Otto-Friedrich-Universität Bamberg
Lehrstuhl für Medieninformatik
Prof. Dr. Andreas Henrich
Leon Martin, Tobias Hirmer, Martin Bullin

VAWi Web-Technologien, Sommersemester 2021

Studienleistung 3

Hinweise

- Die vollständige und korrekte Bearbeitung einer Aufgabe ergibt die volle Punktzahl dieser Aufgabe; in Ausnahmefällen werden halbe Punkte verteilt.
- Für diese Studienleistung erwarten wir einen **Bearbeitungsaufwand von insgesamt etwa 60 Stunden**, da wir von drei Personen pro Team ausgehen. Dies ist als Richtwert zu verstehen und keinesfalls als Unter- oder Obergrenze.
- In Verdachtsfällen behalten wir uns vor, Plagiate von der Benotung auszunehmen. Wir werden in einem solchen Fall Kontakt mit den Betroffenen aufnehmen.
- *Programmieraufgaben* sollen in nachvollziehbar kommentierter Form im Quelltext abgegeben werden. Verwenden Sie keine anderen als die angegebenen Bibliotheken, soweit sie nicht zum Standardsprachumfang gehören. Nicht lauffähige Programme können nicht bewertet werden. Fügen Sie gegebenenfalls eine README-Datei hinzu, die beschreibt, wie man Ihre Lösung kompilieren bzw. ausführen kann.

Abgabe der Lösungen

- *Abgabetermin* **02. August 2021, 23:55 Uhr**
- *Abgabemodus* Einreichung auf der Online-Lernplattform im entsprechenden Kurs; der Server ist erreichbar unter `http://lms.vawi.de/vawi/`.
- *Dateiformat der Abgabe* Die Abgabe erfolgt durch den Upload eines ZIP-Archivs, das alle notwendigen Dateien beinhaltet. Der Dateiname des ZIP-Archivs sollte das Namensschema **nachname1_nachname2_nachname3_03.zip** einhalten.
- *Update der Lösung* Bis zur oben angegebenen Deadline können Sie Ihre Lösung beliebig oft durch eine neue (korrigierte) Fassung ersetzen. Bitte beachten Sie, dass dabei die zuvor hochgeladene Lösung überschrieben wird. Wir haben keine Möglichkeit, alte Fassungen wiederherzustellen!

Wir wünschen Ihnen viel Erfolg
bei der Bearbeitung der Studienleistung!

Aufgabe: Erstellung einer Webanwendung mit dem MEAN-Stack (60 Punkte)

In dieser Teilleistung soll eine Webanwendung mithilfe des MEAN-Stacks realisiert werden. Aufgabe 1 bezieht sich dabei auf die Implementierung des Angular-Frontends, das in Aufgabe 2 an ein von Ihnen mithilfe von Express.js und MongoDB implementiertes Backend angebunden werden soll.

Bei der Anwendung handelt es sich um eine verkleinerte Neu-Implementierung der Anwendung aus SL1, also eine Anwendung bei der Nutzer Fachbegriffe und entsprechende Erklärungen dazu hinterlegen können.

Aufgabe 1: Implementierung des Frontends (40 Punkte)

An das mit Angular zu implementierende Frontend der Anwendung werden die folgenden Anforderungen gestellt:

1. Jedes Terminology-Objekt soll eine ID, den Fachbegriff selbst, die zugehörige Beschreibung, das Kapitel, in dem er behandelt wird, und den Namen des Nutzers, der/die den Eintrag angelegt hat, beinhalten. Die Terminology-Objekte sollen dabei noch nicht in einer Datenbank persistiert werden; diese Funktionalität wird in der zweiten Aufgabe implementiert. Zur temporären Speicherung der Events im Frontend können Sie ein einfaches Array verwenden.
2. Auf einer Übersichtsseite sollen alle Terminology-Objekte als Liste dargestellt werden, wobei diese zunächst nach Kapitel und innerhalb der Kapitel alphabetisch nach ihrem Fachbegriff sortiert sein sollen. Die oben genannten Terminology-Attribute sollen dem Nutzer stets ersichtlich sein.
3. Die Webanwendung soll die Erstellung von neuen Terminology-Objekten über ein Formular auf einer separaten Seite ermöglichen. Zum Formular gelangt man, indem man in der Navigationsleiste *Eintrag hinzufügen* anklickt. Im Formular sollen alle oben genannten Daten eines Terminology-Objekts abgefragt werden. Durch Klicken auf den Button *Sichern* wird das neue Event im Frontend gespeichert, durch Klicken auf den Button *Abbrechen* wird die Aktion abgebrochen. In beiden Fällen soll anschließend automatisch zur Übersichtsseite gewechselt werden.
4. Die Webanwendung soll das Editieren von Terminology-Objekten über dasselbe Formular wie für die Erstellung von Terminology-Objekten ermöglichen. Nach einem Klick auf den *Editieren*-Button eines Terminology-Objekt-Eintrags in der Liste gelangt der Nutzer zum Formular, welches mit den bereits vorhandenen Werten befüllt ist.
5. Terminology-Objekt-Einträge sollen durch Klicken auf den jeweiligen X-Button gelöscht werden können.
6. Zur Gestaltung der Webanwendung soll **ausschließlich** Bootstrap, welches in der Vorlage **bereits eingebunden** ist, (und falls unbedingt nötig selbst geschriebener CSS-Code) verwendet werden. Terminology-Objekt-Einträge sollen je nach Kapitel in einer anderen Farben eingefärbt sein und ihre zugewiesene Farbe behalten. Die weitere Gestaltung soll sich an dem folgenden Mockup orientieren.

7. Ihre Anwendung soll einen Kopfbereich haben, welcher den Titel der Seite, optional ein Logo sowie eine Navigationsleiste beinhaltet. In der Navigationsleiste soll ein Menüpunkt auf die Übersicht und ein weiterer Menüpunkt (Eintrag hinzufügen) auf das Formular zur Erstellung eines neuen Terminology-Objekts leiten. Denken Sie daran, dass der jeweils aktive Menüpunkt hervorgehoben sein soll.

8. Weitere Seiten als die hier beschriebenen müssen Sie nicht erstellen!

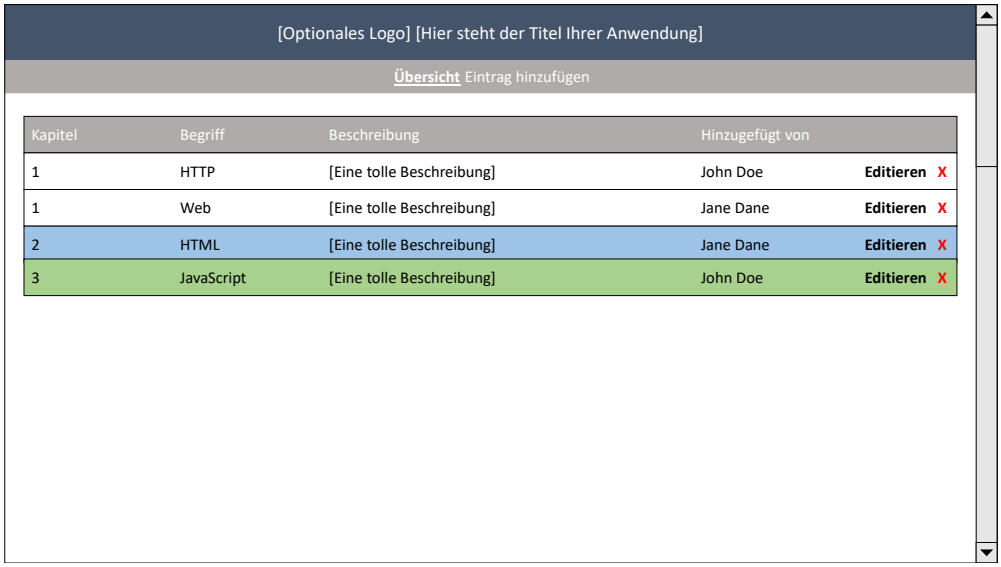

Abbildung 1: Übersichtsseite der Anwendung.

Abbildung 2: Ansicht der Anwendung beim Hinzufügen oder Editieren von Terminology-Objekten.

Verwenden Sie die im Kurs bereitgestellte Vorlage *tl3.zip* als Basis für die Webanwendung. Gewisse Aspekte, hauptsächlich bzgl. der Struktur des Projekts, und eine Klasse für die Terminology-Objekte sowie ein Service, der für die Verwaltung der Terminology-Daten verwendet werden kann, sind dort schon vorimplementiert. Gehen Sie zur Nutzung der Vorlage wie folgt vor:

1. Entpacken Sie die ZIP-Datei
2. Öffnen Sie eine Kommandozeile Ihrer Wahl im entpackten Ordner
3. Installieren Sie die benötigten Node-Module mithilfe des Befehls `npm install`
4. `npm run startApp` macht die Angular-App unter `localhost:4200/` verfügbar
5. Rufen Sie nach dem Starten der Anwendung `localhost:4200/` im Browser auf.

Aufgabe 2: Anbindung an das Backend (20 Punkte)

Für Aufgabe 2 gelten die folgenden Anforderungen:

- Die Terminology-Objekte sollen nun nicht mehr im Frontend in einem Array verwaltet werden. Stattdessen sollen die Terminology-Objekte stets über eine Express-API aus der MongoDB abgefragt werden, wenn sie benötigt werden.
- Bei der Erstellung, bei der Editierung und beim Löschen eines Terminology-Objekts soll eine Anfrage an die Express-API erfolgen, die die entsprechende Änderung in der Datenbank bewirkt.
- Die MongoDB-Datenbank ist erreichbar unter `mongodb.l1n.de`. Die Verbindungs-URI hat folgenden Aufbau: `mongodb://<gn>:<pwd>@mongodb.l1n.de:27017/<gn>`, wobei `<gn>` für die Gruppennummer und `<pwd>` für das Passwort steht. Die Nummern und Passwörter können per Mail angefragt werden (bitte Tobias Hirmer **und** Leon Martin als Adressat angeben).

In der Vorlage ist Express.js bereits eingebunden. Zur Verwendung gibt es die folgenden Hinweise:

- Der Server alleine lässt sich mittels `npm run startServer` starten. Client und Server gemeinsam lassen sich mittels `npm run start` starten.
- Nutzen Sie für Ihre serverseitige Implementierung die Datei `server.js` im Verzeichnis `/server`

Weitere Hinweise zur Teilleistung

- Aufgrund der hohen Komplexität des Projekts und der Vorlage, wird die Einarbeitung in den Code als Teil der Leistung angesehen.
- Ziehen Sie zur Implementierung des Programmablaufs im MEAN-Stack (Angular $\leftrightarrow$ Node.js + Express.js $\leftrightarrow$ MongoDB, s. Übung 8, Folie 8) die Übungsfolien und ggf. die verlinkten Ressourcen zu Rate.
- Nutzen Sie die Möglichkeiten der Konsolenansicht in den Entwicklungswerkzeugen von Firefox und Chrome zum Debuggen Ihres JavaScript- bzw. TypeScript-Codes.
- Testen Sie das von Ihnen implementierte Event Handling auf möglich Fehler, die durch ungewöhnliche Nutzerinteraktionen entstehen könnten.
- Kapseln Sie zusammengehörige Funktionalität in Klassen und Komponenten, um Ihren Code besser zu strukturieren und die Wiederverwendbarkeit von Code-Ausschnitten zu erhöhen. Ihre Anwendung soll aus **mindestens vier Komponenten** bestehen!
- Geben Sie in einer README-Datei an, ob Sie Ihre Umsetzung mit Firefox oder Google Chrome getestet haben. Dokumentieren Sie in dieser README-Datei bitte auch weitere Besonderheiten, z.B. wenn Ihre Umsetzung unvollständig ist oder sich unter bestimmten Umständen fehlerhaft verhält.
- Ihr Abgabe-ZIP-Archiv muss alle zur Ausführung notwendigen Dateien beinhalten, nicht nur solche, die Sie neu erstellt haben.