UE4 – Linked Open Data, Web-Services, https (25 Punkte)

Ziel dieser Übung ist es, externe Services zu integrieren und eine Client-Server Kommunikation über https zu realisieren. Weiters soll ein Einblick in das Deployment und die Optimierung von Webapplikationen genommen werden.

Deadline der Abgabe via TUWEL: **Montag, 05.06.2017 23:55 Uhr**

Nur ein Gruppenmitglied muss die Lösung abgeben.

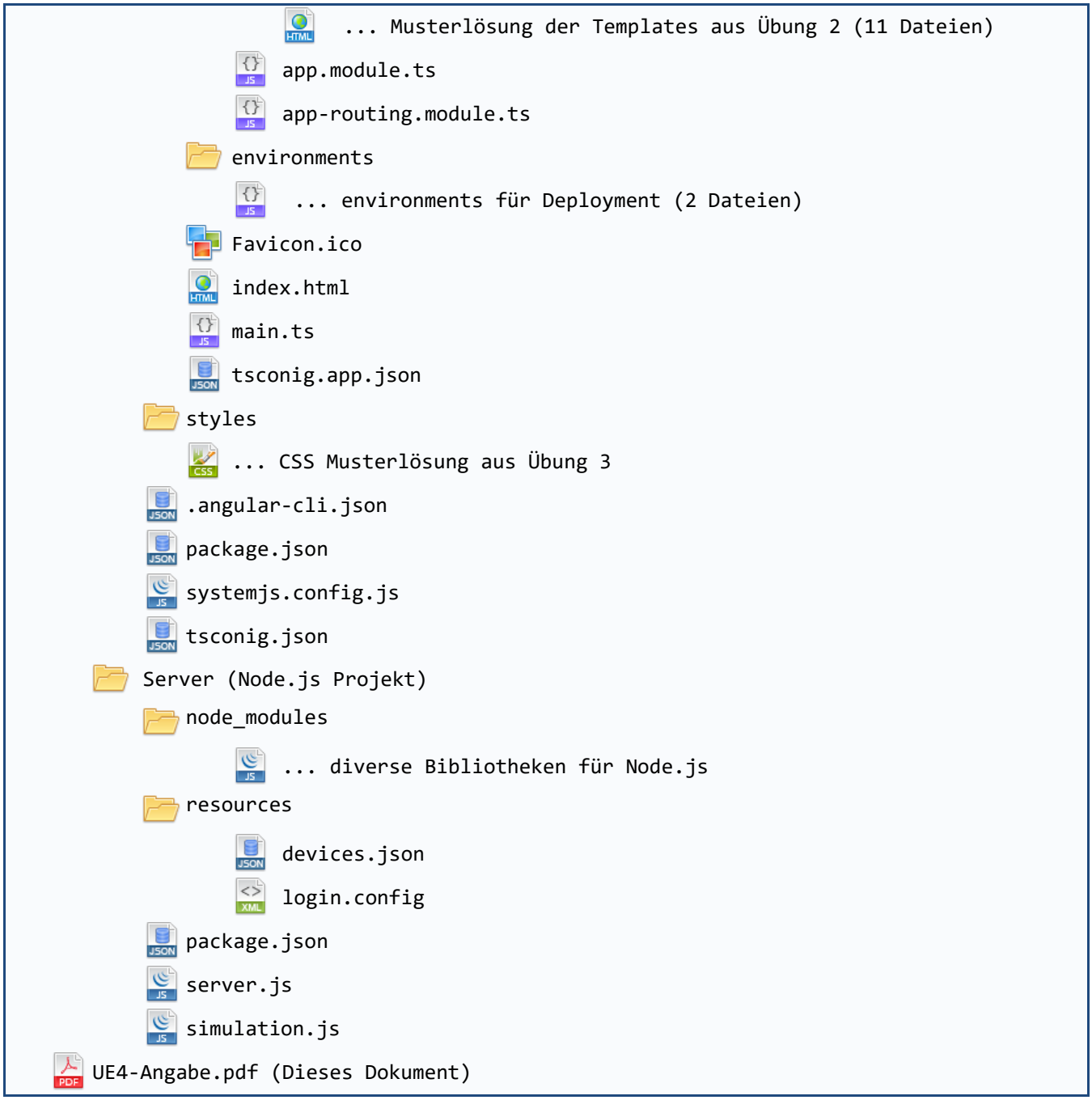
BIG Smart Home

Bei BIG Smart Home handelt es sich um eine visuelle Schnittstelle zur Steuerung und Überwachung von Smart Devices eines vernetzten Haushalts. Autorisierte Personen dürfen dabei die aktuellen Zustände und Werte der Geräte einsehen, so wie auch Einfluss auf diese nehmen. Bei Neuanschaffungen können Geräte einfach ins System eingegliedert werden oder, wenn Geräte ausgemustert werden, ist auch eine Entfernung aus dem System möglich. Über simple Grafiken werden zur besseren Übersicht die aktuellen Zustände der Geräte dargestellt.

Angabe

Diese Angabe umfasst folgende Dateien:

```
UE4-Angabe.zip
    lab4
        Client (Angular 2 Projekt)
            images
                ... Bilder aus Übung 3
            node_modules
                ... diverse Bibliotheken für Angular 2
            scripts
                ... JavaScript Musterlösung aus Übung 3
            src
                app
                    components
                        ... Angular-Components (12 Dateien)
                    model
                        ... Datenmodell aus Übung 3
                    services
                        ... Angular Services (3 Dateien)
                    views
```



Bitte beachten Sie, dass nicht alle der bereitgestellten Dateien bearbeitet werden sollen (siehe Abgabemodalität). Nehmen Sie jedoch keinesfalls Änderungen an den verwendeten Bibliotheken vor. Alle für diese Aufgabe benötigten Bibliotheken sind bereits enthalten und entsprechend eingebunden und bedürfen keines Einschreitens Ihrerseits.

Erweitern Sie die bisherigen Übungen um weitere Funktionalitäten. Sichern Sie Ihre REST-Schnittstelle durch die Verwendung von https etwas ab und verwenden Sie Linked Open Data um die Anzahl der unterschiedlichen Gerätetypen zu steigern. Führen Sie zudem ein Deployment Ihrer Applikation durch und ~~O~~optimieren bzw. ~~V~~verbessern Sie diese mit entsprechenden Tools. Auch in dieser Übung gilt, dass der Großteil aller Stellen, die von Ihnen angepasst werden müssen, zur besseren Erkennung mit „*TODOs*“ markiert und dort erneut kurz (aber nicht vollständig) beschrieben werden. Achten Sie bitte darauf, dass

bei manchen „*TODOs*“ gewisse Anweisungen zu beachten und einzuhalten sind. Bitte verwenden Sie als Ausgangspunkt für Ihre Implementierung die von uns zur Verfügung gestellten Dateien.

Anmerkungen zum Datenmodell

Das Datenmodell zur Darstellung der Geräte ist vorgegeben und soll nicht verändert werden. Die smarten Geräte werden innerhalb des Programms als Devices (definiert in *device.ts*) dargestellt. Ein Gerät kann dabei mehrere Interaktionsmöglichkeiten besitzen, im weiteren auch Steuerungselemente genannt (definiert in *controlUnit.ts*). Über diese können Einstellungen an den Geräten vorgenommen werden. Pro Gerät können mehrere Steuerungselemente existieren, wobei drei unterschiedliche Arten von Steuerungselementen vorhanden sind, welche für die dazugehörigen Datentypen verwendet werden. Folgende Datentypen sind vorhanden (definiert in *controlType.ts*): kontinuierliche Werte (bspw. für Temperatur), diskrete Werte/Enum (bspw. für Rollladen) und boolesche Werte (bspw. für Beleuchtung).

Hauptanforderungen an Ihre Implementierung

- *https-Schnittstelle*: Stellen Sie die REST-Schnittstelle über https zur Verfügung. Erstellen Sie dazu ein https Zertifikat sowie einen dazugehörigen Schlüssel und binden Sie diese in den Server ein. Für die Erzeugung eines Zertifikates können Sie bspw. openssl[1] verwenden, da für diese Übung ein selbstsigniertes Zertifikat ausreichend ist. Die Schnittstelle darf auch weiterhin über http erreichbar sein, sollte jedoch aus Ihrer Angular 2 Applikation via https angesprochen werden. Passen Sie daher auch die URLs innerhalb der clientseitigen Applikation entsprechend an. Der Websocket muss hingegen **nicht** über eine gesicherte Verbindung abgewickelt werden. Beachten Sie bitte auch die Hinweise zu https.
- *Twitter-Anbindung*: Um mit externen Web-Services in Berührung zu kommen, soll beim Anlegen von neuen Geräten auf Twitter eine Nachricht veröffentlicht werden. Das Abschicken des Tweets soll unter Verwendung der twitter[2] Bibliothek für node.js über den Server erfolgen. Dieser soll nach dem Hinzufügen des neuen Gerätes in seine Geräteliste einen neuen Tweet erstellen. Den Inhalt des Tweets erhalten Sie, indem Sie die Funktion getTwitterPublicationString() mit den entsprechenden Parametern in der *server.js* Datei aufrufen. Benutzen Sie für die Kommunikation mit Twitter folgende Zugangsdaten:

Twitter URL[3]:	https://twitter.com/BIGEWA2013
Consumer Key:	GZ6tiy1XyB9W0P4xEJudQ
Consumer Secret:	gaJDlW0vf7en46JwHAOkZsTHvtAiZ3QUd2mD1x26J9w
Access Token:	1366513208-MutXEbBMAVOwrbFmZtj1r4Ih2vcoHGHE2207002
Access Token Secret:	RMPWOePlus3xtURWRVnv1TgrjTyK7Zk33evp4KKyA

  Damit die Anbindung nachvollzogen werden kann, muss im Erfolgsfall der Inhalt des abgeschickten Tweets und im Fehlerfall eine sinnvolle Fehlermeldung auf die Konsole des Servers ausgegeben werden.

[1] https://www.openssl.org/

[2] https://www.npmjs.com/package/twitter

[3] Es wird der Account von 2013 verwendet

- *Linked Open Data*: Um eine höhere Anzahl an möglichen Gerätetypen zu erreichen, sollen Informationen unterschiedlicher Smart Devices über DBPedia ausgelesen werden. Zu diesem Zweck sollen beim ersten Start des Bildschirms zur Anlage eines Gerätes die benötigten Daten ausgelesen und anschließend im Session-Storage gespeichert werden. Diese zusätzlichen Gerätetypen sollen dann in weiterer Folge im Dropdownmenü „Gerätetyp“ auswählbar sein. Lesen Sie dafür aus der Kategorie Home_automation alle Subjects aus, welche ein Gegenstand sind – also dem Typ „Thing“ entsprechen – von einer Firma hergestellt werden – also mindestens einen „is Product of“ Eintrag haben – und ein Bild sowie eine Bezeichnung in deutscher Sprache besitzen. Speichern Sie anschließend die Bezeichnung und die Bild-URL in den Session-Storage und lesen Sie diese an passender Stelle wieder ein. Sie können Ihre SPARQL Query über den von DBPedia zur Verfügung gestellten Endpoint testen[4]. Um Ihre Query aus der Applikation heraus aufzurufen, können Sie diesen Endpoint direkt mit einem Ajax-Request ansprechen. Dabei können Sie Ihre Query direkt als Parameter in der Request URL angeben. Das Ergebnis Ihrer fertigen Query sollte folgende Daten enthalten:

label	url
"Kurzzeitwecker"@de	http://commons.wikimedia.org/wiki/Special:FilePath/Kitchen_timer.jpg?width=300
"Staubsauger"@de	http://commons.wikimedia.org/wiki/Special:FilePath/Vacuum_cleaner.jpg?width=300
"Waschmaschine"@de	http://commons.wikimedia.org/wiki/Special:FilePath/LGwashingmachine.jpg?width=300
"Geschirrspülmaschine"@de	http://commons.wikimedia.org/wiki/Special:FilePath/Dishwasher_with_dishes.JPG?width=300

- *Optimierung und Deployment*: Zu guter Letzt muss eine Applikation auch noch freigegeben werden. Zu diesem Zweck soll die clientseitige Implementierung deployed und optimiert werden. Mit dem Befehl „*ng build --env=prod*“ können Sie einen Großteil der Applikation in ein paar wenige JavaScript Files verpacken, welche Sie dann entsprechend optimieren und deployen sollen. Diese Dateien werden im *dist*-Ordner im Projektverzeichnis des Clients erstellt. Optimieren Sie diese unter zu Hilfenahme von externen Programmen, wobei sich die Optimierung hier rein auf die Verringerung der Dateigrößen bezieht. Es sollen hier sowohl die JavaScript Files sowie auch die CSS-Datei(en) entsprechend optimiert werden. Beispielweise können Sie den JavaScript Code mit Hilfe von uglifyjs[5] komprimieren und somit eine Optimierung bzw. Verbesserung durchführen. Suchen Sie sich hier entsprechende ähnliche Software für Ihre CSS-Files. Die zuvor automatisch erstellten Dateien reichen jedoch nicht aus, um eine endgültiges Deployment durchzuführen. Organisieren Sie Ihre Dateien im *dist*-Ordner daher so, dass es möglich ist, den Inhalt dieses Ordners zu kopieren und direkt in einem Apache Server ohne weitere Änderungen zum Einsatz zu bringen. Testen Sie Ihre Applikation daher auch mit einem echten Apache Server[6]. Bitte beachten Sie dabei jedoch, dass bei einer Standardkonfiguration des Apache Servers zwar die Navigation innerhalb der Applikation möglich ist, ein Neu-Laden einer Seite jedoch zu einem Fehler führt. Sie müssen sich in diesem Fall jedoch nicht um eine korrekte Konfiguration kümmern. Die optimierten und fürs Deployment benötigten Dateien sollen alle im *dist*-Ordner gespeichert und bei der Abgabe hochgeladen werden.

[4] https://dbpedia.org/sparql

[5] http://lisperator.net/uglifyjs/

[6] https://httpd.apache.org/

Testdaten

Verwenden Sie zum Testen Ihrer Implementierung die von uns zur Verfügung gestellten Geräte in der *devices.json* Datei. Diese Geräte werden automatisch beim Starten des Servers eingelesen und in einer JavaScript Variable gespeichert. Bei einem Neustart des Servers gehen daher alle bisherigen Änderungen verloren. Verändern Sie den Inhalt und die Struktur dieser Datei bitte nicht.

Hinweise

Validierung

Der von Angular 2 generierte Code muss nicht auf Validität geprüft werden, sollte jedoch so weit wie möglich valide gehalten werden und den Vorgaben von WAI Conformance Level Double-A entsprechen. Ausgenommen von dieser Regelung sind die Diagramme auf der Detailseite, welche bezüglich WAI-Tauglichkeit nicht berücksichtigt werden müssen.

Entwicklungsumgebung

Es ist Ihnen freigestellt, welche Entwicklungsumgebung Sie für diese Übung verwenden. Beispielsweise bieten sich IntelliJ IDEA Ultimate[7] oder WebStorm[8] an, welche mit den entsprechenden Plugins eine sehr gute Basis für die Entwicklung dieser Aufgabe darstellen.

npm Projekt

Das von uns zur Verfügung gestellte Projekt wird mittels npm[9] administriert, um eine leichtere Verwaltung der Bibliotheken zu ermöglichen. Gleichzeitig wird npm verwendet, um Ihre Angular 2 sowie auch die Node.js Applikation zu starten. Mit dem Befehl „*npm start*“ können Sie in weiterer Folge Ihre clientseitige sowie auch die serverseitige Implementierung starten und ausführen, wobei dieser Vorgang für jeden Teil separat zu erfolgen hat. Sie können diesen Befehl entweder über die Kommandozeile in Ihrem Projektverzeichnis oder direkt aus Ihrer Entwicklungsumgebung ausführen. Stellen Sie vor der Abgabe dieser Aufgabe sicher, dass Ihre fertige Applikation weiterhin auf diese Art gestartet werden kann. Die clientseitige Applikation verwendet zudem noch die Angular CLI[10] um das Deployment zu erleichtern.

https

Vor dem ersten Zugriff auf Ihren Server, welcher bereits Ihr selbst erstelltes Zertifikat eingebunden haben muss, müssen Sie der Verwendung des Zertifikates zustimmen. Wechseln Sie dafür mit Ihrem Browser auf beispielsweise https://localhost:8081 (wenn Ihr https-Server auf Port 8081 läuft) und erlauben Sie die Verwendung Ihres Zertifikates. Ansonsten können die Daten durch Ihre clientseitigen Applikation nicht abgerufen werden.

---

[7] https://www.jetbrains.com/idea/

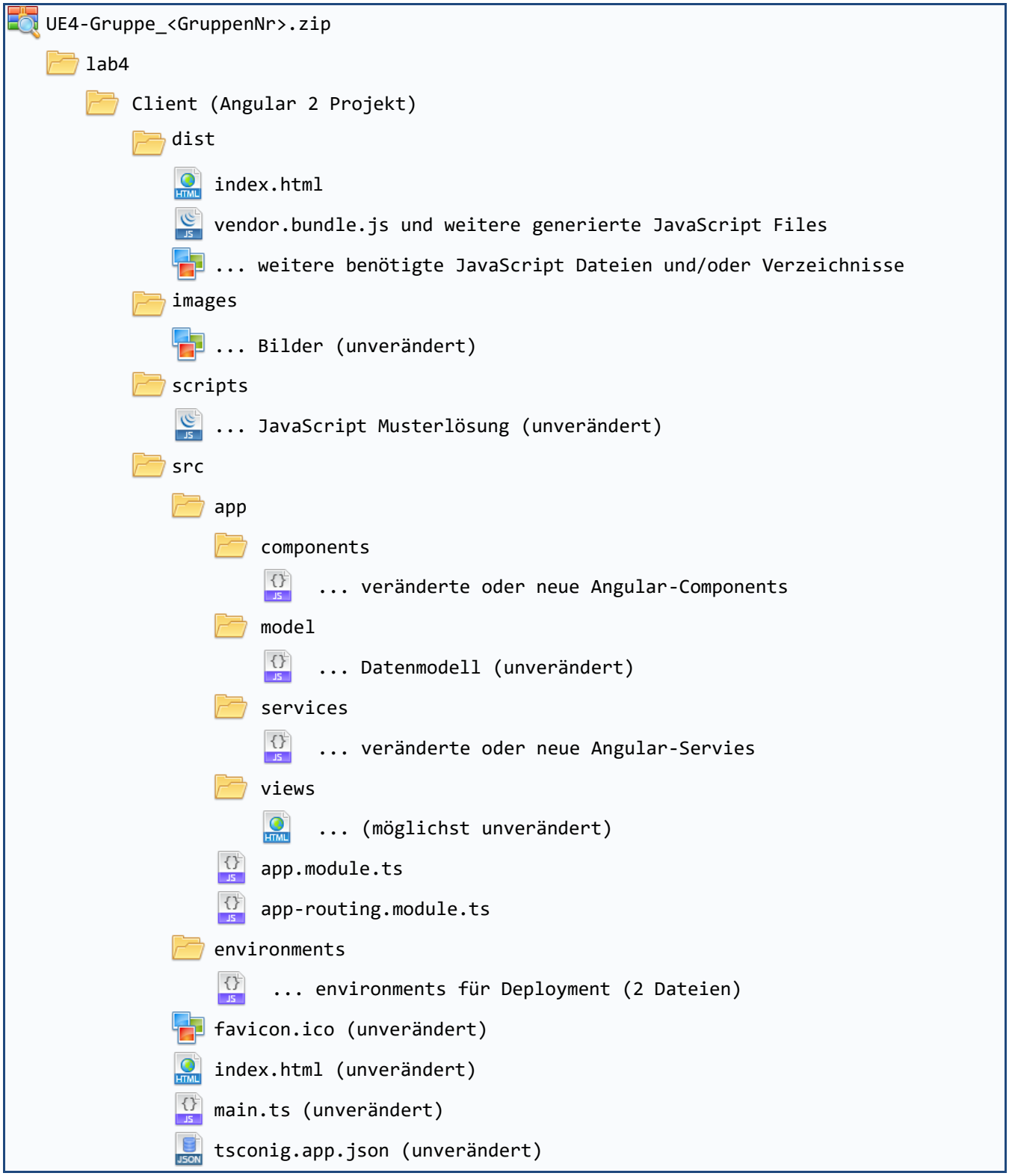
[8] https://www.jetbrains.com/webstorm/

[9] https://www.npmjs.com/

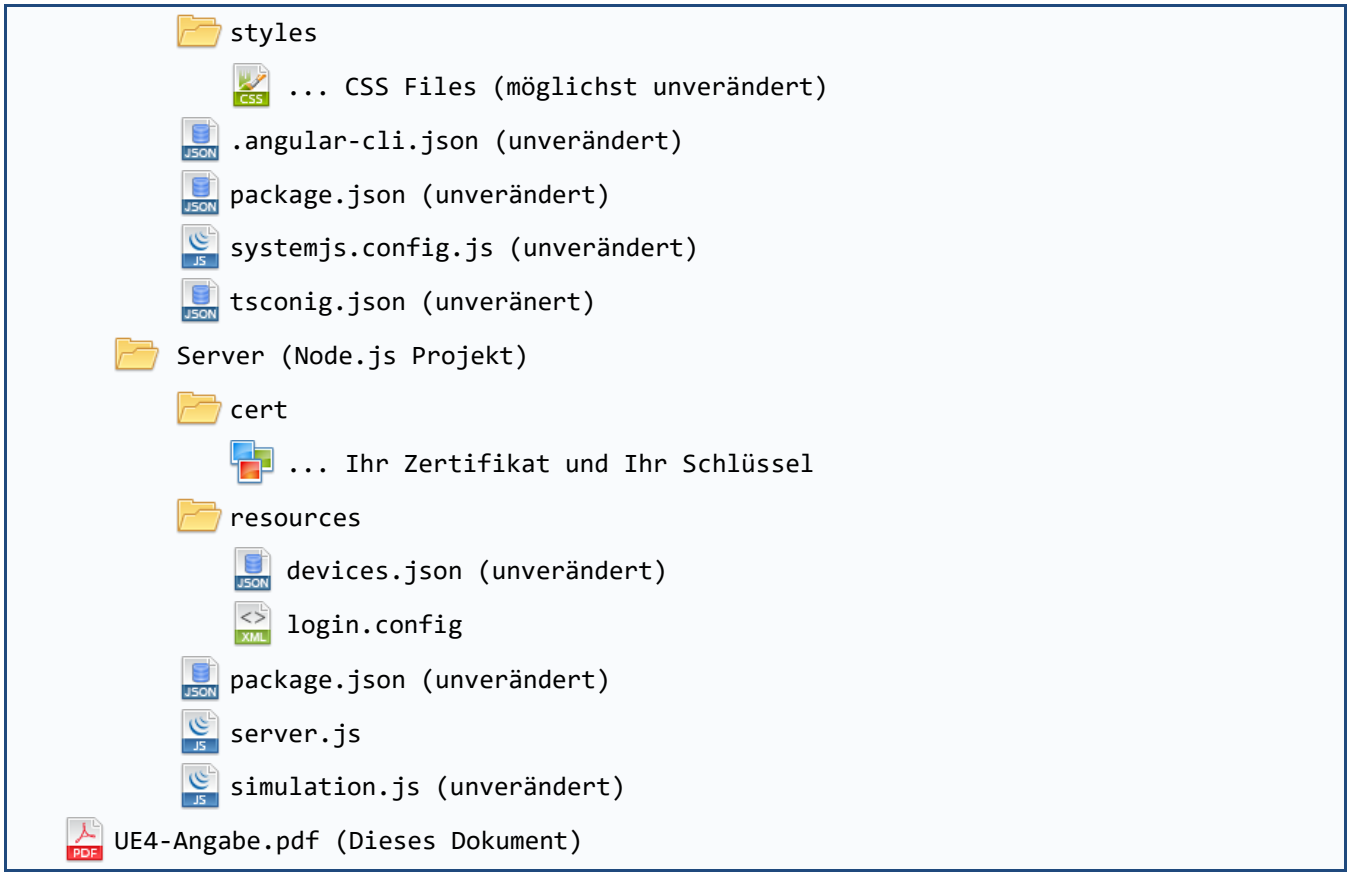
[10] https://cli.angular.io/

Abgabemodalität

Beachten Sie die allgemeinen Abgabemodalitäten des TUWEL-Kurses[11]. Zippen Sie Ihre Abgabe, sodass sie die folgende Struktur aufweist, beachten Sie dabei besonders, dass Sie die Bibliotheken im Verzeichnis node_modules **nicht** abgeben müssen:



[11] https://tuwel.tuwien.ac.at/course/view.php?id=7423



Alle Dateien müssen UTF-8 codiert sein!

ACHTUNG: Wird das Abgabeschema nicht eingehalten, so kann es zu Punkteabzügen kommen!