

## Dokumentation zur betrieblichen Projektarbeit

### Erstellen einer Individuellen Bewerberplattform auf Basis eines Headless CMS

#### PRÜFUNGSBEWERBER

Jamal Harris

Goebensiedlung 14  
56077 Koblenz

Identnummer: 1152593

#### AUSBILDUNGSBETRIEB

Open New Media GmbH  
Simrockstraße 5  
56075 Koblenz

Tel.: 0261 / 30 380-80  
Fax.: 0261 / 30 380-88

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.

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Abgabe: Koblenz, 20.04.2023

---

# Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

## Inhaltsverzeichnis

1.	Einleitung	1		
1.1	Projektbeschreibung	1		
1.2	Projektbegründung	2		
1.3	Projektziel	3		
1.4	Projektabgrenzung	3		
1.5	Projektumfeld	3		
1.5.1	Projektschnittstellen	3		
1.5.2	Entwicklungsumgebung	3		
1.6	Abweichungen zum Projekt-Antrag	4		
2.	Projektplanung	4		
2.1	Ressourcen Planung	4		
2.2	Entwicklungsprozess	4		
2.3	Planung von Maßnahmen zur Qualitätssicherung	5		
2.4	Anstoß-Meeting	5		
2.5	Auswahl von Headless CMS	5		
2.6	Auswahl von Formular Tool	7		
2.7	Auswahl von Tool für Mailversand über SMTP	7		
2.8	Entwerfen von Seitenstruktur & Elementen	7		
3.	Implementierung	8		
3.1	Installation und Grundeinrichtung des Headless CMS	8		
3.2	Erstellen einer Vue.JS Applikation	9		
3.2.1	Installation einer Vue.JS Applikation	9		
3.2.2	Grundeinrichtung Vue.Js Applikation	9		
3.2.3	Schnittstellen zu Headless CMS einrichten	10		
3.2.4	Erstellen von Inhaltstypen in CMS	10		
3.2.5	Auslesen der Seiten-Elemente und Vorbereitung der Daten	11		
3.2.6	Erstellung von Vue-Komponenten	12		
3.2.7	Integration von Formtool	13		
3.3	Mailversand über SMTP	14		
3.3.1	Versand aus Vue.js Applikation (Problem)		<b>Error!</b>	<b>Bookmark not defined.</b>

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

3.3.2	Versand aus Vue.js Applikation (Lösung)	<b>Error!</b>	<b>Bookmark</b>	<b>not defined.</b>
3.3.3	Grundeinrichtung Laravel Projekt	15		
3.3.4	Anbinden von Vue Applikation an Laravel	15		
3.3.5	Abwickeln des Mailversands	16		
3.4	Umsetzung von Design	17		
3.5	Routing	18		
3.6	Abweichungen	19		
3.7	Maßnahmen zur Qualitätskontrolle	19		
4.	Dokumentation	20		
5.	Projekt Abschluss	21		
5.1	Soll-ist-Vergleich	21		
5.2	Fazit	22		
6.	Ausblick	23		
7.	Anhänge	i		
A1.	Tabelle 1: Ressourcen/Kosten Tabelle	i		
A2.	Tabelle 2: Produkt Backlog	i		
A3.	Gantt-Diagramm	ii		
A4.	Use-Case-Diagramm	ii		
A5.	Seitenbaum	iii		
A6.	Ausschnitt der groben Gestaltung (Teil 1)	iv		
A7.	Ausschnitt grober Gestaltung (Teil 2)	<b>Error! Bookmark not defined.</b>		
A8.	Anhang 6: Strapi Backend	<b>Error! Bookmark not defined.</b>		
A9.	Anhang 7: Erstellen von Beispiel Element in Strapi	vii		
A10.	Beispiel Seite mit Überschrift, Text und Bild BE und Ausschnitt von JSON-Antwort	ix		
A11.	App.js	xi		
A12.	ViteConfig.js	xii		
A13.	Ausschnitt aus Mail-Model	xiii		
A14.	Anhang 12: Mail Controller	xiv		
A15.	Anhang 13: Sequenzdiagramm E-Mails	xv		
A16.	Anhang 14: Ausschnitte Design Umsetzung (Teil 1)	xvi		
A17.	Vue Router	xvii		

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

A18. Anhang 16: Erstellen von Seiten Map xx

A19. Projektantrag xxiii

A20. Formblatt für das Projekt xxiv

# 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.

## 1.1 Projektbeschreibung

Um die Rekrutierung neuer Mitarbeiter zu optimieren, soll eine individuelle Bewerberplattform für ONM entwickelt werden, welche im Nachgang auf der TYPO3<sup>1</sup>-Webseite der Agentur verlinkt werden soll, um die bestehende Lösung zu ersetzen. Die Daten sollen über ein Headless CMS<sup>2</sup> bezogen werden. Ein Headless CMS ist ein CMS ohne Frontend (Daten werden nur gemanagt und können über Abfragen bezogen werden, es liegt kein User-Interface zur Darstellung der Daten vor).

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 One-Pager<sup>3</sup>, 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.

---

<sup>1</sup> Name eines Content-Management Systems

<sup>2</sup> CMS

<sup>3</sup> Startseite die alles beinhaltet, ohne Unterseiten

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Zudem soll es eine Übersicht über die verfügbaren Stellenangebote, die nach Beschäftigungsart gruppiert wird, 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.

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 auf der Plattform 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

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 ersetzen.

Mein Projekt dient des Weiteren, 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

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 einen das Erstellen einer Bewerberplattform, die sich aus einem Headless CMS, in dem Daten gepflegt werden, und einer Vue.js Applikation, die die Daten dynamisch ausliest und daraus ein Front-End erstellt, zusammensetzt. Diese Lösung soll im Nachgang die bestehende Lösung zur Rekrutierung von Mitarbeitern über die Homepage ersetzen. Des Weiteren besteht das Ziel einer Evaluation des Einsatzes 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

Es wird noch nicht das fertige Endprodukt, sondern die Basis für weitere Iterationen geschaffen. Ein Deployment wird in der Projektphase noch nicht stattfinden.

## 1.5 Projektumfeld

### 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 dieses Endpunktes die Informationen auslesen, und ein Frontend daraus erstellen.

Enge Schnittstellen stellen auch die Betriebsleitung als Auftragsgeber, die Personalverantwortliche 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 Entwickeln wurde ein Desktop-Computer und verschiedene Software verwendet, genauere Spezifikationen sind der Tabelle in Anhang [A1 \(Tabelle 1: Ressourcen/Kosten\)](#)



Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

[Tabelle](#)) der Spalte „Ressource“ unter den Punkten „Hardware“ und „Software“ zu entnehmen.

## 1.6 Abweichungen zum Projekt-Antrag

Durch Probleme, die in der Entwicklungsphase festgestellt wurden, wurde sich dazu entschieden, die Vue.js Applikation an ein Laravel-Projekt anzubinden, hierauf wird unter [3.3.1 \(Problem Mailversand\)](#) und [3.3.2 \(Lösung Mailversand\)](#) genauer eingegangen. Dieser zusätzliche Aufwand konnte jedoch durch Zeitgewinne, diese werden unter [5.1 \(Soll-ist-Vergleich\)](#) genauer erläutert, wieder ausgeglichen werden, weshalb die Gesamtprojektzeit sich nicht geändert hat.

## 2. Projektplanung

### 2.1 Ressourcen-Planung

Im Anhang [A1 \(Tabelle 1: Ressourcen/Kosten Tabelle\)](#) befindet sich eine Auflistung aller Ressourcen, die zum Entwickeln der Plattform in dem von der IHK vorgegebenen Zeitraum von 80h verwendet werden. Die Ressourcen werden den entstehenden Kosten gegenüber gestellt. Bei der Auswahl der verwendeten Software wurde auf Open-Source-Lizenzen<sup>4</sup> gesetzt, weshalb in diesem Bereich keine weiteren Kosten entstehen.

### 2.2 Entwicklungsprozess

Zum Entwickeln der Plattform 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 definiert 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 primär alle Haupt-Anforderungen, sekundär die weiteren Anforderungen der Priorität nach 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 kann Anpassungen von bestehenden Elementen (iterativer Charakter, besonders im Bereich der Gestaltung) sowie neue Anforderungen (inkrementell) enthalten. Für den Bearbeitungszeitraum

---

<sup>4</sup> Lizenz mit der Software kostenlos genutzt, verändert und vertrieben werden kann

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

wurden insgesamt drei Meetings eingeplant. Die Zeitplanung wurde in einem Gantt-Diagramm visualisiert, dieses befindet sich im Anhang [A3 \(Gantt-Diagramm\)](#).

Ein agiler Ansatz ist für dieses Projekt besonders gut geeignet, da intern noch keine Erfahrung mit Headless CMS besteht, und so deshalb wichtig ist auf neue Herausforderungen und Erkenntnisse 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.

## 2.3 Planung von Maßnahmen zur Qualitätssicherung

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. Zudem soll ein GitHub<sup>5</sup> Repository angelegt werden, das zur Versionskontrolle dient.

## 2.4 Anstoß-Meeting

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 Produkt-Backlog zusammengefasst, dieses befindet sich im Anhang [A2 \(Tabelle 2: Produkt Backlog\)](#). Der Produkt-Backlog dient als Sammlung aller Anforderungen, die die Plattform erfüllen soll, und ist nach Priorität sortiert. Auch wurde mir eine Sammlung von Texten/Inhalten für die einzelnen Seiten übergeben. Die Interaktionen zwischen Beteiligten und dem System wurden in einem Use-Case-Diagramm dargestellt. Diese befindet sich im Anhang [A4 \(Use-Case-Diagramm\)](#).

## 2.5 Auswahl von Headless CMS

Als zu verwendendes CMS wurde sich für Strapi entschieden. Es erfüllt alle Anforderungen, befindet sich unter einer Open-Source-Lizenz (siehe [2.1 Ressourcen-Planung](#)), kann selbst gehostet werden, was ein großer Vorteil in Bezug auf DSGVO-Konformität und Unabhängigkeit von Anbietern mit sich bringt. Zudem bietet es ein

---

<sup>5</sup> Plattform für Kollaboration und Versionskontrolle von Software

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Verwaltungstool für hochgeladene Medien und eine intuitive Benutzeroberfläche für Redakteure. Auch werden Daten ausschließlich im JSON-Format<sup>6</sup> bereitgestellt, was sich gut mit Vue.js kombinieren lässt, da Vue.js auf JavaScript basiert.

---

<sup>6</sup> Java Script Objekt Notation, Art von JavaScript Objekte darzustellen

## 2.6 Auswahl von Formular Tool

Nach einer Internetrecherche wurde sich 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<sup>7</sup>, und hat eine ansprechende Funktionsweise. Zudem ist es möglich, eigene Formular-Feld-Validationen<sup>8</sup> zu erstellen.

## 2.7 Auswahl von Tool für Mailversand über SMTP

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

## 2.8 Entwerfen von Seitenstruktur & Elementen

In Anhang [A5 \(Seitenbaum\)](#) befindet sich ein Entwurf für eine erste simple Seitenstruktur, bei welcher jede Seite als direkte Unterseite der Startseite angelegt wird, in Form eines Seiten-Baumdiagramms.

Auf Basis der vorhanden Informationen aus dem Product-Backlog, wurden nun grobe Entwürfe für Inhalts- und Seitenelemente angefertigt und gestaltet, ein Ausschnitt hiervon befinden sich in Anhang [A6 \(Ausschnitt der groben Gestaltung\)](#).

---

<sup>7</sup> Formulare, die in meherer Schritte aufgeteilt sind (wie Teil-Formulare)

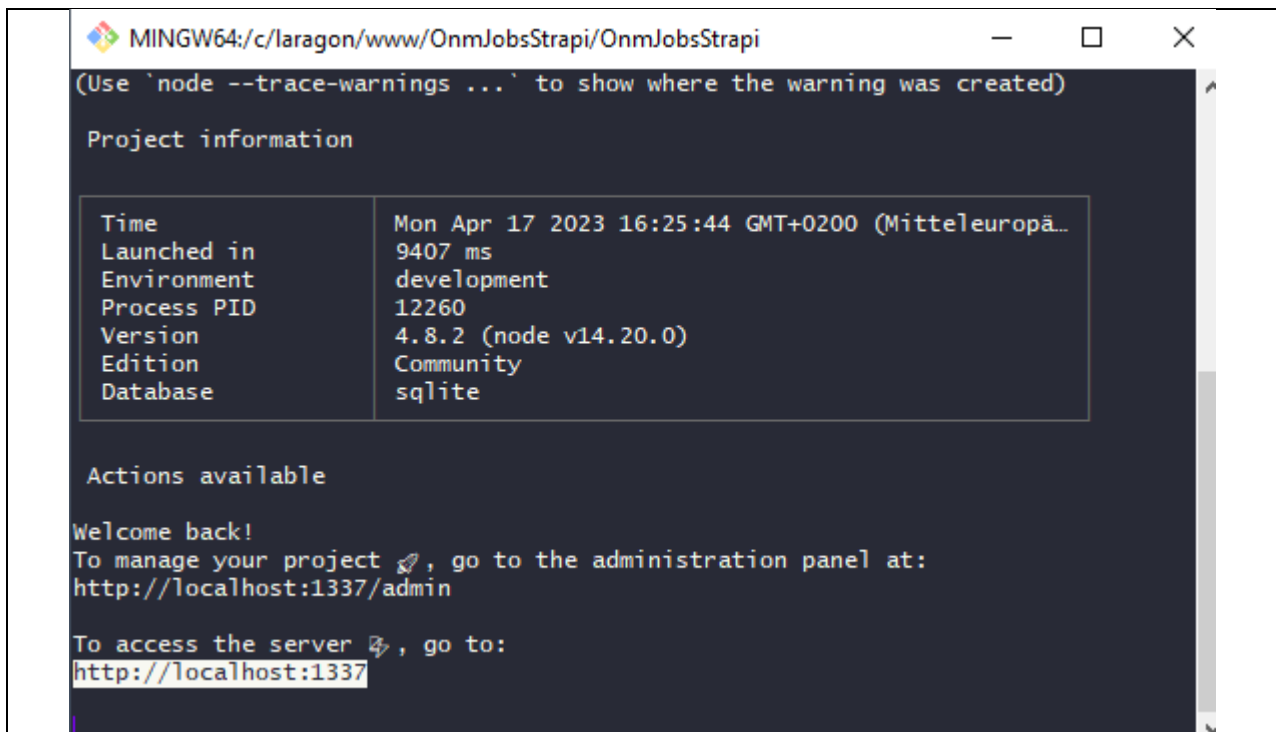
<sup>8</sup> Methodik um Eingaben der Nutzer auf Plausibilität/Korrektheit zu prüfen

## 3. Implementierung

### 3.1 Installation und Grundeinrichtung des Headless CMS

Die Installation von wird wie in der Dokumentation beschrieben mit dem Kommando „`npx create-strapi-app@latest strapiJobs`“ in einem CLI<sup>9</sup> 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 1 dargestellt.



```
MINGW64:/c:/laragon/www/OnmJobsStrapi/OnmJobsStrapi
(Use 'node --trace-warnings ...' to show where the warning was created)

Project information

Time                Mon Apr 17 2023 16:25:44 GMT+0200 (Mittleeuropä...
Launched in        9407 ms
Environment         development
Process PID        12260
Version            4.8.2 (node v14.20.0)
Edition            Community
Database           sqlite

Actions available

Welcome back!
To manage your project , go to the administration panel at:
http://localhost:1337/admin

To access the server , go to:
http://localhost:1337
```

Abbildung 1

Die Applikation ist nun im Browser lokal unter der in dem CLI ausgegebenen Adresse 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 [A7 \(Screenshot von Strapi Backend\)](#) befindet.

---

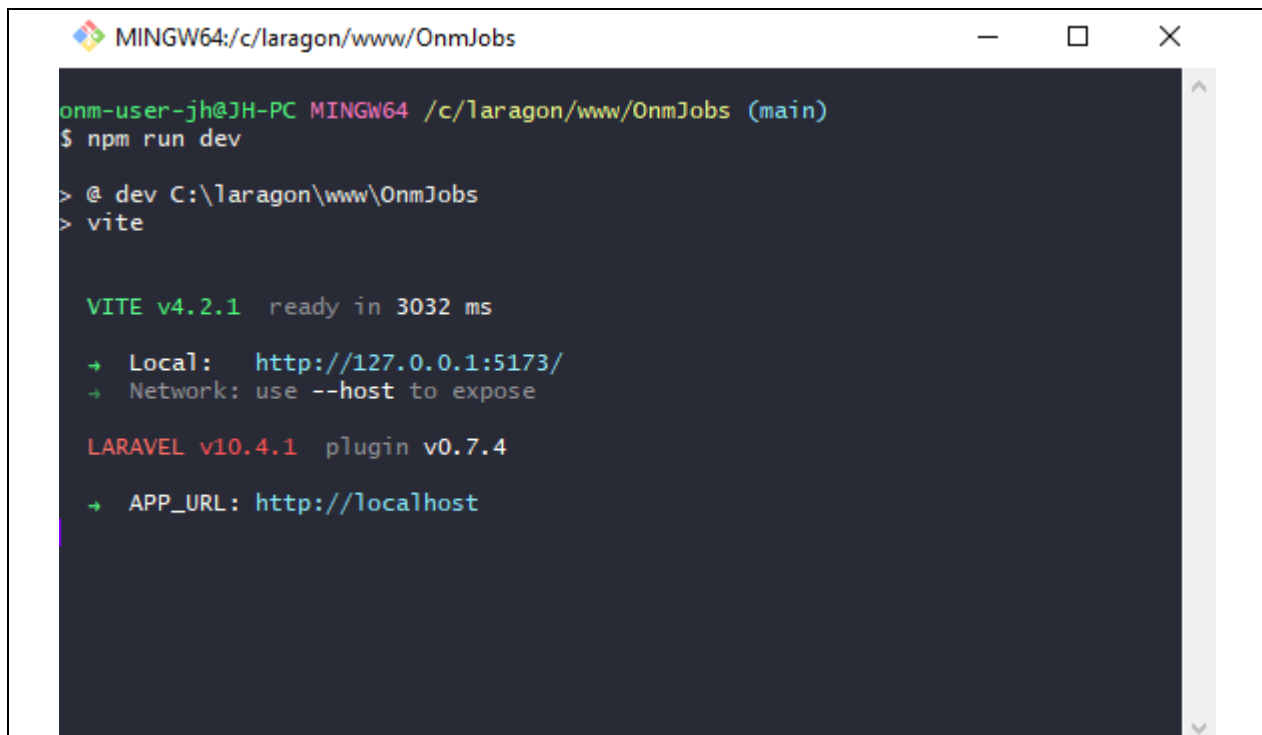
<sup>9</sup> Command Line Interface, beschreibt die Schnittstelle zwischen Mensch und Computer Befehle in Textform einzugeben

## 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, und ein Ordner erstellt, der die Vue.js Applikation enthält.

Als Nächstes werden mit dem CLI-Kommando „`npm install`“ Abhängigkeiten installiert. Um das Projekt lokal zu hosten, startet man einen Entwicklungsserver mit dem CLI-Kommando „`npm run dev`“. Die Ausgabe des Befehls im CLI ist in Abbildung 2 zu sehen.



```
MINGW64:/c:/laragon/www/OnmJobs
onm-user-jh@JH-PC MINGW64 /c:/laragon/www/OnmJobs (main)
$ npm run dev

> @ dev C:\laragon\www\OnmJobs
> vite

VITE v4.2.1 ready in 3032 ms
  → Local:   http://127.0.0.1:5173/
  → Network: use --host to expose

LARAVEL v10.4.1 plugin v0.7.4
  → APP_URL: http://localhost
```

Abbildung 2

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 entsprechenden Views über eine statische Navigation verlinkt. Dies diente lediglich als temporäre 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 <sup>10</sup>, mit dem HTTP Requests aus der App heraus getätigt werden können. Eine Beispiel Codeauszug ist in Abbildung 3 zu sehen. Hier wird die URL angefragt, und bei einer Antwort werden die Daten über die Funktion „loadContent“ zu einem Array hinzugefügt.

```

axios.get(this.dataSrcURL).then((response) => {
  this.pageObject = response.data.data.attributes.PageContent
  this.pageObject.forEach(el =>
    this.loadContent(el)
  )
})

```

Abbildung 3

### 3.2.4 Erstellen von Inhaltstypen in CMS

Als Nächstes wurden Teststrukturen angelegt, welche aus Texten und Bildern bestanden. Um diese Teststrukturen abzufragen, muss ein Sammel-Typ<sup>11</sup> erstellt werden. Dieser wurde unter dem Namen „Seite“ angelegt, und kann als Behälter dienen, in dem die erstellten Inhaltselemente in einer dynamischen-Zone<sup>12</sup> platziert werden können.

Hier habe ich Dummy-Daten<sup>13</sup> 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 Inhaltstypen auf Basis der entworfenen Elemente (siehe 2.8 Entwerfen von Seitenstruktur & Elementen) im Headless CMS erstellt. Im Anhang A8 (Erstellen von Beispiel-Element in Strapi) wird dieser Prozess beispielhaft für eine vereinfachte Überschrift Komponente dargestellt.

<sup>10</sup> versendet Http Anfragen und nimmt Antworten entgegen (bsp.Web-Browser)],

<sup>11</sup> Behälter indem erstelle und native Komponenten platziert werden können

<sup>12</sup> Behälter, in dem eine undefinierte Menge von diversen Inhaltselementen in beliebiger Reihenfolge erfasst

<sup>13</sup> Daten ohne Bedeutung, nur zum Testen

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Als Nächstes kann die Komponente zu der dynamischen-Zone des Seiten-Objektes hinzufügen, und wie in Abbildung 4 verwendet werden.



Abbildung 4

Nun kann das Seiten-Element über den entsprechenden Endpunkt mit der Seiten ID abgefragt werden. Ein Beispiel aus dem Backend von einer Seite mit einem Inhaltselement vom Typ „Überschrift“ sowie einem Inhaltselement vom Typ „Text und Bild“, sowie ein Ausschnitt der Antwort auf die Anfrage des API-Endpunktes ist in Anhang A9 ([Beispiel Seite mit Elementen und Ausschnitt von JSON-Antwort auf Anfrage](#)) zu sehen.

### 3.2.5 Auslesen der Seiten-Elemente und Vorbereitung der Daten

Nun wurden innerhalb der Vue.js Applikation die Daten über „Axios“ promise-based<sup>14</sup> abgefragt, die Inhaltselemente verarbeitet und die Daten zu einem von der Vue-Komponente erstellten Datenobjekt mit dem Namen „content“ vom Typ Array hinzugefügt. Um die Inhalte nun auszugeben wurde ein Vue-for-Loop (v-for=„Inhalt in Array“) mit Konditionen verwendet, es folgt eine vereinfachte Darstellung:

```
<div v-for="inhalt in inhaltsArray" >
  <div v-if="inhalt['__component']=='beispiel.element'">
    //Gib Komponente von beispiel.element mit Informationen
    //aus der Abfrage aus
  </div>
  //...weiter Komponenten
</div>
```

So wird später für jedes Inhaltselement die entsprechende Vue-Komponente gerendert, und die Daten können über die „Slot“-Tags zugeordnet, oder als Parameter für konditionelle Zwecke genutzt werden (siehe [3.2.6 Erstellung von Vue-Komponenten](#)).

<sup>14</sup> basiert auf dem Konzept das etwas zu einem Zeitpunkt einen Wert haben wird (nicht von Anfang an hat)



### 3.2.6 Erstellung von Vue-Komponenten

Nun wurde für die Inhaltstypen aus dem Headless CMS (siehe [2.8](#) Entwerfen von Seitenstruktur & Elementen) 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 5.



Abbildung 5

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

### 3.2.7 Integration von Formtool

Das ausgewählte Framework für die Formulare (siehe [2.6 Auswahl von Formular Tool](#)) kann als Node-Package mit dem Befehl „npm install @formkit/vue“ installiert werden.

Es wurden des Weiteren eine Formkit-Erweiterung namens „FormkitMultiStep“ für Multi-Step-Formulare sowie eine Erweiterung mit Themes<sup>15</sup> mit dem Namen „themes“ für ein Design des Formulars hinzugefügt.

Um es in der Vue Applikation verfügbar zu machen, wurde es nun noch im Startpunkt der Vue Applikation integriert, wie in Anhang [A10 \(App.js\)](#) zu sehen ist.

Ein simples beispielhaftes Formular mit Codeauszug neben der Frontend-Ausgabe ist in Abbildung 6 verdeutlicht.

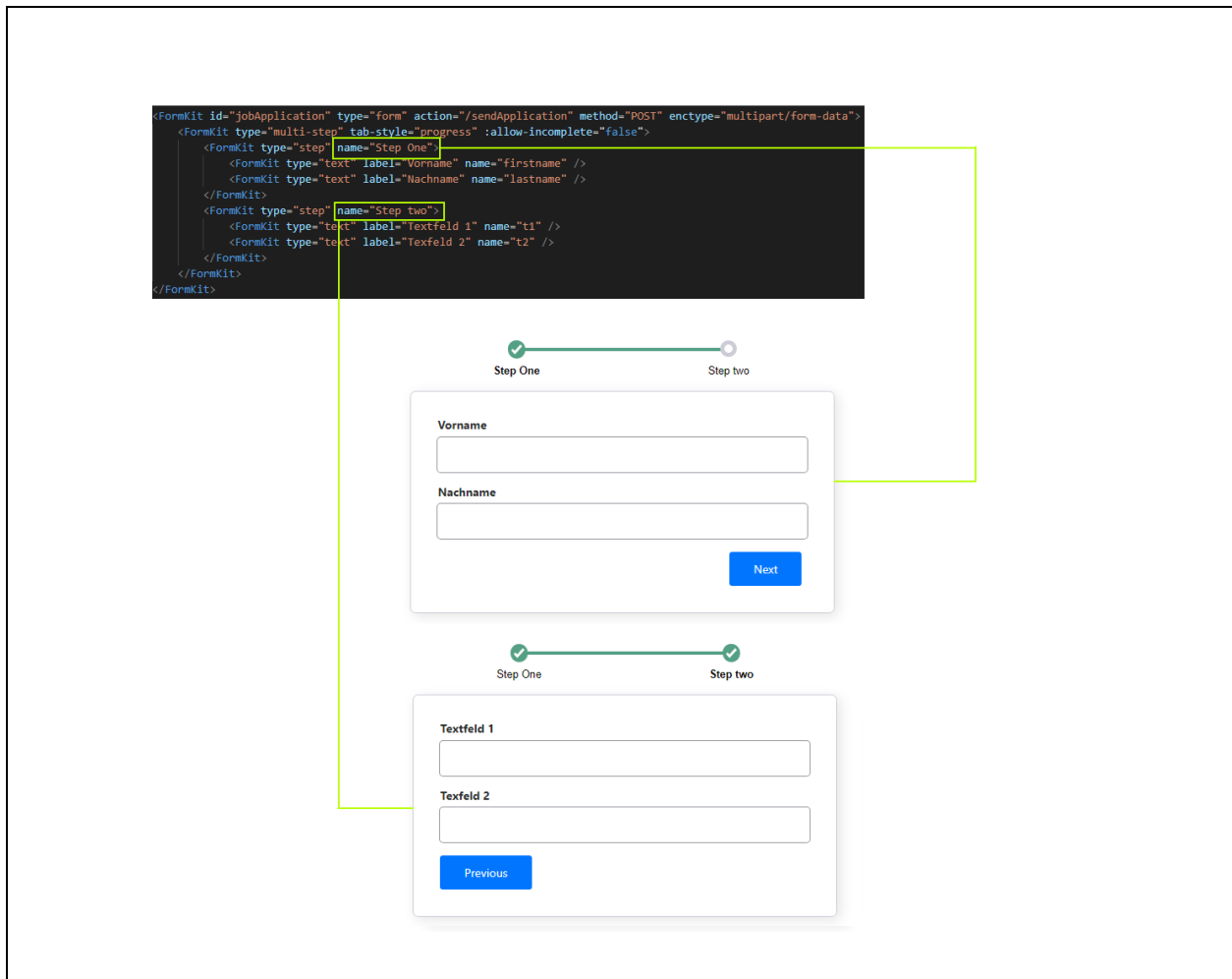


Abbildung 6

<sup>15</sup> Zusammen greifendes Grund Design der Komponenten

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

### 3.3 Mailversand über SMTP<sup>16</sup>

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 [3.3.1 \(Problem Mailversand\)](#) genauer eingegangen.

#### 3.3.1 Problem Mailversand

Das Node-Package „NPMJS“ (siehe [2.7 Auswahl von Tool für Mailversand über SMTP](#)) kann in einem CLI mit dem Befehl „`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 Lizenzkosten zu vermeiden sind ([2.1 Ressourcen-Planung](#)) ist diese Option nicht geeignet.

#### 3.3.2 Lösung Mailversand

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

---

<sup>16</sup>Simple Mail Transfer Protocol, Protokoll zum austausch von Emails

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

### 3.3.3 Grundeinrichtung Laravel Projekt

Um das Laravel Projekt zu initialisieren wurde der Befehl „composer create-project laravel/laravel OnmJobs“ in einem CLI ausgeführt. Nun kann man in das Verzeichnis wechseln und über ein CLI den Befehl „php artisan serve“ ausführen, um das Projekt lokal zu hosten. Die Abbildung 7 zeigt die Ausgabe des CLI. Das Projekt ist jetzt über die in der Ausgabe zu sehende URL zu erreichen.

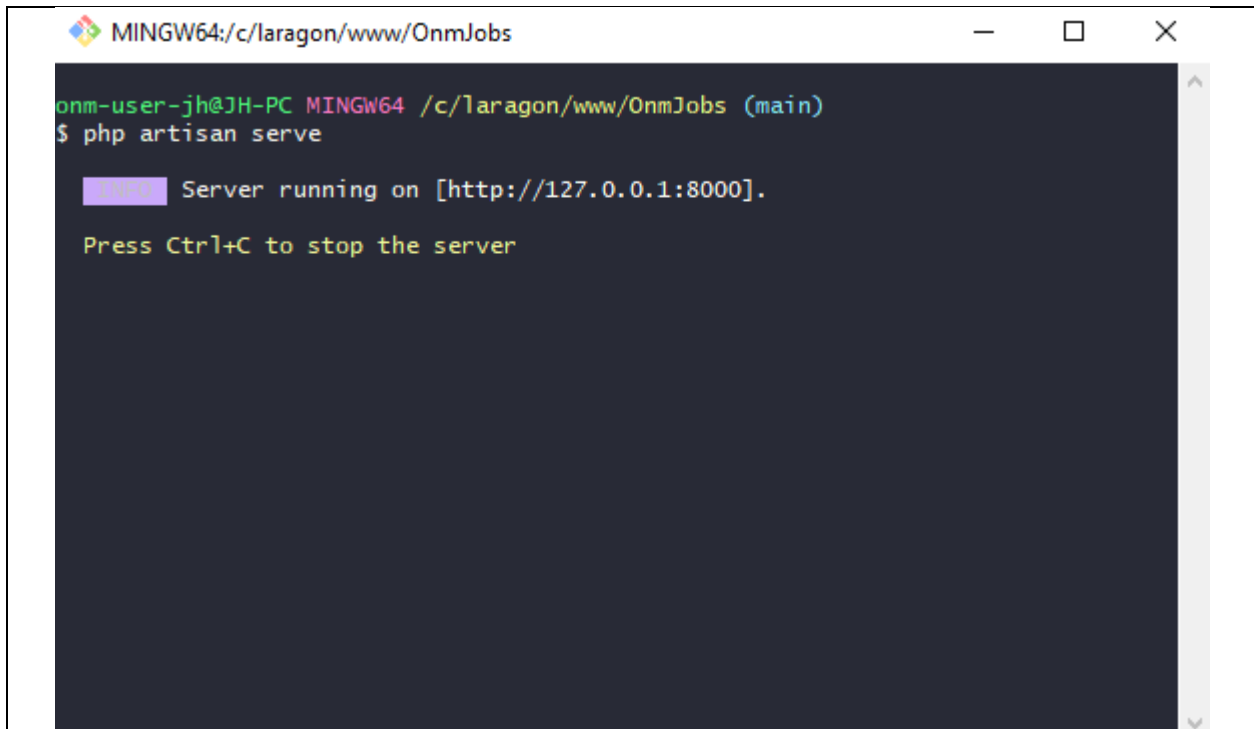
A screenshot of a terminal window titled "MINGW64:/c:/laragon/www/OnmJobs". The terminal shows the command "php artisan serve" being executed. The output is "INFO: Server running on [http://127.0.0.1:8000]". Below this, it says "Press Ctrl+C to stop the server". The terminal has a dark background with light-colored text. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Abbildung 7

### 3.3.4 Anbinden von Vue Applikation an Laravel

Um die erstellte Vue Applikation an das Laravel Projekt anzubinden, wurde nun zunächst das Node Package „@vitesjs/plugin-vue“ über ein CLI installiert. Als Nächstes wurde die Datei „Vite.config.js“<sup>17</sup> (siehe [A11 ViteConfig.js](#)) um das Plugin und somit eine Vue.js Applikation erweitert.

Hieraufhin habe ich die bereits erstellte 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

---

<sup>17</sup> Konfigurations Datei des Build-Tools

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Haupt JavaScript Datei des Laravel Projektes wurde nun die Vue.js Applikation auf den Container mit der ID „App“ gemounted<sup>18</sup>.

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 Frontend der Seite darstellt.

### 3.3.5 Abwickeln des Mailversands

Laravel arbeitet nach dem MVC-Prinzip<sup>19</sup>, also wurde ein Model, eine View und ein Controller erstellt, um aus einem ausgefüllten Bewerbungsformular, ein E-Mail zu erstellen, welche automatisch an die Personalverantwortliche versendet wird.

Zunächst wurde ein Model mit dem Namen „ApplicationMail.php“ durch das CLI-Kommando „`php artisan make:mailable ApplicationMail`“ erzeugt. In dieser Klasse werden Inhalte sowie Anhänge über eine erstellte View formatiert zu einem via E-Mail zu versendendem Objekt verarbeitet. Im Anhang [A12 \(Ausschnitt aus Mail-Model\)](#) befindet sich ein Code Ausschnitt dieser Klasse, von der Stelle, an der die Daten an die View bezogen wird.

Um den Mailversand durchzuführen, wurde nun mit dem CLI-Kommando „`php artisan make:controller SendMailController`“ ein neuer Controller mit dem Namen „SendMailController“ zu dem Laravel Projekte hinzugefügt. Dieser nimmt die Informationen aus dem Bewerbungsformular entgegen, und verwendet das „ApplicationMail“-Model, um eine E-Mail daraus zu erstellen, und zu versenden. Ein Codeauszug der Controller-Klasse befindet sich im Anhang [A13 \(Mail Controller\)](#).

Als Nächstes wurde noch eine Laravel Route hinzugefügt, die auf den „SendMailController“ deutet, und das „action“-Attribut<sup>20</sup> des Testformulars (siehe [3.3 Mailversand über SMTP](#)) wurde auf diese Route gedeutet. So werden die Formulardaten über das POST-Array<sup>21</sup> an den Mail-Controller übergeben.

Nun wurde der Mailversand für einen Test-SMTP Server konfiguriert und darauf getestet.

---

<sup>18</sup> in einen Behälter „eingefügt“

<sup>19</sup> Model-View-Controller Teilt die Logic von der Darstellung und dem zu Erzeugenden Objekt

<sup>20</sup> entscheidet darüber, über welche URL beim versenden des Formulars angesprochen wird

<sup>21</sup> PHP-Array mit Variablen die über die POST-Methode versendet werden

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Darauf hin wurden die Formular-Felder validiert, und mit Fehlermeldungen je nach Validations-Ergebniss über Angabe der Validations-Regel angepasst, wie in im Abbildung 8 zu sehen ist , und ein simpler eigener Validator für den Datei-Typ geschrieben. Dieser ist in Abbildung 9 zu sehen und wird über die Datei App.js (siehe [A10 App.js](#)) als Validations-Regel integriert.

```
<FormKit type="text" label="Vorname" name="firstname" validation="required|alpha" :validation-messages="{
  alpha: 'der Vorname darf keine Zahlen enthalten',
  required: 'dies ist ein Pflichtfeld'
}" />
```

Abbildung 8

```

8   var file = function (node) {
9     var update = (node.value) //updates node value when different file is entered
10    return (node['_value'][0]['name'].split('.').pop().toLowerCase()=='pdf')
11  }
12  file.force = true
13  export default file
```

Abbildung 9

Der Prozess des Mailversands wurde zur Veranschaulichung in einem Sequenz-Diagramm dargestellt, diese befindet sich im Anhang [A14 \(Sequenzdiagramm E-Mails\)](#).

### 3.4 Umsetzung von Design

Als Nächstes wurde das Styling für die unter [3.2.6 \(Erstellung von Vue-Komponenten\)](#) erstellten Komponenten anhand der Entwürfe unter [2.8 \(Entwerfen von Seitenstruktur & Elementen\)](#), sowie das Design der Seitenelemente umgesetzt. Zu sehen ist ein Ausschnitt des Ergebnisses in den Anhängen [A15 \(Ausschnitt Design-Umsetzung Frontpage\)](#) sowie [A16 \(Design Umsetzung Bewerbungs-Seite\)](#).

### 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-Parameter von einzelnen Seiten anzupassen wurde im Backend zu dem Seitentyp eine Option für ein wählbares URL-Segment eingefügt, dies ist in Abbildung 10 zu sehen.

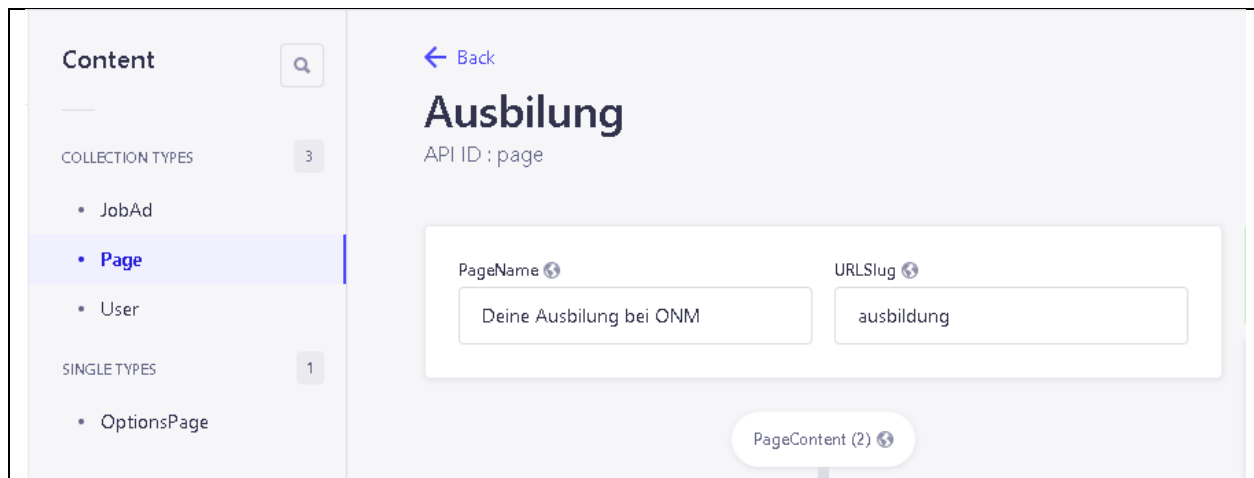


Abbildung 10

Im Vue Router wurde die URL um einen Parameter erweitert. Dieser ist in Anhang [A17 \(Vue Router\)](#) zu sehen. Über den Parameter wird dann die ID des Seiten-Objektes über eine in der Vue.js Applikation durch eine Abfrage der Seitenelemente erstellte Map <sup>22</sup> bestimmt. Wenn im Backend des Headless CMS kein URL-Segment für eine Seite spezifiziert wird, wird der Seiten-Name, welcher einmalig ist, aus den Attributen der Seite verwendet. Dies wird in einem Codeauszug in Abbildung [A18 \(Erstellen von Seiten Map\)](#) verdeutlicht.

Um die Navigation dynamisch zu gestalten, wurde als Nächstes eine Options-Seiten-Komponente als Sammel-Typ erstellt. Hier sollen Elemente/Informationen gepflegt werden, die auf allen Seiten gleich sind (Beispielsweise Logo, Navigation, Footer). Für die Navigation werden Relationen<sup>23</sup> zu bestehenden Seiten als Inhaltselemente verwendet, um diese an die Vue.js Applikation zu übergeben, in welcher daraus Navigations-Menüs erstellt werden.

<sup>22</sup> Equivalent zu einem Array, bei dem jedem Schlüssel ein Wert zugeordnet wird

<sup>23</sup> Beziehungen zwischen Objekten

### 3.6 Abweichungen

Wie unter [3.3.1 \(Problem Mailversand\)](#) und [3.3.2 \(Lösung Mailversand\)](#) erklärt, wurde sich dazu entschieden, die Vue.JS Applikation an ein Laravel Backend anzubinden, um den Mailversand abzuwickeln.

### 3.7 Maßnahmen zur Qualitätskontrolle

[Todo: Code tests, code review durch die Leitung der Anwendungsentwicklung und Vue Experten wie in planung [\[verlinken\]](#) angesetzt...]

Zudem wurde ein Gitlab für Versionskontrolle bei Anpassungen und Erweiterungen, sowie einer übersichtlichen Darstellung von Änderungen im Code, um bei Problemen auf vorherige Stände zurück wechseln zu können, eingerichtet.



## 4. Dokumentation

Dieses Dokument wurde im Laufe des Projektes angefertigt, und nach Abschluss überarbeitet und ergänzt. Die Kundendokumentation wurde im Anschluss unter Zuhilfenahme dieses Dokumentes erstellt. Ein Auszug der Kundendokumentation befinden sich in Anhang [A19 \(Ausschnitt Kundendokumentation\)](#).

Die Entwicklerdokumentation wurde im während des Projektes in Form von Kommentaren, und im Anschluss unter Zuhilfenahme dieses Dokumentes verfasst, und mit einer lokalen Installationsanleitung im Gitlab hinterlegt. Ein Ausschnitt der Dokumentation ist in Anhang [A20 \(Ausschnitt der Entwicklerdokumentation\)](#) zu sehen.

## 5. Projekt Abschluss

### 5.1 Soll-ist-Vergleich

Arbeitsschritt	Zeitangabe Projektantrag	Zeit verwendet
<b>Projektplanung</b>		
Auswahl Headless CMS	1 Stunde	1 Stunde
Planung von Seitenstruktur & Einrichtung von Elementen	3 Stunden	1.5 Stunden
Abstimmung	1 Stunde	1 Stunde
<b>Basisgestaltung</b>		
Erstellung von groben Entwürfen für Elemente	3 Stunden	3 Stunden
Abstimmung	1 Stunde	1 Stunden
<b>Einrichten des Headless CMS</b>		
Grundeinrichtung Headless CMS	3 Stunden	3 Stunden
Einrichtung von Rollen im Headless CMS	2 Stunden	1 Stunde
<b>Erstellen einer Vue.JS Applikation</b>		
Grundeinrichtung Vue.js Application	2 Stunden	1 Stunde
Umsetzung der Seitenstruktur	4 Stunden	4 Stunden
Schnittstellen zu Headless CMS einrichten	6 Stunden	4 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	8 Stunden
Erstellen Laravel Projekt und Anbinden der Vue Applikation	0	2 Stunden
Mailversand über SMTP	0	3 Stunden
<b>Tests</b>		
Funktionstests	5 Stunden	3.5 Stunden
Modultests	5 Stunden	5 Stunden
Integrationstests	5 Stunden	5 Stunden
<b>Dokumentation</b>		
Projektdokumentation	10 Stunden	12 Stunden
Dokumentation für Entwickler	3 Stunden	1 Stunde

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Kundendokumentation	2 Stunden	1 Stunden
Zeitplan Ergebnis		
Gesamtzeit	80h	80h

*Zeitplanung*

In Tabelle ist der Zeitplanung aus dem Projektantrag zu sehen. Diese wurde um die Spalte „ist“ erweitert. Zeiteinsparungen sind mit Grün, Zeitüberschreitungen mit Rot markiert.

Wie in 5.1 zusehen ist, gab es zeitliche Abweichungen. Diese werden im Folgenden begründet.

Bei dem Punkt „Planung von Seitenstruktur & Einrichtung von Elementen“ konnte durch eine simple Seitenstruktur und erfolgreicher Abstimmung im Anstoß-Meeting ([siehe 2.4Anstoß-Meeting](#)) Zeit eingespart werden.

Der Schritt „Einrichtung von Rollen im Headless CMS“ konnte durch die hohe Benutzerfreundlichkeit von Strapi ebenfalls schneller durchgeführt als geplant.

Die Schritte „Grundeinrichtung Vue.js Application“ sowie „Schnittstellen zu Headless CMS einrichten“ konnten durch nahtloses Zusammenspiel von Vue.JS und Strapi ebenfalls schneller durchgeführt werden als in der Planung angegeben.

Als bei der Mailversand nicht wie geplant durchgeführt werden konnte, stellte es sich als beste Alternative heraus, die Vue.js Applikation an ein Laravel Backend anzubinden, über welches den Mailversand abgewickelt wird. Hierdurch wurden die Punkte „Erstellen Laravel Projekt und Anbinden der Vue Applikation“ und „Mailversand über SMTP“ hinzugefügt.

Für die Projektdokumentation wurde mehr Zeit aufgewendet als geplant. Hierdurch konnte bei der Kundendokumentation Zeit eingespart werden, Abbildungen und Erklärungen teilweise wieder verwendet werden können. Durch das Verwenden von Kommentaren im Code während des Entwickelns konnte ebenfalls die Entwicklerdokumentation schneller angefertigt werden.

Die Gesamtzeit hat sich trotz dieser Änderungen nicht verändert.

## 5.2 Fazit

Mit dem gewählten Entwicklungsprozess war die Umsetzung des Projektes, trotz Komplikationen einen Erfolg. Die Kommunikation zwischen dem Headless CMS und der Vue.js Applikation ging überraschend leicht von der Hand. Das entstandene Produkt dient für die Geschäftsleitung als zufriedenstellende Basis für den zukünftigen Einsatz zur Rekrutierung neuer Mitarbeiter.

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

Hieraus hat sich auch die Erkenntnis ergeben, dass der Einsatz von Headless CMS eine gute Möglichkeit darstellt, um kleine bis mittelgroße Kundenprojekte umzusetzen. Dies kann also künftig das Angebot der Firma erweitern und die Kundenzufriedenheit durch schlanke, leicht zu aktualisierende Systeme erhöhen.

## 6. Ausblick

Die Plattform wird vor allem im Bereich der Inhaltselemente an Gestaltung und Umfang noch erweitert werden. Zunächst wird die Anwendung auf einem Testserver deployt, auf dem die Redakteurin mit der Inhaltspflege experimentieren, die Zuverlässigkeit des Mailversands garantiert werden, und Anpassungswünsche und Erweiterungen geplant und abgestimmt werden können.

Wenn die Plattform bereit für den Livegang ist, wird sie auf der Webseite der Agentur verlinkt, um die bestehende Lösung zu ersetzen.

## 7. Anhänge

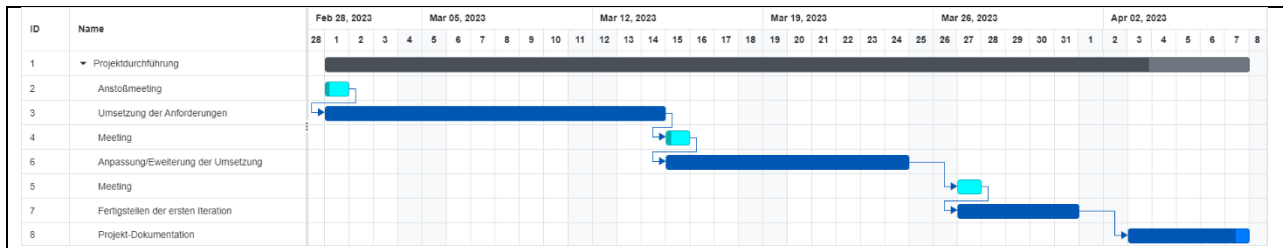
### A1. Tabelle 1: Ressourcen/Kosten Tabelle

Ressource	Kosten
Hardware	
Desktop Computer (Windows 10 Pro, 64-Bit-Betriebssystem, Intel(R) Core(TM) i5-3570 CPU 3.40GHz Prozessor, 16GB RAM)	Stromverbrauch ausrechnen?
Software	
Visual-Studio Code (v1.72.1)	keine Kosten
Node JS (v16.14.2)	keine Kosten
Node Package Manager (v6.14.7)	keine Kosten
Vue.js (v3.2.45)	keine Kosten
Personal	
Auszubildener (80h)	
Anwendungsentwickler (2h)	
Personalverantwortliche (2h)	
Redaktionsverantwortliche (2h)	
Firmenleitung (2h)	

### A2. Tabelle 2: Produkt Backlog

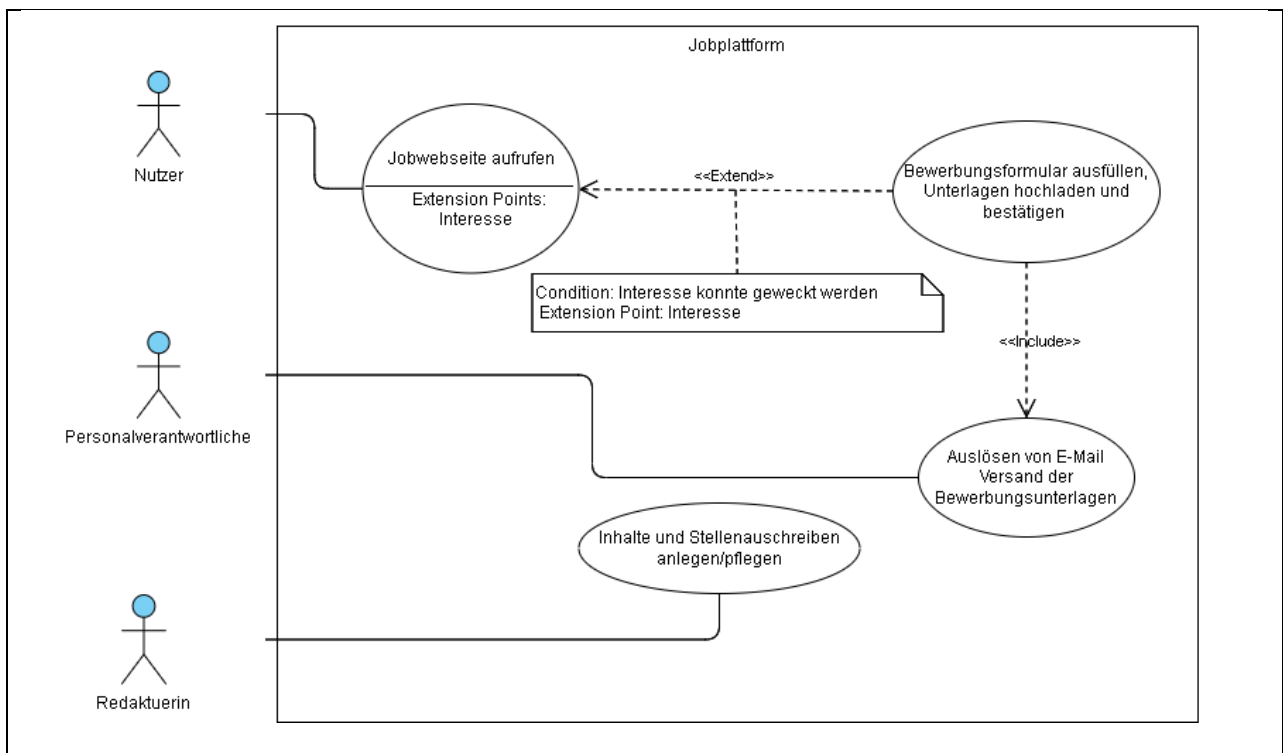
Anforderung	Priorität (1-5, niedrig zu hoch)
Datenverwaltung über Headless CMS	5
Umsetzung in Vue.Js	5
Automatischer Mailversand mit hochgeladenen Anhängen	5
Einbindung von Bildern und Videos	4
Gegliederte Übersicht der Stellenangebote	4
Multi Step Formular	3
Seperater Bereich für Ausbildung	2
Übersichtliche Gestaltung	1
Freundlicher Charakter	1

### A3. Gantt-Diagramm



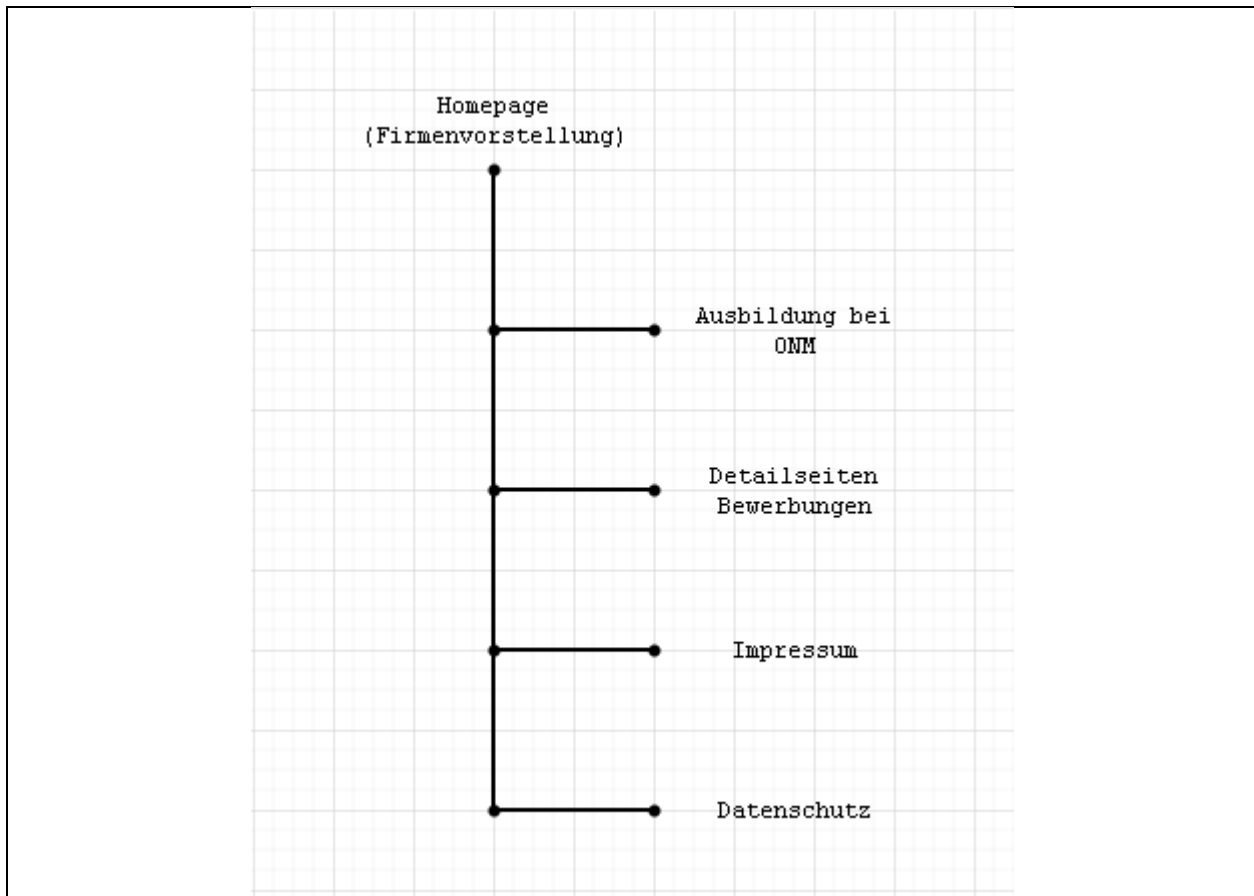
*Gant Diagramm der Projektplanung*

### A4. Use-Case-Diagramm



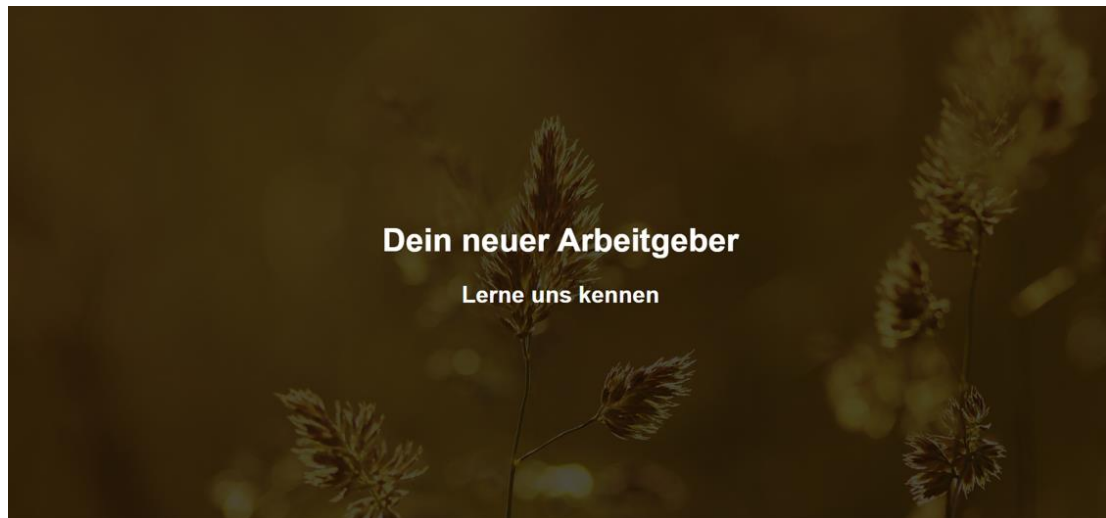
*Use-Case-Diagramm zur Nutzerinteraktion*

## A5. Seitenbaum



*Seitenbaum der Webanwendung*

## A6. Ausschnitt der groben Gestaltung



### Was macht ihr so und für wen?

#### Individuell digitalisieren

Bei ONM triffst du auf 19 kreative und motivierte Köpfe unterschiedlicher Fachrichtungen. Zusammen entwickeln wir innovative Projekte insbesondere für die Hotellerie, aber auch für diverse weitere Branchen. Unser Leistungsspektrum reicht von der Beratung über die Konzeption und das Design bis zur technischen Umsetzung und Betreuung digitaler Lösungen. Unter der Marke hotelsuite entwickeln wir hochkomplexe Lösungen für die Hotelbranche. Unsere Leidenschaft ist es, den Hoteldirektvertrieb zu optimieren, die Kundenbindung durch professionelle Gastkommunikation zu verbessern und aufwendige Prozesse zu digitalisieren. Von unserem Standort Koblenz aus betreuen wir wunderbare Individualhotels aus ganz Deutschland, die allesamt hohe Ansprüche an ihre digitalen Lösungen stellen. Unsere Kunden entscheiden sich für ONM, da sie eine individuelle Lösung benötigen. Immer dann, wenn Standard nicht genügt, besondere Gegebenheiten mit einbezogen werden müssen, dann sind wir gefragt.



[zur Kundenübersicht](#)



#### Passe ich zu euch?

Bei ONM erwarten dich vielfältige Projekte und unterschiedlichste Aufgabengebiete. Bei uns arbeiten Menschen, die eine Leidenschaft für Technologie in der Hotellerie teilen. Wir möchten unseren Kunden und deren Gästen das bestmögliche Erlebnis bieten. Wer bei uns arbeitet, liebt die immer neuen Herausforderungen, die unsere Projekte mit sich bringen. Hier gleicht kein Tag dem anderen. Jeder hier ist mit vollem Einsatz bei der Sache. Motiviert, verantwortungsvoll und zuverlässig. Leute, die mitdenken und eigene Ideen entwickeln, sind bei uns richtig. Ob extrovertierte Kundenvertreter oder stille Nerds, die aus Einsen und Nullen etwas komplett Neues schaffen, wir suchen Teamplayer mit Charakter.

#### Wie arbeitet ihr? Was zeichnet eure Arbeitsweise aus? Was ist euch wichtig?

Bei ONM findest du flache Strukturen und einen lockeren, kollegialen und respektvollen Umgang miteinander. Die Beziehung im Team untereinander ist uns genauso wichtig, wie das gute und partnerschaftliche Verhältnis zu unseren Kunden. Wir sind davon überzeugt, dass offene Kommunikation ein wichtiger Teil des Erfolgs ist. Wir fördern Eigenverantwortung, Kreativität und Teilhabe und wollen immer besser werden. Auch du darfst, kannst, sollst dich einbringen! Bei uns gibt es keine Überstundenkultur. Sollte es doch einmal vorkommen, werden alle Überstunden selbstverständlich bezahlt. Da Arbeit aber bekanntlich nicht alles ist, feiern wir auch gerne zusammen und sind auch ganz privat super nette, interessante Menschen.





## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

### Aktuelle Stellen

### Stellen alternative Darstellung

Anwendungsentwickler (m/w/d) 15.01.2033

Anwendungsentwickler Azubi (m/w/d) 15.01.2033

Mediengestalter (m/w/d) 15.01.2033

Mediengestalter Azubi(m/w/d) 15.01.2033

### Unsere Benefits

**Wissensdurst?**

Wir fördern die Teilnahme an Weiterbildungen, Konferenzen, etc. und bieten die Nutzung von Online-Weiterbildungsmedien

Early Bird oder Nachteule?

Homeofficemöglichkeit

Ein bisschen Spaß muss sein

**Außerdem:**

- Keine Überstundenkultur
- Kaffee, Tee, Wasser for free & leckeres Eis im Sommer
- Dein Gehaltsextra Givve-Firmenkreditkarte mit steuerfreiem Sachbezug
- Corporate Benefits Nutze vergünstigte Mitarbeiterkonditionen bei namhaften Markenanbietern
- Lebensqualität in Koblenz: Wir bieten dir einen modernen Arbeitsplatz mit aktuellster Hard- und Software in direkter Nähe zum Rhein. D.h. ein kurzer Entspannungspaziergang ist immer drin.
- Super nette Kunden aus der Hotellerie und unterschiedlichen Branchen

Und last, but most important...

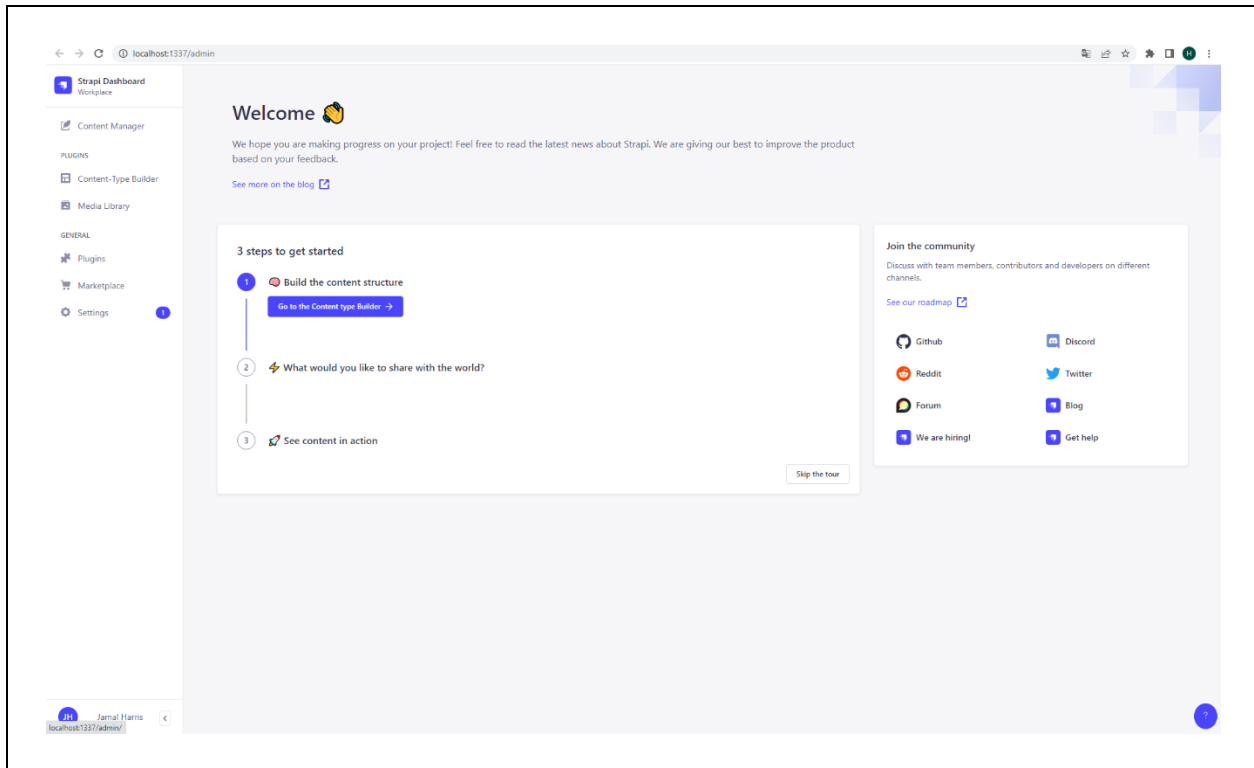
...Die besten Kolleg:innen der Welt und immer ein offenes Ohr, falls es irgendwo klemmt

Screenshot von Ausschnitt grober Gestaltung

© Open New Media GmbH | Jamal Harris | 1152593

v

## A7. Screenshot von Strapi Backend



*Screenshot von Strapi Backend*

## A8. Erstellen von Beispiel-Element in Strapi

The screenshot shows the 'Create a component' dialog in Strapi. It has a title bar with a Strapi icon and a close button. Below the title bar, there's a section titled 'Configurations' with a subtitle 'A type for modeling data'. There are two tabs: 'BASIC SETTINGS' (active) and 'ADVANCED SETTINGS'. Under 'BASIC SETTINGS', there are two input fields: 'Display name' with the value 'Beispiel Überschrift' and 'Select a category or enter a name to create a new one' with a dropdown menu showing 'text'. At the bottom, there are 'Cancel' and 'Continue' buttons.

The screenshot shows the 'Select a field for your component' dialog in Strapi. It has a title bar with the component name 'Beispiel Überschrift (Text - Beispiel Überschrift)' and a close button. Below the title bar, there are two tabs: 'DEFAULT' (active) and 'CUSTOM'. The main area displays a grid of field options. The 'Text' field is highlighted with a green oval. The fields include: Text (Small or long text like title or description), Rich text (A rich text editor with formatting options), Number (Numbers (integer, float, decimal)), Date (A date picker with hours, minutes and seconds), Boolean (Yes or no, 1 or 0, true or false), Relation (Refers to a Collection Type), Email (Email field with validations format), Password (Password field with encryption), Enumeration (List of values, then pick one), Media (Files like images, videos, etc), and JSON (Data in JSON format). At the bottom, there is a 'Component' field (Group of fields that you can repeat or reuse).

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

*Screenshots Erstellung Beispiel-Element*

Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

## A9. Beispiel Seite mit Elementen und Ausschnitt von JSON-Antwort auf Anfrage

PageContent (2)

Abschnittsüberschrift - Hallo Welt

Hauptüberschrift

Hallo Welt

Unterüberschrift

Tschüss, Welt!

Text

Ich bin ein Text

Text und Bild - Hallo Welt zwei


Überschrift

Hallo Welt zwei

Text

Text, der zum Bild gehört

Bild



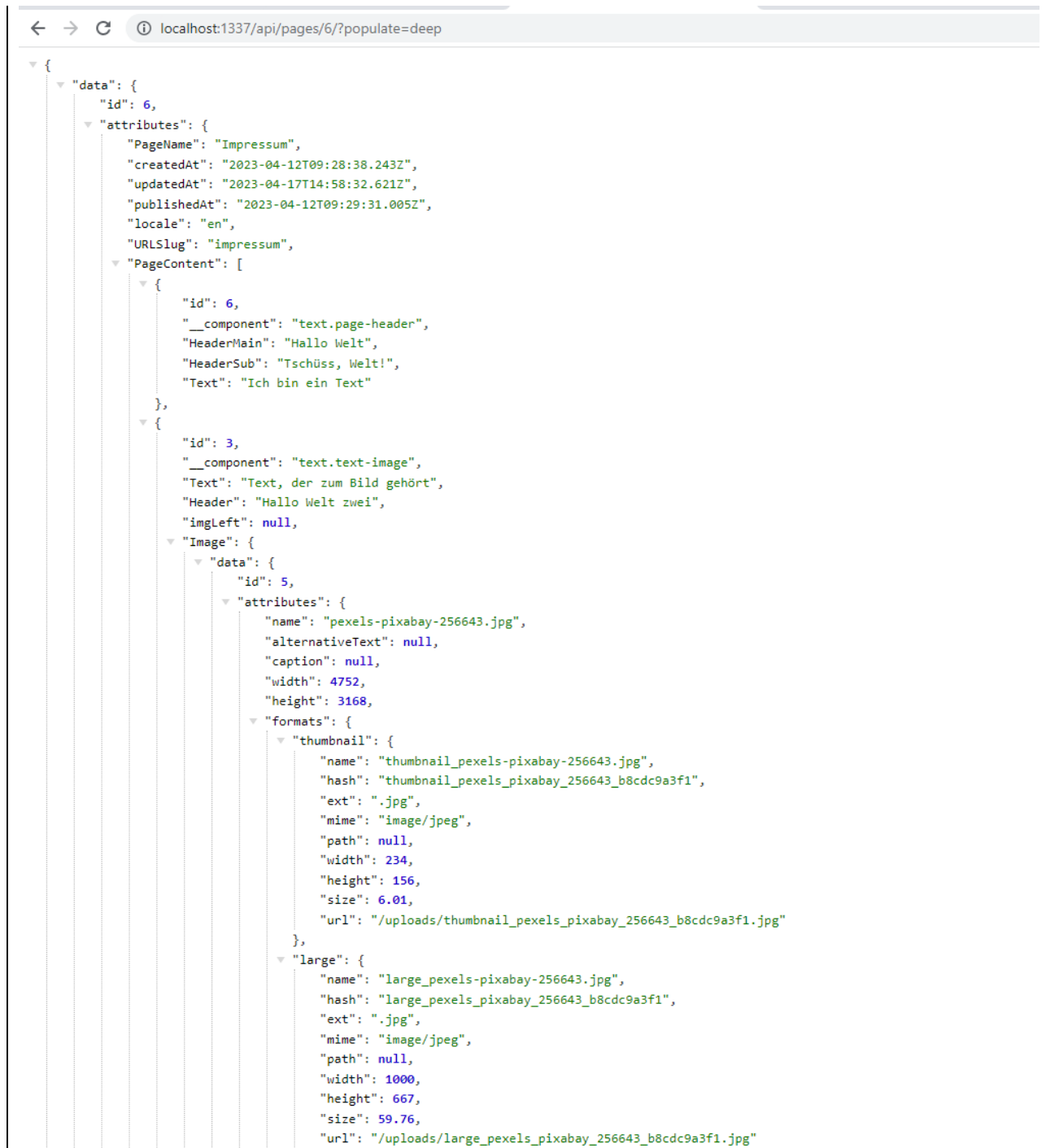
pexels-pixabay-256643.jpg

Bild Links von Text ausrichten?

FALSETRUE

+ Add a component to PageContent

## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS



Screenshot einer Beispielseite mit Überschrift, Text, Bild und JSON Antwort auf Anfrage

## A10.App.js

```
resources > js > JS app.js > ...
jamal harris, 2 days ago | 2 authors (You and others)
1  import { createApp } from 'vue'
2  import App from './VueApp/App.vue'
3  import router from './VueApp/router'
4
5  /* import the fontawesome core */
6  import { library } from '@fortawesome/fontawesome-svg-core'
7  /* import font awesome icon component */
8  import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'
9  /* import specific icons */
10 import { faTwitter, faFacebook, faLinkedin, faXing } from '@fortawesome/free-brands-svg-icons'
11 import { faX } from '@fortawesome/free-solid-svg-icons' | jamal harris, 2 days ago • wip ...
12
13
14 /* add icons to the library */
15 library.add(faTwitter, faFacebook, faLinkedin, faXing, faX )
16
17
18 //form validation
19 import file from './VueApp/custom-form-rules/file'
20
21 import './VueApp/assets/main.css'
22
23 // FormKit imports
24 import { plugin as formKitPlugin, defaultConfig } from '@formkit/vue'
25 import { createMultiStepPlugin } from '@formkit/addons'
26 import '@formkit/themes/genesis'
27 import '@formkit/addons/css/multistep'
28
29 const app = createApp(App).component('font-awesome-icon', FontAwesomeIcon)
30
31 app.use(router)
32 app.use(formKitPlugin, defaultConfig({
33   plugins: [createMultiStepPlugin()],
34   rules: { file }
35 })))
36 ,
37 app.mount('#app')
38
```

*Codeausschnitt der App.js-Datei*

## A11. ViteConfig.js

```
JS vite.config.js > ...
jamal harris, 2 days ago | 2 authors (jamal harris and others)
1 import { defineConfig } from 'vite';
2 import laravel from 'laravel-vite-plugin';
3 //import vue
4 import vue from '@vitejs/plugin-vue';
5
6
7 export default defineConfig({
8   define: {
9     'process.env': {}
10  },
11  plugins: [
12    vue({
13      template: {
14        compilerOptions: {
15          isCustomElement: (tag) => {
16            return tag.startsWith('Formkit') // (return true)
17          }
18        }
19      }
20    }), //ad vue to laravel as plugin
21    laravel({
22      input: ['resources/css/app.css', 'resources/js/app.js'],
23      refresh: true,
24    }),
25  ],
26 });
27
28
29 //vite will watch your resources/js/app.js file and resources/css/app.css file: npm run dev
```

Codeausschnitt der ViteConfig.js-Datei



## A12. Ausschnitt aus Mail-Model

```

100  /**
101   * Get the message content definition.
102   */
103  public function content(): Content
104  {
105      return new Content(
106          view: 'mail.email',
107          with: [
108              'firstname' => $this->firstname,
109              'lastname' => $this->lastname,
110              'email' => $this->email,
111              'phone' => $this->phone,
112              'html_css' => $this->html_css,
113              'php' => $this->php,
114              'mysql' => $this->mysql,
115              'js' => $this->js,
116              'frontendFrameworks' => $this->frontendFrameworks,
117              'wordpress' => $this->wordpress,
118              'typo3' => $this->typo3,
119              'bonusText' => $this->bonusText,
120              'jobapplication' => $this->jobapplication
121          ],
122      );
123  }
124
125  /**
126   * Get the attachments for the message.
127   *
128   * @return array<int, \Illuminate\Mail\Mailables\Attachment>
129   */
130  public function attachments(): array
131  {
132      return [
133          Attachment::fromPath($this->file->path())
134              ->as($this->file->getClientOriginalName())
135              ->withMime($this->file->getClientMimeType())
136      ];
137  }
138  }
139  }

```

Codeausschnitt des Mail-Models

## A13. Mail Controller

```
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;
use App\Mail\ApplicationMail;
...
class SendMailController extends Controller
{
    public function sendmail(Request $request)
    {
        //set in conf
        $to_email = "i@onm.de";
        Mail::to($to_email)->send(new ApplicationMail(
            $request->input('firstname'),
            $request->input('lastname'),
            $request->input('email'),
            $request->input('phone'),

            $request->input('html_css'),
            $request->input('php'),
            $request->input('mysql'),
            $request->input('js'),

            $request->input('frontendFrameworks'),

            $request->input('wordpress'),
            $request->input('typo3'),

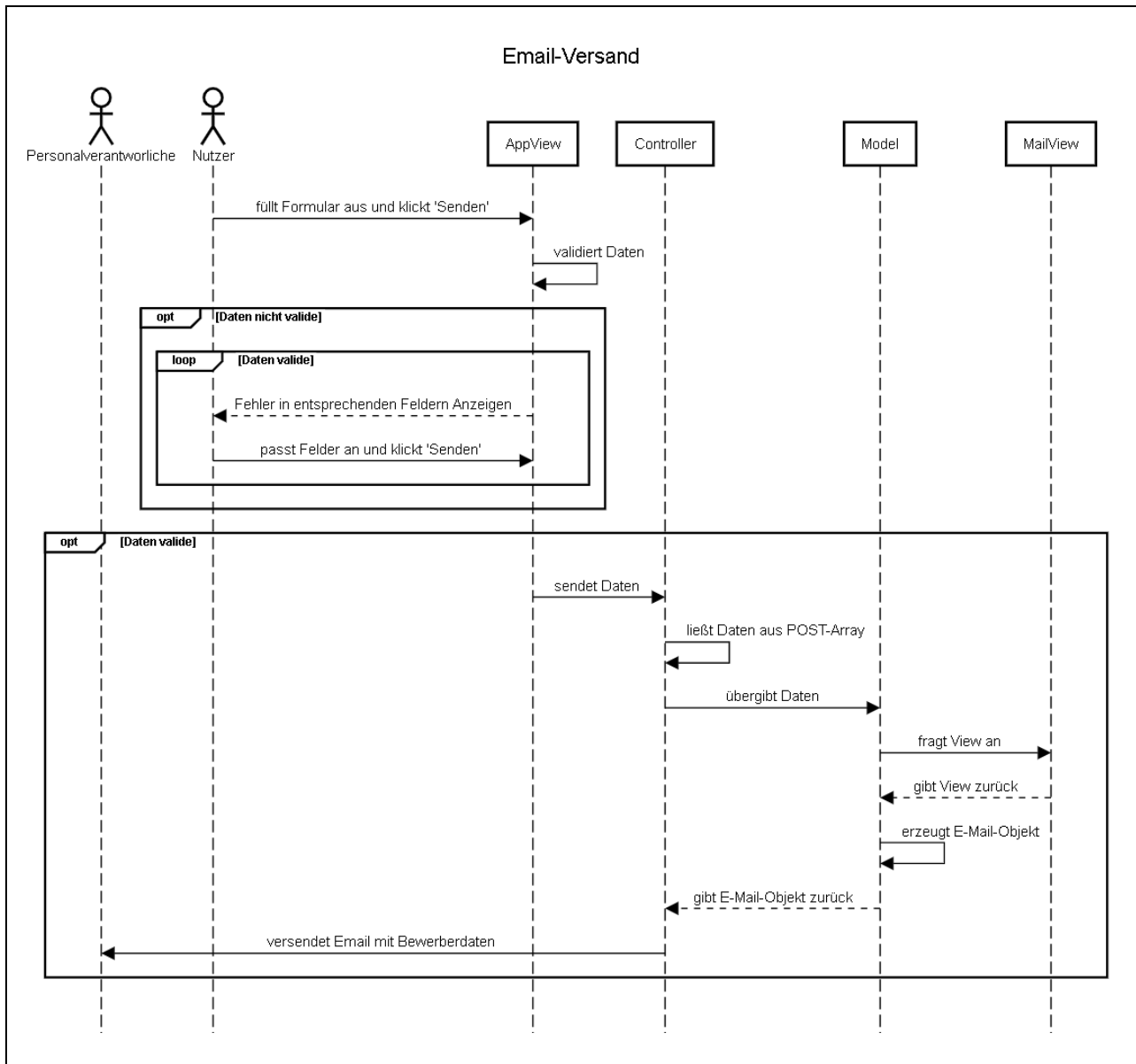
            $request->file('file'),
            $request->input('bonusText'),

            $request->input('jobapplication')
        ));
        return redirect('/')->withErrors(['msg' => 'The Message']);
    }

    //The email sending is done using the to method on the Mail facade
    //Mail::to('test@onm.de')->send(new ApplicationMail($name, $content));
}
```

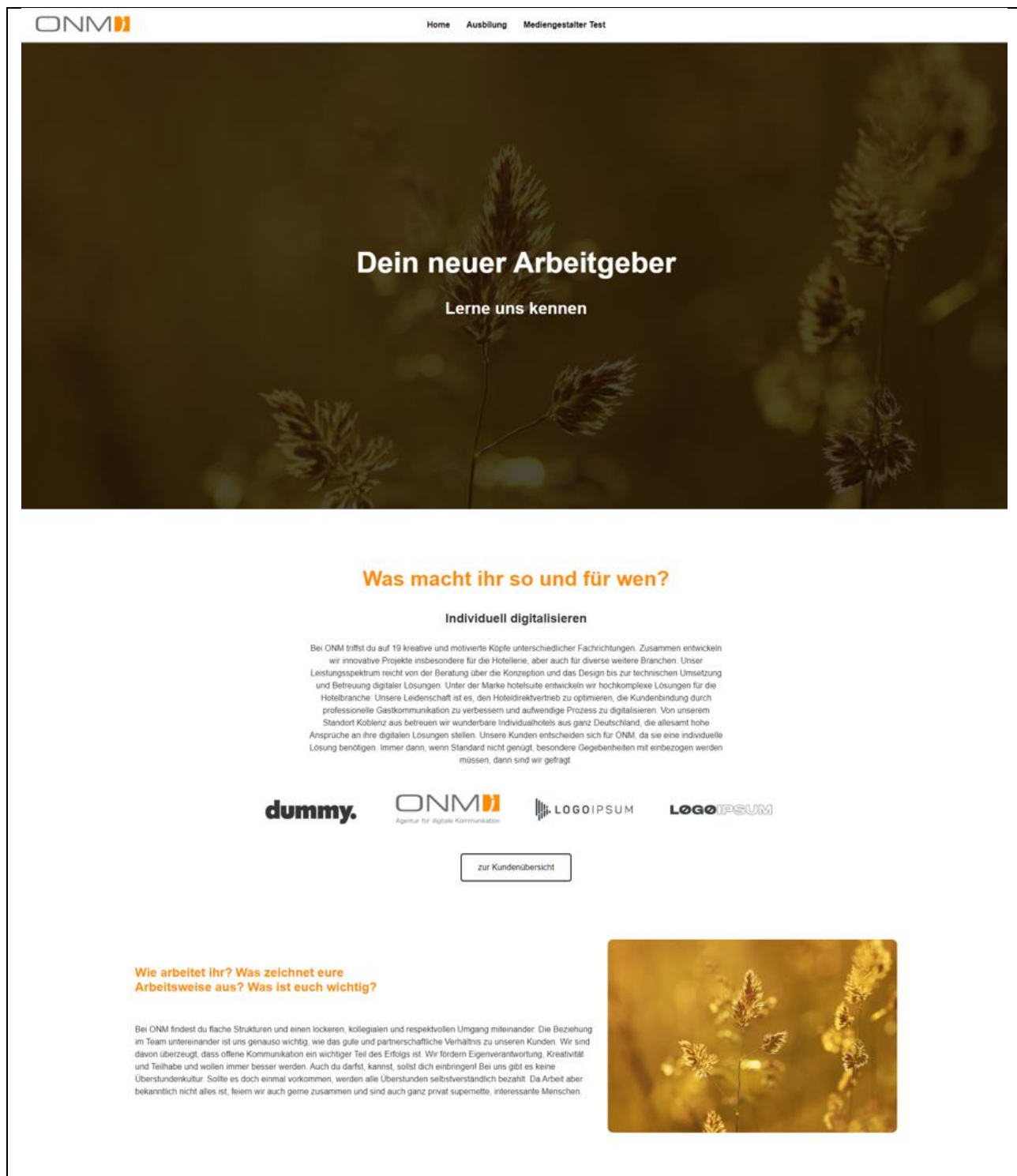
Codeausschnitt des Mail-Controllers

## A14. Sequenzdiagramm E-Mails



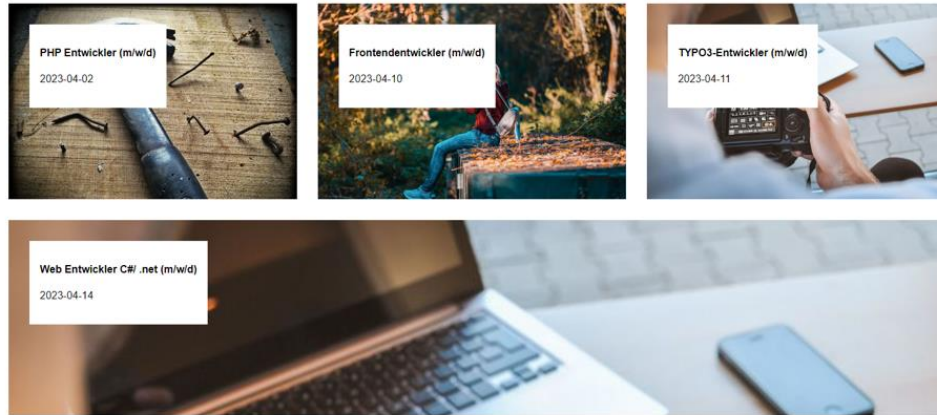
Sequenzdiagramm des Email-Versands

## A15. Ausschnitt Design-Umsetzung Frontpage



## Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

### Festeinstellungen



### Ausbildung



### Stellen alternative Darstellung

Fachinformatiker für Anwendungsentwicklung (m/w/d)	2023-04-25
Mediengestalter Digital- und Print (m/w/d)	2023-04-10
Frontendentwickler (m/w/d)	2023-04-10
PHP Entwickler (m/w/d)	2023-04-02
TYPO3-Entwickler (m/w/d)	2023-04-11
Web Entwickler C#/.net (m/w/d)	2023-04-14

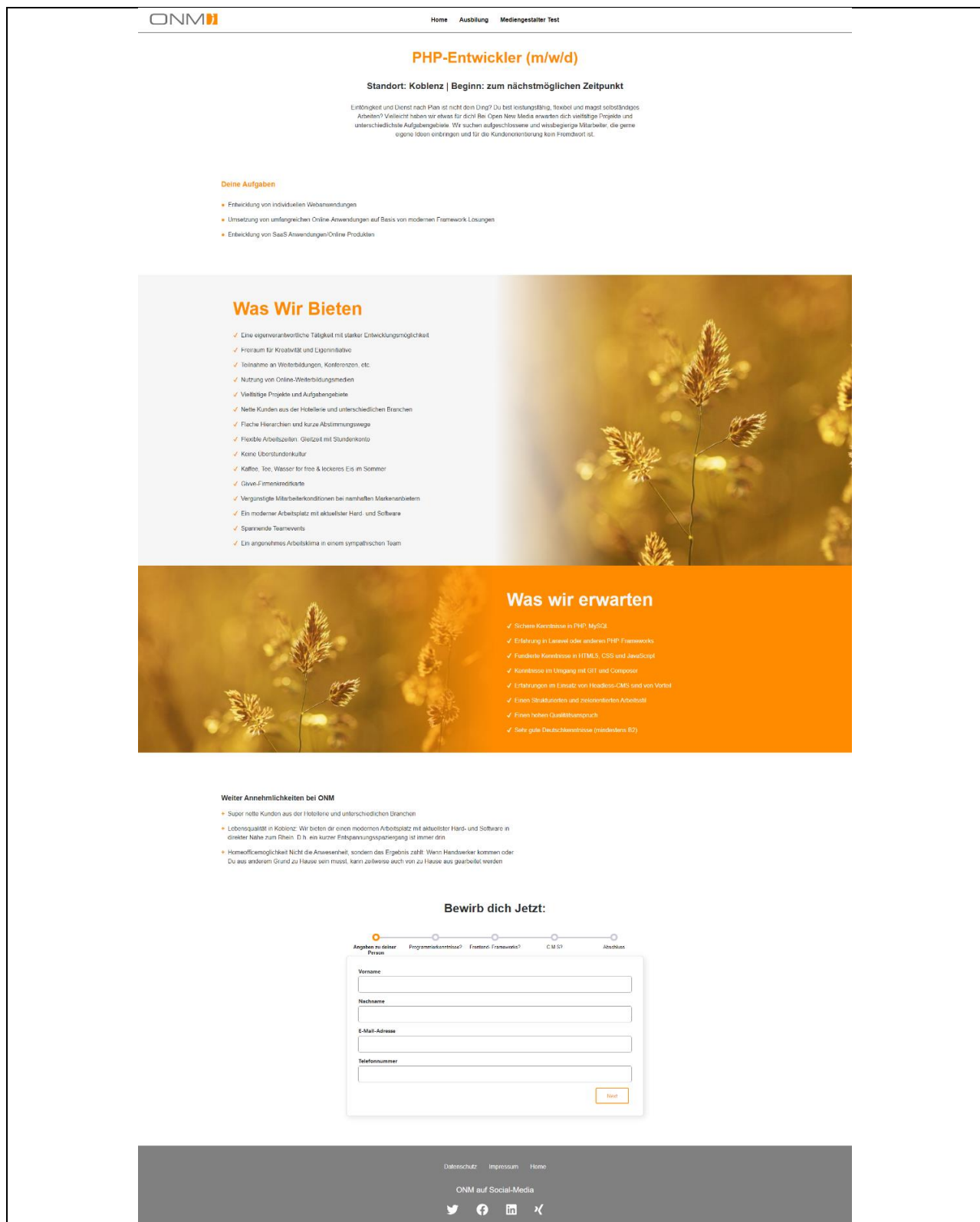
[Datenschutz](#) [Impressum](#) [Home](#)

ONM auf Social-Media



Ausschnitt Design Umsetzung Frontpage

## A16.Design Umsetzung Bewerbungs-Seite



Screenshot Design Umsetzung Bewerbungs-Seite

## A17. Vue Router

```
resources > js > VueApp > router > JS index.js > routes
1  import { createRouter, createWebHistory } from 'vue-router'
2  import HomeView from '../views/HomeView.vue'
3  import PageNotFound from '../views/PageNotFound.vue'
4
5
6  //Dynamic routing: make HomeConst to pass to vue router with dymaic information
7  // const Home = {
8  //   template: HomeView,
9  //   use: template instead
10 // }
11 // these are passed to `createRouter`
12 const routes = [
13   // dynamic segments start with a colon
14   { path: '/:pageName', component: HomeView },
15   { path: '/', component: HomeView, props: { HomePage: 1 } },
16   {
17     path: '/404', component: PageNotFound
18   },
19   {
20     path: '/*:pathMatch(.*)**', component: PageNotFound
21   }
22 ]
23 const router = createRouter({
24   history: createWebHistory(import.meta.env.BASE_URL),
25   routes: routes
26 })
27
28 export default router
29
```

*Codeausschnitt aus der Vue-Router-Datei*

## A18. Erstellen von Seiten Map

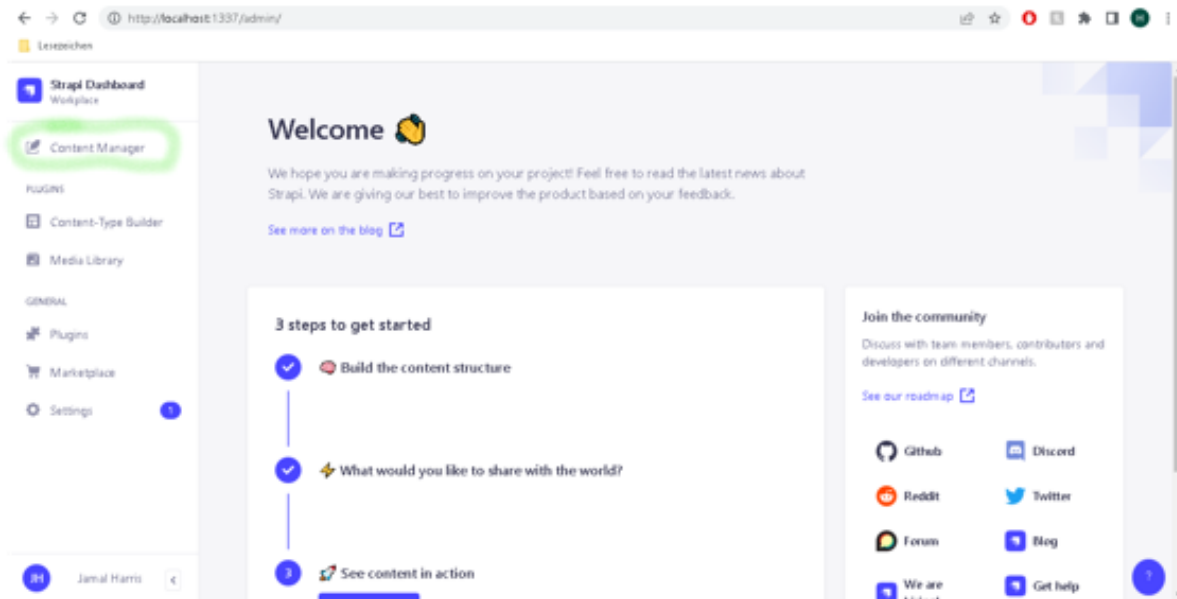
```
224     axios.get(this.dataSrcURL).then((response) => {  
225         this.pages = response.data.data  
226         // var map = new Map() in data return object  
227         this.pages.forEach((el) => {  
228             // this.pages[el.id] = el.attributes.PageName or URL slug if it is set  
229             if (el.attributes.URLSlug != null && el.attributes.URLSlug != "Default") {  
230                 this.map.set(el.attributes.URLSlug, el.id);  
231             } else {}  
232             this.map.set(el.attributes.PageName, el.id);  
233         })  
234     })
```

*Screenshot der Erstellung der Seitenmap*

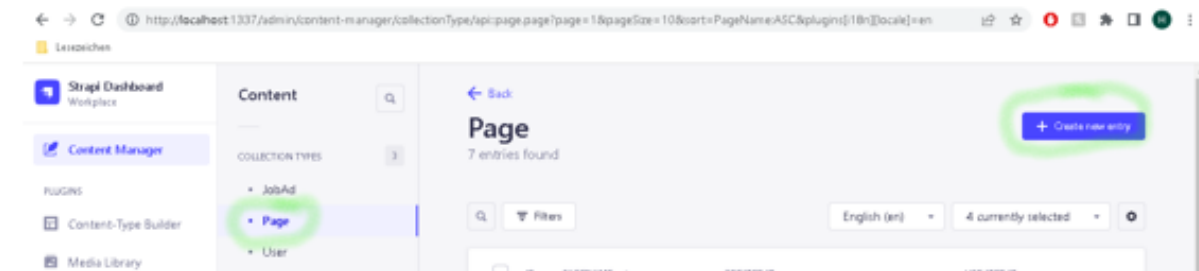


## A19. Ausschnitt Kundendokumentation

Nach Erfolgreicher Anmeldung können nun Seiten erstellt werden. Um Seiten Elemente zu erstellen, wählen sie im Backend den Punkt Content Manager aus:



Nun werden sie zur folgenden Ansicht geleitet, auf der sie in der rechten oberen Ecke auf den Button „+ Create new entry“ betätigen müssen:



Screenshot von Ausschnitt der Kundendokumentation

## A20. Ausschnitt der Entwicklerdokumentation


Requires onmJobs Strapi Instance

### Installation

1. Use Bash to navigate choosable to projekt location
2. Enter `git clone@[THIS_REPOSITORY URL]` use HTTPS or SSH depending on requirements
3. Enter projekt root, install dependencies via command "npm Install"
4. Install onmJobsStrapi Strapi Instance if not done yet
5. In projekt root, run command "php artisan serve" to start php local development server
6. In projekt root, run command "npm run develop" to serve the Vue.js application
7. Enter project root of onmJobsStrapi project, enter start local development server by running CLI command "npm run develop"

### The Strapi Backend

1. Once the onmJobsStrapi projekt is running on a local Server, visit the url shown in CLI
2. If you havent logged in before, your are prompted to create a Admin-User Account.
3. You will gain Access to the Dashbord. From here you can navigate to either the Content Manager, to edit content, or the the content-type-builder, where you can edit exsisting components or create new ones. Every available Option will be visible for you (as an Admin) on the left sidebar:



Screenshot von Ausschnitt der Entwickler Dokumentation

## A21. Projektantrag



Erstellen einer individuellen Bewerberplattform auf Basis eines Headless CMS

## **A22. Formblatt für das Projekt**

