dafür ist das Marketing verantwortlich und für das SAP-System ist die IT-Abteilung verantwortlich. Der Sales ist für CRM verantwortlich und das Versprechen von MACH ist, dass ich das alles zusammenführen kann und dann ein Bild habe von meinem Kunden, von meinen Besuchern, von meinen Lieferanten, von wem auch immer und denen biete ich allen am Ende eine Digital Experience an.

00:44:10 Agentur-Experte 1

Ja, da müssen aber die Abteilung anfangen, miteinander zu arbeiten. Und wer führt das Boot und die meisten Organisationen haben diese erste Aufgabe der Digitalisierung gerade mal so bestanden. Die größte Aufgabe, die sie in den letzten Jahren hatten, war Homeoffice zu organisieren mit Remote Work.

00:44:34 Agentur-Experte 1

Und jetzt? Sind die Unternehmen wirklich kundenzentriert? Wüsste ich nicht. Also die wenigsten tun etwas, weil ihre Kunden das von ihnen erwarten, sondern manche tun etwas, weil der andere Wettbewerber, das auch so macht. Und ja. Schwierig. Also du hörst ein wenig Frust bei mir heraus. Ich mache seit über 25 Jahren Digitalisierungsprojekte, wo es immer darum ging Produkte zu den Besuchern der Webseite zu bringen und Interaktionen und Käufe und Abwicklung und alles zu ermöglichen. Und manchmal denke ich mir, ich bin in so einer Zeitschleife.

00:45:28 Agentur-Experte 1

Ja, weil es immer wieder am Grundsätzlichen fehlt, am Verständnis. Klar hat das Mobiltelefon unheimlich viel dazu beigetragen, dass das so sehr nutzbar geworden ist. Nichtsdestotrotz merkst du, da gibt es immer Glas, also Wände, wo du nach oben nicht weiterkommst. Ja und, das ist schwierig.

00:45:57 Forscher

Okay.

00:45:58 Agentur-Experte 1

Also ich hab ein letztes Beispiel. Ich habe einen Führerschein, einen alten Führerschein, den musste ich erneuern, um einen Scheckkartenführerschein zu bekommen.

Und um das zu machen, musste ich einen Termin ausmachen, wo die Terminverfügbarkeit erratisch war. Also ich musste auf eine Webseite gehen und dann gab es entweder Termine oder keine Termine so. Dann musste ich ein Passfoto machen lassen. Dann musste ich vor Ort eine Gebühr bezahlen. Und dann hieß es: „Ja, ihr Führerschein, bis zu 4 Wochen kann es dauern.“ Die Führerscheinherstellung ist eigentlich ein digitaler Prozess. Ja, die Sammeln das alles vor Ort ein und dann wird das eingeschickt. Bei der Bundesdruckerei wird das alles zusammengeführt und die verschicken das. Was ich nicht hatte, nachdem ich diesen Antrag gestellt habe und das hätte ich erwartet, dass ich eine E-Mail-Bestätigung zum Erhalt bekomme, dass ich eine Tracking Nummer nach Versand bekomme, mit der ich prüfen kann, wo mein Führerschein sich aktuell befindet, in welchem Zustand, etc. PP. Ich hätte vielleicht am Ende sogar einen digitalen Führerschein bekommen, den ich in einer App schon hätte sofort nutzen können. So, und das ist eigentlich, wo ich sage: Wo ist das Problem? Das Problem ist: Menschen wollen an vielen Stellen Prozesse nicht digitalisieren. Die Technologie ist seit 20 Jahren ja.

00:48:03 Forscher

War das in Berlin? Denn das klingt so nach Berlin.

00:48:09 Agentur-Experte 1

Nein, das ist hier in Frankfurt am Main. Ich hab schon einen digital lesbaren Personalausweis seit 6, 7 Jahren. Ich konnte während der Pandemie einmal ein polizeiliches Führungszeugnis, das ich für eine Ausschreibung gebraucht habe, vollständig online erhalten, nichtsdestotrotz das Zertifikat oder das Führungszeugnis war auf Papier und das hat wieder diesen Bruch gemacht. Wo ist das Problem zu sagen: Hier unter dieser URL, mit dem Passwort, liebe Anfordernde, hier ist das digital signierte Zertifikat von Herrn Dobberkau, sein Führungszeugnis.

00:48:58 Agentur-Experte 1

Die Leute denken nicht in Web-Dimensionen. Die Leute denken, ich nehme einen Papierprozess, digitalisiere den dabei und dabei bloß keine Menschen einsparen, denn das ist ja nicht in Ordnung. Dass diese Menschen aber auf der anderen Seite überall für anderes gebraucht werden, das ist ja der Sinn von Informatik. Die soll eigentlich die Menschen befreien von der unnötigen Arbeit.

00:49:40 Forscher

Ja, da ist Potenzial, auch mit standardisierten Schnittstellen, worüber Behörden dann zusammenarbeiten können.

00:49:44 Agentur-Experte 1

Ja, auch Praktiken und Methoden und, dass man irgendwann mal sagt: Hey, das verdammte Web hat ja dafür auch schon die Sachen, mit Zertifikatsunterschriften et cetera PP. Na ja.

00:50:05 Agentur-Experte 1

Gut, letzte Frage vielleicht für dich, oder?

00:50:10 Forscher

Eine abschließende Frage noch. Ich bin gleich mit dem nächsten Gesprächspartner verabredet. Denkst du, dass MACH-basierte CMS in Zukunft traditionelle CMS verdrängen werden?

00:50:30 Agentur-Experte 1

Nein, also ich denke, es wird ein Nebeneinander geben. An manchen Stellen wird es vielleicht sogar zu einer Symbiose kommen, weil, wie gesagt, eben traditionell CMS werden von den Prinzipien der MACH-Welt lernen und umgekehrt werden die MACH-basierten Systeme natürlich auch versuchen möglichst viele Dinge auch zu bündeln.

00:51:08 Agentur-Experte 1

Es ist wie, wenn du mich fragst, was wird überleben? Bilder, Puzzles, Legosteine oder fertige gebaute Playmobil- oder Barbie-Häuser. Verstehst du, was ich meine? Am Erde werden die Leute mit Lego und Barbie und Playmobil parallel spielen. Also das ist halt einfach so, so sind die Menschen. Die werden sich nicht auf eine einzelne Technologie verlassen und, und, und. Es wird in einer heterogenen Welt weitergehen. Ich sehe halt die einzige Chance in dem Thema, dass die Technologie am Ende Portabilität ermöglichen sollte. Also Portabilität sollte vorhanden sein. Ich meine das Versprechen bei Open Source

ist da und geht darüber hinaus, da heißt die Leute können nicht nur ihre Daten nutzen, sondern auch tatsächlich ihre Anwendung weiterentwickeln, wenn sie die Fähigkeiten dazu haben. Das ist so ein bisschen die Frage, ob die MACH-Technologie es schaffen wird, noch zusätzliche Standards zu setzen oder Standards zu etablieren, die es vielleicht auch schon gibt. Ja.

2.5 Interview D

00:00:02 Forscher

Seit wie vielen Jahren beschäftigst du dich mit Content Management Systemen?

00:00:15 Agentur-Experte 2

Seit ungefähr 18 Jahren. Ich weiß nicht mehr, ob ich 2006 oder 2003 angefangen habe mich mit TYPO3 und damit mit meinem ersten Content Management System zu beschäftigen.

00:00:25 Forscher

OK, das ist ja schon eine ganze Weile. Was hat dich dazu bewegt, in diesem Bereich zu arbeiten?

00:00:32 Agentur-Experte 2

Wir brauchten damals für den AStA der Uni eine neue Website und dann habe ich ein bisschen recherchiert. Also ich wusste, man braucht ein Content Management System dafür und baut das nicht mehr mit handgeschnitzten HTML und da habe ich ein bisschen nach Open Source System recherchiert und bin dann bei TYPO3 gelandet.

00:00:58 Forscher

In welcher Rolle arbeitest du aktuell?

00:01:02 Agentur-Experte 2

Ich bin eine Mischung aus Entwickler, Consultant und Trainer. Also Entwickler bei den Projekten, die ich selbst betreue, Consultant wenn mich Kundinnen oder Kunden für knifflige Probleme dazu holen und Trainer, wenn es darum geht, neue Dinge zu lernen, entweder im Bereich TYPO3-Entwicklung oder im Bereich Führung.

00:01:27 Forscher

Das heißt, TYPO3 ist auch das einzige CMS geblieben, mit dem du Erfahrung gesammelt hast?

00:01:34 Agentur-Experte 2

Ich habe mal ein bisschen in Joomla! reingeguckt. Und wir benutzen für eine andere Website noch WordPress. Also dann eher für einen Blog, aber auch ein bisschen als CMS. Aber hauptsächlich mache ich TYPO3.

00:01:55 Forscher

Sind wir denn in deiner Arbeit schon mal die Stichwörter Microservices, API-First, Cloud-Native oder Headless begegnet, an irgendeiner Stelle?

00:02:07 Agentur-Experte 2

Also ich habe die alle schon mal gehört, wenn auch nicht in dieser Kombination MACH. Im TYPO3 Kontext ist mir nur Headless als einziger Begriff inhaltlich tatsächlich begegnet.

00:02:32 Forscher

OK und inwiefern im TYPO3 Kontext?

00:02:43 Agentur-Experte 2

Entwicklungen TYPO3 Headless zu betreiben, gibt es schon eine Weile und irgendwann haben die Beteiligten diese Extension gebaut und dadurch hatte ich so am Rand ein bisschen mehr davon gehört, auch wenn ich die selbst bisher noch nicht benutzt habe und auch TYPO3 bisher noch nie Headless benutzt oder betrieben habe.

00:03:03 Forscher

OK, also war das noch keine Anforderungen in einem Kundenprojekt?

00:03:09 Agentur-Experte 2

Nein. OK, wir können einfach mal gucken, ob ich trotzdem ein sinnvoller Interviewpartner für dich bin oder dann einfach die falsche Zielgruppe für dein Interview bin.

00:03:50 Forscher

Auf welchen Markt konzentriert sich TYPO3 aktuell?

00:03:56 Agentur-Experte 2

Da es gibt zwei Zielbereiche, der eine, wo es sich historisch drauf konzentriert und als was es auch immer noch vermarktet wird, nämlich als Enterprise Content Management System, also sehr große Systeme, Mehrsprachigkeit, ausgefeilte Rechteverwaltung und in den letzten Jahren auch gezielt mit Fokus auf Non-Profit- oder Regierungsorganisationen. Also Ruanda zum Beispiel benutzt TYPO3 und die deutsche Regierung ja auch, für den Government Website Builder, heißt es glaube ich.

00:04:41 Forscher

Genau, der Government Site Builder. Da hatte TYPO3 kürzlich wieder eine Ausschreibung gewonnen, um den auch weiterzuentwickeln. Da könnte man von einem politischen Einflussfaktor für Innovationsdruck bei TYPO3 sprechen. Fallen dir noch andere Einflussfaktoren ein, wo bei TYPO3 Innovation gefragt ist?

00:05:19 Agentur-Experte 2

Also wir haben starke Konkurrenz durch andere PHP, insbesondere PHP-basierte Content Management Systeme, also wie WordPress oder Drupal und verlieren auch tendenziell Marktanteile an diese Systeme. Das ist also auf jeden Fall Druck da, auch wenn das trotzdem Partnersysteme sind. Also wir arbeiten nicht gegeneinander, aber das ist halt ein

weiterer Druck für TYPO3 auch technologisch besser zu werden. Ansonsten die rechtlichen Anforderungen ab 2025 müssen ja auch Nicht-Regierungs-Sites barrierefrei oder barrierearm sein und da hat sich deswegen auch in TYPO3 sehr viel getan. Das hast du wahrscheinlich mitbekommen. Also du hast wahrscheinlich auch mit CPS (Anm.d.Red.: Akronym für coding. powerful. systems. - Agentur mit Sitz in Berlin) zu tun und die haben das ja auch sehr vorangetrieben da. Ansonsten arbeiten viele auch in Sachen Barrierefreiheit, also Usability, aber weniger aus Druck, sondern einfach, weil sie das wichtig finden. Das heißt, so kommt eher der Anstoß von innen als von außen. Druck klingt immer so nach von außen.

00:06:36 Forscher

Denkst du, dass die Entwicklung hin zu mehr Cloud-basierten Lösungen auch TYPO3 betrifft? Oder ist es eigentlich ein ganz anderer Markt?

00:06:49 Agentur-Experte 2

Nee, da tut sich auch was. Dadurch, dass auch andere Systeme, soweit ich das Mitkriege, Cloud-basiertes Hosting erlauben, entsprechend mit Skalierbarkeit. Da tut sich bei TYPO3 auch was, aber das ist halt noch ein bisschen am Anfang. Also gibt es Lösungen, aber es ist noch nicht so richtig out-of-the-box so weit. Und es gibt natürlich noch Dinge, die einfach technologisch dafür sorgen, dass wir zumindest nicht hinterher hängen. Also Sachen wie, es gibt neue PHP-Versionen, es gibt neue symfony-Versionen, es gibt eine neue Doctrine DBAL-Version und so. Wobei TYPO3 den Anspruch hat mit den aktuellen Versionen zu arbeiten und Upgrades in gemischten Systemen nicht zu blockieren und entsprechend die Performance-Verbesserungen und Features und Sicherheitsfixes mitzunehmen.

00:08:46 Forscher

Wie gut lassen sich denn die aktuellen Anforderungen mit der Architektur von TYPO3 lösen? Gibt es da irgendwo Stellen, wo man vielleicht aneckt mit der aktuellen Architektur, wenn man versucht etwas zu lösen oder lässt sich im Prinzip alles umsetzen?

00:09:07 Agentur-Experte 2

Also die aktuellen Anforderungen lassen sich umsetzen. Einige Sachen waren sehr viel Arbeit, also zum Beispiel der Umstieg auf Doctrine DBAL 4, das war sehr viel Arbeit, weil TYPO3 sehr viel selbst an Datenbankabstraktion betreibt. Und das wird sich auch in Zukunft in die Richtung verändern, dass TYPO3 da weniger selbst Sonderlocken baut, sondern mehr Doctrine DBAL Standardlösungen benutzt. Also damit wir zum einen nicht das selbst erfundene Rad weiter betreiben müssen und zum anderen damit Upgrades auf zukünftige Doctrine DBAL Versionen entsprechend weniger Arbeit sind.

00:09:50 Agentur-Experte 2

Und die anderen Sachen lassen sich lösen. Also architektonisch passt es, unabhängig davon, dass man natürlich noch Sachen anpassen oder noch bauen muss. Also Barrierefreiheit ist schon gut, aber es kann natürlich noch besser werden. Und beim Government Site Builder kenne ich die Anforderungen nicht.

00:10:25 Forscher

Ja, ist zum Beispiel eine Anforderung, dass TYPO3 Software-as-a-Service betrieben werden soll. Da wäre die Frage ist es möglich TYPO3 Software-as-a-Service anzubieten? Oder ist das eher schwierig?

00:10:55 Agentur-Experte 2

Es ist eher schwierig. Also man kann sich wahrscheinlich was bauen, das ganz viele Dinge automatisiert. Also zum Beispiel das Setup zu standardisieren, das geht, aber es geht nicht auf out-of-the-box.

00:11:35 Forscher

Die nächste Frage wäre, ob TYPO3 in Konkurrenz zu MACH-basierten CMS steht.

00:11:47 Agentur-Experte 2

Hast du Beispiele für MACH-basierte CMS?

00:11:52 Forscher

Contentful oder ContentStack zum Beispiel. Ich weiß nicht, ob du schon mal von der MACH Alliance gehört hast.

00:12:05 Agentur-Experte 2

Ne. Da kann ich tatsächlich wenig zu sagen.

00:12:31 Forscher

Dann lassen wir die Fragen diesbezüglich weg. Im zweiten Teil geht es darum welche Ziele man sich für die Weiterentwicklung von TYPO3 setzen könnte. Da verfolgt TYPO3 laut Strategie aktuell keine Ziele in Richtung Cloud-basierter Lösungen. Trotzdem möchte man anpassungsfähig bleiben. Widerspricht sich das oder geht es da um eine Fokussierung auf einen bestimmten Bereich?

00:13:22 Agentur-Experte 2

Ich würde es als Fokussierung ansehen. Ich würde auch trennen dazwischen, ob TYPO3 grundsätzlich Software-as-a-Service-fähig ist und dem Anbieten dessen und das Gleiche bei Cloud-basierten Lösungen. Es ist explizit ausgeschlossen, dass die TYPO3 Association oder die TYPO3 GmbH so etwas anbieten, weil sie explizit keine Konkurrenz zu den Agenturen sein wollen. Das Anbieten würde auf die Agenturen oder Firmen fallen.

00:14:11 Agentur-Experte 2

Bisher war es so, dass es wenig Bestrebungen gab TYPO3 Cloud-basiert anzubieten, oder Software-as-a-Service-basiert. Wenn es so etwas gibt, wäre das üblicherweise so, dass nicht die TYPO3 Association das Beschließen würde, sondern eher, dass irgendjemand sagen würde: „Okay, wir würden das gerne anbieten. Wir bauen eine Extension dafür oder eine Plattform, suchen uns ein Team und stoßen das an, dass es ganz offiziell wird.

00:14:42 Agentur-Experte 2

Das heißt solche neuen Sachen kommen eher aus einem Bedarf und davon, dass jemand einfach mal macht und weniger, dass jemand sagt, OK, wir bräuchten das mal und entwickeln quasi spekulativ in diese Richtung.

00:14:56 Forscher

OK. Lässt sich denn so eine MACH-Architektur (..) Du hast jetzt nicht so viel Erfahrung damit, aber lässt sich das mit Open Source in Einklang bringen oder gibt es Reibungspunkte? Also, MACH bedeutet ja Microservices, API-First, Cloud-native SaaS und Headless. Lässt sich das in Einklang bringen?

00:15:50 Agentur-Experte 2

Ja, klar, grundsätzlich schon. Also der einzige Punkt, wo es eventuell einen Konflikt gibt ist Software-as-a-Service, weil wenn der Code frei verfügbar ist und dann unter einer entsprechend offenen Lizenz steht, dann würde das den Wettbewerbsvorteil von Firmen reduzieren, die genau diesen Code auf ihre Server packen, weil Sie keinen eigenen Code mehr haben, der das Software-as-a Service-mäßige, dann ausmacht.

00:16:23 Agentur-Experte 2

Andererseits, WordPress macht so etwas ähnliches. Also man kann WordPress mieten und das Hosting macht halt (...) Also ich weiß nicht genau welche Organisation dahintersteht, aber man kann einfach WordPress mieten, Software-as-a-Service und das scheint zu funktionieren.

00:16:40 Agentur-Experte 2

Das heißt, in der Praxis vermute ich, dass es kein so großer Widerspruch wäre, weil der Vorteil von Software-as-a-Service ist ja nicht der Kostenvorteil, sondern dass man weniger Arbeit mit dem ganzen Kram hat, indem jemand anders das Hosting und das Provisioning und so weiter für einen übernimmt. Insofern sehe ich da keinen Widerspruch und denke das kann man wunderbar zusammen machen.

00:17:05 Forscher

OK und im letzten Bereich wollte ich fragen, inwieweit denn eigentlich diese MACH-Kriterien, die MACH-Architektur Chancen und Risiken bieten für die Weiterentwicklung von TYPO3. Und da würde ich direkt mal mit dem ersten anfangen. Microservices. Die Microservices Architektur bedeutet das ganze System in kleine unabhängige Dienste zu zerlegen. Das hätte verschiedene Vorteile, verschiedene Nachteile, aber wäre das für TYPO3 eine Option oder würde das eigentlich alles nur viel zu kompliziert machen?

00:18:06 Agentur-Experte 2

Nee, TYPO3 ist, wenn du eine Microservice-basierte Architektur haben willst, nicht das richtige Werkzeug. Also das ist nicht sinnvoll. Also genau, ich kann mir den Käse zwar mit einer Schere schneiden, aber sinnvoll ist es nicht.

00:18:29 Forscher

Das heißt, es würde keine Vorteile bringen in dem Sinne?

00:18:32 Agentur-Experte 2

Genau.

00:18:34 Forscher

Weil das System, so wie es ist, richtig gut funktioniert.

00:18:38 Agentur-Experte 2

Genau, es ist halt ein monolithisches System und es bietet sich einfach nicht an, dass dann in Microservices zu betreiben. Es lässt sich nicht sinnvoll aufsplitten.

00:20:35 Forscher

Was denkst du, wie gut lässt sich TYPO3 in andere Umgebungen integrieren, also beispielsweise in MACH-basierte Umgebungen?

00:21:13 Agentur-Experte 2

Ja, das müsste man programmieren. Und das hängt aber auch davon ab, was du brauchst, also ob du einmalig den Content migrieren möchtest oder den Content synchronisieren möchtest. Aber das musst du dann alles bauen. Also es geht ab Werk so nicht. Und man muss halt auch überlegen, ob das sinnvoll ist, also ob man dafür ein Content Management System wie TYPO3 benutzen möchte.

00:22:06 Forscher

Cloud-native rein vom Begriff her, kann TYPO3 nicht mehr sein, weil es eben schon ein etabliertes System ist, was seit vielen Jahren existiert. Wir haben vorhin auch schon mal drüber gesprochen, dass es schon Anbieter gibt, die TYPO3 Software-As-A-Service anbieten. Wäre es dann sinnvoll, dass man vielleicht so eine Art vorkonfigurierten Container für TYPO3 anbietet, die man dann einfacher in der Cloud bereitstellen kann?

00:22:45 Agentur-Experte 2

Nee, eigentlich nicht. Das harmoniert nicht gut mit der Architektur von TYPO3. Man würde halt eher auf einen PHP-Container setzen oder einen Composer Container.

00:23:03 Agentur-Experte 2

Ich überleg gerade. Meinst du ein Image, dass man herunterlädt oder dass man containerisiert arbeitet und wie Heroku zum Beispiel einfach einen neuen Container hochfährt, wenn man eine neue Version deployen möchte?

00:23:19 Forscher

Die Voraussetzung für so Container ist natürlich, dass man das System auch ein bisschen mehr in Microservices zerlegt.

00:23:28 Agentur-Experte 2

Nee eigentlich nicht, also du kannst einen Container für den Webserver haben, einen für die Datenbank oder so das geht ja. Also du musst nicht alles Ver-Microservisieren, damit das containerisiert Sinn ergibt.

00:23:46 Forscher

Ja, OK, es sind dann halt größere Container. Die Frage ist, ob die dann die Möglichkeiten nutzen können, die in der Cloud zur Verfügung stehen, zur Skalierung beispielsweise. Das war eigentlich meine ursprüngliche Frage, ob Vorlagen sinnvoll wären, dass man dann einfacher ein Setup in die Cloud laden kann, das automatisch funktioniert.

00:24:24 Agentur-Experte 2

Ach so, ja. Das auf jeden Fall. Das wäre auf jeden Fall sinnvoll. Klar.

00:24:34 Forscher

Und fehlen in Bezug auf die Bereitstellung in der Cloud offene Standards?

00:24:43 Agentur-Experte 2

Eventuell. Ich, ich habe keine Ahnung, ob es so welche gibt oder nicht. Insofern kann ich das nicht sagen. Ich hab noch keine gesehen dazu, aber das heißt nicht, dass es keine gibt.

00:25:09 Forscher

Das MACH-Kriterium „Modular Implementation“ ist dem der Microservices sehr ähnlich, in meinem Verständnis, bzw. es baut darauf auf. Ich habe mich gefragt, wäre dieser Punkt überhaupt relevant für TYPO3? Weil TYPO3 ist schon sehr modular durch seine Extensions, durch Composer. Kann man in der Hinsicht noch etwas verbessern?

00:25:48 Agentur-Experte 2

Ja, also man könnte die Abhängigkeiten zwischen den einzelnen System-Extensions reduzieren. Da wird auch gerade dran gearbeitet, sodass man zum Beispiel nicht immer den Extension-Manager oder das Install-Tool mit installieren muss, bei Systemen, wo man die GUI davon nicht braucht. Das wäre wahrscheinlich hilfreich.

00:26:12 Forscher

Okay, dann zur Agilität und Erweiterbarkeit. Die MACH Alliance fordert, dass ein System automatisch Updates und Upgrades durchführt, rückwärts kompatibel ist und dass es eine API-Versionierungsmethodik gibt. Bei TYPO3 gibt es schon die Upgrade Wizards, da gibt es DDEV, was einem hilft, beim Sprung über eine Major-Version hinweg. Sind solche automatischen Updates wünschenswert bzw. warum macht man das noch nicht bei TYPO3?

00:27:13 Agentur-Experte 2

Also TYPO3 ist darauf ausgerichtet, dass man üblicherweise keine Standard-Site-Konfiguration hat, sondern eine angepasste, eventuell mit eigenen Extensions und Minor-Updates könnte man theoretisch machen. Major-Upgrades erfordern üblicherweise immer Handarbeit wegen der Custom-Extensions und wegen der Third-Party-Extensions. Man kann es halbautomatisieren, indem man mit Dependabot oder Renovate die Composer-Abhängigkeiten hochzieht aber die Sachen, wo man tatsächlich Hand anlegen muss, die lassen sich halt per se nicht gut automatisieren. Da kann man sich halt nur Tool-

Unterstützung holen, z.B. mit Rector, in dem man den Code entsprechend halbautomatisiert aktualisiert.

00:31:01 Forscher

Hattest du schon mal Schwierigkeiten, TYPO3 zu skalieren in einer Produktivumgebung?

00:31:13 Agentur-Experte 2

Möchtest du wissen, ob ich die Anforderungen hatte oder ob ich es versucht habe und es schwierig wurde?

00:31:23 Forscher

Genau, ob die Anforderung bestand und ob das dann mit TYPO3 gut funktioniert hat.

00:31:30 Agentur-Experte 2

Wir hatten das mal mit einem Projekt. Aber es ist schon sehr, sehr lange her, da ging es darum horizontal zu skalieren. Das ging dann auch, war aber sehr viel Handarbeit. Also da wurde vor allem mit Primary/Secondary Datenbanken gearbeitet. Das hat funktioniert. War halt viel Handarbeit, das einmal einzurichten.

00:33:08 Forscher

Kommen wir noch zum Punkt Stabilität. Gibt es denn ausreichend Monitoring Tools, das man schauen kann, wie es meiner TYPO3-Instanz gerade geht?

00:33:27 Agentur-Experte 2

In dem Bereich habe ich keinen guten Überblick. Ich weiß, dass Leute Sentry benutzen und dass das eine sehr feine Sache ist. Ich weiß aber nicht, was Leute ansonsten noch benutzen. Also speziell für TYPO3, spezifische Sachen und nicht nur so Sachen, die kucken wieviel Last hat der Webserver gerade oder so. Sentry ist aber auch eher fürs Logging und jetzt weniger, um die Last zu überprüfen. Ich glaube aber das Lastenmonitoring würde ich nicht in TYPO3 selbst machen, sondern würde ich halt irgendwas nehmen, was auf den Webserver kuckt, auf den Datenbank-Server und so.

00:37:21 Forscher

Noch eine Anforderung der MACH Alliance ist „Easy-to-Use“, also schnelle Erlernbarkeit und auch viele verfügbare Trainings. Denkst du, dass bei TYPO3 in der Hinsicht noch viel zu tun ist?

00:37:38 Agentur-Experte 2

Ja, die, das sind 2 Sachen. Es gibt sehr viele Dokumentationen und es gibt auch viele online Trainings und auch Leute, die Trainings anbieten. Gleichzeitig ist TYPO3 ein sehr komplexes System, zu dem man sehr viel lernen kann in verschiedenen Gebieten. Das heißt, dass es Trainings gibt, sorgt nicht dafür, dass es weniger Inhalt ist, den man lernen muss, um bestimmte Dinge machen zu können und wenn man schnell ganz schnell ein System benutzen können will, dann widerspricht sich das ein bisschen dem, dass TYPO3 ein sehr komplexes System ist und auch für komplexe Anforderungen gedacht und dafür ausgelegt ist. Wenn man eher was Kleines hat, dann würde ich auch von jemandem, der

als Consultant unterwegs ist, erwarten, dass die Person dann sagt, OK, in dem Bereich könntest du TYPO3 benutzen, aber ich glaube WordPress würde besser passen.

2.5 Interview E

00:00:02 Forscher

For how many years have you been working with content management systems?

00:00:12 Hosting-Experte 1

We are working with different CMS for more than 15 years. With TYPO3 we are working with for 11 years. Yes. So that's quite long journey.

00:00:44 Forscher

Why did you choose to work with CMS?

00:00:51 Hosting-Experte 1

I'm coming from the technical background. At that time, 15 years ago, PHP was rising, and I chose it because of PHP. That is one reason and 15 years ago, there was very less scope of work. Like you have a framework or CMS and there was nothing more than that. So, you already had less choice about choosing the industry. So, content management was one of the rising industries and it is becoming bigger every year.

00:01:54 Hosting-Experte 1

And PHP is the most popular language among CMS. I know one statistic that almost 70% of websites are in PHP. So, PHP is another reason I love open-source CMS.

00:02:25 Forscher

Yes. Interesting. What was your first CMS?

00:02:32 Hosting-Experte 1

It was WordPress. Yeah, because at the time WordPress was very popular and the first thing you hear when asking for CMS. I worked around one or two years with it.

00:02:53 Forscher

So how did you come across TYPO3?

00:03:05 Hosting-Experte 1

In my first job, I think that was the only company who worked with TYPO3 in the whole of India. So, just by following my job, I got to TYPO3.

00:03:15

Then I just learn and start comparing with other CMS, like WordPress, Joomla!, Drupal there was many other CMS there and I found that it is concretely developed and robust.

These features I really like. From the core perspective. It is very well designed. So, after I left the job, I started with TYPO3 I continued to work with TYPO3 and still am.

00:03:45 Hosting-Experte 1

So, the comparison was quite clear for me. In technical terms, in user related terms, in terms of features TYPO3 of is one of the best.

00:03:58 Hosting-Experte 1

So, we keep continuing with that.

00:04:01 Forscher

What would you say is the biggest strength of TYPO3?

00:04:08 Hosting-Experte 1

You know, if you take any open-source CMS or any commercial one, you will not find that many features within the core, like multi-lingual or multi-site or workspaces or users and roles. If you take a deep look you see TYPO3 has everything you need for an ideal CMS.

00:04:35 Hosting-Experte 1

So that is one strength, in terms of the core features and the second thing is about TYPO3's architecture. It is offering modern technology and robust solutions.

00:04:54 Forscher

What did you specialize in now? What is your current role?

00:05:06 Hosting-Experte 1

Our company is working in two ways with TYPO3. One is to work as Agency5 in TYPO3 enterprise projects and we just connected with Plattform1. You heard a lot of that. So Plattform1 is focusing on small to medium-size businesses. Plattform1 is designed for creating products and solutions which never happened before it started in TYPO3, in my opinion. It is the platform you really need for TYPO3. So, for example if you are talking about WordPress, you will get good solutions for templates, extensions, or something like SaaS. So, I'm focusing on Plattform1 with all these products and services. There are some microservices as well, some SaaS cloud services as well.

00:06:09 Hosting-Experte 1

So, my goal is that TYPO3 can be spread more, if people have a ready to use solutions.

00:06:25 Forscher

Yeah, that's basically why I wanted to talk to you because you offer TYPO3 in a software-as-a-service manner and that's one aspect of MACH architecture. So, I wanted to ask as well in what kind of context did you get in touch with the terms Microservices, API-first, Cloud-native and Headless?

00:07:11 Hosting-Experte 1

Yeah, I think all four initiated very separately, but now they are linked to modern technology. A few things, I can say like cloud-native was driven by AWS. They set a lot of standards with their cloud. So, we work with AWS for a long time, because compared to traditional hosting you can scale it, you can extend it. This is why we work cloud native.

00:07:57 Hosting-Experte 1

The API-First approach, I think, regarding TYPO3, is mostly connected with the Headless extension. On the other hand, many customers of Agency5 have already started with the API-based approach many years ago. It is important for them to have flexibility and to connect applications.

00:08:30 Hosting-Experte 1

Microservices. To be honest, I don’t have much experience with that. I know the basics of it.

00:08:59 Hosting-Experte 1

And the last one is Headless. That's a very trendy and very good thing. If you know your frontend and your backend are separate, you have a great freedom to choose any independent platform for your frontend. For example, if you are looking for an ecommerce website, TYPO3 is not that much capable to make a great ecommerce solution, so you could use the Headless API for TYPO3 and for the e-commerce application you might use Shopware or something like that which is supporting this approach. So, it is helpful for the frontend team to integrate with the backend.

00:09:42 Hosting-Experte 1

And you know, not a single platform needs to provide everything, as long as it can bind with any other solution with the Headless approach.

00:10:33 Forscher

What market is TYPO3 focused on now?

00:10:46 Hosting-Experte 1

That's a that's a very hard question because in my opinion the TYPO3 Community has two types of people to be honest. One type says it is mainly for the community and the other kind of people say that TYPO3 is designed for enterprise, which means customers and projects.

00:11:07 Hosting-Experte 1

And another part of community or people like me believe that TYPO3 is for everybody. Because it is not like that you can create a small site with TYPO3 easily in that. I'm not sure about the community, like how they are thinking. But if we talk about the future road map maybe TYPO3 Association can give a better answer, but in my opinion, TYPO3 should break the image of the enterprise level CMS because that's one of the drawbacks or maybe a reason why we loose customers to other places like, although TYPO3 has the potential to target small and medium business as well, and I think it should.

00:13:25 Forscher

Do you think there's any other factors driving innovation for TYPO3?

00:13:48 Hosting-Experte 1

In recent years, TYPO3 is very innovative. It always supports the latest technology. If we talk about TYPO3 v13, they already started with PHP 8.2 and 8.3, for example. Webhooks was introduced in version 13. Also, they are planning about a new API integration with 3rd-party-applications. So, I'm sure like compared to the first decade in the last decade, you know TYPO3 has 25 years of journey, so if you just forget the first 15 years, in the last 10 years, I can see drastic improvement for sure. There are a lot of things happening to improve the backend.

00:14:59 Hosting-Experte 1

Now, a lot of features are very stable. Considering Usability and technical innovation, it's been getting better and better every year.

00:15:12 Forscher

And do you see in what direction the competitors are moving? Is there a difference?

00:15:29 Hosting-Experte 1

In that context we are a little bit behind you know, to be honest. Like if we talk about competition, for example, WordPress has introduced the API-approach many, many years ago. right? You just set some settings and then the frontend gets the API. So, in TYPO it's not done in the core, but it's introduced as a 3rd-party-solution, so I think that would be something to adopt in the core to enable more solutions.

00:16:52 Forscher

Have you worked with some MACH-based CMS already?

00:17:01 Hosting-Experte 1

Yes, not 100%, but we already worked with the Headless-, the API-First and the cloud-native approach. These three things are already working well with TYPO3. If you look at Plattform1.com, for example, we have 3 to 4 products which concern MACH-based solutions. We have a Headless template. We have SaaS-solution. Our customers are really appreciating it and they are surprised at how we are delivering these products.

00:17:52 Hosting-Experte 1

About Headless for example. We use Vue.JS in the frontend and TYPO3 in the backend with the Headless APIs. It’s very simple and fast. So, it's completely changing the mindset about TYPO3 in that there are endless possibilities. Just like with other CMS.

00:18:22 Hosting-Experte 1

And about TYPO3 SaaS-solution, that's something no one could imagine for a long time. So, it's a little bit hard for us to convince people that it really works well. You don't need to worry about infrastructure or technology innovation or upgrades or something like that. But I think it will take time because for 25 years many agencies have created a mindset

that TYPO3 is just for the custom solution. You just need to give us a requirement, we will do some code and then you can use it. But Software-as-a-Solution is the next thing.

00:19:08 Hosting-Experte 1

For example, if we talk about WordPress, they have a SaaS-solution for years. It's not about business only, but it's also about spreading the word WordPress. So, they know some people don't want to run servers, hire programmers and everything. They just want to do some clicks and publish their site. So that's something we have been trying with TYPO3 for one year now looking forward to people's feedback on it.

00:20:04 Forscher

Yes. Interesting. Do you think there is anything to make TYPO3 more cloud friendly?

00:20:14 Hosting-Experte 1

Yes, for sure. I think in 2016 the TYPO3 Award was started. That was the goal for us. The old CEO and I had a dream to create TYPO3 SaaS-solutions, but unfortunately, it was not done, but yes, there is a lot of potential and the time is saying that it is particularly important.

00:20:44 Hosting-Experte 1

Hopefully, individuals or agencies, or core or association will take this seriously. It would be nice if you can run officially with the TYPO3 Cloud-solution. You know, it is all about trust. If you create something official. It is more impacting to the people.

00:21:13 Forscher

Yeah. Something, something like an official container image, for example. That is something I heard in other interviews.

00:21:21 Hosting-Experte 1

Yes, right.

00:21:24 Forscher

It would be interesting, yes, to host it more easily. Or, I don't know about scaling, in how far is scaling a topic for your clients or for the projects you do. You said you are focusing on the small to medium-sized clients, but do they sometimes have this requirement as well?

00:21:53 Hosting-Experte 1

Yes.

00:21:56 Forscher

How do you solve this?

00:21:57 Hosting-Experte 1

We don't have any big clients. We have small to medium size customers. So scaling is not a bit topic. But it is also not a big issue, because we have a very great infrastructure with

the cloud setting. So, there are not any limitations compared to traditional CMS or compared to traditional hosting infrastructure.

00:22:26 Hosting-Experte 1

If you talk about the big side, for example, the New York Times or whatever, like the big names, they use the cloud, that kind of solution then makes provides the scalability. So, it should not be a problem, because you have the flexibility on the infrastructure of cloud native. So, you do not need to be worried about if you have a bigger site or a bigger customer who has millions of users.

00:22:57 Hosting-Experte 1

It is not like TYPO3 Core has that duty to help manage those numbers of users. So, we have a great platform with TYPO3 CMS, and we just need to connect everything and make sure about cloud integration. I do not think it's a big trouble or challenging with TYPO3 to make it scalable for many users, if you know about the core. You just need to think about the best infrastructure for your CMS and that should not be a problem.

00:27:01 Forscher

Do you think Headless should become a core functionality?

00:27:29 Hosting-Experte 1

It should be part of the core, as an optional feature, not in general. Because the API-First approach does not fit every solution, for example, if you have a simple solution, the traditional CMS, is still the best, right? If you are talking about a small or medium size site.

00:27:50 Hosting-Experte 1

If you are talking about connecting with TYPO3 for example using a frontend technology, like Vue.js or something like that. It should not be 100% mandatory, but it's really needed as a second option. People should have the flexibility.

00:28:34 Forscher

And do you think there should be more native APIs?

00:29:01 Hosting-Experte 1

Yes, we should have more. Like, in version 12, the community already created webhooks to communicate with other APIs. Every platform should communicate with each other to create these microservices or API-based solutions. So, I think that's ongoing well and hopefully we'll have more innovation on that point.

00:29:43 Forscher

Have you ever thought of having multiple clients in one instance of TYPO3?

00:29:53 Hosting-Experte 1

In terms of a SaaS-solution?

00:29:59 Forscher

Yes. You think it could be a promising idea to have more than one client in a TYPO3 instance to share resources for example?

00:30:12 Hosting-Experte 1

Yes, that is an interesting question. We talk internally a lot about this point, but to be honest now we do not have one centralized instance for all the customers because we need to make sure about you the stability and scalability. But we are thinking about the solution. For example, we have a lot of templates and if one TYPO3 instance has one template then we could have 10 or 20 customers on it who all use the same template. That is the beauty of TYPO3, it can be realized easily because we have the user roles and access management in the back end, so we can separate the page trees and they will never realize that it's one instance. Technically it is possible, but we have not implemented that yet, but that should be possible, I'm sure. For ten smaller customers, for example, it should work with one instance. But if you have a bigger size then it is better to keep separate instances.

00:31:35 Forscher

Yeah. At the moment, the user data cannot be separated into different databases, but it could be interesting to share the resources depending on the load within one instance.

00:33:08 Hosting-Experte 1

Yes we can consider a cost-effective solution where we just need to maintain one instance. It can be also helpful to SaaS-solution-providers.

00:33:25 Forscher

And did you ever have a feature that you wanted to be implemented in the core?

00:33:35 Hosting-Experte 1

The community is really doing an excellent job. Like, they have created Headless extension and many others like news. I wish one day it could be part of the core, like it will turn into APIs within the core. You don't need to depend on any other actions.

00:34:05 Hosting-Experte 1

If you talk about the backend usability it should be very user-friendly. The workspace for example is overly complicated now. You could introduce a simple workspace. So, these are the ideas that are coming from brainstorming in community, and people are doing a great job on those points.

00:34:32 Hosting-Experte 1

And I also want to speak about AI. I think, it is one interesting topic that is coming. So, I think that's also an interesting point how TYPO3 thinks about AI technologies.

00:34:53 Hosting-Experte 1

We already created one cool type of extension called “TYPO3 OpenAI” and we are trying and experimenting how can we make automation? Like for example, we have a lot of features like SEO optimization using AI and create a one-click AI page.

00:35:12 Hosting-Experte 1

So, this kind of things we are experimenting and let's see what kind of echo or response we get from people. But that's also one interesting point, I just wanted to add.

00:35:26 Forscher

Yes, good point. I mean, especially for the editors, it would be amazing to have an AI tool integrated in their content management system so they can easily come up with a new text or image.

00:35:45 Hosting-Experte 1

It's getting more and more popular and creates more demanding tasks for content management system.

00:36:00 Forscher

AI speeds up the production of content, I agree. So, we are getting to the end of the interview. Before, I wanted to ask, did you encounter custom content types already with TYPO3. Do you use your own content types? Because that is a key feature in MACH-based CMS. They use individual content types to structure the content more semantically, less site based.

00:37:01 Hosting-Experte 1

In TYPO3, we do not use it because the core has defined enough types. Of course, for some blogs or some custom key we use it. For example, in blog extension to provide a custom page pipeline.

00:37:21 Hosting-Experte 1

2 years ago, we stopped working with STRAPI, but before, we worked 2 years with it. It is one of the famous Open-Source CMS which is based on the Headless approach, and they don't have a frontend interface, they just provide the backend and APIs. So, at that time, I realized that, as you said, that the content type is especially important. You can create a custom content type or custom content element with just few clicks there, and your API is ready for the actual CMS or frontend needs.

00:38:00 Hosting-Experte 1

If we talk about WordPress for example or any story blog, this is the first concept they always bring up, you can create a custom post type or page type or something like that.

00:38:17 Hosting-Experte 1

Well, I think in TYPO3, we never think on that due to our traditional way, but the innovations of CMS say that this is an important topic. The point is that TYPO3 has the ability for custom pages for many years.

00:38:51 Forscher

Yes. Considering this, some improvements have been made in version 13. There will be new ways to create custom content types.

00:39:07 Hosting-Experte 1

The Mask extension is a good example for that, and I think the core is getting connected with those people. In version 13 they are working with content blocks. So, I'm sure that both features will result in a great version 13. If you talk about other CMS, they just have a drag and drop, if you want to create a custom element.

00:39:52 Forscher

OK, so let's get to the end of our interview. I have some final questions. What do you wish for your work with TYPO3 in the future?

00:40:18 Hosting-Experte 1

I really wish TYPO3 will get more popular because it's simply very well deserved. It is one of the CMS that is underestimated or not getting enough popularity, due to the branding or marketing,
I don’t know, but it's been there for more than 20 years it has been the first open-source CMS and I think that is the goal of the community to spread the word about TYPO3. So that's why this is from the heart that TYPO3 deserves more attention.

00:41:07 Forscher

Yeah, that sounds amazing. Do you think the MACH-based CMS will become stronger in the future. Or will traditional CMS like TYPO3, WordPress at some point be irrelevant?

00:41:40 Hosting-Experte 1

My answers are always diplomatic because you know, no one knows what will happen, because if you take a look at first sight, MACH looks very complicated to the common people.

00:41:56 Hosting-Experte 1

So, my point is that MACH-based CMS, MACH-based solutions is the future for the big players and some medium sized businesses. But traditional CMS will stay for a while like 10 years or 15 years for sure, and it’s not a question mark because not everybody is looking for the MACH solution.

00:42:13 Hosting-Experte 1

If you are talking about small to medium sized business, they just want a simple backend and then the site should work well.

00:42:26

But if you talk about the modern solution with enterprise clients or people who really need that kind of function. For example, those who want a mobile application connected to their websites and some other channels. Then I should think about a MACH solution, that's for sure.

00:42:41 Hosting-Experte 1

So, my point is that still it's a rising star, the MACH solutions and the traditional CMS will still be going on for some years. For sure. They won't be replaced.

00:42:59 Forscher

All right, I found another question I wanted to ask you. How could it become easier to learn and to use TYPO3?

00:43:31 Hosting-Experte 1

Ah, yes, that is where TYPO3 can improve a lot. Because if you start any CMS, they have many tutorials, books, video courses or something like that.

00:43:53 Hosting-Experte 1

And TYPO3 is a little bit complex to learn. So, the point is that in recent years the documentation is getting better and better, from installation to the deployment they really have a genuinely nice documentation. But still if you want to start it, it's very challenging to be honest, like for a beginner person without getting help. It is challenging to understand and deeply learn TYPO3. There are some good guidebooks. There are also courses in German, but not in English I think.

00:44:50 Hosting-Experte 1

The most important part is the community. If you love open source than join the Slack channel. I think there is a lot of people always happy to help you.

00:45:04 Hosting-Experte 1

But the learning curves are hard. I am not sure what do you think about this point because if we find a solution for this, then TYPO3 will get more popular among developers.

00:45:27 Forscher

Yes, it would be amazing to make it more popular among developers. So, did I understand you right that there should be more English materials as well?

00:45:50 Hosting-Experte 1

The whole documentation is in English now. But the point is that if you have a student or an intern, it is not enough. We need big tutorials or courses or realistic example projects. It is just installation documentation. It is not about getting trouble and how can you solve it? So, my point is that English materials are getting better and better every year, every day.

00:46:28 Hosting-Experte 1

I just want to give you an example like Wolfgang is one of the persons who runs courses in TYPO3 in German language. So, it would be nice to have an English version that could help people internationally.

00:46:47 Hosting-Experte 1

And I think we should try to make more virtual videos, courses, that are more interactive and helpful to the beginners.

00:47:01 Forscher

Yeah, it is important to be able to grow the community internationally as well. So how do you train your developers, actually? Or do they come prepared already?

00:47:18 Hosting-Experte 1

It is hard to find a TYPO3-developer in Germany, so in India it is almost impossible. We just take PHP-developers and train them internally. There are no other options. So, we have senior resources who always prepare the tutorials. So TYPO3 is a little bit complex to learn for developers, so a mentorship is always needed to understand deeply, for problem solving or help and support or to clear the fundamentals. I think that is the way, most agencies follow, they are tutoring resources in-house.

00:48:04 Forscher

Yes, that's the best way to learn, if you have someone experienced you can ask for help.

00:48:08 Hosting-Experte 1

I have a dream to create a training and workshops. Hopefully, someday we will get the time to do that. That would be beneficial for the community and for spreading TYPO3.

00:50:37 Forscher

Thank you for your expertise and for your time.

00:50:44 Hosting-Experte 1

It's my pleasure. I'm happy to talk with you and your thoughts on this model solutions and I wish you all the best for your thesis.

00:50:54 Forscher

Thank you.

2.6 Interview F

00:00:00 Forscher

Seit wie vielen Jahren beschäftigst du dich mit Content Management Systemen?

00:00:19 Hosting-Experte 2

20. Ich weiß nicht welche Version von TYPO3 2004 aktuell war. Keine Ahnung 3.5 oder so.

00:00:33 Forscher

Das ist nah am ersten öffentlichen Release. Das war, glaube ich, 2002, wenn ich mich recht entsinne.

00:00:41 Hosting-Experte 2

Ja, das könnte hinkommen.

00:00:44 Forscher

OK und wie bist du dazu gekommen in dem Bereich zu arbeiten?

00:00:51 Hosting-Experte 2

Ich habe damals ein Praktikum bei Agentur2 gemacht. Die haben das Hosting damals mitgemacht, bevor es dann ausgegründet wurde, als eigene Firma. Unser Chef hat damals angefangen Internetseiten mit TYPO3 zu bauen. Agentur2 hat zu dem Zeitpunkt so ganz klassisches Agenturgeschäft gemacht.

00:01:32 Hosting-Experte 2

Er hat aber dann irgendwann gemerkt, dass es irgendwie sehr, sehr schwierig ist die Infrastruktur für dieses CMS zu betreiben und ist dann relativ schnell pivotiert auf das Thema Hosting. Da zu diesem Zeitpunkt das niemand so recht angeboten hat. Mein Chef war seinerzeit in der Community sehr, sehr aktiv, hat glaube ich mit eins der ersten populäreren TYPO3 Bücher geschrieben. Das habe ich dann damals im Praktikum auch als erstes in die Hand gedrückt bekommen. Er hat gesagt: „Hier. Arbeite das mal durch! Hier sind zwei Projekte, bei denen du mithelfen kannst. Ich weiß gar nicht mehr, was für Projekte das waren. Ich glaub‘ das war so ein kommunales Nachrichtenmagazin, hier aus dem Ort. Wir mussten damals ihre Seite mit TYPO3 gebaut bekommen. Das war alles wild, wird man heute wahrscheinlich nicht mehr so machen.

00:02:27 Hosting-Experte 2

Die hatten so fiese Content-Integrationsgeschichten, wo ich dann immer die InDesign-Dateien bekommen habe, und dann mussten die irgendwie ins CMS importiert werden. Wo man halt eben so einen Praktikanten ran setzt.

00:02:43 Forscher

Und du hast dich dann mit der Zeit in die Richtung Hosting entwickelt?

00:02:50 Hosting-Experte 2

Genau, das ist immer noch so. Das war 2004, als ich das Schülerpraktikum gemacht habe. Später war ich wieder bei Agentur2 und habe dann Ausbildung und Studium gemacht. Agentur2 hat ihr Geschäftsmodell mit der Zeit in Richtung Hosting verschoben. Mittlerweile würde ich es ein bisschen ganzheitlicher betrachten. Sie sind darauf fokussiert Agenturen dabei zu unterstützen ihre TYPO3-Projekte zu betreiben und nicht nur TYPO3-Projekte. Wir sind hier in allen möglichen CMS-Communities unterwegs.

00:03:29 Forscher

Wie nennt sich deine aktuelle Rolle, in der du arbeitest?

00:03:38 Hosting-Experte 2

Ich hab habe mehrere Hüte auf. Ich habe lange traditionelle Softwareentwicklung gemacht. Ich bin dann irgendwann so ein bisschen in Richtung DevOps/Cloud gegangen, dann irgendwann in Richtung Architektur, und schließlich Richtung Cloud-Architektur.

00:04:01 Hosting-Experte 2

Wir bieten jetzt bei Agentur2 seit grob anderthalb Jahren auch eine Produktpalette an die komplett auf einer Cloud-Plattform basiert. Das war so ein Stück weit mein Baby. Jetzt im Moment geht es für mich so ein bisschen Richtung Developer Relations. Das heißt, wir versuchen im Moment, ja, so Entwickler-Ökosystem rund um unsere Plattform aufzubauen, das ist.

00:04:30 Hosting-Experte 2

Dann geht es halt darum, wie kriegt man das Produkt überhaupt für Entwickler attraktiv, wie kriegst du Entwickler in dieses Ökosystem rein. Developer Experience ist dann ein interessanter Punkt, also ein bisschen analog zur User Experience macht man sich halt auch um Developer Experience Gedanken. Das heißt, wie nehmen Entwickler überhaupt dein Produkt wahr, haben sie die richtigen Tools, kann man da Arbeitsabläufe irgendwie unterstützen? Genau, das ist so im Moment mein Hauptbereich. Ich mache auch ein bisschen Personalentwicklung. Ich bin in der Rolle im Recruiting mit drin.

00:05:11 Forscher

Ja, richtig vielfältig und genau der Richtige, um über MACH-Architektur zu sprechen. Weil du ja, schon erwähnt, mit dem Hosting von TYPO3 in der Cloud viel Erfahrung gesammelt hast. Und da will ich direkt mal fragen: Ist es öfter eine Anforderung, dass Kunden in der Cloud direkt bei großen Anbietern gehostet werden wollen oder wollen sie lieber eine On-Premise-Lösung haben?

00:05:53 Hosting-Experte 2

Das ist, das ist ganz unterschiedlich.

00:05:55 Hosting-Experte 2

Die Kunden, die wirklich On-Prem im eigenen Rechenzentrum hosten wollen, die kommen naturgemäß nicht zu uns.

00:06:03 Forscher

Ich verstehe, und ihr habt dann eure eigenen Server. Ihr nutzt nicht die großen Cloud Anbieter?

00:06:13 Hosting-Experte 2

Nee, genau, wir haben eine komplett eigene Infrastruktur. Wir haben auch ein eigenes Rechenzentrum hier am Standort. Das heißt also die Leute, die jetzt wirklich sagen: „Ich bringe mein eigenes Rechenzentrum mit, und darauf soll die Infrastruktur betrieben werden.“ Das sind traditionell nicht die Gruppe von Kunden, die wir ansprechen.

00:06:37 Hosting-Experte 2

Ich kann das jetzt nicht quantifizieren. Ich habe da keine Daten dafür, aber ich habe das Bauchgefühl, dass die Bereitschaft sich da auf Cloud-Dienstleister zu verlassen, in letzter Zeit auch bisschen gestiegen ist.

00:06:52 Forscher

Sind dir die Begriffe Microservices, API-First, Cloud-native und Headless schon mal begegnet?

00:07:04 Hosting-Experte 2

Da bin ich schon mal drüber gestolpert. Ja.

00:07:07 Forscher

Ist es in Verbindung mit dem Hosting von TYPO3 ein Thema, dass TYPO3 so ein bisschen diese Architektur fehlt, oder funktioniert es ganz gut ohne?

00:07:34 Hosting-Experte 2

Wir können das mal so ein bisschen auseinanderfuddeln. Also bei Microservices ist das prinzipiell erstmal ein Deployment-Modell, von dem ich als User, also als Endanwender, der mit dem System arbeitet, vielleicht gar nicht so viel mitkriege. Das ist dann eher so ein Thema fürs Operating. Also wie betreibe ich die Plattform dann am Ende und es ist ein Thema für den Entwicklungsprozess selbst.

00:08:08 Hosting-Experte 2

Microservices mache ich klassischerweise, wenn ich eine komplexe Fachdomäne habe und versuche, die dann irgendwie auf einzeln abgrenzbare Kontexte herunterzubrechen. Das ist auch häufig organisatorisch motiviert. Im Sinne von: „Wir bauen unsere eigene Software Microservice-orientiert, weil wir dann sagen können: „Wieviel Entwicklungsteams haben wir jetzt? Wir haben jetzt, wir haben jetzt 5, 6 Entwicklungsteams. Die haben alle ihre eigenen Microservices und können da unabhängig parallel dran arbeiten. Das ist so die häufige Motivation dafür.

00:08:49 Hosting-Experte 2

Aus Betriebssicht kommen dann noch so ein paar technische Aspekte rein. Ich kann einzelne Komponenten unabhängig voneinander skalieren oder sowas.

00:08:59 Hosting-Experte 2

Die Anforderung von Kunden, die jetzt einen CMS betreiben möchten, dass das Ganze Microservice-basiert sein muss, habe ich bisher selten vernommen.

00:09:13 Hosting-Experte 2

Was man eher sieht, also beziehungsweise was ich öfter vernehme, ist diese Anforderung, API-First getrieben, beziehungsweise Headless zu sein.

00:09:27 Hosting-Experte 2

Weil sich da der Spieß dann umdreht. Da hast du dann als Endanwender einen spürbaren Mehrwert davon, wenn ein Produkt diese Features bietet. Zum Beispiel, weil du sagst: „Wir haben sehr spezielle Anforderungen, zum Beispiel, wie sich mein Frontend verhält.“ Wir haben viele Kunden, die bauen ihre React-Frontend oder schon allein die Leute, die Multi-Channel-Marketing machen wollen und ihren Content im CMS verwalten wollen, für die ist das Klasse, wenn irgendwie noch eine mobile App davorhängt oder sowas.

00:10:10 Hosting-Experte 2

Die Anforderungen höre ich öfter muss ich gestehen, im TYPO3 Umfeld bekomme ich das ich nicht so oft mit. Ich höre das häufig aus der WordPress Ecke. Also ich weißt nicht, das ist jetzt auch eine subjektive Einzelbeobachtung. Ich habe das Gefühl, die WordPress-Leute machen im Moment ein bisschen mehr Marketing. Die hauen ein bisschen mehr auf die Marketing-Trommel, was ihre Headless-Features angeht. Vielleicht treibt das auch so ein bisschen den Bedarf.

00:10:43 Forscher

Würdest du einem Kunden TYPO3 empfehlen, wenn er die Anforderung Headless hat? Weil es ist möglich das CMS mit einer Extension Headless zu betreiben?

00:11:02 Hosting-Experte 2

Ja, bei Extensions... Wenn es um irgendwelche Drittanbieter-Integrationen geht, ich find das hat immer noch ein anderes Gewicht, als wenn es vom System nativ angeboten wird, denn es ist immer schön, wenn so eine Funktion von einer Drittanbieter Extension angeboten wird. Das ist immer ein gewisses Umsetzungsrisiko, weil ich der Weiterentwicklung dieser Extension ein gewisses Vertrauen entgegenbringen muss.

00:11:39 Hosting-Experte 2

Wenn es ein First-Party-Feature ist, dann weiß ich OK, das ist Teil des CMS selbst. Ich kann mich drauf verlassen, dass (..) Welche LTS-Version von TYPO3 ist jetzt gerade aktuell? 12, glaube ich. Jetzt kommt demnächst irgendwann 13. Wenn ich weiß, ich nutze jetzt in TYPO3 12 dieses Feature, dann weiß ich, das wird in 13 auch noch funktionieren, weil ich darauf vertrauen kann. Wenn es eine Third-Party-Extension ist, dann ist das ein bisschen so ein Glücksspiel. Vielleicht wird die von einem Typen namens Pete in seinem Keller weiterentwickelt. Ich weiß, es ist nicht so, aber trotzdem ist das halt ein offensichtliches Umsetzungsrisiko, wenn man sich auf Third-Partys lässt. Deswegen kann ich Kunden verstehen, die sagen: „Ja, Ich hätte gerne First-Party-Support für welches Feature man dann haben will (..), Headless.

00:12:39 Forscher

Verstehe. Da muss man dann abwägen, ob einem die Stärken von TYPO3 wichtig sind oder eben Headless. Denkst du, dass MACH-basierte CMS in Konkurrenz stehen zu TYPO3 oder ist es eher etwas Parallellaufendes?

00:13:19 Hosting-Experte 2

Ja, spannende Frage. Ich kann mir gut vorstellen, dass es da Schnittmengen gibt.

00:13:34 Hosting-Experte 2

Ich kann mir auch vorstellen, dass sich die Demografie, in der Anwenderschaft ein bisschen verschiebt. Weil bei den Leuten, die jetzt die Anforderung stellen nach MACH-basierten CMS, das Anspruchsniveau ein bisschen höher liegt. Also wenn du nach Microservice-Deployment fragst, wenn du nach frei zugänglichen Schnittstellen fragst, wenn du nach Headless fragst, weil du vier verschiedene Frontends andocken willst, dann ist das ist schon ein etwas gehobenes Anspruchsniveau. Das hast du nicht bei jedem Projekt. Da bist du dann eher im Umfeld der etwas komplexeren Projekte. Du bist bei

größeren Unternehmen, die diese Anforderungen mitbringen oder eben im etwas komplexeren Umfeld und das ist ein Bereich, wo TYPO3 sich auch bewegt und wo TYPO3 sich, glaub ich, jetzt im Moment strategisch rein entwickelt.

00:14:50 Hosting-Experte 2

Ich erinnere mich noch, vor 10, 20 Jahren war das anders. Vor 10, 20 Jahren, hast du auch noch Agenturen gehabt, die dann so die typischen Kleinunternehmen oder Kleinstunternehmen mit TYPO3-Webseiten betreut haben. Keine Ahnung, so Bäckerei oder Friseursalon, denen haben wir von der Agentur eine TYPO3-Webseite verkauft. Oder das kommunale Nachrichtenblatt, von dem ich vorhin erzählt habe, so vor 20 Jahren. Da war das Gang und Gäbe, dass man sowas mit TYPO3 gebaut hat.

00:15:28 Hosting-Experte 2

Das willst du heute wahrscheinlich nicht mehr machen, weil solchen Kunden würdest du entweder ein WordPress verkaufen, denn es ist einfach aufgesetzt, es ist einfacher in der Betreuung. Es ist für die Entwickler einfacher, da ein vordefiniertes Theme zu kaufen und das vielleicht anzupassen. Entweder WordPress oder du schickst die Leute halt gleich zu irgendeinem Homepage Baukasten wie SquareSpace und Co. gibt es ja auch.

00:15:54 Hosting-Experte 2

Das heißt, das ist, glaube ich, ein Segment, wo diese komplexen Systeme wie TYPO3 sich zukünftig nicht mehr so großartig werden positionieren können. Und das merk ich bei TYPO3 auch. Also ich weiß nicht so sehr, wie du da den Markt verfolgst. Im Moment hab ich das Gefühl, TYPO3 fokussiert sich sehr stark aufs Enterprise und auf die öffentliche Hand, öffentliche Verwaltung.

00:16:29 Hosting-Experte 2

Und das sind halt schon so die Segmente, wo es auf technisch komplexe Projekte ankommt, wo Deployment auf einmal eine wichtige Frage ist. Wo auch Headless auf einmal eine Anforderung wird, weil es eben verschiedene Kanäle gibt, über die ich Content ausspielen will und in dem Sinne gibt es da wahrscheinlich (.) Das war jetzt eine sehr, sehr lange Einleitung. (.) In dem Sinne gibt es da wahrscheinlich schon Potenzial, dass TYPO3 in Konkurrenz zu solchen Systemen steht.

00:17:07 Hosting-Experte 2

Jetzt hoffe ich, du konntest folgen. Das war ein recht langer Monolog.

00:17:14 Forscher

Es war schön eingeleitet und durchaus verständlich. Es bestätigt auch ein bisschen meine Annahme, dass da Innovationsdruck herrscht. Auch von politischer Seite, weil man mit TYPO3 zusammenarbeiten will, um öffentliche Webseiten anzubieten, auf denen auch mal viel Traffic in kurzer Zeit auftreten kann oder durch andere Anforderungen, die man gut mit MACH-basierter Architektur lösen kann. Und dass man TYPO3 da vielleicht auch noch ein bisschen weiterentwickeln muss, damit das besser funktioniert. Denkst du denn, man könnte TYPO3 in kleinere Bausteine zerlegen, die sich dann besser elastisch skalieren lassen in einer Umgebung, die sowas ermöglicht?

00:18:27 Hosting-Experte 2

Es gibt verschiedene Ansätze in der Literatur, allein wenn es darum geht, komplexe, monolithische Anwendungen in Microservices zu zerlegen. Also da könnte ich jetzt drei Stunden lang erzählen, weil es das Thema ist, über das ich damals meine Abschlussarbeit geschrieben habe. Da gibt es verschiedene Ansätze dazu. Keiner davon ist einfach und je älter und historisch gewachsener, so ein System ist, desto komplexer wird das. So ein einzelnes, historisch gewachsenes monolithisches System wieder in einzelne Bausteine zu zerlegen, ist ein Prozess, den man sehr, sehr bewusst machen muss, durch den ich auch wieder neue Komplexität in so ein System hineinbringe, die ich dann beherrschbar machen muss.

00:19:30 Hosting-Experte 2

Dann habe ich auf einmal neue Kommunikationswege in so einem Gesamtkonstrukt CMS, die dann über Schnittstellen kommunizieren, die alle über das Netzwerk laufen. Und dann muss ich mir überlegen: Wie gehst du damit um? Wie gehst du mit den erhöhten Latenzen um? Das sind sehr harte, greifbare, technische Probleme, die man dann lösen muss.

00:20:02 Hosting-Experte 2

Man muss sich auf einmal Gedanken über Resilienz-Strategien machen. Also was passiert, wenn, nehmen wir an, du hast einen CMS Core, teilst ihn in 3 Microservices auf, was passiert, wenn die nicht mehr miteinander kommunizieren können, Ich gehe davon aus, dass das Netzwerk nicht verfügbar ist bzw. ich gehe davon aus, dass es ausfällt. Und das sind alles Probleme, die man lösen muss, wenn man ein monolithisches System in Microservices überführen will.

00:20:38 Hosting-Experte 2

Wenn man jetzt auf der grünen Wiese anfängt, fällt das meistens einfacher als ein bestehendes System zu überführen. Beantwortet das die Frage? Wie sind wir da hingekommen?

00:21:02 Forscher

Ja, also es klingt schon so, dass eigentlich die Risiken größer sind als das, was man dadurch gewinnt, dass man sehr viel Aufwand betreiben muss. Ich habe auch Literatur dazu gelesen, wie man das machen könnte. Dass man dann zum Beispiel Stück für Stück Bausteine herauslöst, die dann erstmal parallel laufen zu dem Monolithen und dass das aber schon ein großer Aufwand ist und da ist es die Frage, ob sich das lohnt?

00:21:45 Hosting-Experte 2

Alles, was ich bisher immer gefunden habe, darüber, wie man Monolithen in Microservices aufteilt, dreht sich um Individualsoftware, die du halt, die man an einer Stelle betreibt und nicht um ein Open Source Produkt, was millionenfach in Millionen verschiedenen Setups installiert ist, denn das ist ja wieder zusätzliche Komplexität.

00:22:24 Forscher

Was schränkt eigentlich die Möglichkeiten für automatische Updates bei TYPO3 ein?

00:22:44 Hosting-Experte 2

Erfahrungsgemäß sind das meistens die Third-Party-Komponenten, die die Leute sich installiert haben. Wir haben da ein bisschen Erfahrung damit, weil wir automatische Updates als Service für unsere Kunden anbieten.

00:23:07 Hosting-Experte 2

Wir trauen uns das aber auch nur innerhalb eines Patch-Levels, also von X.Y.1 auf X.Y.2. Da kann man sich meistens gut drauf verlassen. Weil der Hersteller, beziehungsweise die Open Source Entwickler Community innerhalb der Patch-Levels darauf achtet, die Schnittstellen stabil zu halten.

00:23:38 Hosting-Experte 2

Mit Schnittstellen meine ich jetzt nicht irgendwelche REST-APIs oder sowas, sondern tatsächlich die internen Programmierschnittstellen des Systems, also die Programmierschnittstellen die jetzt eine Extension zum Beispiel nutzen würde, um sich ins System einzuklinken. Die können sich verändern, also gerade zwischen Major-Versionen, zum Beispiel von TYPO3 12 auf 13. Da gibt es dann jedes Mal ein großes Dokument, was mit „Migration Guide“ oder „Breaking Changes“ oder ähnlich betitelt ist. Und dann habe ich immer das Risiko, (.) Wir haben es vorhin schon mal kurz tangiert. (.) Dann habe ich immer das Risiko, dass ich Drittanbieter-Extensions verwende, die mit der neuen Version nicht mehr kompatibel sind oder dass ich selbst eine Extension geschrieben habe oder sogar nur ein Site-Package verwende, was Funktionen nutzt, die nicht mehr kompatibel sind bzw. die in der nächsten Version nicht mehr angeboten werden. Das ist das, was im Moment an automatischen Updates so ein bisschen das Hindernis ist, also zumindest zwischen den Major-Versions.

00:24:54 Hosting-Experte 2

Aber mittlerweile bin ich bei TYPO3 tatsächlich relativ schmerzbefreit. Erfahrungsgemäß läuft das eigentlich sehr, sehr stabil.

00:25:05 Forscher

OK, ja und die Rückwärtskompatibilität, sollte man die noch ausweiten? Oder ist das ausreichend so? Also wenn man länger Rückwärtskompatibilität behalten würde, dann könnte man länger automatische Updates realisieren.

00:25:29 Hosting-Experte 2

Könnte man, das hemmt dann natürlich auch ein bisschen die Weiterentwicklung. An sich, Ich muss jetzt gerade kurz überlegen, dass ich nichts Falsches erzähle, aber ich glaube, TYPO3 hat da eine relativ ausgefeilte Deprecation-Strategie. Wenn sich zum Beispiel eine Schnittstelle auf eine nicht abwärtskompatible Art ändert oder ein Feature rausfliegt oder sowas. Dann glaube ich, gibt es immer eine Major-Version Zeit, wo das Ganze dann als Deprecation-Error gemeldet wird, sodass du das als User mitkriegst, aber die Features funktionieren noch und dann ist es in deiner Verantwortung als Betreiber dieser Seite, in die Protokolle reinzukucken und zu schauen, tauchen irgendwelche Meldungen auf, die ich dann beheben kann. Wenn man darauf achtet, funktioniert das alles. Das ist schwierig zu automatisieren.

00:26:32 Forscher

Verstehe. Wie realisiert ihr die Skalierbarkeit von TYPO3? Bei den meisten Seiten ist es wahrscheinlich kein Thema, weil es einfach kleine Webseiten sind. Wenn es doch mal größer wird, was für Möglichkeiten gibt es da?

00:27:05 Hosting-Experte 2

Es kommt darauf an. Ich spare mir jetzt den großen Ausflug in die Theorie. Also man kann vertikal und horizontal skalieren. Es geht beides mit TYPO3. Wir haben Kunden, die skalieren vertikal, die schlagen das Problem dann einfach mit Hardware. Das funktioniert im Prinzip sehr, sehr einfach.

00:27:35 Hosting-Experte 2

Es gibt auch, es gibt auch Ansätze TYPO3 horizontal zu skalieren. Das System unterstützt das, man muss es allerdings bewusst machen. Es gibt ein paar Einstellungen, die man dafür konfigurieren muss. So richtig einfach out-of-the-box funktioniert das nicht. Weil allein, wenn ich in PHP mit Sitzungsinformationen arbeite, dann fängt PHP zum Beispiel an irgendwelche Sitzungsdateien irgendwohin zu schreiben. Das kann man ihm alles abkonfigurieren. Da muss ich als Betreiber dann halt daran denken.

00:28:17 Hosting-Experte 2

TYPO3 fängt an, irgendwelche temporären Cache-Dateien ins Dateisystem zu werfen. Wenn ich mehrere Server parallel betreibe, dann ist das auch keine gute Idee mehr. Dann muss man sich dafür irgendwas anderes überlegen. Das, wie gesagt, das funktioniert alles, das kann man mit entsprechenden Konfigurationsaufwand lösen und ich weiß auch von Setups, wo das so funktioniert.

00:28:41 Hosting-Experte 2

Es ist dann nicht so, dass ich diesen Vorteil habe, wie wenn ich wirklich Microservices nutze, wo ich sage, OK, dieser eine Service ist jetzt gerade hoch ausgelastet, den skalier ich jetzt fünfmal horizontal hoch oder 500-mal horizontal hoch. Sondern es ist halt immer noch ein Monolith. Das heißt, ich nehme, wenn ich horizontal skalieren will, den Monolithen, starte ihn 2mal, 3mal, 4mal. Und dann skaliere ich das System am Stück horizontal.

00:29:36 Forscher

Du meintest vorhin, dass ihr Entwicklung und Operations zusammenführt bei euch, im Sinne von DevOps. Cloud-Native SaaS bedeutet ja unter anderem, dass das Entwicklung und Betrieb Hand in Hand läuft, also dass der Anbieter der Software, auch den Betrieb gewährleistet. Das geht bei TYPO3 eher schlecht, weil es von einer Community entwickelt und dann von jemand anders betrieben wird. Versucht ihr da etwas zu ändern oder anders zu machen?

00:30:27 Hosting-Experte 2

Nein, wir entwickeln nicht am Kern, also zumindest nicht direkt, wir hängen in der TYPO3 Association mit drin. Also wir finanzieren das Ganze mit, aber wir sind jetzt nicht der Entwickler. Das ist auch eine der Stärken von so einem Open Source System, dass du nicht von einem bestimmten Betreiber abhängig bist.

00:30:58 Hosting-Experte 2

Wenn ich irgendeinen SaaS-Anbieter nutze, dann bin ich halt von diesem Betreiber abhängig. Und bei einem Open-Source-System, was du selbst hosten kannst, das ist jetzt nicht auf TYPO3 eingeschränkt diese Argumentation, das gilt jetzt für WordPress, Joomla!, DRUPAL und was es sonst noch alles so gibt, genauso. Da habe ich eine viel größere Vielfalt von Betreibern auf dem Markt, von denen ich mir einen aussuchen kann. Und natürlich steht es mir auch immer noch frei zu sagen: „Ich bin ein Großunternehmen, ich habe mein eigenes Rechenzentrum, ich habe vielleicht spezielle Compliance-Anforderungen oder sowas, Stichwort Finanzindustrie oder Medizin, die mich dazu zwingen, mich nicht von einem Drittanbieter abhängig zu machen. Dann hast du bei den Open-Source-Systemen auch immer noch die Möglichkeit zu sagen: „Ich hab eigene IT-Infrastruktur, die steht bei mir im Haus, darauf betreibe ich den Kram selbst.

00:32:06 Forscher

Ja, das ist dann der berühmte Cloud-Vendor-Lockin, da ist es dann schwierig, wenn man mal umziehen muss. Oder falls der Betreiber an einem bestimmten Zeitpunkt plötzlich nicht mehr operiert, weil eventuell das Geschäftsmodell doch nicht so profitabel war, dann hat man auch ein Problem.

00:32:26 Hosting-Experte 2

Also es ist halt kein proprietäres System. Das ist auch der Vorteil zu vielen Software-as-a-Service Angeboten, die es gibt. Jetzt kann man darüber streiten. Wenn du jetzt bei Agentur2, als Beispiel, ein Paket mit einem vorinstallierten TYPO3 kaufst und wir kümmern uns um die Updates und alles, da kann man jetzt drüber streiten, also mit ein bisschen Interpretationsspielraum ist das auch Software-as-a-Service.

00:33:01 Hosting-Experte 2

Aber am Ende des Tages betreiben wir auch „nur“, in Anführungsstrichen, ein Open-Source-Produkt für dich als User. Es gibt in dem Sinne keinen Lock-In, als dass du dir theoretisch jederzeit deine Daten, einmal exportieren kannst, läufst damit zu einem anderen Anbieter und dann läuft dein TYPO3-Projekt halt da weiter.

00:33:24 Hosting-Experte 2

Oder andersrum du schiebst dein TYPO3-Projekt von deiner Infrastruktur, zu Agentur2 auf den Server und es läuft da weiter?

00:33:35 Forscher

Würde es Vorteile bringen für den Hosting-Anbieter, wenn man TYPO3 noch mehr in Richtung Multi-Mandantensystem entwickelt, dass man sagt eine TYPO3-Instanz kann dann innerhalb eines Servers viele Kunden gleichzeitig bedienen und die Ressourcen effizienter nutzen?

00:34:20 Hosting-Experte 2

Also harte Multi-Mandantenfähigkeit bietet das System im Moment nicht. Was natürlich geht, ist Multi-Site, also, dass ich mehrere Webseiten gleichzeitig über eine TYPO3-Instanz ausliefere. Für so eine richtige Multimandanten-Fähigkeit ist das System nicht ausgelegt.

00:34:46 Hosting-Experte 2

Und spätestens, wenn ich Third-Party-Extensions oder sowas installiere oder spätestens so bald einer meiner Mandanten zum Beispiel die Anforderung hätte, eine bestimmte Third-Party-Extension nutzen zu wollen, dann kriegen die halt alle, weil das das System nicht darauf ausgelegt ist bestimmte Features nur für einzelne Mandanten anzubieten.

00:35:13 Hosting-Experte 2

Aus Betreibersicht hätte es den charmanten Vorteil, dass man dann als Betreiber die Option hätte, die Packdichte zu erhöhen. Das ist dann das Gegenteil der Skalierung, dass ich mehr Kunden mit einer gegebenen Menge an Ressourcen bedienen kann.

00:35:36 Forscher

Ja, so eine Art nachhaltiges CMS, was die Ressourcen optimal ausnutzt. Was sind eigentlich die wichtigsten Trade-Offs, die man da in Kauf nehmen muss, wenn man die MACH-Architektur einsetzt im Vergleich zu einem Monolithen.

00:36:31 Hosting-Experte 2

Ich hangle mich mal gerade von vorn nach hinten durchs MACH durch.

00:36:40 Hosting-Experte 2

Bei Microservices denke ich jetzt vor allem gerade an das Deployment-Modell. Ich denke zum Beispiel gerade an User, die vielleicht sowieso auf irgendeiner Cloud-Plattform unterwegs sind, weil sie vielleicht noch andere Sachen auf irgendwie gearteten Cloud-Plattformen hosten. Das kann auch ein selbst gehostetes Kubernetes sein oder so etwas. Das zählt ein Stück weit mit da rein. Dann würde ich zum Beispiel davon ausgehen, dass man mit einem MACH-basierten CMS vielleicht schon einen intuitiveren Deployment-Weg hat.

00:37:32 Hosting-Experte 2

Niemand, den ich kenne, macht Microservices ohne Container. Ich weiß in der Theorie gibt es das, aber in der Praxis, ich weiß nicht, was deine Wahrnehmung bisher war, habe ich das nie gesehen, dass jemand Microservices ohne Container macht.

00:37:56 Forscher

Ja es ist auch ein grundlegendes Umsetzungsprinzip. Das habe ich bisher in der Theorie auch nur so gelesen.

00:38:07 Hosting-Experte 2

Beziehungsweise Container sind halt meistens die Lösung, um die zusätzliche Komplexität, die man sich durch Microservices ans Bein bindet, beherrschbar zu machen.

00:38:19 Hosting-Experte 2

Allein, das Stichwort: Du hast deine Software und die Betriebsumgebung in einem Artefakt zusammen verpackt. Du kannst Software und Betriebsumgebung zusammen ausrollen. Du bist technologisch unabhängiger, weil das Einzige, was deine Plattform noch können muss, ist Container zu starten, unabhängig was davon läuft, das sind Lösungen für diese ganze zusätzliche Komplexität, die Microservices dich kosten, deswegen geht das in der

Praxis eigentlich immer Hand in Hand, dass solche Systeme in Containern deployt werden.

00:39:09 Hosting-Experte 2

Bei den klassischen Open Source Content Management Systemen, also TYPO3, WordPress und Co., da gibt es nicht diesen klar definierten Deployment-Weg. Da gibt es im Moment nicht dieses eine autoritative Container-Image, was offiziell, wenn man auf Docker Hub ist, mit diesem kleinen blauen Häkchen, also direkt vom Hersteller, bereitgestellt wird. Das gibt es bei TYPO3 im Moment nicht. Das heißt, wenn ich zum Beispiel die Anforderung habe mein CMS in einer Container-Cloud-Umgebung betreiben zu wollen, das heißt nicht, dass es nicht geht, das heißt nur, dass man sich mehr Gedanken darum machen muss, weil es nicht diesen vordefinierten Auslieferungsweg dafür gibt.

00:40:04 Hosting-Experte 2

Ich kann ihn selbst bauen und es gibt auch glaube ich ein Dutzend verschiedener Docker Images für TYPO3, aber es gibt kein Offizielles. Es gibt auch kein offizielles Helm-Chart für TYPO3, Stichwort, wenn ich jetzt TYPO3 auf Kubernetes deployen will. Das heißt, wenn am Ende der Betrieb in einer Cloud-Umgebung mit Containern mein Ziel ist, dann ist der Weg dahin relativ weit.

00:40:37 Hosting-Experte 2

Das zum Thema Microservices. Noch ein anderer Trade-Off: Wenn ich mir ein Headless CMS kaufe, dann muss ich mir mein Frontend dafür selbst bauen. Weil es ist halt Headless und Prinzip bedingt. Vielleicht brauch ich das in manchen Fällen gar nicht. Da ist auch wieder die Frage: Ist dieser Entwicklungsaufwand, den ich dafür auf mich nehme, den Nutzen wert, den es mir am Ende bringt? Das ist, glaube ich, auch eine Frage, die man für jedes Projekt einzeln für sich entscheiden muss.

00:41:33 Hosting-Experte 2

Wenn ich jetzt das kleine mittelständische Unternehmen habe, die am Ende sowieso eigentlich nur eine Webseite wollen und jetzt nicht irgendwie verschiedene Kanäle mit ihrem Content bespielen müssen. Vielleicht geben das die Anforderungen da gar nicht her. Das ist jetzt keine feste Aussage. Das sind alles Trade-Offs, die man eingeht.

00:42:07 Forscher

OK. Gut. Wir haben schon viele meiner Fragen direkt oder indirekt besprochen, lass uns noch zu ein paar abschließenden Fragen kommen. Was wünscht du dir zukünftig für deine Arbeit mit TYPO3?

00:42:42 Hosting-Experte 2

Gute Frage.

00:42:45 Hosting-Experte 2

Ich würde mir tatsächlich wünschen, dass das ganze System Cloud-nativer wird. So ein bisschen habe ich es, glaube ich, gerade schon angerissen mit dem Container Image beziehungsweise mit dem Deployment. Das wäre es schön, wenn es standardisierte

Wege gäbe. Das macht es auch für uns als Betreiber am Ende einfacher. Wir stecken jetzt im Moment sehr, sehr viel Energie und sehr viel Arbeit darein irgendwie eine runde Betriebsumgebung für TYPO3 an den Start zu kriegen. Und das ist ja alles Arbeit, die wir sparen könnten, wenn direkt vom Hersteller jetzt beziehungsweise direkt aus der Developer-Community eine offizielle Betriebsumgebung vorgegeben werden würde.

00:43:33 Forscher

Zu so einem Container Image, also gerade, wenn man jetzt in Richtung Microservices geht, da gehen dann noch so ein paar nachgelagerte Architekturprinzipien mit einher. Ich denke jetzt zum Beispiel gerade, das ist jetzt auch nichts Neues, aber ich denke gerade an diese ganzen Twelve-Factor-App Prinzipien.

00:44:02 Hosting-Experte 2

Das wären zum Beispiel auch ganz viele Sachen, die den Betrieb irgendwie erleichtern würden. Wir treffen im Moment sehr, sehr viel Vorkehrungen, um CMS-Instanzen für unsere Kunden zu konfigurieren.

00:44:18 Hosting-Experte 2

Das sind meistens so ganz, ganz billige Sachen, zum Beispiel dass ich irgendwo eine relationale Datenbank am Laufen habe und das CMS irgendwie wissen muss, wie es sich zu dieser Datenbank verbinden kann. Und das funktioniert im Moment darüber, dass ich dem CMS irgendwelche Konfigurationsdateien hinwerfen muss und sowas, die alle richtig vorinitialisiert sind. Das funktioniert alles irgendwie, aber so der Cloud-native Weg wäre ja eigentlich, dass ich einfach irgendwo eine Umgebungsvariable definieren kann und das System weiß: Ich muss nur in dieser Umgebungsvariablen kucken und da stehen meine Verbindungsdaten drin.

00:44:55 Hosting-Experte 2

Oder was gibt es noch in der Twelve-Factor-App? Wie gehe ich mit Logdaten um zum Beispiel. Was steckt da noch drin? So Fast-Startup/Fast-Shutdown. Wobei das geht über PHP auch schon relativ einfach, aber da ist, glaube ich auch, *long story short,* da ist glaube ich noch ein bisschen Potenzial drin.

00:45:26 Forscher

Okay spannend, dass es auch in die Richtung gehen soll, was die Twelve Factors vorgeben, denn die waren die Grundlage, dass sich dann auch der Begriff „Cloud-native“ herausgebildet hat. Denkst du denn, dass die MACH-basierten CMS die traditionellen in Zukunft verdrängen werden?

00:46:01 Hosting-Experte 2

Verdrängen glaube ich nicht. Aber ich glaube, das wird sich irgendwie einpendeln, weil nach wie vor beide Arten von Produkten ihre Daseinsberechtigung haben.