Deckblatt

Inhaltsverzeichnis

1.Einleitung

Die Agentur, in der ich meine Ausbildung mache, ist die Open New Media GmbH (ONM). ONM ist in der

Webentwicklung tätig, wobei wir Konzeption, Design sowie Realisation übernehmen, um ganzheitliche Lösungen für unsere Kunden zu schaffen. Es wird großer Wert auf Individualisierung gelegt, um verschiedenste Kundenanforderungen zu erfüllen.

Die Agentur hat ihren Hauptsitz in Koblenz und hat insgesamt 18 Mitarbeiter. Den Großteil unseres Kundenstamms macht die Hotellerie aus. Hier bieten wir verschiedene individualisierbare Produkte an, mit denen u.a., Prozesse im Hotel digitalisiert und optimiert werden können. Diese werden von uns als Ergänzung zur bestehenden Softwarelandschaft der Kunden entwickelt.

Ein weiteres Aufgabengebiet der Agentur ist die individuelle Entwicklung und Betreuung von Projekten auf Basis verschiedener Content-Management-Systeme.

* 1. Projektbeschreibung

Um die Rekrutierung neuer Mitarbeiter zu optimieren, soll eine individuelle

Bewerberplattform für ONM entwickelt werden, welche im Nachgang auf der TYPO3-Webseite der Agentur verlinkt werden soll, um die bestehende Lösung zu ersetzten. Die Daten sollen über ein Headless CMS bezogen werden.

Eines der Hauptziele dieser Bewerberseite ist es, die Agentur als attraktiven Arbeitgeber darzustellen, ONM optimal zu präsentieren und potenziellen Bewerbern einen ersten Eindruck zu gewähren.

Die Unternehmensdarstellung soll entweder auf 2-5 Seiten oder auf einer Seite, als „Onepager“, erfolgen. Es sollen mehrere Seitenabschnitte erstellt werden, in denen jeweils Texte, Bilder und Videos integriert werden können. Es soll fest vorgegebene Eingabefelder für Überschriften, Texte und Medien je Seitenabschnitt geben. Diese Seitenabschnitte sollen über das Headless CMS zu verwalten sein.

Zudem soll es eine Übersicht über die verfügbaren Stellenangebote geben. Diese sollen über das

Headless CMS angelegt und verwaltet werden. Hier sollen pro Stellenangebot jeweils Titel,

Beschäftigungsart, Beginn der Beschäftigung, die Aufgaben des Beschäftigten sowie die Erwartungen der Agentur an den Beschäftigten gepflegt werden. Folgende Felder sollen bei allen Stellenangeboten angezeigt werden: Unternehmenskurzprofil, „Unser Angebot“ und „Sonstige Annehmlichkeiten bei ONM“.

Auf der Plattform sollen die Stellenangebote nach Beschäftigungsart gruppiert (z.B. Festanstellung/Ausbildung) aufgelistet werden.

Interessenten sollen auch die Möglichkeit haben, sich direkt über ein Formular bei uns zu bewerben. Die Formulare beinhalten wichtige Felder zur Person, eine Upload-Funktion für Bewerbungsunterlagen und jobspezifische Kenntnisabfragen. Das Bewerbungsformular sowie die hochgeladenen Dateien werden anschließend automatisch per E-Mail an die Personalverantwortliche versendet.

Die Daten sollen im headless CMS eingepflegt, und von einer Vue.js Applikation über eine API im JSON Format ausgelesen werden. Die Applikation muss die Daten von den jeweiligen Endpunkten der API auslesen, Daten verarbeiten und daraus eine strukturierte GUI erstellen.

* 1. Projektziel

Das Ziel ist das Erstellen einer Bewerberplattform, die sich aus einem Headless CMS, in dem Daten gefpflegt werden, und einer Vue.js Applikation, die die Daten Dynamisch ausließt und daraus ein Front-End erstellt zusammensetzt. Das Ergeniss dient zur Evaluation des Einsatztes von Headless CMS in kleineren bis mittelgroßen projekten. [zwei ziele...]

* 1. Projektbegründung

Aktuell sind zahlreiche Kundenprojekte in TYPO3 oder WordPress umgesetzt. Besonders hoch individualisierte TYPO3-Instanzen, können bei Aktualisierungen und Updates enorme Aufwände verursachen. Auf der Kundenseite entstehen somit hohe Kosten ohne einen offen ersichtlichen Mehrwert. Damit die Kundenzufriedenheit hierunter nicht dauerhaft leidet, wurde über Alternativen zu diesen Systemen diskutiert.

Ergebnis dieser Diskussion war, dass wir künftig auf Headless Content-Management-Systeme (CMS) setzen wollen, um kleinere Projekte, deren Anspruch nicht einer TYPO3- oder WordPress-Instanz entspricht, als schlanke und leicht zu aktualisierende Lösung zu realisieren.

Mein Projekt dient, als das erstes Projekt das auf Basis eines Headless CMS umgesetzt werden soll, als Pilot, um Erfahrung im Umgang mit dieser Lösung sammeln, und diese für zukünftige Kundenprojekte evaluieren zu können.

Aktuell gibt es auf der Typo3 Seite von ONM Bereits die Möglichkeit sich über Formulare zu bewerben. Der Aufbau der Seite lässt allerdings die Firmendarstellung nur eingeschränkt zu, und eine Erweiterung würde einen großen Aufwand bedeuten. Zudem sind Daten teilweise nur über Umwege verfügbar, und der Mailversand hat in der Vergangenheit nicht immer zuverlässig funktioniert.

* 1. Projektschnittstellen

Um die im Strapi Backend Eingepflegten Daten bereit zu stellen, wird von Strapi ein API-Endpunkt bereitgestellt, dieser liefert Daten im JSON Format. Die Applikation kann nun über die Abfrage diese Endpunktes die Informationen auslesen, und ein Frontend daraus erstellen.

[Schnittstelle zwischen vue und laravel? Strapi und db?]

Enge Schnittstellen stellen auch die Betriebsleitung als Auftragssgeber, die Personalverantworliche sowie die Redakteurin dar, welche über Agile Methoden in den Entwicklungsprozess integriert werden sollen. Bei Fragen/Problemstellungen kann auf Kollegen und Ausbilder zugegangen werden.

* 1. Projektabgrenzung

Es wird noch nicht das fertige Endprodukt, sonder die basis für weitere iterationen geschaffen.

[Nachlesen]2.Analyse - größten teils im rahmen des Projektantrages durchgführt 2.1 Ist-Analyse2.3 Wirtschaftlichkeitsanalyse

1.6 Abweichungen zum Projekt-Antrag

3. Projektplaung

3.1 Resourcen Planung

Im Anhang [\*erstellen] befindet sich eine Auflistung aller Ressourcen, die zum Entwickeln der Plattform in dem von der IHK vorgegebenen Zeitraum von 80h verwendet werden. Bei der Auswahl der Verwendeten Software wurde auf Open Source Lizensen [\*expl] gestetzt, weshalb in diesem Bereich keine weiteren Kosten entstehen.

3.2 Entwicklungsprozess

Zum entwicklen der Platfform wurde sich für ein Iteratives Vorgehen entschieden, dass an scrum orientiert ist. Es werden zunächst die allgemeinen Anforderungen an die Plattform definert. Darauf hin werden Schritte für einen Arbeitszyklus (angelehnt an SCRUM-Sprint) geplant, der Teile der Anforderungen erfüllt. Die Ergebnisse werden dann in einer Iteration des Projektes vorgestellt und besprochen. Hierauf hin wird aus dem Feedback und den eventuell Angepassten allgemeinen Anforderungen (product-backlog) der nächste Arbeitszyklus geplant.[\*screenshots]

Ein Agiler Ansatz ist für dieses Projekt besonders gut geignet, da intern noch keine Erfahrung mit Headless CMS beteht, und so deshalb wichtig ist auf neue Herausforderungen und Erkenntinsse zu reagieren.

Dass bislang auch nur die mindest Anforderungen an die Plattform definiert sind, und diese in der Zukunft noch wachsen werden, wird ebenfalls durch eine Agile Vorgehensweise unterstützt. Hier können somit auch Erfahrungen bezüglich der Erweiterbarkeit von Headless CMS gesammelt werden.

Ein Iteratives Vorgehen wurde gewählt, da durch die iterationen viel Feedback und Rücksprache gehalten werden kann, was den Agilen Prozess unterstützt. Des weiteren kann mit hoher Wahrscheinlichkeit innerhalb der von der IHK vorgegebenen 80 Stunden noch keine vollwertige Plattform, die dem Firmenstandard entspricht, erstellt werden. Das Ergebniss des Projektes soll quais als Basis für die Weiterentwicklung/Ausarbeitung dienen.

4 Anstoßsitzung

Hier wird ein Meeting mit allen beteiligten (Geschäftsführung, Redakteurin, Personalverantwortliche, Entwickler)gehalten, wobei allgemeine Ideen, Vorstellungen, Inhalte sowie Anforderungen besprochen und diskutiert werden. Die Ergebnisse werden in einem Produkt-Backlog [\*erstellen] (angelehnt an SCRUM) festgehalten. Der Produkt-Backlog dient als Sammlung aller Anforderungen, die die Platform erfüllen soll. Auch wurde mir eine Sammlung von Texten/Inhalten für die einzelnen Seiten übergeben.

5 Entwurfsphase

5.1 Auswahl von Headless CMS

Als zu verwendendes CMS habe ich mich für Strapi entschieden. Es erfüllt alle Anforderungen, befindet sich unter einer Open-Source-Linzenz [3.1 Resourcen Planung], kann selbst gehostet werden, was ein großer Vorteil im Bezug auf DSGVO-Konformität und Unabhängigkeit von Anbietern mit sich bringt, bietet ein Verwatlungstool für hochgeladene Medien und eine Intuitive Benutzeroberfläch für Redakteure. Zudem werden ausschließlich im JSON-Format Bereitgestellt, was sich gut mit Vue.JS kombinieren lässt, da Objekte hier in diesem Format besonders leicht verarbeitet werden können.

5.2 Auswahl von Formular Tool

Zunächst habe ich die Vorschläge der Geschäftsführung geprüft, und mich nach einer Internetrecherche für die Integration von „Formkit“, einem Framework zum Erstellen von Formularen in Vue.JS, entschieden. Es kann als Node-Package installiert und in der vue applikation integriert werden. Es deckt alle nötigen Formular-Felder in seiner Funktionalität ab, bietet Multi-Step-Formulare [\*expl], und hat eine ansprechende Funktionsweise.

5.3 Auswahl von Tool Mailverand über SMTP

Nach einer Internet Recherche habe ich die JavaScript-Library „SMTPJS“ gefunden, und wollte diese für den Mailversand verwenden. Es scheint als einfach zu implementierende Lösung um den Mailversand aus der Vue.JS Applikation abzuwickeln.

5.3 Entwerfen von Seitenstruktur & Elementen

Im Anhang [\*erstellen] befindet sich ein erster Entwurf für eine mögliche Seitenstruktur inform eines Baumdiagrammes.

Auf basis der vorhanden Informationen aus dem Anstoßmeeting, habe ich nun grobe Entwürfe für Inhalts- und Seitenelemente angefertigt und Gestaltet, diese befinden sich im Anhang [\*erstellen].

6 Implentierung

6.1 Installation und Grundeinrichtung des Headless CMS

Die Installation kann wie in der Dokumentation [\*add quelle?] beschrieben mit dem Kommando „npx create-strapi-app@latest projekt-name“ in einem CLI [\*explain] ausgeführt werden. Dies erzeugt eine Strapi Instanz mit dem Namen „projekt-name“.

Um die Anwendung zu starten und local zu hosten, kann man nun den Befehl „npm run develop“ in dem CLI ausführen. Die von diesem Befehl erzeugte Ausgabe wird in Abbildung [\*create] dargestellt.

[Abbildung]

Die Applikation ist nun im Browser lokal unter der in dem CLI ausgegeben andresse erreichbar. Zunächst muss man einen Admin-Nutzer erstellen, mit dem man sich authentifizieren kann. Als nächstes hat man zugriff auf das Backend der Applikation, von dessen Startseite sich ein Screenshot im Anhang [\*create] befindet.

**6.1.2 Erstellen und Abfragen von Teststrukturen**

Als nächstes habe ich Teststrukturen angelegt, welche aus Texten und Bildern bestanden. Dieser Prozess ist in Anhang [\*create] dargestellt. Hier habe ich Dummy-Daten[\*explain] eingepflegt, um diese Testweise Testweise auszulesen. Hierfür müssen über ein Authorisierungs-Plugin von Strapi die Berechtigungen für die Api Endpunkte für die Jeweiligen Typen gesetzt werden. Dies ist in Abbildung [\*create] veranschaulicht:

Abbildung [\*create]

Nun können entsprechende Daten Abgefragt werden, wie in Abbildung [\*create] zu sehen ist.

Abbildung [\*create]

6.2 Erstellen einer Vue.JS Applikation

6.2.1 Installation einer Vue.JS Applikation

Wie in der Dokumentation [\*add quelle?] beschrieben kann die Installation mit dem Kommando „npm init vue@latest“ in einem CLI ausgeführt werden. Nun wird in dem CLI der Installationsguide ausgeführt, in dem unter anderem Name und Grundeinstellungen des Projektes Konfiguriert werden. Dies erzeugt einen Ordner, der die Vue.js Applikation enthählt.

Als nächstes wird mit dem CLI aus dem durch den Befehl erstellten Ordner den befehl „npm install“ ausführen, um Abhängigkeiten zu Installieren. Um das Projekt nun lokal zu hosten startet man das Projekt im CLI über den Befehl „npm run dev“. Die Ausgabe des Befehls im CLI ist im abbildung [\*create] zu sehen.

Abbildung [create]

Die Applikation ist nun im Browser unter der im CLI ausgegebenen Adresse erreichbar.

6.2.2 Grundeinrichtung Vue.Js Applikation

Zunächst wurden Views für die Seiten Angelegt. Daraufhin wurde der Vue-Router zunächst Statisch eingerichtet, und die enstprechenden Views Über eine Statische Navigation verlinkt.

6.2.3 Schnittstellen zu Headless CMS einrchten

Um Daten von den Endpunkten des Headless CMS abzufragen, wurde sich für die verwendung von „axios“ entschieden. Dieses Node Modul fungiert als HTTP Client[\*expl], mit dem HTTP Reqeuests aus der App heraus getätigt werden können. Eine Beispiel Anfrage und mit Anwort ist in Abbildung [\*create] zu sehen.

Abbildung [\*create]

6.2.3 Erstellen von Inhaltstypen in CMS

Im nächsten Schritt wurden dann die Grundinhaltstypen auf Basis der Entwurfenen Elemente (siehe 6.3) als Inhaltstypen Erstellt. Im Anhang [\*create] wird dieser Prozess beispielhaft durchgeführt. Um diese Inhalstelmente zu verwenden, wurde ein Sammel-Typ „Seite“ angelegt, in dem die Erstellten Inhaltselemente in einer Dynamischen-Zone [\*expl] platztiert werden können. [\*populate deep]

Nun kann man das Seiten-Element über einen über den entsprechenden Endpunkt mit der Seiten ID abgefragt werden. Eine Beispiel aus demBackend Seite mit einem Inhaltselement vom Typ „Überschrift“ sowie einem Inhaltselement vom Typ „Text und Bild“ ist im Abbildung [\*create] zu sehen, die Antwort auf die Anfrage des API-Endpunktes ist in Abbildung [\*create] zu sehen

[\*create Abbildungen]

6.2.3 Auslesen der Seiten-Elemente und Vorbereitung der Daten

Nun wurden innerhalb der Vue.js Applikation die Daten über „Axios“ promise-based [\*expl] abgefragt, die Inhalstelement verarbeitet und die Daten zu einem erstellen Array in dem vue Data Object [\*expl] hinzugefügt. Um die Inhalte nun auszugen wurde eine Vue-for-Loop mit Conditionen verwendet:

[CODE]

So wird für jedes Inhaltselement die entsprechende Vue-Komponente gerendert, und die Daten können über die „Slot“-Tags zugeorned, oder als paramter für Konditionelle zwecke genutzt werden.

6.2.3 Erstelleung von Vue-Komponenten

Nun wurde für die Inhaltstypen aus dem Headless CMS [siehe 5.2.3] in der Vue.js Applikation Komponenten angelegt, die für die Darstellung der Daten als Inhaltselemente im Frontend der App verantwortlich sind. Als Beispiel hierfür befindet sich ein Code-Auszug für eine Komponente für den Inhaltstyp „Überschrift“ im Anhang [\*create].

6.2.2 Integration von Formtool

Das Ausgewählte Framework für die Formulare [siehe nicht 3.3] kann als node package mit dem Befehl „npm install @formkit/vue“ installiert werden.

Es wurden des weiteren ein Formkit-Addon names „FormkitMultiStep“ für Multi-Step-Formulare [\*explain] und sowie ein Formkit -Addon mit Themes [\*explain] mit dem Namen „themes“ für ein Design des Formulares hinzugefügt.

Um es in der Vue Applikation verfügbar zu machen, wurde es nun noch im Startpunkt der Vue Applikation integriert, wie in Anhang EinstiegspunktVueApplikation [\*create] zu sehen ist.

Eine simples Beipielhaftes Formular mit Codeauszug neben der Frontend-Ausgabe ist in in Abbildung [\*create] verdeutlicht [einzeichnen step one, darunte step two, parallel zu code].

6.3.3 Mailversand über SMTP (\*@bop)

Nachdem ein simples Testformular angelegt war, war es nun an der Zeit die Daten aus dem Formular zu versenden. Hierbei konnte leider nicht wie zunächst geplant vorgegangen werden, herauf wird unter 6.3.3.1 genauer eingegangen.

**6.3.3.1 Versand aus Vue.JS Applikation (problem)**

-Test von Node Plugin zum Mailversand (ergebniss: Versand nur über einen Bestimmten Anbieter von SMTP-Servern möglich)

-Suche nach alternativen -> Einbindung der Vue js Applikation in PHP Projekt

**5.3.3.2 Auswahl und Einrichtung von PHP Applikation um app zu hosten (Lösung)**

-Grundeinrichtung Laravel Projekt (es besteht interne expertise, habe bereits erste positive erfahrungen in schulprojekt machen können)

-Route zu Vue.JS Applikation hinzufügen

**5.3.3.3 Abwickeln des Mailversands**

-Route für Mailverand hinzufügen, controller erstellen um request zu handeln (PAP?)

-Erstellen von Mailable klasse in laravel (Klassendiagramm?)

-senden von emails an test SMTP

-Erweiterung von Testformular um Datei feld

-Erweiterung des Controllers und der Mailable klasse zum Handlen einer Datei

(diagramm?)

-Erstellen von Test-Validatoren in Laravel, erweiterung des Controllers zum validieren

-Problem mit Fehlerausgabe (unschöne kommunikation zwischen JS/PHP)

-Erstellen von Test-Validatoren in für Formkit (Vue.JS) 🡪validierung mit eigenen Bedingungen/Fehlerausgaben

-...

5.3.4 Umsetzung von Grobem Design

-Erstellung grobes Design für elemente

-Erstellung grobes Desing für Header/footer

5.3.5 Routing

-Dynamisches auslesen von Seiten um Router dynamisch agieren zu lassen

-Navigation dynamisch erweitern (code, be, vorher nacher screenshot?)

-Relationen zwischen inhaltselementen und Jobanzeigen herstellen und verarbeiten

-Relationen zwischen Jobanzeigen und Seiten herstellen

-Integration der Ausgelesenen Daten pro seite um Jobanzeigen inhalte

-router diagramm?

....

6? Abnahme/Deploiment QS

7 Dokumentation

8 Fazit