

PRAXISAUFGABE

**Dokumentation des Projekts
TravelPoster für das Wahlfach
Microservices**

VON

Felix Waldbach & Marius Kiessling

Abgabedatum: 11. Juni 2018
Matrikelnummer, Kurs: 8215018, /*1234567*/, TINF16B
Dozent: Dr. Ingolf BUTTIG

Abkürzungsverzeichnis

Ajax Asynchronous JavaScript and XML

MVC Model-View-Controller

UI User Interface

Inhaltsverzeichnis

1	Einleitung	5
2	Aufgabenstellung	6
3	Architektur und Grundlagen	7
3.1	Projektaufbau	7
3.2	MongoDB	7
3.3	Express	7
3.4	React	8
3.5	Docker	8
4	Umsetzung	9
4.1	Frontend	9
4.1.1	Bildupload	9
4.1.2	Senden der Informationen an das Backend	9
4.1.3	Bildanzeige	9
4.2	Backend	9
4.2.1	Speichern der Posts	9
4.2.2	Senden der Informationan an das Frontend	10
5	Fazit	11
5.1	Erweiterungsmöglichkeiten	11
	Literaturverzeichnis	11

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Struktur der Arbeit

Kapitel 2

Aufgabenstellung

Kapitel 3

Architektur und Grundlagen

3.1 Projektaufbau

3.2 MongoDB

MongoDB ist ein dokument-orientiertes, Open-Source Datenbank-Programm. Dabei speichert MongoDB Daten in JSON-Dokumenten, sodass Datenstrukturen sich von Zeit zu Zeit verändern können und von Dokument zu Dokument unterschiedlich sind. Ein JSON-Dokument könnte also beispielsweise so aussehen: ... Ein wichtiges Feature ermöglicht es, Attribute ineinander zu verschachteln. Im Idealfall können die Dokumente also stets auf den Anwendungscode angepasst werden. Mithilfe von Queries werden diese Daten erreicht und verändert. Dieses Datenbank-Schema zeichnet sich also vor allem durch Flexibilität aus und dadurch, dass es sehr einfach zu lernen ist. Wird MongoDB beispielsweise in einem NodeJS-Projekt verwendet, sieht eine beispielhafte Query so aus: ... Hierbei muss jedoch zunächst überhaupt eine Verbindung zu der Datenbank hergestellt werden: ... Grundvoraussetzung ist, dass diese Datenbank und die sogenannte Collection, vergleichbar mit einer Relation bei einem relationalen Datenbankmodell, überhaupt existiert.

3.3 Express

Bei Express handelt es sich um ein serverseitiges Web-Application-Framework, welches oft als Standard-Server-Framework für Node.js bezeichnet wird. Es enthält eine Menge Features und erleichtert so das schnelle Aufsetzen eines Node.js-Servers und ermöglicht es, diesen Server schnell um Features zu erweitern. So ermöglicht der „Express-Router“

beispielsweise das Definieren von REST-Routen, auf die mithilfe von GET-, POST, PUT und DELETE-Methoden zugegriffen werden kann. Mithilfe des Express-Routers kann das Verhalten definiert werden, das abgearbeitet werden soll, wenn auf eine der Routen zugegriffen wird. Inzwischen bauen viele Node.js-Framework auf Express auf, da es sich für Single-Page, Multi-Page sowie für mobile und webbasierte Applikationen eignet. So gibt es beispielsweise den häufig verwendeten MERN-Stack, bestehend aus MongoDB, Express, React sowie Node.js, wie er auch in diesem Projekt verwendet wird. Weitere Beispiele sind KeystoneJS oder Kraken. Dank Express und Node.js wird Javascript mittlerweile nicht mehr nur für Frontend-Entwicklung von Webseiten verwendet, sondern es werden auch immer häufiger serverseitige Komponenten mithilfe von Javascript realisiert. Es unterstützt außerdem das MVC-Pattern, um Applikationen nach einem bestimmten Schema zu modellieren und ist betriebssystem unabhängig.

3.4 React

React ist eine Javascript-Bibliothek für das Entwickeln von interaktiven Benutzeroberflächen. Mithilfe von View-Komponenten, die jeweils eine `render()`-Methode enthalten, werden einzelne Teile dieser React-UI realisiert. React kümmert sich vor allem um effizientes Rendering und das Updaten der Webanwendung. Die `render()`-Methode verarbeitet Input-Daten und gibt den HTML-Code zurück, der letztendlich angezeigt, also gerendert, wird. Eine hilfreiche Funktionalität von React ist der sogenannte State. Über diesen werden Daten verarbeitet und bei jeglicher Änderung der State-Variable wird die Seite neu geladen, die `render()`-Methode also erneut ausgeführt. Die state-Variable wird beispielsweise so verwendet: ...

3.5 Docker

Kapitel 4

Umsetzung

4.1 Frontend

4.1.1 Bildupload

4.1.2 Senden der Informationen an das Backend

4.1.3 Bildanzeige

Jedes Mal, wenn die Seite aktualisiert wird, sollen anhand der aktuellen Informationen in der Datenbank alle Posts angezeigt werden. Hierfür wird ein GET-Request an die Route `/post/get/all` an das Backend geschickt. Das Resultat enthält den Namen des Verfassers sowie das dazugehörige Bild in einem JSON-Objekt.

4.2 Backend

4.2.1 Speichern der Posts

Die Posts eines Benutzers werden mithilfe von `cors()`-Headern vom separaten Frontend an das Backend gesendet. Diese bestehen im Grunde lediglich aus dem Namen des Verfassers des Posts und dem Namen der Datei, in der das zugehörige Bild gespeichert wird. Empfängt das Backend einen POST-Request an die Route `/post/fileupload`, wird der mitgesendete Post in der dahinterliegenden MongoDB-Collection gespeichert. Zusätzlich wird das Bild, welches im base64-Format gesendet wird, in PNG-Format umgewandelt und in einer Datei gespeichert. Der Name dieser Datei wird letztendlich in der Datenbank zusammen mit dem Namen des Verfassers gespeichert. Um die Einträge in der Collecti-

on zu speichern, wird eine Query verwendet, die der Collection einen neuen Eintrag mit einem "name und einem "filenameAttribut hinzufügt.

4.2.2 Senden der Informationan an das Frontend

Empfängt das Backend einen GET-Request an die Route /post/get/all", soll es alle bisher in der Collection gespeicherten Posts im JSON-Format an das Frontend zurückschicken. Hierfür wird eine Query verwendet, die die "postCollection ohne jegliche Bedingung durchsucht, sodass alle Einträge gefunden werden: ...

Kapitel 5

Fazit

5.1 Erweiterungsmöglichkeiten

Dieses Projekt bietet eine gute Basis für zukünftige, umfangreichere Projekte. Es könnte gar zu einer umfangreichen Blog-Webseite erweitert werden. Hierfür könnte beispielsweise ein User-Authentifizierungssystem implementiert werden. Da MongoDB bereits zum Speichern der Posts verwendet wird, könnte die zugrundeliegende Datenbank einfach um eine User-Collection erweitert werden. Im Zuge dieser Erweiterung würde es sich weiterhin eignen, durch Session-Handling und Cookies das Benutzererlebnis zu verbessern, sodass man sich beispielsweise nur einmalig anmelden muss und dann jederzeit seine eigenen Posts ansehen kann. Weiterhin könnte ein Follower-System implementiert werden, bei dem ein Benutzer in der Lage ist, anderen Benutzern zu folgen und so auch die Posts eben dieser anderen Benutzer anzusehen. Ein Benutzer sieht also schlussendlich nur die Posts von anderen Benutzern, die ihn interessieren. Des Weiteren können die Posts umfangreicher und flexibler gestaltet werden, beispielsweise durch Link-Unterstützung, dem Hinzufügen einer aussagekräftigen Beschreibung oder auch dem Posten von Videos. Travelposter zeichnet sich also in der Umsetzung besonders durch flexible Erweiterbarkeit aus.

Literaturverzeichnis

- [1] MongoDB <https://www.mongodb.com/what-is-mongodb>
MongoDB, Inc.
Stand 03.06.2018
- [2] Frameworks aufbauend auf Express <https://expressjs.com/en/resources/frameworks.html>
expressjs contributors
Stand 03.06.2018
- [2] Express.js <https://www.upwork.com/hiring/development/express-js-a-server-side-javascript-framework/>
Upwork
Stand 03.06.2018
- [2] ReactJS <https://reactjs.org/>
Upwork
Stand 03.06.2018