IHK Abschluss-Projekt 2017

Jeremy Seipelt

6. Dezember 2017

Für den Beruf des Fachinformatiker der Anwendungsentwicklung

# **Agentur Website**
## Ein Full-Stack Web-Developer Projekt

Auszubbildner:
Jeremy Seipelt
Pichelsdorferstr. 72
13595 Berlin

Ausbildungsbetrieb:
zmb GmbH
Scharnhorstr. 112
11234 Berlin

# Inhaltsverzeichnis

# 1 Vorwort

Diese Projektdokumentation ist für die Abschlussprüfung der IHK für den Beruf des Fachinformatikers der Anwendungsentwicklung entstanden und beschreibt Vorgehensweisen des Verfassers.
Ausbildungsbetrieb ist die zmb GmbH, ein KMU mit Standort in Berlin Mitte. Die zmb vertreibt, supportet, und entwickelt ihre eigene E-Commerce Plattform, genannt CORE, und geht Projektarbeiten im B2B Bereich für alle gängigen Shopsysteme nach. Dabei legt die zmb Wert darauf alle Wünsche des Kunden, von der Bereitstellung eigener API über Onboarding bis hin zu Zahlungsweisen, zu erfüllen. Dem Kunden werden auch individuelle Lösungen angeboten.

## 1.1 Beschreibung des Projektes

Für die Agenturarbeiten der zmb, soll eine neue Internet-Präsenz aufgebaut werden.

Die neue Seite wird eine SPA um die Seite für den Kunden modern und schnell zu machen. Der Server muss dafür die SPA bereits kompiliert versenden, da Suchmaschinen sich nur die gesendete HTML Datei ansehen. wird ein SSR-Setup auf dem Server benötigt. Im Verlaufe des Projekts wird ein Server gemietet, aufgesetzt und eine Website erstellt auf der potentielle Kunden mit der zmb in Kontakt treten können.

## 1.2 Ziel des Projektes

Am Ende des Projekts soll eine SPA entstehen die mithilfe von SSR auch ein hohes SEO Ranking erreicht wird. Dabei sollen namenhafte Geschwindigkeitsindexe die Seite mit maximalen Punkten bewerten. Die Inhalte der SPA werden über ein selbst auf dem Server installiertes CMS-System eingetragen. Am Ende entsteht eine Website die auch ohne Hilfe von HTML sowie JS und CSS gepflegt werden kann.

## 1.3 Umfeld des Projektes

Das Projekt wurde vom COO Simeon Gerodetti beantragt.
Herr Simeon Gerodetti ist in der zmb für das operative Geschäft, für das Marketing und den Online-Auftritt der zmb zuständig. Des weiteren kümmert er sich um die Partner im B2B Bereich, sorgt für Kundenaufträge des Unternehmens.

Jovan Gerodetti ist der Softwareentwickler für Frontend, sowie der Spezialist in Angular. Er baut neue UIs, Schnittstellen für CORE und stellt eine Beratende Funktion für das

Projekt dar.

Für das Projekt sind die Zuständigkeiten wie folgt:
- Simeon Gerodetti für Design
- Jovan Gerodett für Implementierung

- die Abnahme des Projektes wird von Simeon Gerodetti vollzogen.

### 1.4 Projektentscheidung

Das Hauptgeschäft besteht aus dem Verkauf und der Entwicklung der Inhouse E-Commerce Lösung 'CORE', welche von 3 Personen gewartet wird. 'Core' wird bisher nur als SaaS Lösung eingesetzt und deshalb liegt der administrative Aufwand bei der zmb.

Den weiteren Schritt in die Richtung der Agenturarbeit geht die zmb mit der zmb.agency Website, dessen Unterbau durch dieses Projekt entsteht.

### 1.5 Beschränkungen des Projektes

Das Projekt beschränkt sich auf das Aufsetzen, sowie das Implementieren einer kleineren Version der Agentur Website, die schon alle technischen Voraussetzungen und ein Bruchteil des Inhaltes enthält. Deswegen wird der Großteil der Layout und CSS Arbeiten, die im größeren Umfang ein Teil sind, das Anlegen von Inhalten für die Seite nicht in die Projektzeit eingerechnet.

## 2 Projekt-Planung

Das Projekt wurde auf 70h für die Anforderungen der IHK angesetzt.

### 2.1 Projektverlauf

Ein Übersicht der Hauptphasen sind der Tabelle 1 zu entnehmen. Eine detaillierte Ansicht der Zeitplanung ist in der B.1.1 auf Seite 13 hinterlegt.

### 2.2 Ressourcenplanung

Bei der Auswahl der Ressourcen wurde Wert auf Open Source Lösungen gelegt, um Projektkosten möglichst gering zu halten. Die Liste der benutzten Software ist aufSeite 15 zu finden.

| Phase | Dauer in Stunden |
|---|---|
| Analyse | 8 |
| Entwurf | 7 |
| Implementierung | 23 |
| Testen | 13 |
| Dokumentation | 8 |
| Gesammt | 64 |

Tabelle 1: Übersicht der Zeitplanung

### 2.3 Entwicklungsprozess

Die Entwicklung der Website wird per Wasserfall-Modell abgewickelt. Dabei werden die einzelnen Phasen des Projektverlaufs nacheinander durchlaufen. Nach jeder Phase wird mit Simeon Gerodetti Rücksprache über den Stand der Dinge gehalten, Jovan Gerodetti übernimmt dessen Rolle in der Implementierung.
Das Wasserfall-Modell bietet sich für kleine Software-Projekte an, die einen klaren Endzustand haben. Da die Anforderungen schon niedergeschrieben sind, können diese in der Entwurfsphase mit einbezogen werden.

Eine Agile Entwicklung ist somit nicht Sinnvoll, und würde nur den Aufwand erhöhen.

## 3 Analysephase

### 3.1 Ist-Analyse

Die zmb ist nur als Firma für den Vertrieb und Wartung von Shop Systemen bekannt, allen voran ihr eigenes 'CORE'. Die Seite informiert nicht über die Agenturarbeit, die die zmb anbietet. Zu Beginn des Projektes steht nur ein gemieteter AWS Server zur Verfügung, der eine minimale Version der Linux Distribution Ubuntu als Betriebssystem nutzt.

### 3.2 Wirtschaftlichkeit

Aufgrund des Umstieges auf Agenturarbeit, muss auch eben diese im Internet beworben werden. Momentan kommen Aufträge nur aus dem Partnernetzwerk und nicht auf dem direkten Weg zur zmb, deswegen ist die neue Internetpräsenz zwangsweise erforderlich.

### 3.2.1 Kosten

Die Ressourcenplanung besteht aus Personal und Sachkosten.

Für das Projekt werden 3h für Beratung durch einen Senior Developer eingerechnet.

Für die Entwicklung wird auf Open Source Lösungen gesetzt. Eine genau Liste der Verwendeten Software auf dem Server wird in Tabelle 15 aufgelistet

| **Personalkosten** | | **Sachkosten** | |
|---|---|---|---|
| Entwickler | 70*10€ = 700€ | AWS Lightsail Server | 5€/mtl |
| COO | 2*30€ = 60€ | Domain | 20€/Jahr |
| Beratung durch Seniorentwickler | 3*30€ = 90€ | Software Lizenzen | 0€ |
| Gesammt Fix | 850€ | Gesammt Monatlich | ~7€ |

Tabelle 2: Kostenplanung

### 3.2.2 Amortisation

Die Seite amortisiert sich durch die Anzahl an Stunden Agenturarbeit die durch sie vergeben wurde. Zur Berechnung der Stunden werden die Kosten aus der Ressourcenplanung genommen. Für Agenturarbeit wird eine Pauschale von 100 €/h genommen.

$$x = 850/100 = 8.5$$

Somit Rechnet sich die Seite bereits nach 9h Agenturarbeit. Das entspricht 9 Kunden im Worst-Case-Szenario die durch die Seite Agenturarbeit buchen.

## 3.3 Anforderungen

Angefordert wurde eine neue Website.

Diese Websie muss höchste Punktzahl in verschiedenen Tempo Benchmarks bringen, um den Besucher zu zeigen das die zmb modernste Technologien beherrscht.

Dabei soll eine SPA entstehen, damit der Benutzer wenn er auf der Seite navigiert nicht den DOM bei jedem Request neu berechnen muss. Zur Erstellung der SPA soll Angular verwendet werden, da es sich hierbei um eine bereits im Unternehmen etabliertes Framework handelt.

Damit Crawler etc unsere Seite/SPA indexieren können muss die Seite bereits auf dem

Server vorgerechnet werden um einen fertigen DOM an den Crawler zu senden. Der Prototyp soll dem Mockup von Simeon Gerodetti ähneln.
Das Mockup ist auf Seite **??** einzusehen.
Für den Anfang wird nur das Hero Image, sowie das Kontaktformular auf der Landingpage benötigt.

### 3.3.1 SSR PoC

Zur Analyse gehörte auch der Beweis das der SSR Prozess funktioniert. Die von Angular bereitgestellte Projekt Repo[1] konnte den Vorgang des serverseitigen Renderns, zur Zufriedenstellung Jovan Gerodettis, vorzeigen.

# 4 Entwurfsphase

Die Anforderungen verlangen eine Website, dadurch ist die Wahl der Zielplattform der Webbrowser[2]. Dabei verwenden wir soviel es geht TypeScript[3], da es sich hierbei um eine in der Firma etablierte Sprache handelt. TypeScript wird deswegen sowohl auf Client als auch Server Ebene verwendet und vorher zu JavaScript kompiliert.

## 4.1 Server

Der Server ist ein von Amazon gebuchter Lightsail[4] Server, der über ein Linux OS mit Ubuntu als Distribution verfügt. Die auf Seite 15 für den Server markierten Programme sollen auf dem Server installiert werden.

### 4.1.1 CMS

Geplant ist ein CMS System auf dem Server (Drupal) mit Datenbank(MYSQL).
Dabei soll für jede Komponente der Website, die auf Seite 16 referenziert wird, ein Inhaltstyp erstellt werden.
Die Inhalte sollen über ein REST-Modul in Drupal dann, per REST API, für die SPA erreichbar sein. Diese Inhalte werden in dem JSON[5] Format versendet. Geplant sind vorerst folgende REST Schnittstellen:

[1] https://github.com/angular/universal-starter
[2] Alle Webbrowser ab IE9
[3] https://www.typescriptlang.org/
[4] https://amazonlightsail.com/
[5] https://www.json.org/

| Schnittstelle | Endpunkt |
|---|---|
| Inhalte einer bestimmten Seite | **export/pages/SEITENNAME** |
| Inhalte aller Seiten | **export/pages/all** |
| Inhalte einer bestimmten Sektion | **/export/paragraphs/ID** |
| Inhalte aller Sektionen | **export/paragraphs/all** |
| Inhalte der Kopfzeile | **export/headbar** |

Tabelle 3: REST-API

### 4.1.2 HTTP/HTTPS

Der Server soll über HTTP/HTTPS erreichbar sein. Der Server muss unterscheiden, ob der Benutzer die Agentur SPA anfragt, oder das CMS System. In Anbetracht der weiteren Entwicklung wurde eine zweite Instanz der Webseite erstellt, um Produktions Website von der Entwicklung Website zu trennen. Deswegen werden auf dem Server folgende Routen eingestellt:

| Anwendung | Route |
|---|---|
| Drupal | **/drupal** |
| SPA/Agentur Website | **/** |
| SPA/Entwicklung | **/dev** |

Tabelle 4: Routen

### 4.1.3 Node

Um Javascript auf einem Server ausführen zu können, muss ein Node Server vorhanden sein. Der Node Server muss jede auf ihm zeigende Route abfangen und die zugehörige Route der SPA bereitstellen. Hier findet der eigentliche SSR Prozess statt. Dem Node Server wird ganze SPA bereitgestellt. Dafür wird das Express Framework verwendet welches HTTP Anfragen an den Node Server verarbeitet. Damit Node mit der SPA funktioniert muss ein AOT Compiler die Anwendung für den Server kompilieren.

## 4.2 Single Page Application

Das Framework welches für die SPA verwendet wird ist Angular[6], welches die zmb seit Jahren produktiv einsetzt. Bei Angular wird der Fokus auf wiederverwendbare Komponenten gesetzt die, wie native DOM-Element, in das HTML eingebunden werden.

Dafür wurde der generelle Seiten-Aufbau in der Grafik auf Seite 16 festgelegt. Deswe-

[6] https://angular.io

gen werden Komponenten entworfen, die für einzelne Passagen der Website stehen. Um die Rest-API vom CMS konsumieren zu können, benötigen wir noch ein Service[7]. Der Entwurfsplan der SPA ist auf der Seite 17 hinterlegt.

# 5 Implementierungsphase

## 5.1 Server

### 5.1.1 Aufsetzen des Servers

Der gebuchte Lightsail Server wird nur mit einer minimalen Installation ausgeliefert. Als erstes werden die benötigten Programme installiert. Dabei werden die unter Server auf Seite 15 aufgeführten Programme über den Ubuntu Package Manager apt[8] installiert. Für die einfache Arbeit mit dem Server wurden symbolische Links auf wichtige Serverressourcen in das Home Verzeichnis des Users angelegt, wie auf dem Sreenshot auf Seite 23 .

### 5.1.2 NGINX

Der HTTPS Teil des Servers wird von NGINX gesteuert. Nach der Installation des NGINX, muss der NGINX Service gestartet werden. Sobald der NGINX läuft kann der Server konfiguriert werden. Da Drupal und Node auf dem selben Server liegen, müssen für beide unterschiedliche Routen in der Konfiguration angelegt werden wie auf Seite 23. Die komplette NGINX Konfiguration ist im Extra Source Code Anhang vorzufinden.

### 5.1.3 CMS

Nach dem Download von Drupal und der einrichtung von MySQL, wird Drupal über einen Aufruf im Webbrowser installiert. Die benötigten Module wurden eingespielt und die Inhaltstypen angelegt. Die Inhaltstypen mit ihren Feldern sind auf Seite 21 ff. zu sehen.
Die REST Schnittstelle ist über das REST UI Modul von Drupal erstellt worden, ein Screenshot der erstellten Schnittstellen ist auf Seite 20 zu sehen.

### 5.1.4 Node

Neben Node wurde noch PM2[9] installiert um den Node Prozess bei Abstürzen automatisch neu zu starten. Die Installation von Pm2 sowie aller von der SPA benötigten Pakete erfolgt über den Node Package Manager. Dieser installiert automatisch alle Pakete die als

[7] https://angular.io/guide/dependency-injection
[8] https://wiki.ubuntuusers.de/APT/
[9] http://pm2.keymetrics.io/

Abhängigkeit in der *package.json* angegeben wurden.
Der Inhalt der package.json ist dem Source Code zu entnehmen.

### 5.2 SPA

Für die SPA wurde zuerst ein Grundgerüst bestehend aus einem Root Modul erstellt.
Die Seitenkomponenten werden im Drupal Modul deklariert. Dieses Modul besteht aus mehreren Komponenten und einem Drupal Service.

Die Komponenten der Sektionen bekommen ihre Informationen über die Page Komponente. Über das Angular Data Binding[10] kann das Template mit den erwarteten Daten im Dekorierer der Komponente eingepflegt werden.

Jede Komponente bekommt eine eigene *.css* Datei, damit die CSS Regeln nur auf diese greifen. Das Aussehen der Seite wurde in SASS implementiert, welches CSS um Konzepte aus der Programmierung, wie Variablen, erweitert. Mithilfe der Developer Tools von Google Chrome konnte CSS für die Website erstellt werden, ohne die SPA neu zu kompilieren.

Die SPA wird für den Client als eine *bundle.js* Datei ausgeliefert mithilfe von Webpack[11].
Am Ende wurde noch der Taskrunner Gulp hinzugefügt, damit die benötigten Kommandos zum Bauen der SPA auf ein minimum reduziert werden.
Die Einstellungen Gulps sind im Source Code hinterlegt.

## 6 Abschlussphase

In folgenden werden die abschließenden Arbeiten beschrieben die den pagespeed und die responsiveness der SPA behandeln.

### 6.1 Pagespeed

Um den maximalen Pagespeed zu erreichen, musste auf dem Server der Cache und gzip aktiviert werden, die gesendeten Dateien wurden vorher nicht komprimiert oder gecacht.
Die Performanz des Audits konnte aufgrund der Limitierungen der benutzen Pakete/Libraries/Frameworks nur auf 69/100(zufriedenstellend) erhöht werden.

### 6.2 Responsiveness

Die Responsiveness der Website sollte durch die Verwendung von relativen Größen sichergestellt werden. Diese Annahme erwies sich jedoch als falsch, denn was auf einem großem

[10] https://angular.io/guide/template-syntax#binding-syntax-an-overview
[11] https://github.com/webpack/webpack

Bildschirm groß ist, ist dann auf einem kleinen Bildschirm klein. Deswegen wurde um die Werte adjustieren zu können dem SASS feste Breiten, für verschiedene Endgeräte, als Variablen angelegt.
Die einzelnen Komponenten wurden dann auf Handy/Laptop/Computergröße fein justiert.

# 7 Dokumentation

Die Dokumentation besteht aus zwei Teilen einer Projektdokumentation und einer Entwicklerdokumentation.

## 7.1 Projektdokumentation

Die Projektdokumentation besteht aus einer genauen Beschreibung der für die Ausführung des Projektes benötigten Phasen, sowie Information zur in der Phase benutzten Technologie. Die Dokumentation wird mit LaTeXgeschrieben um ein einheitliches Layout zu gewähren.

## 7.2 Entwickerdokumentation

Die Entwicklerdokumentation besteht aus einer automatisch generierten Doku unter Verwendung von Compodoc[12], welcher alle Informationen aus dem Sourcecode bezieht und diese für Entwickler aufbereitet. Diese Dokumentation ist aus dem Sourcecode für alle Entwickler kompilierbar. Screenshot im Anhang B.3.5 auf Seite 22.

# 8 Endstand

Das Projekt wurde mit einer funktionstüchtigen Website abgeschlossen. Die Maximale Anzahl an Stunden wurde nicht überschritten und der gesammte Planungs Puffer wurde aufgebraucht. Eine genaue Übersicht der Benötigten Zeit ist auf Seite 14 zu finden.

## 8.1 Ist/Soll-Zustand

Der Ist-Zustand wird in den folgenden Abschnitten mit den Anforderungen(Soll-Zustand) verglichen.

### 8.1.1 Geschwindigkeit

Die Website schneidet mit höchsten Punktzahlen bei den ausgewählten Benchmarks ab. Die Berichte für Pingdom und PageSpeed Insight sind auf Seite 18/19. Die Google Audits

[12] https://compodoc.github.io/compodoc/

für Performance konnten nicht im Zeitrahmen des Projekts in den grünen Bereich gebracht werden. Übersicht des Audits ist auf Seite 20 zu sehen.

### 8.1.2 Layout

Das fertige Layout ist auf 3 verschiedenen Endgeräten auf Seite 23 zu sehen. Die Navigation in der Kopfzeile fehlt, da es noch keine Navigation auf der Seite gibt. Problematisch ist die weiße Überschrift, welche wenn sie die Erde berührt nur schwer zu Lesen ist.

### 8.1.3 SPA

Die geschriebene SPA ist in der Lage die Seite in gewünschter Form anzuzeigen. Die Screenshots aus dem Layout Teil sind direkt in der SPA entstanden. Die SPA muss noch von ungenutzten Funktionen, die während der Implementierung entstanden sind, gesäubert werden. Der Taskrunner Gulp führt noch eine eigenartige Reihenfolge der Aufgaben aus. Es funktioniert, doch die Ausgaben der Zeiten einzelner Schritte ergeben wenig Sinn.

### 8.1.4 SSR

Das serverseitige Rendern funktioniert, jedoch entsteht ein Flackern wenn der Serverseitige DOM, durch den Angular Bootstrapping auf dem Client verworfen wird. Zwar wurde das Preboot Modul nicht in den Anforderungen aufgelistet, jedoch sollte es innerhalb der Projektzeit implementiert werden, um den zuvor genannten Fehler zu beheben.

### 8.1.5 CMS

Das CMS Drupal ist auf dem Server installiert und ist über eine eigene URL auf der Domain der Agenturseite freigeschaltet. Inhalte werden bereits für die Seite erstellt und gepflegt. Alle Inhalte aus dem Layout wurden über Drupal eingestellt.
Die Möglichkeit auch das CSS zu editieren existiert aus dem CMS heraus noch nicht.

## 8.2 Quo Vadis ?

Nach Abschluss ist noch eine Menge für die Seite zu tun. Das CMS ist zwar vorhanden, besitzt aber kaum Einträge oder Übersetzungen in andere Sprachen. Somit muss noch ein Übersetzung Service in Angular gebaut und Inhalte erstellt werden. Das Bootstraping der SPA verwirft den DOM zurzeit, deswegen ist der Einbau des PrebootModuls gerade an oberster Priorität. Geplant ist auch das Verwenden von Service Workers[13]. Eine Version der Seite soll Google Amp[14] unterstützen.

[13] https://developers.google.com/web/fundamentals/primers/service-workers/
[14] https://www.ampproject.org/

## 8.3 Fazit für zukünftige Projekte

Das Benutzen einer AOT kompilierten Angular Application für serverseitiges Rendering ist noch eine relativ neue Technologie und ist deshalb noch nicht gut Dokumentiert. Es gibt kaum Best Practices oder Tutorials. Wenn man Pech hat ist der Fehler der gerade auf dem Bildschirm angezeigt wird, der erste seiner Art.
Für die Zukunft sollte man für dringende einsatzfähige Software auf etablierte und getestete Systeme setzen, um nicht vor einer Sackgasse zu Enden. Beim nächsten Projekt sollte der etablierte Teil stärker sein, als der neue Technologie Teil, um keine hohe Zeiten bei der Fehlersuche zu verschwenden.

# A Akronyme

## A.1 Allgemein

**B2B** Business to Business

**KMU** Kleines Mittelständisches Unternehmen

## A.2 Technologie

**SSR** Server Sided Rendering

**SPA** Single Page Application

**DOM** Document Object Model

**CMS** Content Management System

**AOT** Ahead of time

**SASS** Super Awsome Style Sheets

**API** Application Programming Interface

**UI** User Interface

# B Abbildungen

## B.1 Tabellen

### B.1.1 Detaillierte Übersicht der Aufgaben

| Teilaufgabe | Zeit in Stunden |
|---|---|
| Analyse | 8 |
| Serverseitiges Rendern | 4 |
| Belesung | 1 |
| PoC anhand einer Implementierung in Angular mit Typescript | 2 |
| Anforderungen an den Server | 1 |
| Drupal | 4 |
| Einarbeitung ins System | 2 |
| Benötigte Module | 1 |
| Benötigte Inhaltstypen | 1 |
| Entwurf | 7 |
| SSA | 4 |
| Entwurf der Klassen | 2 |
| Entwurf der Componenten Templates | 2 |
| Entwurf für Drupal | 3 |
| Inhaltstypen | 1 |
| REST-API | 2 |
| Implementierung | 28 |
| Webserver | 8 |
| Webserver buchen + hochfahren + konfigurieren | 1 |
| Nginx installieren + konfigurieren | 2 |
| Mysql installieren + konfigurieren | 1 |
| PHP/FastCGI installieren + konfigurieren | 1 |
| Drupal installieren + konfigurieren | 2 |
| Node/PM2 für SSA installieren | 1 |
| Drupal | 7 |
| Inhaltstypen implementieren | 3 |
| REST-API implementieren | 2 |
| Content erstellen | 2 |
| SSA in Angular mit Typescript | 13 |
| Umschreiben der PoC Implementierung als SSA(aus der Analyse) | 1 |
| Implementierung der geplanten Klassen | 2 |
| Anpassung der Componenten Templates + SASS | 5 |
| Main.js für den Node erstellen | 2 |
| Compiler einstellen(NGC+TSC+SASS) | 1 |
| Taskrunner(Gulp) einstellen | 2 |
| Testen | 13 |
| Pagespeed testen* | 10 |
| Responsiveness testen | 3 |
| Dokumentation | 8 |
| Projektdokumentation | 6 |
| Entwicklerdokumentation | 2 |
| Gesammte geplante Projektdauer | 64 |
| Maximale Projektdauer | 70 |
| Puffer | 6 |
| | |
| *Der Pagespeed wird sobald die Website auf dem Server ist getestet.<br>Hier werden ggf. Pagespeed Optimierungen implementiert, wenn sie erforderlich sind.<br>(Caching,Minify,Uglyfy,Compression etc) | |

### B.1.2 Zeitplan

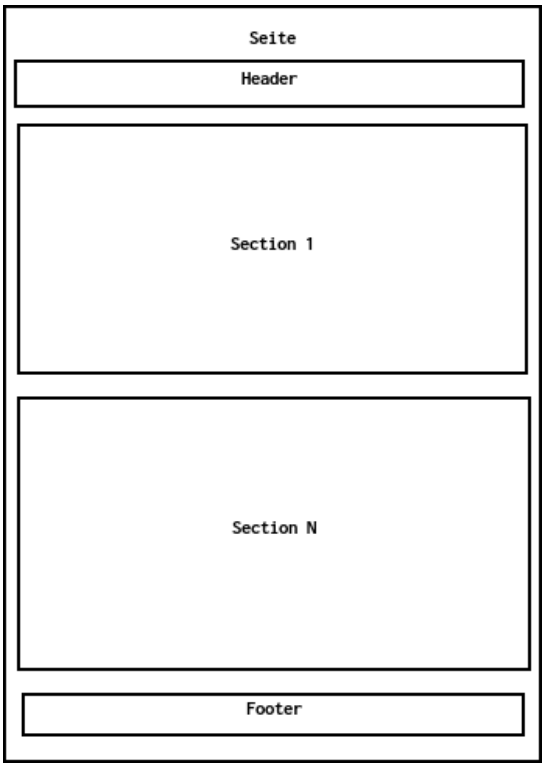
| Teilaufgabe | Zeit in Stunden | Benötigt in Stunden |
|---|---|---|
| Analyse | 8 | 10 |
| Serverseitiges Rendern | 4 | 5 |
| Belesung | 1 | 1 |
| PoC anhand einer Implementierung in Angular mit Typescript | 2 | 3 |
| Anforderungen an den Server | 1 | 1 |
| Drupal | 4 | 5 |
| Einarbeitung ins System | 2 | 2 |
| Benötigte Module | 1 | 2 |
| Benötigte Inhaltstypen | 1 | 1 |
| Entwurf | 7 | 4 |
| SSA | 4 | 2 |
| Entwurf der Klassen | 2 | 1 |
| Entwurf der Componenten Templates | 2 | 1 |
| Entwurf für Drupal | 3 | 2 |
| Inhaltstypen | 1 | 1 |
| REST-API | 2 | 1 |
| Implementierung | 28 | 40 |
| Webserver | 8 | 12 |
| Webserver buchen + hochfahren + konfigurieren | 1 | 1 |
| Nginx installieren + konfigurieren | 2 | 3 |
| Mysql installieren + konfigurieren | 1 | 2 |
| PHP/FastCGI installieren + konfigurieren | 1 | 1 |
| Drupal installieren + konfigurieren | 2 | 4 |
| Node/PM2 für SSA installieren | 1 | 1 |
| Drupal | 7 | 6 |
| Inhaltstypen implementieren | 3 | 2 |
| REST-API implementieren | 2 | 3 |
| Content erstellen | 2 | 1 |
| SSA in Angular mit Typescript | 13 | 22 |
| Umschreiben der PoC Implementierung als SSA(aus der Analyse) | 1 | 3 |
| Implementierung der geplanten Klassen | 2 | 3 |
| Anpassung der Componenten Templates + SASS | 5 | 6 |
| Main.js für den Node erstellen | 2 | 3 |
| Compiler einstellen(NGC+TSC+SASS) | 1 | 4 |
| Taskrunner(Gulp) einstellen | 2 | 3 |
| Testen | 13 | 8 |
| Pagespeed testen* | 10 | 4 |
| Responsiveness testen | 3 | 4 |
| Dokumentation | 8 | 8 |
| Projektdokumentation | 6 | 15 |
| Entwicklerdokumentation | 2 | 1 |
| Gesammte geplante Projektdauer | 64 | 70 |
| Maximale Projektdauer | 70 | 70 |
| Puffer | 6 | 0 |
| | | |
| *Der Pagespeed wird sobald die Website auf dem Server ist getestet.<br>Hier werden ggf. Pagespeed Optimierungen implementiert, wenn sie erforderlich sind.<br>(Caching,Minify,Uglyfy,Compression etc) | | |

### B.1.3 Programme

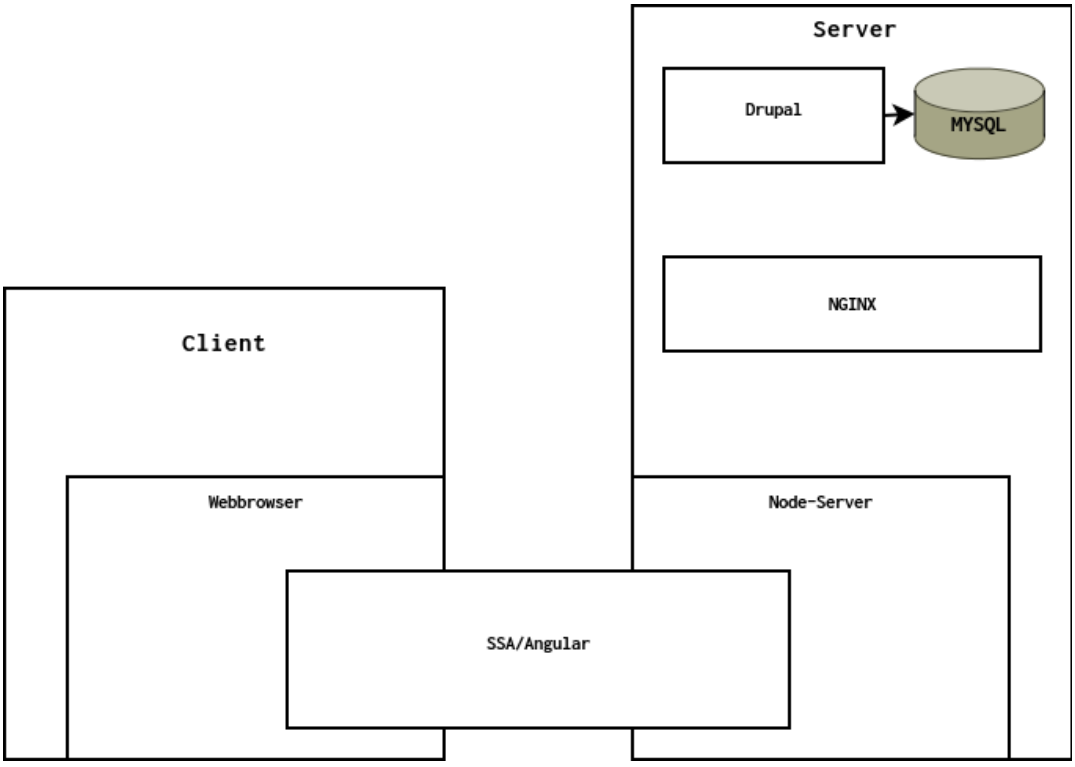
| Architektur | Programm | Verwendung |
|---|---|---|
| Server | | |
| | MYSQL | Datenbak |
| | Drupal | CMS |
| | NGINX | HTTP/HTTPS |
| | Node | Javascript im Backend |
| | Pm2 | ProzessManager |
| Entwicklung | | |
| | Node | Testen der Application |
| | Atom | IDE |
| | Chrome | Webbrowser |
| Alle Programme verwenden eine Open Source Lizenz | | |

## B.2 Grafiken

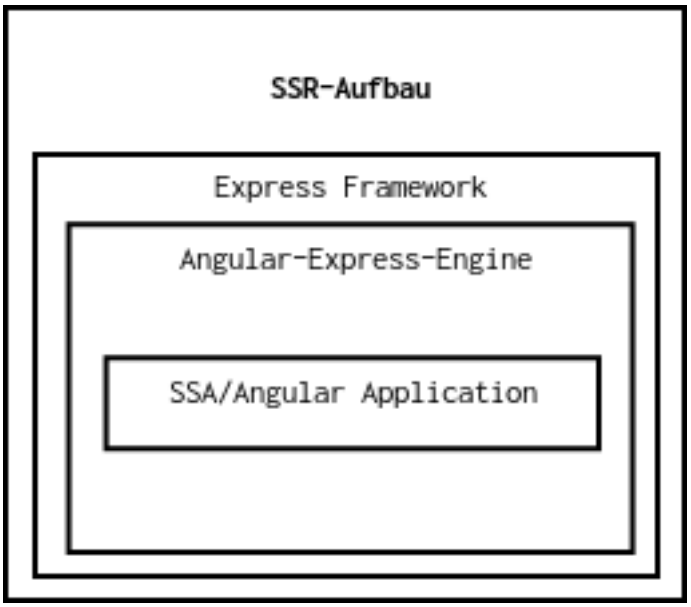
### B.2.1 Aufbau der Seite



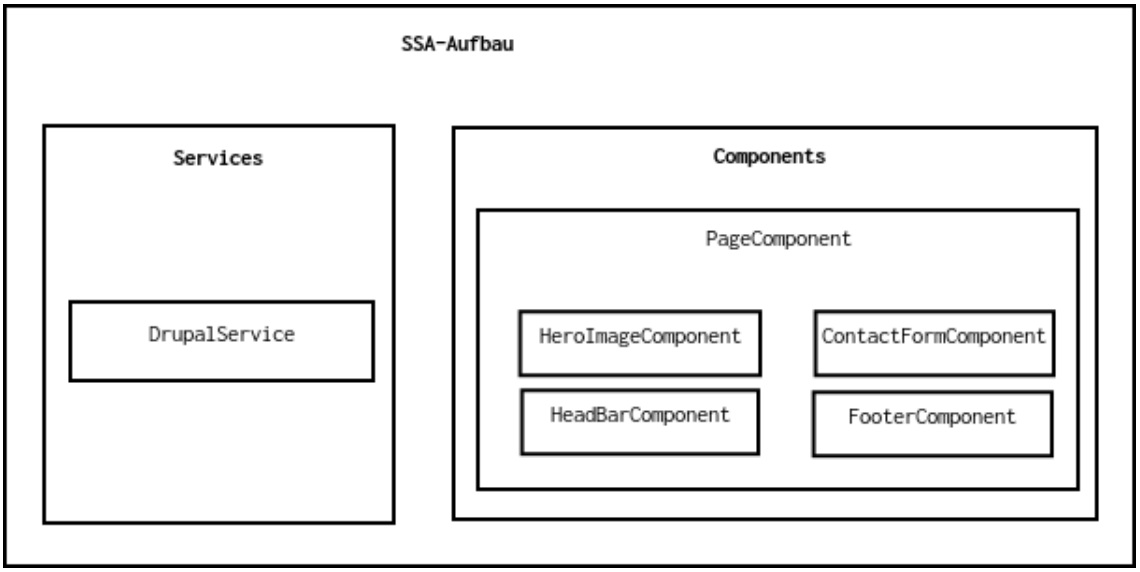

### B.2.2 Projekt Architektur



### B.2.3 SSR Node Entwurf




### B.2.4 SPA Entwurf




### B.2.5 Website Mockup

Aufgrund der Größenlimitierung der PDF muss das Bild im Webbrowser Aufgerufen werden.
https://goo.gl/qfjDmC

### B.2.6 PageSpeed Insights

PageSpeed Tools > Insights

LEITFÄDEN REFERENZ BEISPIELE SUPPORT

## PageSpeed Insights

http://52.29.161.188/agency/ ANALYSIEREN

Mobil Desktop

**Good**

100 / 100


Gut gemacht. Auf dieser Seite werden die meisten Best Practices zur Leistung umgesetzt, sodass eine gute Nutzererfahrung gewährleistet sein sollte.

**Glückwunsch! Es wurden keine Probleme gefunden.**

Antwortzeit des Servers reduzieren
Ihr Server hat schnell geantwortet. Weitere Informationen zur Optimierung der Serverantwortzeit

Bilder optimieren
Ihre Bilder wurden optimiert. Weitere Informationen zum Optimieren von Bildern

Browser-Caching nutzen
Sie haben das Browser-Caching aktiviert. Empfehlungen für das Browser-Caching

CSS reduzieren
Ihre CSS-Ressource wurde reduziert. Weitere Informationen zum Reduzieren von CSS-Ressourcen

HTML reduzieren
Ihre HTML-Ressource wurde reduziert. Weitere Informationen zum Reduzieren von HTML-Ressourcen

JavaScript reduzieren
Ihre JavaScript-Ressource wurde reduziert. Weitere Informationen zum Reduzieren von JavaScript-Ressourcen

JavaScript- und CSS-Ressourcen, die das Rendering blockieren, in Inhalten "above the fold" (ohne Scrollen sichtbar) beseitigen
Sie haben keine Ressourcen, die das Rendering blockieren. Weitere Informationen zum Entfernen von Ressourcen, die das Rendering blockieren

Komprimierung aktivieren
Die Komprimierung ist aktiviert. Weitere Informationen zum Aktivieren der Komprimierung

Sichtbare Inhalte priorisieren
Die Inhalte "above the fold" (ohne Scrollen sichtbar) wurden ordnungsgemäß priorisiert. Weitere Informationen zum Priorisieren sichtbarer Inhalte

Zielseiten-Weiterleitungen vermeiden
Auf Ihrer Seite sind keine Weiterleitungen vorhanden. Weitere Informationen zum Vermeiden von Zielseiten-Weiterleitungen

** Die Ergebnisse werden 30 Sekunden lang im Cache gespeichert. Wenn Sie Änderungen an Ihrer Seite vornehmen, warten Sie 30 Sekunden, bevor Sie den Test erneut durchführen.*

** Mit diesem Test wird überprüft, ob auf einer Seite gängige Best Practices zur Leistung umgesetzt werden. Eine hohe Punktzahl weist auf eine gute Nutzererfahrung hin, ist jedoch keine Garantie dafür. Weitere Informationen.*

### B.2.7 Pingdom

pingdom

FULL PAGE TEST DNS HEALTH LOG IN SIGN UP

## Pingdom Website Speed Test

Enter a URL to test the load time of that page, analyze it and find bottlenecks.

URL: http://52.29.161.188/agency/

Test from: Stockholm, Sweden

START TEST

DOWNLOAD HAR SHARE RESULT

Sign up for **free** to test your site every minute

SIGN UP FOR FREE

### Summary


| Performance grade | Load time | Faster than |
|---|---|---|
| A 100 | 484 ms | 97 % of tested sites |

| Page size | Requests | Tested from |
|---|---|---|
| 1.5 MB | 6 | Stockholm on Nov 8 at 17:54 |

### Performance insights

| GRADE | SUGGESTION |
|---|---|
| A 100 | Avoid bad requests |
| A 100 | Leverage browser caching |
| A 100 | Minimize redirects |
| A 100 | Minimize request size |
| A 100 | Remove query strings from static resources |
| A 100 | Serve static content from a cookieless domain |
| A 100 | Specify a cache validator |
| A 100 | Specify a Vary: Accept-Encoding header |

### Response codes

## B.3 Screenshots

### B.3.1 Audit


### B.3.2 REST Schnittstellen Drupal

| | | |
|---|---|---|
| HeadBar | headbar | REST export (/export/headbar) |
| Pages | pages | REST export (/export/pages) |
| Paragraphs | paragraphs | REST export (/export/paragraphs) |

### B.3.3 Drupal Inhalte

#### Seite als Inhaltstyp

| Beschriftung | Systemname |
|---|---|
| Beschreibung | field_description |
| Paragraph | field_paragraph |
| SEO-Titel | field_seo_titel |
| URL | field_url |

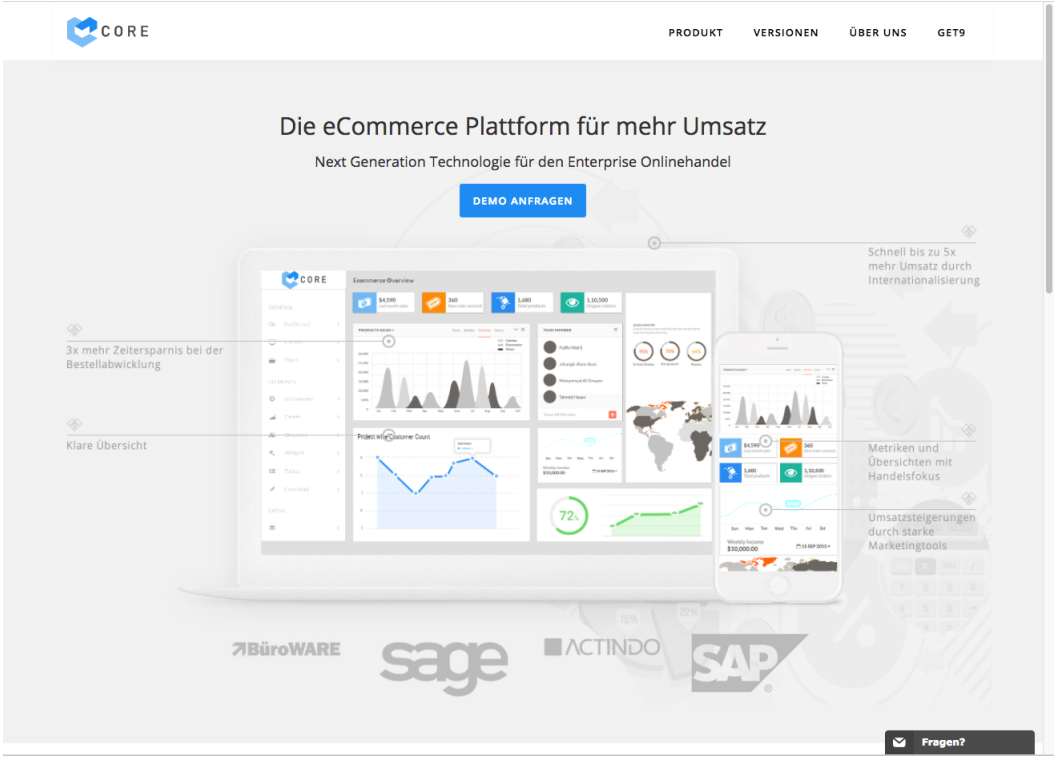
#### Formular Sektion als Inhaltstyp

| Beschriftung | Systemname |
|---|---|
| Subtext | field_subheader |
| webform | field_webform |
| Überschrift | field_header |

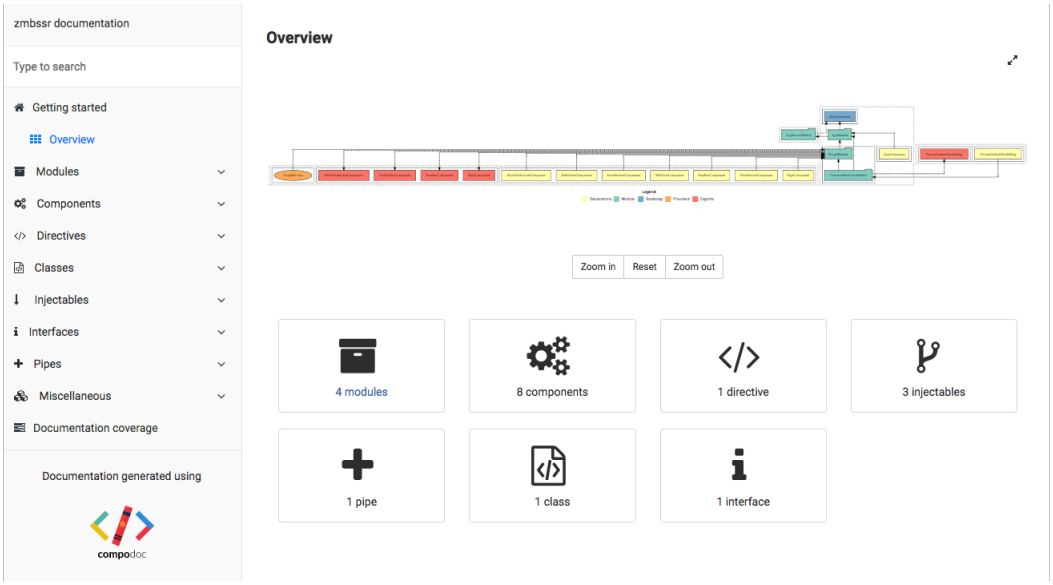
#### Hero als Inhaltstyp

| Beschriftung | Systemname |
|---|---|
| Hintergrundbild | field_backgroundimage |
| Subtext | field_subheader |
| Überschrift | field_header |

### B.3.4 Alte Website




### B.3.5 Entwickler Dokumentation



### B.3.6 Ordner Struktur


### B.3.7 Responsive Design


# C Code Beispiele

## C.1 NGINX Konfiguration Auszug

```
location  / {
        proxy_pass http://localhost:1337/;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

location /drupal/ {
        index index.php;
        root /data/www/drupal;
    }
```