

Über diese Vorlage

Diese L^AT_EX-Vorlage wurde von Stefan Macke¹ als Grundlage für die Projektdokumentationen der Auszubildenden zum Fachinformatiker mit Fachrichtung Anwendungsentwicklung bei der ALTE OLDENBURGER Krankenversicherung entwickelt. Nichtsdestotrotz dürfte sie ebenso für die anderen IT-Berufe² geeignet sein, da diese anhand der gleichen Verordnung bewertet werden.

Diese Vorlage enthält bereits eine Vorstrukturierung der möglichen Inhalte einer tatsächlichen Projektdokumentation, die auf Basis der Erfahrungen im Rahmen der Prüfertätigkeit des Autors erstellt und unter Zuhilfenahme von ROHRER UND SEDLACEK [2011] abgerundet wurden.

Sämtliche verwendeten Abbildungen, Tabellen und Listings stammen von GRASHORN [2010].

Download-Link für diese Vorlage: <http://fiae.link/LaTeXVorlageFIAE>

Auch verfügbar auf GitHub: <https://github.com/StefanMacke/latex-vorlage-fiae>

Lizenz



Dieses Werk steht unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz.³



Namensnennung Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.⁴

Weitergabe unter gleichen Bedingungen Wenn Sie das lizenzierte Werk bzw. den lizenzierten Inhalt bearbeiten oder in anderer Weise erkennbar als Grundlage für eigenes Schaffen verwenden, dürfen Sie die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch oder vergleichbar sind.

¹Blog des Autors: <http://fachinformatiker-anwendungsentwicklung.net>, Twitter: @StefanMacke

²z. B. IT-Kaufleute, Fachinformatiker mit Fachrichtung Systemintegration usw.

³<http://creativecommons.org/licenses/by-sa/4.0/>

⁴Die Namensnennung im L^AT_EX-Quelltext mit Link auf <http://fiae.link/LaTeXVorlageFIAE> reicht hierfür aus.

Inhalt der Projektdokumentation

Grundsätzlich definiert die [REGIERUNG DER BUNDESREPUBLIK DEUTSCHLAND](#) [1997, S. 1746]⁵ das Ziel der Projektdokumentation wie folgt:

"Durch die Projektarbeit und deren Dokumentation soll der Prüfling belegen, daß er Arbeitsabläufe und Teilaufgaben zielorientiert unter Beachtung wirtschaftlicher, technischer, organisatorischer und zeitlicher Vorgaben selbständig planen und kundengerecht umsetzen sowie Dokumentationen kundengerecht anfertigen, zusammenstellen und modifizieren kann."

Und das [BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG](#) [2000, S. 36] ergänzt:

"Die Ausführung der Projektarbeit wird mit praxisbezogenen Unterlagen dokumentiert. Der Prüfungsausschuss bewertet die Projektarbeit anhand der Dokumentation. Dabei wird nicht das Ergebnis – z. B. ein lauffähiges Programm – herangezogen, sondern der Arbeitsprozess. Die Dokumentation ist keine wissenschaftliche Abhandlung, sondern eine handlungsorientierte Darstellung des Projektablaufs mit praxisbezogenen, d.h. betriebüblichen Unterlagen. Sie soll einen Umfang von maximal 10 bis 15 DIN A 4-Seiten nicht überschreiten. Soweit erforderlich können in einem Anhang z. B. den Zusammenhang erläuternde Darstellungen beigelegt werden."

Außerdem werden dort die grundlegenden Inhalte der Projektdokumentation aufgelistet:

- Name und Ausbildungsberuf des Prüfungsteilnehmers
- Angabe des Ausbildungsbetriebes
- Thema der Projektarbeit
- Falls erforderlich, Beschreibung/Konkretisierung des Auftrages
- Umfassende Beschreibung der Prozessschritte und der erzielten Ergebnisse
- Gegebenenfalls Veränderungen zum Projektantrag mit Begründung
- Wenn für das Projekt erforderlich, ein Anhang mit praxisbezogenen Unterlagen und Dokumenten. Dieser Anhang sollte nicht aufgebläht werden. Die angehängten Dokumente und Unterlagen sind auf das absolute Minimum zu beschränken.

In den folgenden Kapiteln werden diese geforderten Inhalte und sinnvolle Ergänzungen nun meist stichwortartig und ggfs. mit Beispielen beschrieben. Nicht alle Kapitel müssen in jeder Dokumentation vorhanden sein. Handelt es sich bspw. um ein in sich geschlossenes Projekt, kann das Kapitel 1.5: Projektbegrenzung entfallen; arbeitet die Anwendung nur mit XML-Dateien, kann und muss keine Datenbank beschrieben werden usw.

⁵Dieses Dokument sowie alle weiteren hier genannten können unter <http://fiae.link/LaTeXVorlageFIAEQuellen> heruntergeladen werden.

Formale Vorgaben

Die formalen Vorgaben zum Umfang und zur Gestaltung der Projektdokumentation können je nach IHK recht unterschiedlich sein. Normalerweise sollte die zuständige IHK einen Leitfaden bereitstellen, in dem alle Formalien nachgelesen werden können, wie z. B. bei der [IHK OLDENBURG \[2006\]](#).

Als Richtwert verwende ich 15 Seiten für den reinen Inhalt. Also in dieser Vorlage alle Seiten, die arabisch nummeriert sind (ohne das Literaturverzeichnis und die eidesstattliche Erklärung). Große Abbildungen, Quelltexte, Tabellen usw. gehören in den Anhang, der 25 Seiten nicht überschreiten sollte.

Typographische Konventionen, Seitenränder usw. können in der Datei `Seitenstil.tex` beliebig angepasst werden.

Bewertungskriterien

Die Bewertungskriterien für die Benotung der Projektdokumentation sind recht einheitlich und können leicht in Erfahrung gebracht werden, z. B. bei der [IHK DARMSTADT \[2011\]](#). Grundsätzlich sollte die Projektdokumentation nach der Fertigstellung noch einmal im Hinblick auf diese Kriterien durchgeschaut werden.

Prüfungsteil A

Prüfling (private Anschrift):	Ausbildungsbetrieb:
-------------------------------	---------------------

Bestätigung über durchgeführte Projektarbeit

diese Bestätigung ist mit der Projektdokumentation einzureichen

Ausbildungsberuf (bitte unbedingt angeben):

Projektbezeichnung:

Projektbeginn: _____	Projektfertigstellung: _____	Zeitaufwand in Std.: _____
----------------------	------------------------------	----------------------------

Bestätigung der Ausbildungsfirma:

Wir bestätigen, dass der/die Auszubildende das oben bezeichnete Projekt einschließlich der Dokumentation im Zeitraum

vom: _____ bis: _____ selbständig ausgeführt hat.

Projektverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

Ausbildungsverantwortliche(r) in der Firma:

Vorname	Name	Telefon	Unterschrift
---------	------	---------	--------------

Eidesstattliche Erklärung:

Ich versichere, dass ich das Projekt und die dazugehörige Dokumentation selbständig erstellt habe.

Ort und Datum: _____ Unterschrift des Prüflings: _____



Abschlussprüfung Winter 2022/23

Fachinformatiker für Anwendungsentwicklung
Dokumentation zur betrieblichen Projektarbeit

Nuxt3 Typo3 Skeleton

Webbasiertes Headless Content-Management-System

Abgabetermin: Dortmund, den 23.04.2015

Prüfungsbewerber:

Lukas Röding
Hohe Straße 8
44139 Dortmund



Ausbildungsbetrieb:

bits & likes GmbH
Rheinische Str. 171
44147 Dortmund

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Contents

List of Figures	III
List of Tables	IV
Listings	V
Abkürzungsverzeichnis	VI
1 Einleitung	1
1.1 Projektumfeld	1
1.2 Projektziel	1
1.3 Projektbegründung	1
1.4 Projektschnittstellen	2
1.5 Projektabgrenzung	2
2 Projektplanung	2
2.1 Projektphasen	2
2.2 Abweichungen vom Projektantrag	6
2.3 Ressourcenplanung	6
2.4 Entwicklungsprozess	6
3 Analysephase	6
3.1 Ist-Analyse	6
3.2 Wirtschaftlichkeitsanalyse	7
3.2.1 Projektkosten	7
3.2.2 Amortisationsdauer	7
3.3 Nutzwertanalyse	8
3.4 Anwendungsfälle	9
3.5 Qualitätsanforderungen	9
3.6 Nuxt 2 vs Nuxt 3	9
3.6.1 Warum wurde sich trotzdem für Nuxt 3 entschieden?	10
4 Entwurfsphase	10
4.1 Zielplattform	10
4.2 Architekturdesign	12
4.3 Entwurf der Benutzeroberfläche	13
4.4 Datenmodell	14
4.5 Geschäftslogik	15
4.6 Maßnahmen zur Qualitätssicherung	15
4.7 Pflichtenheft/Datenverarbeitungskonzept	16

5	Implementierungsphase	16
5.1	Docker-Setup	16
5.2	Ausspielen der Typo3 Daten	16
5.3	Skeleton-Extension / Layout	18
5.4	Implementierung der Benutzeroberfläche	20
5.5	Implementierung der Geschäftslogik	24
6	Abnahmephase	24
7	Einführungsphase	24
8	Dokumentation	25
9	Fazit	25
9.1	Soll-/Ist-Vergleich	25
9.2	Lessons Learned	25
9.3	Ausblick	26
	Literaturverzeichnis	27
	Eidesstattliche Erklärung	28
A	Anhang	i
A.1	Detaillierte Zeitplanung	i
A.2	tt_content.php	ii
A.3	ext_localconf.php	iii
A.4	[...slug].vue	iv
A.5	navigation.ts	v
A.6	SlideMenu.vue	vi
A.7	Datenbankmodell	vii
A.8	Oberflächenentwürfe	ix
A.9	Screenshots der Anwendung	xi
A.10	Entwicklerdokumentation	xii
A.11	Testfall und sein Aufruf auf der Konsole	xiv
A.12	Klasse: ComparedNaturalModuleInformation	xv
A.13	Klassendiagramm	xviii
A.14	Benutzerdokumentation	xix

List of Figures

1	Vereinfachtes ER-Modell	14
2	Prozess des Einlesens eines Moduls	15
3	Datenbankmodell	viii
4	Liste der Module mit Filtermöglichkeiten	ix
5	Anzeige der Übersichtsseite einzelner Module	x
6	Anzeige und Filterung der Module nach Tags	x
7	Ausspielen der Komponenten im Frontend	xi
8	Pflegen der Komponenten im Backend	xi
9	Aufruf des Testfalls auf der Konsole	xv
10	Klassendiagramm	xviii

List of Tables

1	grobe Zeitplanung	3
2	detaillierte Zeitplanung	5
3	Kostenaufstellung	7
4	Entscheidungsmatrix Nuxt3	12
5	Entscheidungsmatrix DaisyUI	12
6	Entscheidungsmatrix Typo3	12
7	Entscheidungsmatrix TS	13
8	Soll-/Ist-Vergleich	26

Listings

1	Testfall in PHP	xiv
2	Klasse: ComparedNaturalModuleInformation	xv

Abkürzungsverzeichnis

BAL	bits & likes GmbH
CMS	Content-Management-System
DSGVO	Datenschutz-Grundverordnung
SEO	Search-Engine-Optimization
TS	TypeScript
JS	JavaScript
API	Application Programming Interface
EPK	Ereignisgesteuerte Prozesskette
ERM	Entity-Relationship-Modell
HTML	Hypertext Markup Language
UML	Unified Modeling Language
XML	Extensible Markup Language

1 Einleitung

1.1 Projektumfeld

Die bits & likes GmbH(BAL) ist eine Fullservice-Digitalagentur aus Dortmund. Sie bietet sowohl Onlinemarketing als auch Webseitenentwicklung an. Aktuell beschäftigt sie ca. 50 Mitarbeiter, die Tendenz ist steigend.

Die Idee für das Projekt kam bei der Entwicklung einer Website für die Murtfeldt Kunststoffe GmbH & Co. KG(Murtfeldt). Historisch nutzte Murtfeldt Typo3 als Content-Management-System(CMS). BAL entwickelte mit Nuxt 2 eine Headless Lösung für diese neue Website. Dieser Technologiestack soll nun mit der neuesten Version von Typo3(Version 11.5 zum Zeitpunkt des Projekts) als auch der neusten Version von Nuxt(Nuxt 3)⁶ neu entwickelt werden. Damit zukünftige Murtfeldt Projekte als auch andere Projekte(z.B. BAL Firmenwebsite, andere Kundenprojekte, ..) eine Vorlage(Skeleton) haben, mit welcher sie umgesetzt werden können.

1.2 Projektziel

Das Ziel des Projektes ist es eine Typo3 Nuxt 3 Skeleton zu erstellen, auf dessen Basis neue Projekte umgesetzt werden können. Dazu müssen folgende Punkte erreicht werden:

- Content aus dem Typo3 System im JSON-Format ausspielen.
- Potentielle Erweiterungen an den JSON-Daten um alle technischen Voraussetzungen zu erfüllen
- Content im Nuxt 3 Frontend auslesen, verarbeiten und ausspielen. Dies beinhaltet sowohl den Content, als auch die Navigation

1.3 Projektbegründung

Bis jetzt klonete BAL immer alte Websites auf dem Typo3 & Nuxt2 Technologiestack um neue Websites zu entwickeln. Dies bedeutete, dass viel Content, als auch alter Code erstmal händisch aus dem Projekt gelöscht werden musste. Zusätzlich musste die Datenbank bereinigt werden, damit die neue Website, keine Altlasten / kritische Daten aus dem vorherigen Projekt mit übernimmt. Die Alternative war, ein komplett neues Typo3 und Nuxt2 Projekt aufzusetzen und jedes mal viel Code / Template-Anpassungen neu schreiben. Dieser unnötige Aufwand soll mit diesem Projekt umgangen werden. Dadurch entstehen sowohl Kostenersparnisse durch kürzere Entwicklungszeiten, als auch weniger potentielle Fehler/Komplikationen mit alten Code/Daten von anderen Projekten.

Ein Vorteil von Typo3, gegenüber anderen CMS ist, dass Typo3 sich stark auf den deutschen Markt fokussiert hat. Das bedeutet, dass es für Typo3 schnelle Anpassungen an die deutsche Gesetzgebung

⁶Vgl. [NUXT 3 \[2022\]](#).

gibt. Dies ist beispielsweise relevant für die DSGVO. Dazu haben Kunden häufiger bereits Expertise in Typo3, was Zeit und Kosten in der Einarbeitung spart.

Nuxt hat als Vorteil, dass es serverseitig gerendert wird. Dies führt zu schnelleren Ladegeschwindigkeiten der Seite. Dies verbessert die Search-Engine-Optimization-Performance(SEO-Performance). Zusätzlich hat BAL bereits viel expertise in Vue und alten Nuxt Versionen. Weswegen die Einarbeitung in die Technologie einfacher ist.

1.4 Projektschnittstellen

Die finale Nuxt Anwendung interagiert nur mit der Typo3 Schnittstelle. Projekte, die mit dem Skeleton entwickelt werden, können aber weitere Schnittstellen einbinden. Im Fall von BAL würde dies beispielsweise oft eine Schnittstelle zu einem Shopware-System(E-Commerce Plattform) sein. Zukünftige Iterationen des Skeletons könnten diese auch als Standard beinhalten.

Die Mittel für das Projekt werden von BAL zur Verfügung gestellt. Da die Entwickler von BAL auch die zukünftigen Nutzer des Projekts sind. Dementsprechend muss das Projekt auch den Entwicklern später präsentiert und eine Dokumentation für diese geschrieben werden. Projekte, welche aus dem Skeleton entwickelt werden, würden aber in der Zukunft Kunden präsentiert werden und Teile des Skeletons enthalten.

1.5 Projektabgrenzung

Das Projekte enthält folgende features nicht:

- Ein umgesetztes Frontend-Design
- Alle gängigen Typo3-Komponenten im Frontend eingebaut(nur die Basics)
- Anbindungen zu anderen technischen Schnittstellen außer Typo3

2 Projektplanung

2.1 Projektphasen

- Verfeinerung der Zeitplanung, die bereits im Projektantrag vorgestellt wurde.

Die Bearbeitung der Projektarbeit fand vom 23.09.2022 - 07.10.2022 statt. Die Tagesarbeitszeit war standardmäßig 8 Stunden. Gearbeitet wurde an allen Tagen, bis auf den 28.09.2022. Der Prüfungsbewerber hatte an dem Tag Berufsschule.

Die im Projektantrag geplante Zeitplanung sah wie folgt aus:

Tabelle 1 zeigt die grobe Zeitplanung des Projektes.

Projektphase	Geplante Zeit
Analysephase	10 h
Entwurfsphase	5 h
Implementierungsphase	55 h
Einführungsphase	5 h
Erstellen der Dokumentation	5 h
Gesamt	80 h

Table 1: grobe Zeitplanung

Es wurde ebenfalls eine detaillierte Zeitplanung erstellt. In ihr werden die jeweiligen Projektphasen in weitere kleinere Teilabschnitte unterteilt. Dies dient sowohl dem besseren Verständnis des Projektes und liefert ebenfalls einen Handlungsfaden um das Projekt abzuarbeiten. Die Teilabschnitte der Projektphasen haben ebenfalls eine geplante Zeit, welche zusammen addiert die Zeit der jeweiligen Projektphase ergibt.

Tabelle 2 zeigt die detaillierte Zeitplanung des Projektes.

Analysephase	10 h
1. Analyse des Ist-Zustands	2 h
2. Wirtschaftlichkeitsanalyse	1 h
3. Nuxt2 vs Nuxt3, welche Nuxt3 features sollen implementiert werden?	7 h
Absprache mit senior Developer	
Entwurfsphase	19 h
1. Prozessentwurf	2 h
2. Datenbankentwurf	3 h
2.1. ER-Modell erstellen	2 h
2.2. Konkretes Tabellenmodell erstellen	1 h
3. Erstellen von Datenverarbeitungskonzepten	4 h
3.1. Verarbeitung der CSV-Daten	1 h
3.2. Verarbeitung der SVN-Daten	1 h
3.3. Verarbeitung der Sourcen der Programme	2 h
4. Benutzeroberflächen entwerfen und abstimmen	2 h
5. Erstellen eines UML-Komponentendiagramms der Anwendung	4 h
6. Erstellen des Pflichtenhefts	4 h
Implementierungsphase	29 h
1. Anlegen der Datenbank	1 h
2. Umsetzung der HTML-Oberflächen und Stylesheets	4 h
3. Programmierung der PHP-Module für die Funktionen	23 h
3.1. Import der Modulinformationen aus CSV-Dateien	2 h
3.2. Parsen der Modulquelltexte	3 h
3.3. Import der SVN-Daten	2 h
3.4. Vergleichen zweier Umgebungen	4 h
3.5. Abrufen der von einem zu wählenden Benutzer geänderten Module	3 h
3.6. Erstellen einer Liste der Module unter unterschiedlichen Aspekten	5 h
3.7. Anzeigen einer Liste mit den Modulen und geparsten Metadaten	3 h
3.8. Erstellen einer Übersichtsseite für ein einzelnes Modul	1 h
4. Nächtlichen Batchjob einrichten	1 h
Abnahmetest der Fachabteilung	1 h
1. Abnahmetest der Fachabteilung	1 h
Einführungsphase	1 h
1. Einführung/Benutzerschulung	1 h
Erstellen der Dokumentation	9 h
1. Erstellen der Benutzerdokumentation	2 h
2. Erstellen der Projektdokumentation	6 h
3. Programmdokumentation	1 h
3.1. Generierung durch PHPdoc	1 h
Pufferzeit	2 h
1. Puffer	2 h
Gesamt	70 h

Table 2: detaillierte Zeitplanung

2.2 Abweichungen vom Projektantrag

- Sollte es Abweichungen zum Projektantrag geben (z. B. Zeitplanung, Inhalt des Projekts, neue Anforderungen), müssen diese explizit aufgeführt und begründet werden.

2.3 Ressourcenplanung

Für die Umsetzung des Projektes wurden folgende Ressourcen benötigt:

- Computer
- Internetzugang
- Büroraum
- Quellcode von Typo3 und Nuxt3 (kostenlos übers Internet verfügbar)
- Personelle Ressourcen: senior Developer für Rückfragen

2.4 Entwicklungsprozess

Bei der Entwicklung des Projektes wurde ein Wasserfall Entwicklungsprozess benutzt. Dies bedeutet dass die Projektphasen linear und ohne Rückschritte nacheinander abgearbeitet wurden. Die vordefinierten Projektphasen ergaben sich aus dem Projektantrag.

3 Analysephase

3.1 Ist-Analyse

- Wie ist die bisherige Situation (z. B. bestehende Programme, Wünsche der Mitarbeiter)?
- Was gilt es zu erstellen/verbessern?

Zum Zeitpunkt des Projektstarts hatte BAL mehrere Projekte mit Typo3 in Verbindung mit Nuxt2 erstellt. Das genutzte Typo3 System war nicht auf der aktuellen Version. Dazu bestand noch kein vorhandenes Skeleton. Diese Situation gilt es zu verbessern. Dies bedeutet, sowohl die neuste Version von Nuxt, als auch von Typo3 zu nutzen, um ein Skeleton zu erstellen.

3.2 Wirtschaftlichkeitsanalyse

BAL schätzt, dass durch das erstellte Skeleton ca. 20 Stunden Arbeit, pro umgesetztes Projekt, eingespart werden können. Bei durchschnittlichen Entwicklerkosten von 45€ pro Stunde (30€ Stundenlohn + 15€ Ressourcen⁷) ergibt dies eine Ersparnis von $20 \cdot 45€ = 900€$ pro Projekt. Damit sich das Skeleton für die Firma lohnt, müssen die Projektkosten kleiner als die Einsparungen. Dafür müssen die Projektkosten berechnet werden und überprüft werden, wie oft das Skeleton wahrscheinlich genutzt wird.

3.2.1 Projektkosten

Die Kosten für die Durchführung des Projekts setzen sich sowohl aus Personal-, als auch aus Ressourcenkosten zusammen. Ein Auszubildender der bits & likes GmbH verdient im dritten Lehrjahr 1450 €.

$$8 \text{ h/Tag} \cdot 220 \text{ Tage/Jahr} = 1760 \text{ h/Jahr} \quad (1)$$

$$1450 \text{ €/Monat} \cdot 12 \text{ Monate/Jahr} = 17400 \text{ €/Jahr} \quad (2)$$

$$\frac{17400 \text{ €/Jahr}}{1760 \text{ h/Jahr}} \approx 9.89 \text{ €/h} \quad (3)$$

Es ergibt sich also ein Stundenlohn von 9.89 €. Die Durchführungszeit des Projekts beträgt 80 Stunden. Für die Nutzung von Ressourcen⁸ wird ein pauschaler Stundensatz von 15 € angenommen. Für die anderen Mitarbeiter wird pauschal ein Stundenlohn von 30 € angenommen. Eine Aufstellung der Kosten befindet sich in Tabelle 3 und sie betragen insgesamt 2621.2 €.

Vorgang	Zeit	Kosten pro Stunde	Kosten
Entwicklungskosten	80 h	$9.89 \text{ €} + 15 \text{ €} = 24,89 \text{ €}$	1991.2 €
Fachgespräch	3 h	$30 \text{ €} + 15 \text{ €} = 45 \text{ €}$	135 €
Abnahmetest	1 h	$30 \text{ €} + 15 \text{ €} = 45 \text{ €}$	45 €
Anwenderschulung	10 h	$30 \text{ €} + 15 \text{ €} = 45 \text{ €}$	450 €
			2621.2 €

Table 3: Kostenaufstellung

3.2.2 Amortisationsdauer

Bei Einsparungen von ca. 900 € pro zukünftigen Projekt und Projektkosten von 2621.2 € ergibt sich ein Amortisationsdauer von $2621.2 \text{ €} / 900 \text{ €} = 2,921$ Projekten. Aufgerundet sind dies 3 Projekte.

⁷Räumlichkeiten, Arbeitsplatzrechner etc.

⁸Räumlichkeiten, Arbeitsplatzrechner etc.

BAL muss also 3 Projekte mit dem Skeleton entwickeln, damit sich das Projekt/Skeleton gelohnt hat. Aktuell sind bei BAL drei Projekte in Planung, welche das Skeleton nutzen werden. Dazu gehört die eigene Firmenwebsite, als auch zwei Kundenprojekte. Diese werden im Laufe des nächsten Jahres abgearbeitet. Dadurch wird sich das Projekt bis Ende 2023 amortisiert haben. Ein genaueres Datum kann zum aktuellen Zeitpunkt nicht ermittelt werden, da die zukünftigen Projekte noch keinen festen Zeitplan haben. Wahrscheinlich werden im Laufe der nächsten Jahren aber auch weitere Projekte mit dem Skeleton umgesetzt. Das Projekt hat sich also finanziell für BAL gelohnt.

3.3 Nutzwertanalyse

Da es noch kein Skeleton gab, kann ein Vorher-/Nacher-Vergleich zwischen einem alten Skeleton und dem neuen nicht geschehen. Es kann jedoch verglichen werden, ob es vom Nutzen ist ein Skeleton zu haben oder nicht. Dafür gibt es zwei Alternativen.

- Jedes Projekt schreibt den gesamten Code neu. Es wird mit einem leeren Typo3 und Nuxt 3 gestartet
- Es werden wie vorher alte Projekte geklont und auf deren Basis neue Projekte umgesetzt

Dafür wurde folgende Nutzwertanalyse erstellt:

Eigenschaft	Gewichtung	Skeleton	kein Code	alte Projekte klonen
Einarbeitungszeit	3	5	2	3
Fehleranfälligkeit	3	5	4	2
initiale Kosten	2	2	5	5
Gesamt:	8	34	22	25
Nutzwert:		4,25	2,75	3,125

Das Skeleton hat die geringste Einarbeitungszeit(dadurch den höchsten Wert, da hohe Einarbeitungszeit schlecht ist). Dies liegt daran, dass beim Skeleton eine kleine Codebasis besteht, welche die Verbindung zum Typo3 schon übernimmt. Ein Frontend Entwickler kann sich beispielsweise an das Projekt setzen und bekommt direkt seine Daten aus dem Typo3. Gleichzeitig besteht aber kein anderer Code, welcher zu Komplikationen/Verwirrung führen könnte. Geklonte Projekte würden zwar auch schon ihre Daten aus dem Typo3 bekommen, aber neue Entwickler müssten sich erstmal in die große bestehende Codebase einarbeiten. Wenn kein vorheriger Code genutzt wird, ist die Codebase zwar am geringsten(leeres Typo3 und Nuxt 3). Dies führt aber dazu, dass sich ein neuer Entwickler erstmal damit beschäftigen muss, wie er an die Daten des Typo3 kommt und diese verarbeitet. Das Skeleton ist auch am wenigsten fehleranfällig. Bei geklonten Projekten können alte Codeabschnitte mit neuen auf unerwartete Art und Weisen miteinander interagieren. Neuer Code kann zwar theoretisch fehlerfrei geschrieben werden, aber ein Skeleton, welches häufiger genutzt wird, wird auch häufiger getestet. Der größte Nachteil eines Skeletons gegenüber den Alternativen sind die Kosten. Die beiden anderen Alternativen haben initial keine/sehr geringe Kosten. Da sich die Kosten des Skeletons

aber in einem geringen Rahmen halten (Kapitel 3.2: Wirtschaftlichkeitsanalyse) und die Kosten durch zukünftige Vorteile ausgeglichen werden, schlägt dies nicht so sehr ins Gewicht. Es wurden zusätzlich unterschiedliche Technologien verglichen um so ein Skeleton zu erstellen. Genauere Analysen zum Nutzwert der Technologien finden sie in Kapitel 4.2: Architekturdesign.

3.4 Anwendungsfälle

Das Skeleton kann theoretisch für sämtliche Websites genutzt werden, welche ein CMS nutzen. Natürlich muss analysiert werden, ob dies auch sinnvoll wäre. Typo3 ist ein Enterprise CMS, was bedeutet, dass es sich auf große Firmen spezialisiert hat. Für kleinere Projekte wäre es vielleicht sinnvoller ein anderes CMS zu nutzen. Nicht jedes Projekt braucht auch eine custom Lösung im Frontend. Für einige Projekte wäre es vielleicht sinnvoller vorhandene Software zu nutzen. Dies kann ebenfalls ein Typo3 mit vorgegebenen Theme sein, oder ein komplett anderes CMS.

3.5 Qualitätsanforderungen

Da verschiedene Projekte mit dem Skeleton umgesetzt werden, ist es sehr wichtig, dass das Skeleton flexibel ist. Da unterschiedlichen Projekte, zumindest teilweise, von verschiedenen Entwicklern umgesetzt werden, muss der Code leicht zu verstehen sein. Der Code muss also leicht erweiterbar und leserlich sein. Natürlich darf die Ladezeit der Website auch nicht sonderlich groß sein, damit die SEO-Performance nicht zu schlecht wird. Dies ist wichtig, damit mit dem Skeleton entwickelte Projekte über google / anderen Suchmaschinen gefunden werden können. Zusätzlich erhöht eine schnelle Ladezeit die Benutzerfreundlichkeit.

3.6 Nuxt 2 vs Nuxt 3

Es wurde sich entschieden für das Projekt Nuxt 3 zu nutzen. Bis jetzt hatte BAL Nuxt 2 und Vue 3 genutzt um sich mit Headless-Typo3 Systemen zu verbinden. Der Grund für die Entscheidung liegt unter anderem der Erweiterung der Wissensbasis der Firma. Eine genauere Analyse, warum sich für Nuxt im allgemeinen und nicht für andere Frontend Webframeworks entschieden wurde, finden sie in Kapitel 4.2: Architekturdesign. Zum Zeitpunkt der Projektarbeit war Nuxt 3 noch in der Beta. Am 16.11.2022 wurde die erste stable Version von Nuxt veröffentlicht. Dies ist natürlich erstmal ein Nachteil, da Betaversionen von Programmen häufiger von Bugs geplagt sind. Zusätzlich werden in Betaversionen häufiger stärkere Veränderungen durchgeführt, welche zu Problemen bei bestehenden Programmcode führen könnten. Nuxt 2 ist seit Jahren offiziell veröffentlicht und BAL hat bereits mehrere Projekte erfolgreich mit Nuxt 2 abgeschlossen. Zusätzlich hat BAL bereits Nuxt 2 erfolgreich mit einem Headless-Typo3 verbunden. Es wäre also sehr viel einfacher das Skeleton mit Nuxt 2 umzusetzen. Es gibt also folgende gute Gründe Nuxt 2 anstelle von Nuxt 3 zu nutzen:

- Programmcode aus vorherigen Projekten kann genutzt werden. -> Zeitersparnis

- Wissensbasis in Nuxt 2 vorhanden. -> potentiell weniger Bugs / schnelleres Entwickeln
- Programm kann schneller von Arbeitskollegen verstanden werden -> potentiell schnellere Entwicklungszeit / Einarbeitungszeit bei ersten neuen Projekten mit Skeleton

3.6.1 Warum wurde sich trotzdem für Nuxt 3 entschieden?

Nuxt 3 biete gegenüber von Nuxt 2 einige Vorteile. Der erste Vorteil ist, dass Typescript unterstützt wird. Zwar konnte Typescript auch in Nuxt 2 und Vue 2 Projekten genutzt werden, es war aber immer mit Komplikationen verbunden. Nuxt 3(und Vue 3) unterstützen dies von Anfang an, was zu einer erleichterten Nutzung von Typescript führt. Selbst wenn kein Typescript genutzt wird, beinhaltet Nuxt 3 einige Vorteile gegenüber von Nuxt 2. Beispielsweise hat Nuxt 3 eine komplett neue Server-Engine bekommen. Diese hat den Namen Nitro. Nitro hat dank dynamischen Code-Splitting eine bessere Leistung als Nuxt 2. Es gibt zusätzlich viele neue Features, wie die neue Composition API, automatische Imports, verbesserte Devtools, etc. Aus diesen Gründen scheint Nuxt 3 langfristig die bessere Entscheidung für die Entwicklung neuer Projekte zu sein, als Nuxt 2. Die Projektarbeit wäre trotzdem wahrscheinlich sehr viel schneller und einfacher mit Nuxt 2 umzusetzen gewesen. Aus den oben genannten Gründen ist ein Umstieg auf Nuxt 3 aber absehbar. Deswegen entschied sich BAL das Projekt in Nuxt 3 umzusetzen. Die Wissensbasis von Nuxt 3 soll mithilfe des Projektes erweitert und der Grundstein für zukünftige Nuxt 3 Projekte gelegt werden.

4 Entwurfsphase

4.1 Zielplattform

Als Programmiersprache wurde TypeScript(TS) ausgewählt⁹. TS ist ein Superset von JavaScript(JS). Dies bedeutet, dass jeder JS-Code in TS funktioniert, aber nicht jeder TS-Code in JS funktioniert. TS erweitert JS um Typen. Dies bedeutet, dass Variablen feste Datentypen(string, number, etc.) haben können. Dies hat den Vorteil, dass TS beim kompilieren Fehler schmeißt, falls eine Variable einen Wert bekommt, welche sie nicht haben sollte. TS wird zu JS kompiliert, da Browser aktuell nur JS und WebAssembly unterstützen. Dadurch werden vorallem große Projekte weniger fehleranfällig. Das Skeleton selber ist nicht so ein großes Projekt, aber es werden große Projekte damit entwickelt werden. Deswegen ist es sinnvoll bereits mit TS zu starten.

Als Framework für TS wurde sich für Nuxt3 entschieden. Nuxt3 ist ein serverseitig gerendertes Javascript-Framework mit TypeScript Support. Nuxt3 bietet viele Features, welche Webentwicklung einfacher machen. Dazu zählt beispielsweise reactivity. Dies bedeutet, dass das Framework automatisch Variablen in der Darstellung anpassen kann, wenn diese ihren Wert ändern. Viele andere Javascript-Frameworks werden clientseitig gerendert. Das bedeutet, dass der Nutzer das HTML erst rendern

⁹Vgl. [MICROSOFT \[2022\]](#).

muss. Nuxt3 macht dies bereits auf dem Server, wodurch die Seite beim Nutzer schneller angezeigt wird. Dies wirkt sich auch vorteilhaft auf SEO aus.

Als UI-Framework wurde DaisyUI genutzt. Für die Umsetzung des Skeletons war die Nutzung eines UI-Frameworks nicht wirklich nötig. Insgesamt musste während der Entwicklung des Skeleton nicht viel CSS / Styling genutzt werden. Das Framework liefert vorgestylte Komponenten und CSS-Klassen, welche die zukünftige Entwicklung beschleunigen sollen. DaisyUI basiert auf Tailwind, weswegen alle Tailwind Klassen auch in dem Projekt genutzt werden können.

Um Typo3 besser testen zu können, wird Typo3 in einem Docker-Container ausgeführt. Docker ist eine Open-Source-Tool, welches es erlaubt Software in abgekapselten Umgebungen/Containern laufen zu lassen. Ein Docker-Container ist Softwarepaket, was alle nötigen Werkzeuge enthält um ein Programm laufen zu lassen. Docker ermöglicht es verschiedenste Programme (andere Programmiersprachen, Systemanforderungen, Libraries, etc.) in unterschiedlichen Containern laufen zu lassen. Beispielsweise kann auf einem Rechner/Server in einem Docker-Container Typo3 laufen, während in einem anderen Nuxt läuft. Docker-Container können schnell hoch und runter gefahren werden, was testen von neuem Code, oder neustarten von Systemen sehr schnell macht.

Sonst wurden die klassischen Webtechnologien HTML und CSS genutzt.

Genauere Vergleiche zu anderen Frameworks / Technologien, werden im folgenden Kapitel analysiert.

4.2 Architekturdesign

Anhand folgender Bewertungsmatrix, wurde sich für Nuxt3 als JS-Framework entschieden:

Eigenschaft	Gewichtung	Nuxt3	Vue	Angular	React/Next
Dokumentation	3	4	5	5	5
Firmenwissen	5	4	5	2	3
Serverseitig	10	1	0	0	1
Einarbeitungszeit	2	5	5	3	4
Gesamt:	20	52	50	31	48
Nutzwert:		2,6	2,5	1,55	2,4

Table 4: Entscheidungsmatrix Nuxt3

Als Optionen wurden alle Frameworks genommen, mit welchen BAL bereits Erfahrungen gesammelt hat. Eigenentwicklungen wurden von Anfang an ausgeschlossen, da diese zu aufwendig sind. Andere Frameworks hätten eine zu lange Einarbeitung/Umschulung für potentiell zu wenig Nutzen gehabt.

Anhand folgender Bewertungsmatrix, wurde sich für DaisyUI als UI-Framework entschieden:

Eigenschaft	Gewichtung	Buefy	DaisyUI/Tailwind	Vuetify	Standard CSS
Dokumentation	5	4	3	5	0
Nuxt3 Support	3	4	2	5	3
Firmenwissen	3	5	5	5	2
Anpassbarkeit	2	3	2	3	5
Features	2	3	2	3	0
Gesamt:	17	65	52	73	21
Nutzwert:		3,82	3,06	4,29	1,24

Table 5: Entscheidungsmatrix DaisyUI

Anhand folgender Bewertungsmatrix, wurde sich für Typo3 als CMS entschieden:

Eigenschaft	Gewichtung	Akelos	CakePHP	Symfony	Eigenentwicklung
Dokumentation	5	4	3	5	0
Reengineirung	3	4	2	5	3
Generierung	3	5	5	5	2
Testfälle	2	3	2	3	3
Standardaufgaben	4	3	3	3	0
Gesamt:	17	65	52	73	21
Nutzwert:		3,82	3,06	4,29	1,24

Table 6: Entscheidungsmatrix Typo3

Anhand folgender Bewertungsmatrix, wurde sich für TS als Programmiersprache entschieden:

Eigenschaft	Gewichtung	TS	JS
Nuxt3 Support	4	5	5
Skalierbarkeit	4	5	4
Komplexität	3	3	4
Gesamt:	11	49	48
Nutzwert:		4.45	4.36

Table 7: Entscheidungsmatrix TS

Beide Programmiersprachen werden von Nuxt3 standardmäßig unterstützt. TS benötigt mehr Zeilen Code und fügt einen weiteren Layer an Komplexität zu dem Projekt hinzu. Dafür werden Fehler früher entdeckt und das Projekt lässt sich leichter skalieren. Diese Faktoren werden von BAL mehr wertgeschätzt, weswegen sich für TS entschieden wurde.

Navigation: Standardmäßig gibt die Headless-Extension von Typo3 keine Navigationsstruktur aus, weswegen noch Anpassungen am Template gemacht werden mussten. Es wurde sich entschieden, dass alle Seiten(und deren Unterseiten) in die Navigation ausgespielt werden, welche einer bestimmten Kategorie zugewiesen werden. Kategorien sind standardmäßig bei Seiten in Typo3 vorhanden und können mit kleinen Template-Anpassungen ausgespielt werden. Die Headless Extension bietet dafür Funktionalitäten.

Layout / Spalteneinteilung: Standardmäßig hat Typo3 keine Tools um Komponenten in Spalten/Bereiche aufzuteilen. Es gibt viele Standardkonfigurationen, welche dies ermöglichen, aber ein komplett leeres Typo3 hat diese Funktionalität nicht. Deswegen wird eine eigene kleine Extension für Typo3 geschrieben, welche diese Funktionalität liefert.

4.3 Entwurf der Benutzeroberfläche

- Entscheidung für die gewählte Benutzeroberfläche (z. B. GUI, Webinterface).
- Beschreibung des visuellen Entwurfs der konkreten Oberfläche (z. B. Mockups, Menüführung).
- Ggfs. Erläuterung von angewendeten Richtlinien zur Usability und Verweis auf Corporate Design.

Frontend Das Ziel der Benutzeroberfläche des Skeletons ist, dass Komponenten dargestellt werden und zwischen Seiten des Typo3 navigiert werden kann. Es muss noch keine gute Benutzeroberfläche erstellt werden, welche Kunden nützen können. Stattdessen muss es für Entwickler einfach und flexibel sein, eine gute Benutzeroberfläche für Kunden, aus der Benutzeroberfläche des Skeletons, zu entwickeln. Dafür werden die Komponenten und die Navigation stylistisch sehr minimalistisch gehalten und primär ihre Logik erstellt. Die Navigation befindet sich klassisch am Anfang der Website und unter ihr wird der Content der Seite gerendert. Dies wurde auch hier angewandt.

Backend Die Benutzeroberfläche des Backends/CMS ist größtenteils durch Typo3 vorgegeben. Es werden lediglich die Komponenten erweitert, so dass gesteuert werden kann, wie viel Platz die Komponenten einnehmen. Dafür werden für die einzelnen Bildschirmgrößen(Desktop, Tablet, Mobile) Selects eingebaut, welche die Größen als Optionen haben.

Menüführung Die Hauptnavigationenpunkte, welche aus dem Typo3 ausgespielt werden, werden oben in der Menüleiste angezeigt. Ihre Kinder werden ebenfalls ausgespielt, aber erst angezeigt, wenn durch die Navigation navigiert wird. Wenn einer der Hauptnavigationenpunkte angeklickt wird, wird überprüft, ob dieser Children(Unterseiten) hat oder nicht. Wenn dies nicht der Fall ist, wird direkt die Seite von dem Hauptnavigationenpunkt aufgerufen. Wenn dies der Fall ist, wird das Sidemenu aufgerufen. Das Sidemenu enthält sowohl den zuletzt angeklickten Navigationspunkt, als auch alle Children von ihm. Zusätzlich gibt es einen Button, mit welchem zurück navigiert werden kann. Wenn der zuletzt angeklickte Navigationspunkt hier nochmal angeklickt wird, wird dessen Seite aufgerufen. Wenn ein Child angeklickt wird, wird wieder überprüft, ob es Children hat oder nicht. Falls nein, wird die Seite aufgerufen, falls ja wird das Child zum zuletzt angeklickten Navigationspunkt. Eine Darstellung der Menüführung finden sie in Abbildung xyz

Beispiel Beispielentwürfe finden sich im A.8: Oberflächenentwürfe auf Seite ix.

4.4 Datenmodell

- Entwurf/Beschreibung der Datenstrukturen (z. B. ERM und/oder Tabellenmodell, XML-Schemas) mit kurzer Beschreibung der wichtigsten (!) verwendeten Entitäten.

Beispiel In Abbildung 1 wird ein Entity-Relationship-Modell (ERM) dargestellt, welches lediglich Entitäten, Relationen und die dazugehörigen Kardinalitäten enthält.

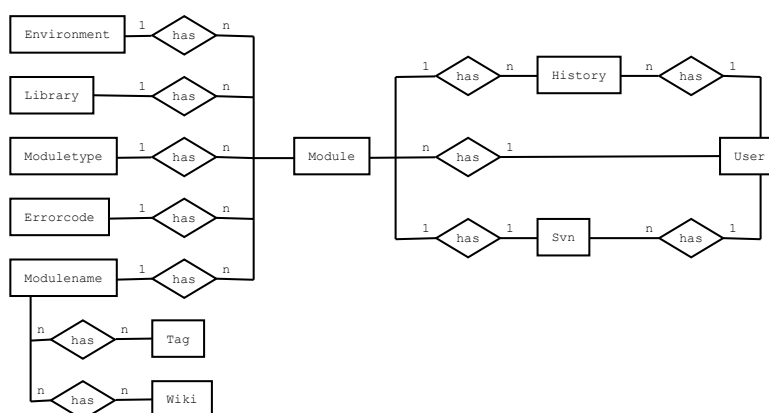


Figure 1: Vereinfachtes ER-Modell

ERM erstellen

Content:

Navigation:

4.5 Geschäftslogik

- Modellierung und Beschreibung der wichtigsten (!) Bereiche der Geschäftslogik (z. B. mit Komponenten-, Klassen-, Sequenz-, Datenflussdiagramm, Programmablaufplan, Struktogramm, Ereignisgesteuerte Prozesskette (EPK)).
- Wie wird die erstellte Anwendung in den Arbeitsfluss des Unternehmens integriert?

Beispiel Ein Klassendiagramm, welches die Klassen der Anwendung und deren Beziehungen untereinander darstellt kann im A.13: Klassendiagramm auf Seite xviii eingesehen werden.

Abbildung 2 zeigt den grundsätzlichen Programmablauf beim Einlesen eines Moduls als EPK.

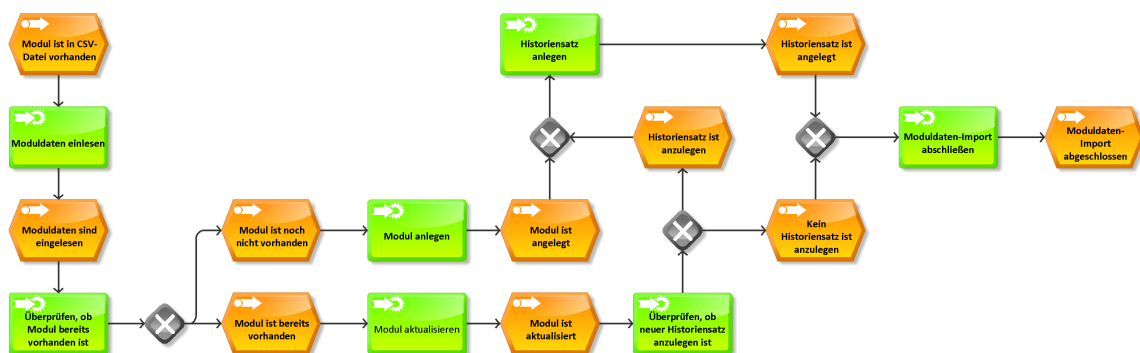


Figure 2: Prozess des Einlesens eines Moduls

4.6 Maßnahmen zur Qualitätssicherung

- Welche Maßnahmen werden ergriffen, um die Qualität des Projektergebnisses (siehe Kapitel 3.5: Qualitätsanforderungen) zu sichern (z. B. automatische Tests, Anwendertests)?
- Ggfs. Definition von Testfällen und deren Durchführung (durch Programme/Benutzer).

Um die SEO von dem Skeleton abschätzen zu können kann Lighthouse von Google verwendet werden. Lighthouse analysiert die Seite bezüglich SEO und liefert abhängig davon einen Wert von 0 – 100. Falls es SEO Schwierigkeiten gibt, werden diese hier erläutert. Da noch nicht viel Content auf den Seiten gepflegt ist, kann es sein, dass es dadurch zu Abzügen kommt. Dies wird sich aber dann automatisch anpassen, wenn mit dem Skeleton eine vernünftige Website erstellt wird.

Wie gut der Code bezüglich Flexibilität und Leserlichkeit ist, lässt sich am besten durch Feedback von Arbeitskollegen erfassen. Da bereits Projekte mit dem Skeleton in Umsetzung sind und mehrere Arbeitskollegen des Prüflings mit dem Code arbeiten, wurde viel Feedback generiert. Dieses viel größtenteils positiv aus.

4.7 Pflichtenheft/Datenverarbeitungskonzept

- Auszüge aus dem Pflichtenheft/Datenverarbeitungskonzept, wenn es im Rahmen des Projekts erstellt wurde.

Beispiel Ein Beispiel für das auf dem Lastenheft (siehe Kapitel ??: ??) aufbauende Pflichtenheft ist im ??: ?? auf Seite ?? zu finden.

5 Implementierungsphase

5.1 Docker-Setup

Als Image für das aktuellste Typo3 Docker-Setup wurde das Image von Martin Helmich genutzt (<https://github.com/martin-helmich/docker-typo3>). Ein Docker-Image ist eine Datei, welche aus Anweisungen besteht, welche vollständige und ausführbare Version einer Anwendung erstellt. Diese Anweisungen werden in einer docker-compose.yml Daten gespeichert und dann mit dem Befehl docker-compose up ausgeführt. Nach dem Ausführen des Befehls, wird eine Typo3 Instanz erstellt und gestartet. Abhängig von dem angegebenen Port, kann dann das Typo3 über Localhost aufgerufen werden. Nach dem ausführen der Anweisungen, wird dort der Installations-Screen(siehe Abbildung) von Typo3 angezeigt. Nach dem Installieren, kann ausgewählt werden, ob das Typo3 mit einer leeren Startseite gestartet werden soll, oder weitere Konfigurationen installiert/ausgeführt werden sollen. Um das Skeleton möglichst minimal zu halten, wird das Typo3 mit einer leeren Startseite initialisiert.

5.2 Ausspielen der Typo3 Daten

Damit die Typo3 Daten im JSON-Format ausgespielt werden, wird die Headless-Extension (<https://extensions.typo3.org/extension/headless>) installiert. Danach muss die Extension zum Template hinzugefügt werden, damit Typo3 weiß, dass es das Template dementsprechend anpassen muss. Typo3 spielt nun die Daten im JSON-Format aus. Das JSON muss noch angepasst werden, damit eine Navigationsstruktur damit abgebildet werden kann. Dafür werden folgende Erweiterungen am Template vorgenommen.

```
1 lib.page = JSON
2 lib.page {
3   fields {
4     navigation {
5       fields {
6         main {
7           dataProcessing {
8             10 = FriendsOfTYPO3\Headless\DataProcessing\MenuProcessor
9             10 {
10              special = categories
11              special.value = 1
12              levels = 7
13              as = main
14              expandAll = 1
15              includeSpacer = 1
16              titleField = nav_title // title
17              dataProcessing {
18                10 = FriendsOfTYPO3\Headless\DataProcessing\FilesProcessor
19                10 {
20                  references.fieldName = media
21                  as = media
22                }
23              }
24              overwriteMenuLevelConfig {
25                stdWrap.cObject {
26                  100 = TEXT
27                  100.field = uid
28                  100.wrap = , "uid":|
29                }
30              }
31            }
32          }
33        }
34      }
35    }
36  }
37 }
```

Durch diese Anpassungen erhält das page Objekt die property navigation. Die Eigenschaft navigation hat wiederum die Eigenschaft main. Mit der Hilfe des Menuprocessor von der Headless Extension,

werden in die main Eigenschaft alle Seiten + Unterseiten geladen, welche der 1. Kategorie zugewiesen werden (special = categories & special value = 1). Falls Bilder für die Seiten gepflegt sind, werden diese ebenfalls durch den FilesProcessor ausgespielt. Diese werden in dem Skeleton aber nicht verwendet. Dies dient nur zum Nutzen von potentiellen zukünftigen Projekten. Die Anpassungen können sowohl in der Skeleton-Extension stattfinden, als auch im Typo3 Backend.

5.3 Skeleton-Extension / Layout

Typo3 ermöglicht es Entwicklern das Typo3 mit eigenen Extensions zu erweitern. Diese Extensions müssen einer bestimmten Struktur folgen, welche von Typo3 vorgegeben ist. Die genaue Form lässt sich in der Typo3 Dokumentation finden.

<https://docs.typo3.org/m/typo3/reference-coreapi/main/en-us/ExtensionArchitecture/Index.html>

Datenmodell: Alle Komponenten, die gepflegt werden können, sollen für Desktop, Tablet und Mobile verschiedene Breiten gepflegt bekommen können. Ein Text soll beispielsweise nur ein Drittel der Bildschirmbreite auf Desktop haben, aber 50% auf einem Tablet und die volle Breite auf einem Mobiltelefon. Dafür erhalten alle Elemente drei Felder in der Datenbank:

```
1 CREATE TABLE tt_content (
2     tx_responsive_mobile int(11) DEFAULT '100' NOT NULL,
3     tx_responsive_tablet int(11) DEFAULT '100' NOT NULL,
4     tx_responsive_desktop int(11) DEFAULT '100' NOT NULL,
5 );
```

Anpassungen an bestehende Tabellen in der Datenbank werden in der ext_tables.sql Datei vorgenommen. Diese Datei wird automatisch beim Installieren von Extensions ausgelesen und die Anpassungen durchgeführt. Felder, welche als Input für Inhaltselemente dienen, werden in Typo3 in der tt_content Tabelle gespeichert. Diese Felder müssen noch mit angemessenen Auswahlmöglichkeiten versehen werden und dann allen Elementen zugewiesen werden.

Backend: Die Inputfelder werden als Selects dargestellt. Die Auswahlmöglichkeiten werden im Code vorgegeben.

```
1 "tx_responsive_mobile" => Array (
2     "exclude" => 1,
3     "label" => 'Mobile',
4     "config" => Array (
5         'type' => 'select',
6         'renderType' => 'selectSingle',
7         'items' => [
8             ['25%', '25'],
```

```

9         ['33%', '33'],
10        ['50%', '50'],
11        ['66%', '66'],
12        ['75%', '75'],
13        ['100%', '100']
14    ],
15    'default' => '100',
16    'size' => 1,
17    'maxitems' => 1,
18    )
19    ),

```

Analog zum oberen Code, werden für Tablet und Desktop ebenfalls Konfigurationen vorgenommen und im Array \$tempColumns gespeichert. Welche Eigenschaften Inputfelder brauchen und welche Optionen es bei den Inputfeldern gibt, lässt sich in der Typo3 Dokumentation(<https://docs.typo3.org/m/typo3/reference-tca/main/en-us/Columns/Index.html>) nachlesen. Um die Inputfelder nun allen Elementen hinzuzufügen, wird die ExtensionManagementUtility Klasse aus dem Typo3 Core genutzt.

```

1 \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addTCAcolumns("tt_content",
    $tempColumns,1);
2 \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addToAllTCAtypes('tt_content',
    '--div--;Responsive,tx_responsive_mobile,tx_responsive_tablet,
    tx_responsive_desktop','','after:addToAllTCAtypes');

```

Die erste Zeile Code speichert die Inputfelder ab, die zweite Zeile Code, fügt sie allen Inhaltselementen hinzu. --div--;Responsive sorgt dafür, dass diese in einem eigenen Tab mit dem Namen Responsive dargestellt werden. Danach wird aufgelistet werden Inputfelder hinzugefügt werden.

Ausspielen ans Frontend: Damit die gepflegten Daten auch ans Frontend ausgespielt werden, muss das Template von Typo3 um diese erweitert werden. Extensions haben dafür im Pfad Configuration->TypoScript die Datei setup.typoscript. Hier können Anpassungen ans Template gepflegt werden, diese Anpassungen werden erst aktiv, wenn man die Extension dem Typo3 Template zuweist.

```

1 lib.appearance {
2     fields {
3         responsive_desktop = TEXT
4         responsive_desktop {
5             field = tx_responsive_desktop
6         }
7         responsive_tablet = TEXT .....
8     }

```

9 }

Die gepflegten Daten werden nun in der Eigenschaft `appearance` ausgespielt und können von Nuxt3 ausgelesen werden.

Sections: Um die Elemente auf einer Seite in Bereiche auszuteilen, wurde ein komplett neues Inhaltselement erstellt. Dieses wurde als Section bezeichnet. Sections können Hintergrundfarben haben, wodurch Abschnitte im Frontend farblich voneinander getrennt dargestellt werden können. Dafür wurde ebenfalls in der `tt_content` Tabelle ein Feld angelegt. In dem Fall mit dem Namen `tx_bal_column_color`. Damit das neue Inhaltselement in Typo3 genutzt werden kann, muss die `tt_content.php` Datei um folgenden Code erweitert werden. A.2: `tt_content.php` auf Seite ii

Die Datei lässt sich im Pfad `Configuration->TCA->Overrides` der Extension finden.

Damit das neue Element auch in der Auswahl von Elementen angezeigt wird, muss das Element noch in der `ext_localconf.php` Datei im Root Verzeichnes der Extension hinzugefügt werden.

A.3: `ext_localconf.php` auf Seite iii

Übersetzungen für den Namen und die Beschreibung der Section wurden in einer `.xlf` Datei gepflegt.

5.4 Implementierung der Benutzeroberfläche

- Beschreibung der Implementierung der Benutzeroberfläche, falls dies separat zur Implementierung der Geschäftslogik erfolgt (z. B. bei HTML-Oberflächen und Stylesheets).
- Ggfs. Beschreibung des Corporate Designs und dessen Umsetzung in der Anwendung.
- Screenshots der Anwendung

Nuxt3 aufsetzen um DaisyUI installieren Um Nuxt3 nutzen zu können, muss eine aktuelle Version von Node.js auf der Betriebssystem installiert werden. Zum installieren von Nuxt3 und DaisyUI wurde der Node Package Manager genutzt.

Dynamische Seiten Alle Seiten der Nuxt3 Anwendungen befinden sich im `pages` Ordner. Da Nuxt3 nicht weiß, wie der Seitenbaum von Typo3 aussieht und dieser sich ständig ändert, können die einzelnen Seiten nicht per Hand eingepflegt werden. Damit Nuxt3 trotzdem weiß welcher Code bei den unterschiedlichen Seiten ausgeführt werden soll, gibt es eine Catch-all Route. Eine Datei/Seite die den Namen `[...slug].vue` erhält, fungiert als Catch-all Route. Dies bedeutet, dass alle Seiten die nicht gepflegt sind, durch diese Datei verarbeitet werden. Wenn aber beispielsweise eine Shop-Seite angelegt werden soll, die ihre Daten durch eine E-Commerce Plattform bekommt. Dann kann im `pages` Ordner eine `shop.vue` Datei hinterlegt werden, welche dann für den Pfad `beispielseite.de/shop` die Daten darstellt. Analog kann dies für alle möglichen Seiten geschehen.

Verarbeitung der Typo3 Daten In der `nuxt.config.ts` Datei können generelle Konfigurationen am Nuxt3 vorgenommen werden. In ihr wurde die URL des Typo3 gespeichert. Dies hat den Vorteil, dass die URL in der Anwendung als Variable genutzt werden kann. Das heißt, dass bei späteren Entwicklungen lediglich die Variable einen anderen Wert bekommen muss und Nuxt automatisch alle Daten von einem anderen Typo3 bezieht.

```
1 export default defineNuxtConfig({
2   runtimeConfig: {
3     public: {
4       typo3: 'http://localhost'
5     }
6   },
7 })
```

Die Variable wird darauf hin in der `[...slug].vue` Datei genutzt um die Daten vom Typo3 zu laden. Dies geschieht im `<script setup>` Tag. Nuxt3 Dateien können sowohl einen Script Tag haben, welcher Clientseitig ausgeführt wird, als auch ein Script Tag, welcher Serverseitig ausgeführt wird (hat den Zusatz `setup`). Da die Ladezeit minimal gehalten werden soll, werden die Typo3 Daten hier Serverseitig geladen. Zum Laden der Daten wird die Nuxt3 eigene `useAsyncData()` Funktion genutzt. Mit ihr können Daten asynchron geladen werden. Zusätzlich hat sie viele Optionen um die Anfrage anzupassen. Eine der Optionen ist es den Cache auszuschalten. Für Entwicklungszwecke wird der Cache hier noch ausgeschaltet. Damit Änderungen im Typo3 direkt sichtbar werden. Wenn das Projekt in Production geht, sollte die Zeile `initialCache: false` entfernt werden. Dadurch speichert Nuxt3 die Daten im Cache und die Seite wird schnell geladen. Je nach Website kann programmiert werden, dass sich der Cache häufiger oder weniger häufiger leert.

```
1 const runtimeConfig = useRuntimeConfig()
2 const route = useRoute();
3 const {data} = await useAsyncData(
4   "pageData",
5   () => $fetch(runtimeConfig.typo3 + route.fullPath),
6   {initialCache: false}
7 )
```

Im `route` Objekt ist der Pfad der aktuellen Website gespeichert. `route.fullPath` würde bei der Seite `beispieleite.de/beispiel1` dem Wert `beispiel1` entsprechen. Die Daten aus dem Typo3 (hier das `data` Objekt) wurden danach weiter verarbeitet. Der erste Schritt der Datenverarbeitung war es die Daten in Kategorien zu zerlegen. Dafür wurden vier Kategorien/Variablen angelegt. Die Variable `breadcrumbs` bekam die Daten bezüglich der Breadcrumbs der Seite. Die Variable `Content` bekam alle Daten bezüglich der Inhaltselemente auf der bestimmten Seite. Die Variable `mainNavigation` bekam alle Daten der Hauptnavigation. Als letztes bekam die Variable `metaData` alle Metadaten der Website. Die Breadcrumbs wurden bereits automatisch durch die Headless Extension generiert und sind im

Datenobjekt vorhanden. Das gleiche gilt für die Metadaten und dem Großteil der Inhaltselemente. Die Navigationsdaten finden sich in der Eigenschaft main, welche zum Objekt navigation gehört, welches wiederum zum Objekt page gehört. Diese Logik wurde in 5.2 angelegt.

```
1 let breadcrumbs = data.value.breadcrumbs;
2 let content = data.value.content.colPos0;
3 let mainNavigation = data.value.page.navigation.main;
4 let metaData = data.value.meta;
```

Mit der Nuxt3 Funktion useHead() können die Metadaten an die Website weiter gegeben werden. Für die Weiterverarbeitung der Breadcrumbs, die Navigation und den Content wurden eigene Komponenten erstellt. Diese erhalten die Variablen als Eigenschaften/Properties/Props zugewiesen. Da Nuxt3 einen automatischen Import von Komponenten hat, müssen diese lediglich im Template gepflegt werden und nicht zusätzlich importiert werden. Dafür wurden die Komponenten im components Ordner gepflegt, damit Nuxt3 diese automatisch importieren kann. Den vollständigen Code der [...slug].vue finden Sie unter A.4: [...slug].vue auf Seite iv.

Setup Navigation Die Logik der Navigation wurde in eine Nuxt3 Composable ausgelagert. Die Navigation muss mehrere Aufgaben erfüllen. Den gesamten Code des Composables finden Sie unter A.5: navigation.ts auf Seite v.

- öffnen/schließen des Slidemenu
- anzeigen von allen Seiten
- navigieren auf Seiten

Für das öffnen und schließen des Slidemenu wurden im Composable die Funktionen open() und close() erstellt. Der State, ob das Slidemenu geöffnet oder geschlossen ist, wird über die Variable navigationOpened gesteuert. Dafür wird die neue Nuxt3 eigene Funktion useState() genommen. Mit ihr lässt sich sehr einfach der State über die gesamte Anwendung verwalten. Der useState Funktion wird dabei ein Key übergeben, mit dem überall in der Anwendung auf den State zugegriffen werden kann. Zusätzlich kann der Wert des States optional initialisiert werden. Hier wird natürlich der Initialwert auf false gesetzt. Die Navigation soll sich erst öffnen, wenn der Nutzer der Website das möchte. Analog wurde jeweils ein State für die gesamte Navigation(fullNavList), die aktuelle im Slidemenu angezeigte Navigation(currentNavList) und dem Parent der aktuell angezeigten Navigation(previousNavItem) erstellt.

Aufteilung Navigation Die Navigation wurde in zwei Teile aufgeteilt, der erste Teil sind die Navigationspunkte, welche in der Navigationsbar der Website angezeigt werden. Sie sind die Seiten, welche im Typo3 die Kategorie "main" zugewiesen bekommen haben. Der zweite Teil ist das Slidemenu, das Slidemenu wird geöffnet, wenn ein Element in der Navigationsbar angeklickt wird und es Unterseiten(Children) hat.

Im Slidemenu wird die `currentNavList` dargestellt. Initial sind dies die Unterseiten, des angeklickten Navigationselements. Die Navigationsbar wird in der Komponente `TopNavigation.vue` dargestellt und das Slidemenu in der Komponente `SlideMenu.vue`.

TopNavigation.vue `TopNavigation.vue` ist eine sehr einfache Komponente. Sie nutzt das `navigation Composable` um auf den State `fullNavList` zuzugreifen. Mit `v-for`, einer `vue/nuxt` internen Funktionalität, lässt sich im Template durch die Elemente durchiterieren. Mit `@click` lassen sich Funktionen aufrufen, wenn ein Element angeklickt wird. In diesem Fall ist es die `open()` Funktion des `composables`.

```
1 <div v-for="navItem in navigation.fullNavList.value"
2 :key="navItem.uid" @click="navigation.open(navItem)">
3   {{navItem.title}} </div>
```

Es wurde noch einfaches Styling mit `DaisyUI` hinzugefügt, dies hat aber keinen Einfluss auf die Logik der Komponente und wird in zukünftigen Projekten wahrscheinlich überschrieben werden. Deswegen wird es hier nicht mit angezeigt.

open() Die `open()` Funktion besteht aus einem `if else` Statement. Den Programmcode finden sie im `Navigation Composable A.5: navigation.ts` auf Seite v (`const open = ...`). Wenn das Navigationselement(`navItem`), welches als Parameter angegeben wurde, keine Unterseiten hat, dann wird die Seite des `navItems` aufgerufen. Dafür wird der `Nuxt-Router` genutzt. Mit ihm kann programmatisch die Route geändert werden. Falls das Navigationselement Unterseiten hat, dann wird die aktuell angezeigt Navigation gleich den Unterseiten gesetzt. Als Parent der Navigation wird das `Navigationsitem` gesetzt und das Slidemenu, was diese Informationen darstellt, wird geöffnet.

Slidemenu.vue Den kompletten Code des Slidemenu finden Sie im Anhang unter `A.6: SlideMenu.vue` auf Seite vi. Das Slidemenu zeigt die aktuelle Navigation an, welche aus Unterseiten eines anderen Navigationselement besteht. Dafür nutzt sie die gleiche Logik wie die Navigationsbar. Es wird genauso durch alle Elemente von `currentNavList` iteriert, wie es bei der Navigationsbar für `fullNavList` passiert ist. Der Unterschied liegt in der Funktion, welche bei `@click` hinterlegt wird. Es wird die Funktion `navigate()` aufgerufen, welche im `Navigation Composable` hinterlegt ist. Die Funktion `navigate()` ähnelt der Funktion `open()` sehr. Beide nehmen ein Navigationselement als Parameter. Beide Funktionen haben die gleichen `if else` Bedingungen. Der Unterschied liegt in dem Code, welcher nach der jeweiligen Bedingung ausgeführt wird. Wenn das Navigationselement keine Unterseiten hat, dann wird wie bei `open()` zur Seite navigiert. Zusätzlich wird aber auch das Slidemenu mit der `close()` Funktion geschlossen. Theoretisch könnte hier auch der State von `navigationOpened` direkt auf `false` gesetzt werden. Falls sich die `close()` Logik aber erweitert, muss dies dann nicht an mehreren Orten angepasst werden. Falls das Element Unterseiten hat, wird die aktuelle angezeigte Navigation den Unterseiten gleichgesetzt. Ebenfalls wie bei `open()` Funktion wird auch der Parent der Navigation gleich dem Parameter gesetzt. Es wird aber nicht das Slidemenu geöffnet, da es ja bereits geöffnet ist.

Finale Anwendung Screenshots der finalen Anwendung inklusive der gepflegten Dummy-Inhalte der Typo3 Seite befinden sich im A.9: Screenshots der Anwendung auf Seite xi.

5.5 Implementierung der Geschäftslogik

- Beschreibung des Vorgehens bei der Umsetzung/Programmierung der entworfenen Anwendung.
- Ggfs. interessante Funktionen/Algorithmen im Detail vorstellen, verwendete Entwurfsmuster zeigen.
- Quelltextbeispiele zeigen.
- Hinweis: Wie in Kapitel 1: Einleitung zitiert, wird nicht ein lauffähiges Programm bewertet, sondern die Projektdurchführung. Dennoch würde ich immer Quelltextausschnitte zeigen, da sonst Zweifel an der tatsächlichen Leistung des Prüflings aufkommen können.

Beispiel Die Klasse `ComparedNaturalModuleInformation` findet sich im A.12: Klasse: `ComparedNaturalModule` auf Seite xv.

6 Abnahmephase

- Welche Tests (z. B. Unit-, Integrations-, Systemtests) wurden durchgeführt und welche Ergebnisse haben sie geliefert (z. B. Logs von Unit Tests, Testprotokolle der Anwender)?
- Wurde die Anwendung offiziell abgenommen?

Beispiel Ein Auszug eines Unit Tests befindet sich im A.11: Testfall und sein Aufruf auf der Konsole auf Seite xiv. Dort ist auch der Aufruf des Tests auf der Konsole des Webservers zu sehen.

7 Einführungsphase

- Welche Schritte waren zum Deployment der Anwendung nötig und wie wurden sie durchgeführt (automatisiert/manuell)?
- Wurden ggfs. Altdaten migriert und wenn ja, wie?
- Wurden Benutzerschulungen durchgeführt und wenn ja, Wie wurden sie vorbereitet?

8 Dokumentation

- Wie wurde die Anwendung für die Benutzer/Administratoren/Entwickler dokumentiert (z. B. Benutzerhandbuch, API-Dokumentation)?
- Hinweis: Je nach Zielgruppe gelten bestimmte Anforderungen für die Dokumentation (z. B. keine IT-Fachbegriffe in einer Anwenderdokumentation verwenden, aber auf jeden Fall in einer Dokumentation für den IT-Bereich).

Beispiel Ein Ausschnitt aus der erstellten Benutzerdokumentation befindet sich im A.14: Benutzerdokumentation auf Seite xix. Die Entwicklerdokumentation wurde mittels PHPDoc¹⁰ automatisch generiert. Ein beispielhafter Auszug aus der Dokumentation einer Klasse findet sich im A.10: Entwicklerdokumentation auf Seite xii.

9 Fazit

9.1 Soll-/Ist-Vergleich

- Wurde das Projektziel erreicht und wenn nein, warum nicht?
- Ist der Auftraggeber mit dem Projektergebnis zufrieden und wenn nein, warum nicht?
- Wurde die Projektplanung (Zeit, Kosten, Personal, Sachmittel) eingehalten oder haben sich Abweichungen ergeben und wenn ja, warum?
- Hinweis: Die Projektplanung muss nicht strikt eingehalten werden. Vielmehr sind Abweichungen sogar als normal anzusehen. Sie müssen nur vernünftig begründet werden (z. B. durch Änderungen an den Anforderungen, unter-/überschätzter Aufwand).

Beispiel (verkürzt) Wie in Tabelle 8 zu erkennen ist, konnte die Zeitplanung bis auf wenige Ausnahmen eingehalten werden.

9.2 Lessons Learned

- Was hat der Prüfling bei der Durchführung des Projekts gelernt (z. B. Zeitplanung, Vorteile der eingesetzten Frameworks, Änderungen der Anforderungen)?

¹⁰Vgl. PHPDOC.ORG [2010]

Phase	Geplant	Tatsächlich	Differenz
Entwurfsphase	19 h	19 h	
Analysephase	9 h	10 h	+1 h
Implementierungsphase	29 h	28 h	-1 h
Abnahmetest der Fachabteilung	1 h	1 h	
Einführungsphase	1 h	1 h	
Erstellen der Dokumentation	9 h	11 h	+2 h
Pufferzeit	2 h	0 h	-2 h
Gesamt	70 h	70 h	

Table 8: Soll-/Ist-Vergleich

9.3 Ausblick

- Wie wird sich das Projekt in Zukunft weiterentwickeln (z. B. geplante Erweiterungen)?

Literaturverzeichnis

Bundesministerium für Bildung und Forschung 2000

BUNDESMINISTERIUM FÜR BILDUNG UND FORSCHUNG: Umsetzungshilfen für die neue Prüfungsstruktur der IT-Berufe / Bundesministerium für Bildung und Forschung. Version: Juli 2000. <http://fiae.link/UmsetzungshilfenITBerufe>. Bonn, Juli 2000. – Abschlussbericht. – 476 S.

Grashorn 2010

GRASHORN, Dirk: Entwicklung von NatInfo – Webbasiertes Tool zur Unterstützung der Entwickler / Alte Oldenburger Krankenversicherung AG. Vechta, April 2010. – Dokumentation zur Projektarbeit

IHK Darmstadt 2011

IHK DARMSTADT: *Bewertungsmatrix für Fachinformatiker/innen Anwendungsentwicklung*. <http://fiae.link/BewertungsmatrixDokuDarmstadt>. Version: März 2011

IHK Oldenburg 2006

IHK OLDENBURG: *Merkblatt zur Abschlussprüfung der IT-Berufe*. <http://fiae.link/MerkblattDokuOldenburg>. Version: Mai 2006

Microsoft 2022

MICROSOFT: *TypeScript - TypeScript is a strongly typed programming language that builds on JavaScript*. Version: 2022. <https://www.typescriptlang.org/>, Abruf: 17.10.2022

Nuxt 3 2022

NEXT 3: *Nuxt 3 - Nuxt 3 is an open source framework making web development simple and powerful*. Version: 2022. <https://v3.nuxtjs.org/>, Abruf: 17.10.2022

phpdoc.org 2010

PHPDOC.ORG: *phpDocumentor-Website*. Version: 2010. <http://www.phpdoc.org/>, Abruf: 20.04.2010

Regierung der Bundesrepublik Deutschland 1997

REGIERUNG DER BUNDESREPUBLIK DEUTSCHLAND: *Verordnung über die Berufsausbildung im Bereich der Informations- und Telekommunikationstechnik*. <http://fiae.link/VerordnungITBerufe>. Version: Juli 1997

Rohrer und Sedlacek 2011

ROHRER, Anselm ; SEDLACEK, Ramona: *Cleverer Tipps für die Projektarbeit - IT-Berufe: Abschlussprüfung Teil A*. 5. Solingen : U-Form-Verlag, 2011 <http://fiae.link/ClevererTippsFuerDieProjektarbeit>. – ISBN 3882347538

Sensio Labs 2010

SENSIO LABS: *Symfony - Open-Source PHP Web Framework*. Version: 2010. <http://www.symfony-project.org/>, Abruf: 20.04.2010

Eidesstattliche Erklärung

Ich, Lukas Röding, versichere hiermit, dass ich meine **Dokumentation zur betrieblichen Projektarbeit** mit dem Thema

Nuxt3 Typo3 Skeleton – Webbasiertes Headless Content-Management-System

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Dortmund, den 23.04.2015

LUKAS RÖDING

A Anhang

A.1 Detaillierte Zeitplanung

Analysephase	10 h
1. Analyse des Ist-Zustands	2 h
2. Wirtschaftlichkeitsanalyse	1 h
3. Nuxt2 vs Nuxt3, welche Nuxt3 features sollen implementiert werden?	7 h
Absprache mit senior Developer	
Entwurfsphase	19 h
1. Prozessentwurf	2 h
2. Datenbankentwurf	3 h
2.1. ER-Modell erstellen	2 h
2.2. Konkretes Tabellenmodell erstellen	1 h
3. Erstellen von Datenverarbeitungskonzepten	4 h
3.1. Verarbeitung der CSV-Daten	1 h
3.2. Verarbeitung der SVN-Daten	1 h
3.3. Verarbeitung der Sourcen der Programme	2 h
4. Benutzeroberflächen entwerfen und abstimmen	2 h
5. Erstellen eines UML-Komponentendiagramms der Anwendung	4 h
6. Erstellen des Pflichtenhefts	4 h
Implementierungsphase	29 h
1. Anlegen der Datenbank	1 h
2. Umsetzung der HTML-Oberflächen und Stylesheets	4 h
3. Programmierung der PHP-Module für die Funktionen	23 h
3.1. Import der Modulinformationen aus CSV-Dateien	2 h
3.2. Parsen der Modulquelltexte	3 h
3.3. Import der SVN-Daten	2 h
3.4. Vergleichen zweier Umgebungen	4 h
3.5. Abrufen der von einem zu wählenden Benutzer geänderten Module	3 h
3.6. Erstellen einer Liste der Module unter unterschiedlichen Aspekten	5 h
3.7. Anzeigen einer Liste mit den Modulen und geparsten Metadaten	3 h
3.8. Erstellen einer Übersichtsseite für ein einzelnes Modul	1 h
4. Nächtlichen Batchjob einrichten	1 h
Abnahmetest der Fachabteilung	1 h
1. Abnahmetest der Fachabteilung	1 h
Einführungsphase	1 h
1. Einführung/Benutzerschulung	1 h
Erstellen der Dokumentation	9 h
1. Erstellen der Benutzerdokumentation	2 h
2. Erstellen der Projektdokumentation	6 h
3. Programmdokumentation	1 h
3.1. Generierung durch PHPdoc	1 h
Pufferzeit	2 h
1. Puffer	2 h
Gesamt	70 h

A.2 tt_content.php

```
1 <?php declare(strict_types=1);
2
3 defined('TYPO3_MODE') || die();
4
5 // static TypoScript
6 (static function () {
7     \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addPlugin(
8         array(
9             'LLL:EXT:bal_skeleton/Resources/Private/Language/Tca.xlf:bal_column.
              wizard.title',
10             'tx_bal_column',
11             'EXT:bal_skeleton/Resources/Public/Icons/ContentElements/stage.png'
12         ),
13         'CType',
14         'bal_skeleton'
15     );
16     $temporaryColumn = array(
17         'tx_bal_column_color' => array (
18             'exclude' => 1,
19             'label' => 'LLL:EXT:bal_skeleton/Resources/Private/Language/Tca.xlf:
              bal_column.color.title',
20             'config' => array (
21                 'type' => 'input',
22                 'renderType' => 'colorpicker',
23                 'size' => 10,
24             )
25         ),
26     );
27
28     \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addTCAcolumns(
29         'tt_content',
30         $temporaryColumn
31     );
32
33     $GLOBALS['TCA']['tt_content']['types']['tx_bal_column'] = array(
34         'showitem' => '
35             --palette--;LLL:EXT:frontend/Resources/Private/Language/locallang_ttc.xml
              :palette.general;general,
36             tx_bal_column_color,
```

```
37     ');  
38 } ) ();
```

A.3 ext_localconf.php

```
1 <?php  
2 defined('TYPO3_MODE') || die('Access denied.');
```



```
3  
4 use TYPO3\CMS\Extbase\Utility\ExtensionUtility;  
5  
6 call_user_func(  
7     function () {  
8         // wizards  
9         \TYPO3\CMS\Core\Utility\ExtensionManagementUtility::addPageTSConfig(  
10             'mod.wizards.newContentElement.wizardItems.common {  
11                 elements {  
12                     tx_bal_column {  
13                         iconIdentifier = bal_column  
14                         title = LLL:EXT:bal_skeleton/Resources/Private/Language/Tca.xlf:  
15                             bal_column.wizard.title  
16                         description = LLL:EXT:bal_skeleton/Resources/Private/Language/Tca.  
17                             xlf:bal_column.wizard.description  
18                         tt_content_defValues {  
19                             CType = tx_bal_column  
20                         }  
21                     }  
22                 }  
23             }  
24  
25             $iconRegistry = \TYPO3\CMS\Core\Utility\GeneralUtility::makeInstance(\TYPO3\  
26                 CMS\Core\Imaging\IconRegistry::class);  
27  
28             $iconRegistry->registerIcon(  
29                 'bal_column',  
30                 \TYPO3\CMS\Core\Imaging\IconProvider\SvgIconProvider::class,  
31                 ['source' => 'EXT:bal_skeleton/Resources/Public/Icons/ContentElements/  
stage.svg']  
);
```

```
32   }  
33 );
```

A.4 [...slug].vue

```
1  <script setup lang="ts">  
2    const runtimeConfig = useRuntimeConfig()  
3    const route = useRoute();  
4    const {data }= await useAsyncData(  
5      "pageData",  
6      () => $fetch(runtimeConfig.typo3 + route.fullPath),  
7      {initialCache: false}  
8    )  
9    let breadcrumbs = data.value.breadcrumbs;  
10   let content = data.value.content.colPos0;  
11   let mainNavigation = data.value.page.navigation.main;  
12   let metaData = data.value.meta;  
13  
14   useHead({  
15     title: metaData.title,  
16     meta: [  
17       { name: 'description', content: metaData.description ? metaData.description :  
18         '' }  
19     ]  
20   })  
21 </script>  
22 <template>  
23   <div>  
24     <template v-if="mainNavigation"><Navigation :typo3-navigation="  
25       mainNavigation"></Navigation></template>  
26     <div class="container mx-auto xl">  
27       <template v-if="breadcrumbs"><Breadcrumbs :typo3breadcrumbs="breadcrumbs  
28         "></Breadcrumbs></template>  
29       <template v-if="content"><TypoView :typo3-content="content"></TypoView></  
30         template>  
31       <h3 v-else>Noch kein Content gepflegt!</h3>  
32     </div>  
33   </div>  
34 </template>
```

A.5 navigation.ts

```
1 export const useNavigation = () => {
2   const fullNavList = useState('fullNavList', () => []);
3   const navigationOpened = useState('navigationOpened', () => false);
4   const currentNavList = useState('currentNavList', () => []);
5   const previousNavItem: {value: navigationItem | Record<string, never>} = useState
      ('previousNavItem');
6   const router = useRouter();
7
8   const open = (navItem: navigationItem) => {
9     if (!navItem.hasSubpages) {
10       router.push(navItem.link)
11     } else {
12       previousNavItem.value = navItem;
13       currentNavList.value = navItem.children;
14       navigationOpened.value = true;
15     }
16   }
17   const close = () => {
18     navigationOpened.value = false;
19   }
20   const navigate = (navItem: navigationItem) => {
21     if (!navItem.hasSubpages) {
22       close();
23       router.push(navItem.link)
24     }
25     else {
26       previousNavItem.value = navItem;
27       currentNavList.value = navItem.children;
28     }
29   }
30   const back = () => {
31     const parent = findParent(fullNavList.value, previousNavItem.value.uid);
32     if (parent && parent.children) {
33       previousNavItem.value = parent;
34       currentNavList.value = parent.children;
35     } else {
36       previousNavItem.value = {};
37       currentNavList.value = fullNavList.value;
38     }
39   }
40 }
```

```
39   }
40   const findParent = (nav: navigationItem[], id: number, potentialParentNav?:
     navigationItem) => {
41     let parent: navigationItem;
42     for (let index = 0; index < nav.length; index++) {
43       const item = nav[index];
44       if (item.uid === id) {
45         parent = potentialParentNav;
46         break;
47       }
48       else {
49         if (item.children && item.children.length > 0) {
50           parent = findParent(item.children, id, item);
51           if (parent) {
52             break;
53           }
54         }
55       }
56     }
57     return parent;
58   }
59   return {fullNavList, navigationOpened, open, close, currentNavList,
     previousNavItem, navigate, back};
60 }
```

A.6 SlideMenu.vue

```
1 <script setup>
2   const {fullNavList, navigationOpened, close, currentNavList, previousNavItem,
     navigate, back} = useNavigation();
3 </script>
4 <template>
5   <div class="slidemenu-container" :class="{ 'active' : navigationOpened}">
6     <div class="flex">
7       <div class="text-xl font-bold cursor-pointer" v-if="fullNavList !=
         currentNavList" @click="back()">back</div>
8       <div class="text-xl font-bold pl-6 grow" v-if="previousNavItem">
9         <NuxtLink :to="previousNavItem.link" @click="close()">{{
           previousNavItem.title}}</NuxtLink>
10      </div>
```

```
11     <div class="text-xl font-bold cursor-pointer" @click="close()">X</div>
12   </div>
13   <hr>
14   <div class="navbar">
15     <div class="btn btn-ghost normal-case text-xl" v-for="navItem in
16       currentNavList" :key="navItem.uid" @click="navigate(navItem)">
17       {{navItem.title}}
18     </div>
19   </div>
20 </template>
21 <style>
22   .slidemenu-container {
23     position: absolute;
24     height: 100vh;
25     width: 0px;
26     transition: width 0.5s;
27     background: white;
28     overflow-x: hidden;
29     z-index: 100;
30   }
31   .slidemenu-container.active {
32     width: 400px;
33   }
34 </style>
```

A.7 Datenbankmodell

ER-Modelle kann man auch direkt mit L^AT_EX zeichnen, siehe z. B. <http://www.texample.net/tikz/examples/entity-relationship-diagram/>.

Figure 3: Datenbankmodell

A.8 Oberflächenentwürfe

The screenshot shows a web browser window with the address bar displaying `http://natinfo.intranet/module/`. The page contains a filter section at the top and a table of modules below.

Filter:
Source- and Catalog-Information from Environment:
Library: Select
Filter

Source:
Date: Select
Username: Select
[+][+][+][+]

Catalog:
Date: Select
Username: Select
[+][+][+]

Module	Library	Sourcen	Catalog	Source-User	Source-Date	Catalog-User	Catalog-Date
DGTEST	UTILITY	+	+	Grashorn	01.04.2010	Grashorn	02.04.2010
EINTEST	SYSTEM	-	+	Mustermann	01.04.2010	Grashorn	02.04.2010
NOCHEINS	UTILITY	○	-	Grashorn	05.04.2010	Grashorn	05.04.2010
MANUEL	SYSTEM	○	+	Grashorn	01.03.2010	Grashorn	01.03.2010
RESET	CON	-	+	Mustermann	02.03.2010	Mustermann	02.03.2010
TESTER	SYSTEM	+	-	Grashorn	01.02.2010	Grashorn	01.02.2010
...

Figure 4: Liste der Module mit Filtermöglichkeiten

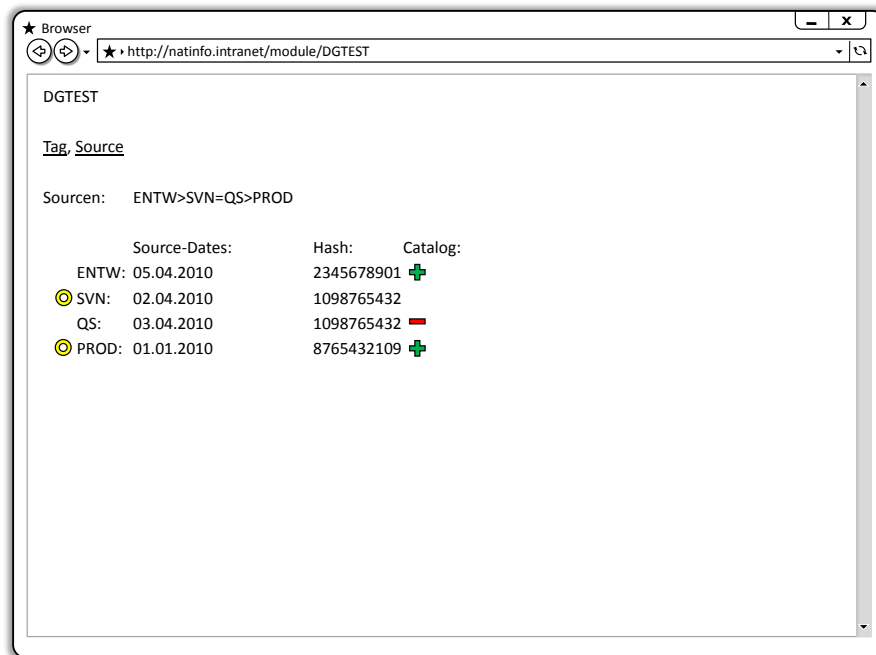


Figure 5: Anzeige der Übersichtsseite einzelner Module

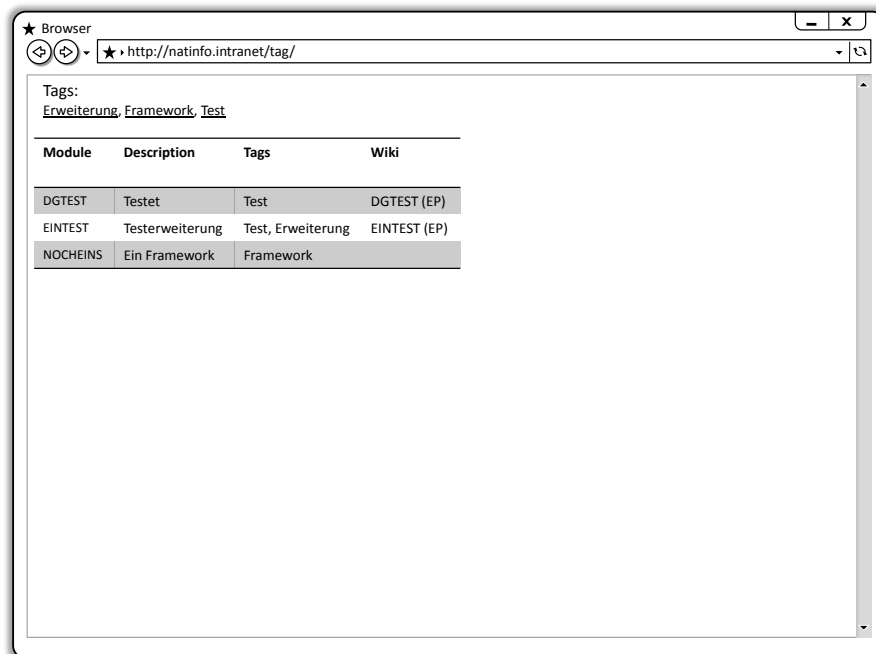


Figure 6: Anzeige und Filterung der Module nach Tags

A.9 Screenshots der Anwendung



Figure 7: Ausspielen der Komponenten im Frontend

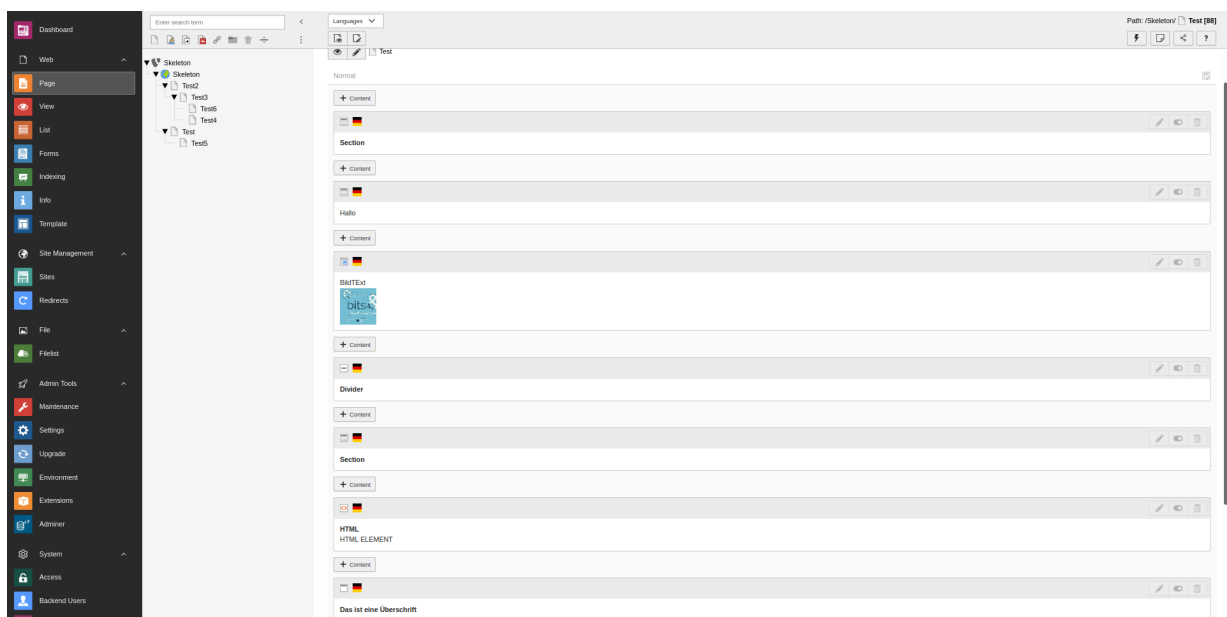


Figure 8: Pflegen der Komponenten im Backend

A.10 Entwicklerdokumentation

lib-model

[class tree: lib-model] [index: lib-model] [all elements]

Packages:
lib-model

Files:
Naturalmodulename.php

Classes:
Naturalmodulename

Class: Naturalmodulename

Source Location: /Naturalmodulename.php

Class Overview

```
BaseNaturalmodulename
|
--Naturalmodulename
```

Subclass for representing a row from the 'NaturalModulename' table.

Methods

- [__construct](#)
- [getNaturalTags](#)
- [getNaturalWikis](#)
- [loadNaturalModuleInformation](#)
- [__toString](#)

Class Details

[line 10]
Subclass for representing a row from the 'NaturalModulename' table.

Adds some business logic to the base.

[[Top](#)]

Class Methods

constructor [__construct](#) [line 56]

```
Naturalmodulename __construct( )
```

Initializes internal state of Naturalmodulename object.

Tags:

see: parent::__construct()
access: public

[[Top](#)]

method [getNaturalTags](#) [line 68]

```
array getNaturalTags( )
```

Returns an Array of NaturalTags connected with this Modulename.

Tags:

return: Array of NaturalTags
access: public

[\[Top \]](#)

method getNaturalWikis [line 83]

```
array getNaturalWikis( )
```

Returns an Array of NaturalWikis connected with this Modulename.

Tags:

return: Array of NaturalWikis
access: public

[\[Top \]](#)

method loadNaturalModuleInformation [line 17]

```
ComparedNaturalModuleInformation  
loadNaturalModuleInformation( )
```

Gets the ComparedNaturalModuleInformation for this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

method __toString [line 47]

```
string __toString( )
```

Returns the name of this NaturalModulename.

Tags:

access: public

[\[Top \]](#)

Documentation generated on Thu, 22 Apr 2010 08:14:01 +0200 by [phpDocumentor 1.4.2](#)

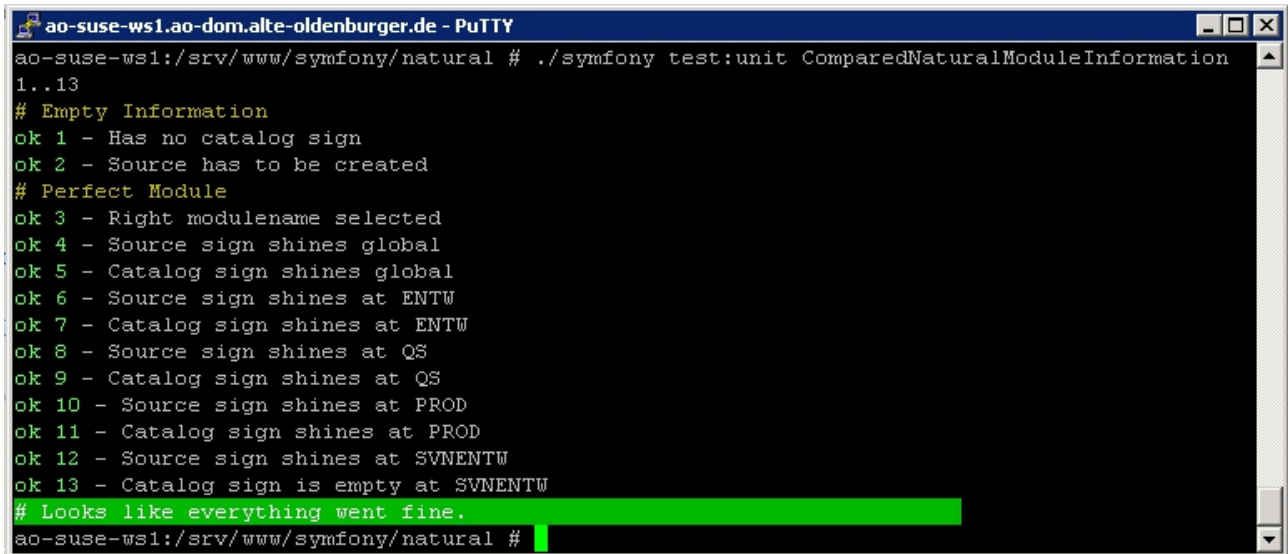
A.11 Testfall und sein Aufruf auf der Konsole

```

1 <?php
2 include(dirname(__FILE__).'/../bootstrap/Propel.php');
3
4 $t = new lime_test(13);
5
6 $t->comment('Empty Information');
7 $emptyComparedInformation = new ComparedNaturalModuleInformation(array());
8 $t->is($emptyComparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, '
    Has no catalog sign');
9 $t->is($emptyComparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_CREATE, '
    Source has to be created');
10
11 $t->comment('Perfect Module');
12 $criteria = new Criteria();
13 $criteria->add(NaturalmodulenamePeer::NAME, 'SMTAB');
14 $moduleName = NaturalmodulenamePeer::doSelectOne($criteria);
15 $t->is($moduleName->getName(), 'SMTAB', 'Right modulename selected');
16 $comparedInformation = $moduleName->loadNaturalModuleInformation();
17 $t->is($comparedInformation->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign
    shines global');
18 $t->is($comparedInformation->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign
    shines global');
19 $infos = $comparedInformation->getNaturalModuleInformations();
20 foreach($infos as $info)
21 {
22     $env = $info->getEnvironmentName();
23     $t->is($info->getSourceSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Source sign shines at ' . $env);
24     if ($env != 'SVNENTW')
25     {
26         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::SIGN_OK, 'Catalog sign shines at ' .
            $info->getEnvironmentName());
27     }
28     else
29     {
30         $t->is($info->getCatalogSign(), ComparedNaturalModuleInformation::EMPTY_SIGN, 'Catalog sign is empty
            at ' . $info->getEnvironmentName());
31     }
32 }
33 ?>

```

Listing 1: Testfall in PHP



```

ao-suse-ws1.ao-dom.alte-oldenburger.de - PuTTY
ao-suse-ws1:/srv/www/symfony/natural # ./symfony test:unit ComparedNaturalModuleInformation
1..13
# Empty Information
ok 1 - Has no catalog sign
ok 2 - Source has to be created
# Perfect Module
ok 3 - Right modulename selected
ok 4 - Source sign shines global
ok 5 - Catalog sign shines global
ok 6 - Source sign shines at ENTW
ok 7 - Catalog sign shines at ENTW
ok 8 - Source sign shines at QS
ok 9 - Catalog sign shines at QS
ok 10 - Source sign shines at PROD
ok 11 - Catalog sign shines at PROD
ok 12 - Source sign shines at SVNENTW
ok 13 - Catalog sign is empty at SVNENTW
# Looks like everything went fine.
ao-suse-ws1:/srv/www/symfony/natural #
  
```

Figure 9: Aufruf des Testfalls auf der Konsole

A.12 Klasse: ComparedNaturalModuleInformation

Kommentare und simple Getter/Setter werden nicht angezeigt.

```

1 <?php
2 class ComparedNaturalModuleInformation
3 {
4     const EMPTY_SIGN = 0;
5     const SIGN_OK = 1;
6     const SIGN_NEXT_STEP = 2;
7     const SIGN_CREATE = 3;
8     const SIGN_CREATE_AND_NEXT_STEP = 4;
9     const SIGN_ERROR = 5;
10
11     private $naturalModuleInformations = array();
12
13     public static function environments()
14     {
15         return array("ENTW", "SVNENTW", "QS", "PROD");
16     }
17
18     public static function signOrder()
19     {
20         return array(self::SIGN_ERROR, self::SIGN_NEXT_STEP, self::SIGN_CREATE_AND_NEXT_STEP, self::SIGN_CREATE, self::SIGN_OK);
21     }
22
23     public function __construct(array $naturalInformations)
24     {
25         $this->allocateModulesToEnvironments($naturalInformations);
  
```

```

26     $this->allocateEmptyModulesToMissingEnvironments();
27     $this->determineSourceSignsForAllEnvironments();
28 }
29
30 private function allocateModulesToEnvironments(array $naturalInformations)
31 {
32     foreach ($naturalInformations as $naturalInformation)
33     {
34         $env = $naturalInformation->getEnvironmentName();
35         if (in_array($env, self::environments()))
36         {
37             $this->naturalModuleInformations[array_search($env, self::environments())] = $naturalInformation;
38         }
39     }
40 }
41
42 private function allocateEmptyModulesToMissingEnvironments()
43 {
44     if (array_key_exists(0, $this->naturalModuleInformations))
45     {
46         $this->naturalModuleInformations[0]->setSourceSign(self::SIGN_OK);
47     }
48
49     for ($i = 0; $i < count(self::environments()); $i++)
50     {
51         if (!array_key_exists($i, $this->naturalModuleInformations))
52         {
53             $environments = self::environments();
54             $this->naturalModuleInformations[$i] = new EmptyNaturalModuleInformation($environments[$i]);
55             $this->naturalModuleInformations[$i]->setSourceSign(self::SIGN_CREATE);
56         }
57     }
58 }
59
60 public function determineSourceSignsForAllEnvironments()
61 {
62     for ($i = 1; $i < count(self::environments()); $i++)
63     {
64         $currentInformation = $this->naturalModuleInformations[$i];
65         $previousInformation = $this->naturalModuleInformations[$i - 1];
66         if ($currentInformation->getSourceSign() <> self::SIGN_CREATE)
67         {
68             if ($previousInformation->getSourceSign() <> self::SIGN_CREATE)
69             {
70                 if ($currentInformation->getHash() <> $previousInformation->getHash())
71                 {
72                     if ($currentInformation->getSourceDate('YmdHis') > $previousInformation->getSourceDate('YmdHis'))
73                     {
74                         $currentInformation->setSourceSign(self::SIGN_ERROR);
75                     }
76                 }
77             }
78         }
79     }
80 }

```

```

76         else
77         {
78             $currentInformation->setSourceSign(self::SIGN_NEXT_STEP);
79         }
80     }
81     else
82     {
83         $currentInformation->setSourceSign(self::SIGN_OK);
84     }
85 }
86 else
87 {
88     $currentInformation->setSourceSign(self::SIGN_ERROR);
89 }
90 }
91 elseif ($previousInformation->getSourceSign() <> self::SIGN_CREATE && $previousInformation->
    getSourceSign() <> self::SIGN_CREATE_AND_NEXT_STEP)
92 {
93     $currentInformation->setSourceSign(self::SIGN_CREATE_AND_NEXT_STEP);
94 }
95 }
96 }
97
98 private function containsSourceSign($sign)
99 {
100     foreach($this->naturalModuleInformations as $information)
101     {
102         if ($information->getSourceSign() == $sign)
103         {
104             return true;
105         }
106     }
107     return false ;
108 }
109
110 private function containsCatalogSign($sign)
111 {
112     foreach($this->naturalModuleInformations as $information)
113     {
114         if ($information->getCatalogSign() == $sign)
115         {
116             return true;
117         }
118     }
119     return false ;
120 }
121 }
122 ?>

```

Listing 2: Klasse: ComparedNaturalModuleInformation

A.13 Klassendiagramm

Klassendiagramme und weitere UML-Diagramme kann man auch direkt mit \LaTeX zeichnen, siehe z. B. <http://metauml.sourceforge.net/old/class-diagram.html>.

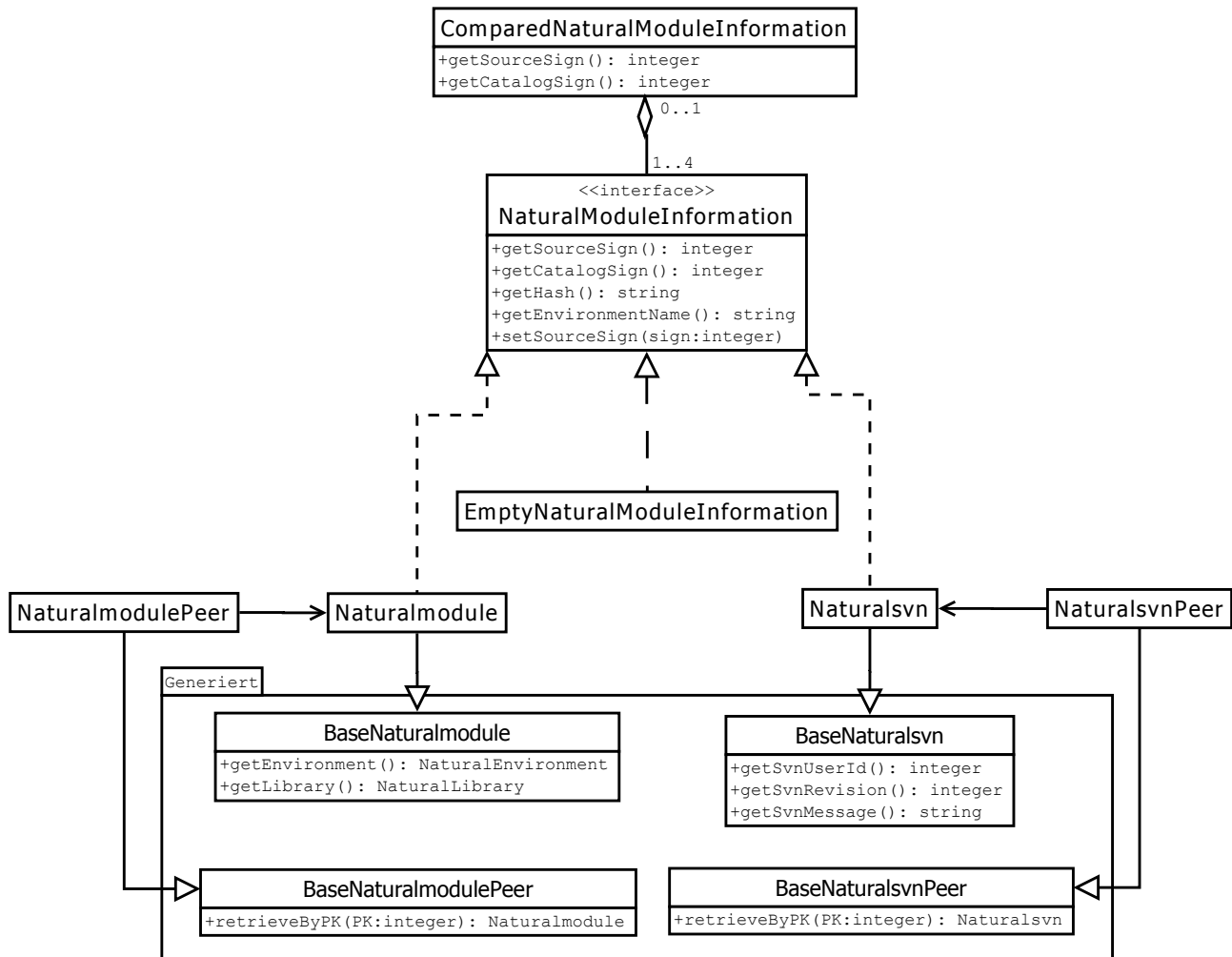







Figure 10: Klassendiagramm

A.14 Benutzerdokumentation

Ausschnitt aus der Benutzerdokumentation:

Symbol	Bedeutung global	Bedeutung einzeln
	Alle Module weisen den gleichen Stand auf.	Das Modul ist auf dem gleichen Stand wie das Modul auf der vorherigen Umgebung.
	Es existieren keine Module (fachlich nicht möglich).	Weder auf der aktuellen noch auf der vorherigen Umgebung sind Module angelegt. Es kann also auch nichts übertragen werden.
	Ein Modul muss durch das Übertragen von der vorherigen Umgebung erstellt werden.	Das Modul der vorherigen Umgebung kann übertragen werden, auf dieser Umgebung ist noch kein Modul vorhanden.
	Auf einer vorherigen Umgebung gibt es ein Modul, welches übertragen werden kann, um das nächste zu aktualisieren.	Das Modul der vorherigen Umgebung kann übertragen werden um dieses zu aktualisieren.
	Ein Modul auf einer Umgebung wurde entgegen des Entwicklungsprozesses gespeichert.	Das aktuelle Modul ist neuer als das Modul auf der vorherigen Umgebung oder die vorherige Umgebung wurde übersprungen.