Deckblatt

Inhaltsverzeichnis

1.Ausgangssituation

1.1 Analyse des Projektauftrags (ist analyse, siehe projektnatrag) [1.1 Projektbeschreibung]

1.2 Ableitung der Projektziele und Projektabgenzung [1.2/1.3/1.4]

1.3 Darstellung des Projektumfeldes und der betrieblichen schnittstellen [1.5]

2.Resourcen/-Ablaufplan

2.1 Projektmanagement, Terminplanung, Ablaufplan (inkl. Meilensteine) [2.1/2.2]

2.2 Plaunung der Qualitätssicherung (projektbezogen und technisch) [2.3]

(Soll Zustand)

3. Durchführung und Auftagsbeschreibung

3.1 Prozess-Schnitte und vorgehensweise, was habe ich gemacht? [6]

( Progarmmablaufplan, ER-Diagramme usw.)

3.2 Abweichunge Anpassung, Entscheidungen [6.7]

3.3 Maßnhmen zur Qualitätskontrolle (Projektbez. Und tech.)

4.Projektabchluss

4.1 Soll-ist-Vergleich (Abweichung, Anpassungen)

4.2 Reflexion/Fazit

4.3 Optimierungsmöglickeiten/Ausblick

1.Einleitung 1.Ausgangssituation

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 1.1 Analyse des Projektauftrags

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.

1.2 Projektbegründung 1.2 Ableitung der Projektziele

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.

Auch die Webseite der Agentur ist in Typo3 umgesetzt, und bietet die Möglichkeit, sich über Formulare zu Bewerben. Möglichkeiten zur Firmendarstellung sind hier nur bedingt gegeben, der Mailversand funktioniert nur unzuverlässig und die Gestaltung der Nutzeroberfläche entspricht nicht dem Firmenstandard. Eine Erweiterung wäre sehr zeitaufwendig, zudem ist ein Relaunch der Webseite in Sprache.

Da mein Unternehmen hohen Wert auf Rekrutierung neuer Mitarbeiter legt, soll in meinem Projekt eine eigenständige Bewerberplattform erstellt werden, die im Nachgang auf der Webseite verlinkt wird, um die Bestehende Lösung zu ersetzten.

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.

1.3 Projektziel 1.2 Ableitung der Projektziele

Da die auf der Typo-3-Webseite der Agentur bestehende Lösung zur Rekrutierung neuer Mitarbeiter zum einen nur einen unzuverlässigen Mailversand, durch den Bewerbungen teilweise verloren gehen, sowie keine ansprechende Nutzer-Oberfläche bietet, und eine Erweiterung der bestehenden-Typo3 Instanz sehr Zeitaufwendig wäre, ist das Ziel zum einem 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. Diese Lösung soll im Nachgang die bestehende Lösung zur Rekrutierung von Mitarbeitern über die Homepage erstetzten. Des weiteren besteht das Ziel einer Evaluation des Einsatztes von Headless CMS in kleineren bis mittelgroßen Projekten, die auf der Auswertung der Ergebnisse basiert. Dies könnte uns langfristig ermöglichen die Kundenzufriedenheit dadurch zu erhöhen, dass weniger Kosten bei Updates entstehen.

1.4 Projektabgrenzung Projektabgenzung

Es wird noch nicht das fertige Endprodukt, sonder die Basis für weitere Iterationen geschaffen. Ein Deployment wird in der Projektphase noch nicht stattfinden.(Abgrenzung deployment?)

1.5 Projektumfeld 1.3 Darstellung des Projektumfeldes

1.5.1 Projektschnittstellen

Um die im Strapi Backend Eingepflegten Daten bereitzustellen, 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?]

1.3 der betrieblichen schnittstellen

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.

1.5.2 Entwicklungsumgebung

Zum Entwicklen wurde ein Desktop-PC Verwendet, die Spezifikationen der Hardware befinden sich im Tabelle [erstellen]. Die zum entwickeln verwendeten Software ist in Tabelle [erstellen] aufgeführt.

1.6 Abweichungen zum Projekt-Antrag

2. Projektplanung 2Resourcen/-Ablaufplan

2.1 Resourcen Planung 2.1 Projektmanagement, Terminplanung, Ablaufplan (

Im Anhang [\*impr] 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 Lizenzen [\*fn:Lizenz mit der Software kostenlos genutzt, verändert und vertrieben werden kann] gestetzt, weshalb in diesem Bereich keine weiteren Kosten entstehen.

2.2 Entwicklungsprozess 22.1 Projektmanagement, Terminplanung, Ablaufplan

Zum entwicklen der Platfform wurde sich für ein Inkrementelles Vorgehen entschieden, dass an Scrum orientiert ist. Es werden zunächst die allgemeinen Anforderungen an die Plattform in einem Anstoß-Meeting definert und einem Dokument, angelehnt an den Produkt-Backlog von Scrum, festgehalten. Darauf hin werden Schritte für einen Arbeitszyklus (angelehnt an einen SCRUM-Sprint) geplant, der als erster Zyklus den Großteil der Anforderungen der nach Priorität geordnet abarbeitet. Die Ergebnisse werden dann in einer Iteration des Projektes in einem weiteren Meeting vorgestellt und besprochen. Hierauf hin wird aus dem Feedback und den eventuell Angepassten allgemeinen Anforderungen, die im Product Backlog ergänzt werden können, der nächste Arbeitszyklus geplant. Dieser enhält Anpassungen von bestehenden Elementen (iterativer Charakter, besonders im Beriech der Gestaltung) sowie neue Anforderungen (inkrementell). Für den Bearbeitungszeitrum wurden insgesamt drei Meetings eingeplant. Die Zeitplanung wurde in einem Gantt-Diagramm visualisert, dieses befindet sich im Anhang [\*create Gantt Diagramm Projektplanung].

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

Dass bislang noch nicht alle Anforderungen an die Plattform definiert sind, und diese wahrscheinlich in der Zukunft noch wachsen werden, wird ebenfalls durch eine inkrementelle/agile Vorgehensweise unterstützt.

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 als Basis für zukünftige Iterationen (Weiterentwicklung/Ausarbeitung der Plattform in Zyklen) dienen.

2.2 Planung von Maßnahmen zur Qualitätssicherung Plaunung der Qualitätssicherung (projektbezogen und technisch) [2.3]

Die Qualität soll zum einen Software-seitig durch Tests, als auch durch Code-Reviews durch die Leitung der Anwendungsentwicklung sowie interne Experten in Vue.JS gesichert werden. Zu dem soll ein Gitlab Repository angelegt werden, das zur Versionskontrolle dient.

2.3 Anstoß-Meeting [2.3 Soll zustand]

Hier wurde ein Meeting mit allen beteiligten (Geschäftsführung, Redakteurin, Personalverantwortliche, und dem Entwickler) gehalten, wobei allgemeine Ideen, Vorstellungen, Inhalte sowie Anforderungen besprochen und diskutiert wurden. Die Ergebnisse wurden in einem Dokument festgehalten, dieses befindet sich im Anhang [\*impr ProduktBacklog] . Der Produkt-Backlog dient als Sammlung aller Anforderungen, die die Platform erfüllen soll, und ist nach Priorität sortiert. Auch wurde mir eine Sammlung von Texten/Inhalten für die einzelnen Seiten übergeben.

2.6 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 [2.1 Resourcen Planung], kann selbst gehostet werden, was ein großer Vorteil im Bezug auf DSGVO-Konformität und Unabhängigkeit von Anbietern mit sich bringt, bietet ein Verwaltungstool für hochgeladene Medien und eine Intuitive Benutzeroberfläch für Redakteure. Zudem werden Daten ausschließlich im JSON-Format Bereitgestellt, was sich gut mit Vue.js kombinieren lässt, da Objekte in Vue/JavaScript in diesem Format standardmäßig verarbeitet werden.

2.7 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 [\*fn: Formulare, die in meherer Schritte aufgeteilt sind (wie Teil-Formulare)], und hat eine ansprechende Funktionsweise. Zudem ist es möglich eigene Formular-Feld-Validatoren [\*fn:Methodik um Eingaben der Nutzer auf Plausibilität/Korrektheit zu prüfen] zu erstellen.

2.8 Auswahl von Tool für Mailverand über SMTP

Nach einer Internet Recherche habe ich die JavaScript-Library „SMTPJS“ gefunden, und mich dazu entschieden, diese als als einfach zu implementierende Lösung für den Mailversand aus der Vue.js Applikation zu nutzen.

2.9 Entwerfen von Seitenstruktur & Elementen

Im Anhang [\*erstellen] befindet sich ein Entwurf für eine erste simple Seitenstruktur, bei welcher jede Seite als direkte Unterseite der Startseite angelegt wird ,in Form eines Baumdiagrammes.

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

3 Implentierung 3. Durchführung und Auftagsbeschreibung, 3.1 Prozess-Schnitte und vorgehensweise, was habe ich gemacht? [6]

3.1 Installation und Grundeinrichtung des Headless CMS

Die Installation wird wie in der Dokumentation beschrieben mit dem Kommando „npx create-strapi-app@latest strapiJobs“ in einem CLI [\*fn: Command Line Interface, beschreibt die Schnittstelle zwischen Mensch und Computer Befehle in Textform einzugeben] ausgeführt. Dies erzeugt eine Strapi Instanz mit dem Namen „strapiJobs“.

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.

3.2 Erstellen einer Vue.JS Applikation

3.2.1 Installation einer Vue.JS Applikation

Wie in der Dokumentation 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.

3.2.2 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. Dies diente lediglich als Temporärer Lösung.

3.2.3 Schnittstellen zu Headless CMS einrichten

Um Daten von den Endpunkten des Headless CMS abzufragen, wurde sich für die verwendung von „Axios“ entschieden. Dieses Node Modul fungiert als HTTP Client[\*fn:versendet Http Anfragen und nimmt Antworten entgegen (bsp.Web-Browser)], mit dem HTTP Reqeuests aus der App heraus getätigt werden können. Eine Beispiel Anfrage sowie die Anwort in einer Browser-Console [\*expl?]ist in Abbildung [\*create] zu sehen.

Abbildung [\*create]

3.2.4 Erstellen von Inhaltstypen in CMS

Als nächstes wurden Teststrukturen angelegt, welche aus Texten und Bildern bestanden. Dieser Prozess ist in Anhang [\*create] dargestellt. Um diese Teststrukturen abzufragen, muss ein Sammel-Typ[\*fn:Behälter indem erstelle und native Komponenten platziert werden können] erstellt werden. Dieser wurd e unter dem namen „Seite“ angelegt, in dem die erstellten Inhaltselemente in einer dynamischen-Zone [\*fn: Behälter, in dem eine Undefinierte menge von diversen Inhaltselementen in beliebiger Reihenfolge erfasst] platztiert werden können.

Hier habe ich Dummy-Daten[\*fn: Daten ohne bedeutung, nur zum testen] eingepflegt, um diese Testweise auszulesen.

Um Bilder aus den Dynamischen Zonen Auszulesen wurde Strapi mit dem CLI Befehl „npm install strapi-plugin-populate-deep“ um ein Plugin erweitert, das es erlaubt verschachtelte Inhalte aus dynamischen Zonen abzufragen.

Im nächsten Schritt wurden dann die Inhalstypen auf Basis der Entwurfenen Elemente (siehe 6.3) als im Headless CMS erstellt. Im Anhang [\*create] wird dieser Prozess beispielhaft für eine Überschrift durchgeführt.

Nun kann das Seiten-Element über den entsprechenden Endpunkt mit der Seiten ID abgefragt werden. Eine Beispiel aus demBackend von einer 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]

3.2.5 Auslesen der Seiten-Elemente und Vorbereitung der Daten

Nun wurden innerhalb der Vue.js Applikation die Daten über „Axios“ promise-based [\*fn: basiert auf dem Konzept das etwas zu einem Zeitpunkt einen Wert haben wird (nicht von Anfang an hat)] abgefragt, die Inhalstelement verarbeitet und die Daten zu einem von der Vue-Komponente erstellten Datenobjekt vom Typ Array hinzugefügt. Um die Inhalte nun auszugeben wurde ein Vue-for-Loop mit Conditionen verwendet:

[CODE]

So wird später 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.

3.2.6 Erstelleung von Vue-Komponenten

Nun wurde für die Inhaltstypen aus dem Headless CMS [siehe 2.8] 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 aus dem Render-Loop und der entsprechenden Komponente für den Inhaltstyp „Überschrift“ in Abbildung [\*create in Abfrageloop und Vue komponente].

3.2.7 Integration von Formtool

Das Ausgewählte Framework für die Formulare [siehe 2.7] 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 und sowie ein Formkit -Addon mit Themes [\*fn: Zusammen greifendes Grund Design der Komponenten] 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 Dateiname app js[\*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].

3.3 Mailversand über SMTP (\*expl) [Simple Mail Transfer Protocol, Protokoll zum austausch von Emails]

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 [6.3.1 link ] genauer eingegangen.

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

Das node Package „NPMJS“ [5.3] kann in einem CLI mit dem Befehehl „npm install smtpjs“ Installiert werden. Als während der Konfiguration des Plugins Probleme auftraten, wurde nach einer Internetrecherche festgestellt, dass aufgrund von Spam und Missbrauch nur noch Elastic-Email als Service Provider für SMTP-Dienste zugelassen ist. Da zum einen ein SMTP-Server zur Verfügung steht, und zum anderen Software sowie Lizenkosten zu vermeiden sind [3.1] ist diese Option nicht geeinget.

**3.3.2 Versand aus Vue.JS Applikation (Lösung)**

Nach weiterer Internetrecherche und Rücksprache mit Kollgen stelle es sich als die Beste Option heraus, die Vue.js Applikation an ein PHP Backend anzubinden, über das der Mailverand abgewickelt werden kann. Um das PHP Backend umzusetzten wurde sich für das PHP-Framework „Laravel“ entschieden. Hierin besteht intern Expertise und auch ich konnte hiermit in Schulischen-Projekten schon positive Erfahrungen sammeln.

**3.3.3 Grundeinrichtung Laravel Projekt**

Um ein Laravel Projekt zu inititalisieren wird der Befehl „composer create-project laravel/laravel ProjektName“ in einem CLI ausgeführt. Nun kann man in das Verzeichnis wechseln und über ein CLI den Befehl „php arstian serve“ ausführen, um das Projekt local zu hosten. Die Abbildung [\*create] zeigt die Ausgabe des CLI. Das Projekt ist nun über die in der Ausgabe zu sehende URL zu erreichen.

Abbildung [\*create]

**3.3.4 Anbinden von Vue Applikation an Laravel**

Um die Erstellete Vue Applikation an das Laravel Projekt anzubinden wurde nun zunächst das Node Package „@vitesjs/plugin-vue“ über ein CLI installiert. Nun wurde die Datei „Vite.config.js“[\*fn: Konfigurations Datei des Build-Tools] (siehe Anhang [\*create]) um das Plugin und somit eine Vue.js Applikation erweitert.

Hieraufhin habe ich die bereits Erstelle Applikation an die in Laravel entstandene Ordner Struktur angepasst, und in das Laravel Projekt eingefügt.

Als nächstes wurde eine Laravel View erstellt, die einen Container mit einer ID von „App“ enthält. In der Haupt JavaScript Datei [better expr.] wurde nun die Vue.JS Applikation auf den Container mit der ID „App“ gemounted [\*fn: in einen Behälter „eingefügt“].

Im Router der PHP Application wurden nun alle Routen auf die Laravel View geleitet, die den Container mit der ID „App“ enthält, und die somit das Fronted der Seite darstellt.

[screenshots? Router->Appjs -> Vue application]

**3.3.5 Abwickeln des Mailversands**

Zunächst wurde eine Mailable-Klasse mit dem Namen „ApplicationMail.php“ durch das CLI-Kommando „php artisan make:mailable ApplicationMail“ erzeugt. In dieser Klasse werden Betreff, Empfänger, Inhalte sowie Anhäge über eine Laravel-View formatiert zu einem via Email [\*better expr] versendbaren Objekt verarbeietet. Ein Code Ausschnitt von dieser Klasse befindet sich im Anhang [\*create] .

Um den Mailversand durchzuführen wurde nun mit dem CLI-Kommando „php artisan make:controller SendMailController“ ein neue Klasse mit dem Namen „SendMailController“ zu den Controllern des Laravel Projektes hinzugefügt. Diese erweitert die standard „Controller“ Klasse von Laravel, und verwendet die erstellte Mailable-Klasse um mit um ein Mailable-Objekt aus den Eingaben des Nutzers im Formular [ref 4.5 form erstellung] zu erstellen. Ein Codeauzug der Controller-Klasse befindet sich im Anhang[\*create].

Als nächstes wurde noch eine Laravel Route hinzugefügt, die auf den „SendMailController“ deutet, und das „action“-Attribut [\*fn: entscheided darüber, über welche URL beim versenden des Formulares angeprochen wird] des Testformulars [ref formerstellung] wurde auf diese Route gesetzt.

Als nächstest wurde der Mailvesand für einen Test-SMTP Server konfiguriert und darauf getestet.

Nun wurden die Formular felder validiert, und mit Fehlermeldungen je nach validations Ergebniss angepasst über Anagabe der Validations-Regel angepasst, wie in im Abbildung[\*create] zu sehen ist , und ein simpler eigener Valdiator für den Datei-Typ geschrieben. Dieser ist in Abbildung [\*create zu sehen] und wird über die Datei App.js [anhang] als Valdationsregel integeriert.

3.4 Umsetzung von Design

Als nächstes wurde das Styling für die unter [ref erstellung vue komponenten] erstellen Komponenten anhand der Entwürfe unter [erstllung entwurf fuer content], sowie das Design der Seitenelemente umgesetzt. Zu sehen ist das Ergebnis in Anhang [\*create].

Anhang [\*create]

3.5 Routing

Im nächsten Schritt wurde das Routing in der Vue Applikation so angepasst, dass Verschiedenen Seiten nun über die gleiche View in der Vue Applikation dargestellt werden, und eine Dynamische Seiten-Navigation eingesetzt werden konnte.

Um den URL-Paramter von einzelnen Seiten anzupassen wurde im Backend zu dem Seitentyp eine Option für ein wählbares URL-Segment eingefügt, dies ist in Abbildung[\*create] zu sehen.

[\*create]

Im Vue Router wurde die URL um einen Parameter erweitert. Dies ist in Abbildung [\*create] zu sehen.Über diesen Paramter wird dann die ID des Seiten-Objektes über eine in der Vue.js Applikation durch eine Abfrage der Seitenelemente erstelle Map [\*expl] bestimmt. Ein Code-Auszug ist in Abbildung [\*create] zu sehen. Wenn im Backend des Headless CMS kein URL-Segment für eine Seite spezifiert wird, wird der Seiten-Name, welcher Einmalig ist, aus den Attributen der Seite verwendet. Dies wird in einem Codeauszug in Abbildung [\*create] verdeutlicht.

Um die Navigation Dynamisch zu gestalten wurde als nächstest eine Options-Seiten-Objekt erstellt. Hier sollen Elemente/Informationen gepflegt werden, die auf allen Seiten gleich sind (Beispielsweise Logo, Navigation, Footer). Für die Navigation werden Relationen [\*expl] zu bestehenden Seiten als Inhalstelemente verwendet, um diese an die Vue.js Applikation zu übergeben, in welcher daraus Navigations Menüs erstellt werden.

3.7 Abweichungen 3.2 Abweichunge Anpassung, Entscheidungen [6.7]

Wie unter [link] erklärt, wurde sich dazu entschieden die Vue.JS Applikation an ein Laravel Backend anzubinden, um den Mailversand abzuwickeln. [mehr]

3.8 Maßnahmen zur Qualitätskontrolle Maßnhmen zur Qualitätskontrolle (Projektbez. Und tech.)

[Code tests? code review durch Chefentwickler und Vue Experten] wie in planung [verlinken] angesetzt...

Gitlab einrichtung für Versionskontrolle bei Anpassungen und Erweiterungen, um bei Problemen auf vorherigen Stand zurück wechseln zu können

4 Dokumentation

5 Projekt Abschluss 4.Projektabchluss

5.1 soll-ist-vergleich 4.1 Soll-ist-Vergleich (Abweichung, Anpassungen)

In Tabelle ist der Zeitplanung aus dem Projektantrag zu sehen. Diese Wurde um die Spalte „Zeit verwendet“ erweitert.

|  |  |  |
| --- | --- | --- |
| Arbeitsschritt | Zeitangabe Projektantrag | Zeit verwendet |
| **Projektplanung** | | |
| Auswahl Headless CMS | 1 Stunde | 1 Stunde |
| Planung von Seitenstruktur & Einrichtung von Elementen | 3 Stunden | 1.5 Stunden |
| Abstimmung | 1 Stunde | 1Stunde |
| Basisgestaltung | | |
| Erstellung von groben Entwürfen für Elemente | 3 Stunden | 3 Stunden |
| Abstimmung | 1 Stunde | 1 Stunden |
| Einrichten des Headless CMS | | |
| Grundeinrichtung Headless CMS | 3 Stunden | 3 Stunden |
| Einrichtung von Rollen im Headless CMS | 2 Stunden | 1 Stunde |
| Erstellen einer Vue.JS Applikation | | |
| Grundeinrichtung Vue.js Application | 2 Stunden | 1 Stunde |
| Umsetzung der Seitenstruktur | 4 Stunden | 4 Stunden |
| Schnittstellen zu Headless CMS einrichten | 6 Stunden | 5 Stunden |
| Erstellen von Inhaltstypen im Headless CMS | 8 Stunden | 8 Stunden |
| Erstellung von Objekten aus Daten von Headless CMS | 8 Stunden | 8 Stunden |
| Darstellung der erstellten Objekte | 8 Stunden | 10 Stunden |
| Anbindung Applikation an Laravel | 0 | 2 Stunden |
| Mailversand über SMTP | 0 | 3 Stunden |
| Test | | |
| Funktionstests | 5 Stunden |  |
| Modultests | 5 Stunden |  |
| Integrationstests | 5 Stunden |  |
| Dokumentation | | |
| Projektdokumentation | 10 Stunden | 12 Stunden |
| Dokumentation für Entwickler | 3 Stunden | 1 Stunde |
| Kundendokumentation | 2 Stunden | 2 Stunden |
| Zeitplan Ergebnis | | |
| Gesamtzeit | 80h |  |

Abweichungen Erklären

5.2 Fazit 4.2 Reflexion/Fazit

5.3 Ausblick 4.3 Optimierungsmöglickeiten/Ausblick