UE2 – Client-seitiges JavaScript, Angular 2, SVG (25 Punkte)

Ziel dieses Übungsbeispiels ist es, clientseitige Technologien kennenzulernen und einzusetzen. Dazu soll das Beispiel aus Übung 1 in eine Angular 2 Applikation überführt werden. Zusätzlich sollen mit Hilfe von jQuery einfache Animationen auf Scalable Vector Grafiken (SVG) durchgeführt werden.

Deadline der Abgabe via TUWEL: **Sonntag, 23.04.2017 23:55 Uhr**

Nur ein Gruppenmitglied muss die Lösung abgeben.

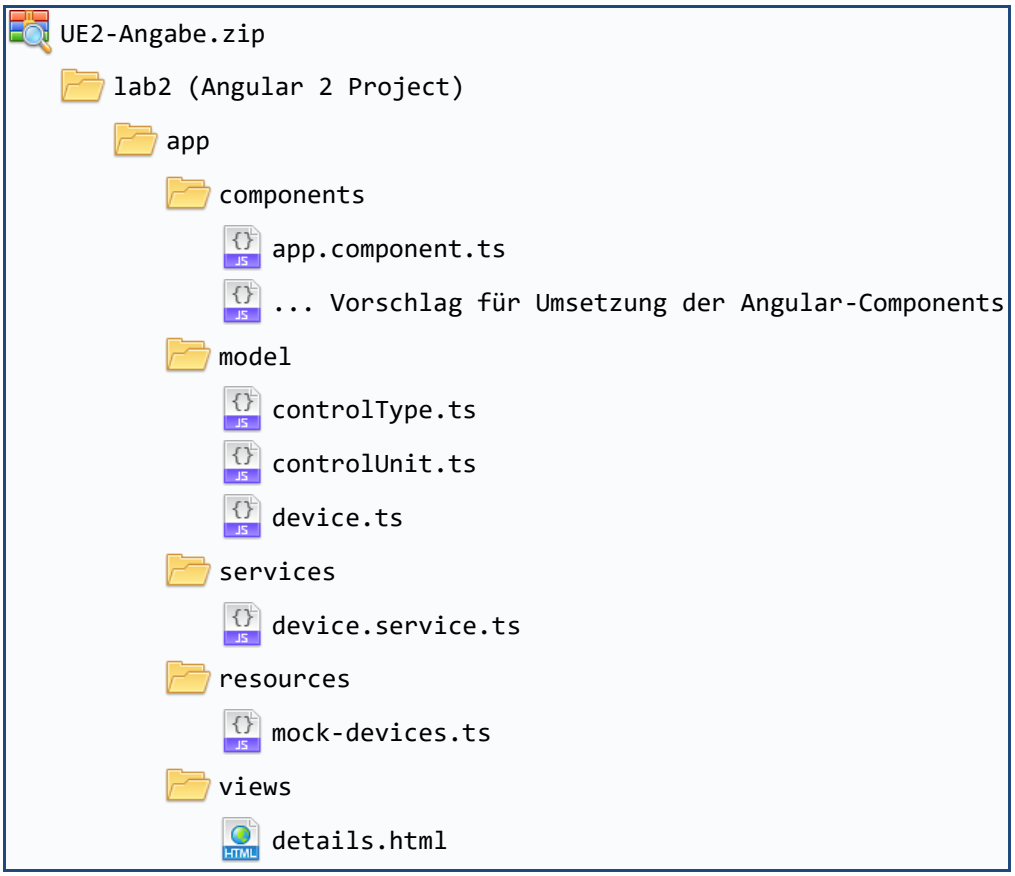
BIG Smart Home

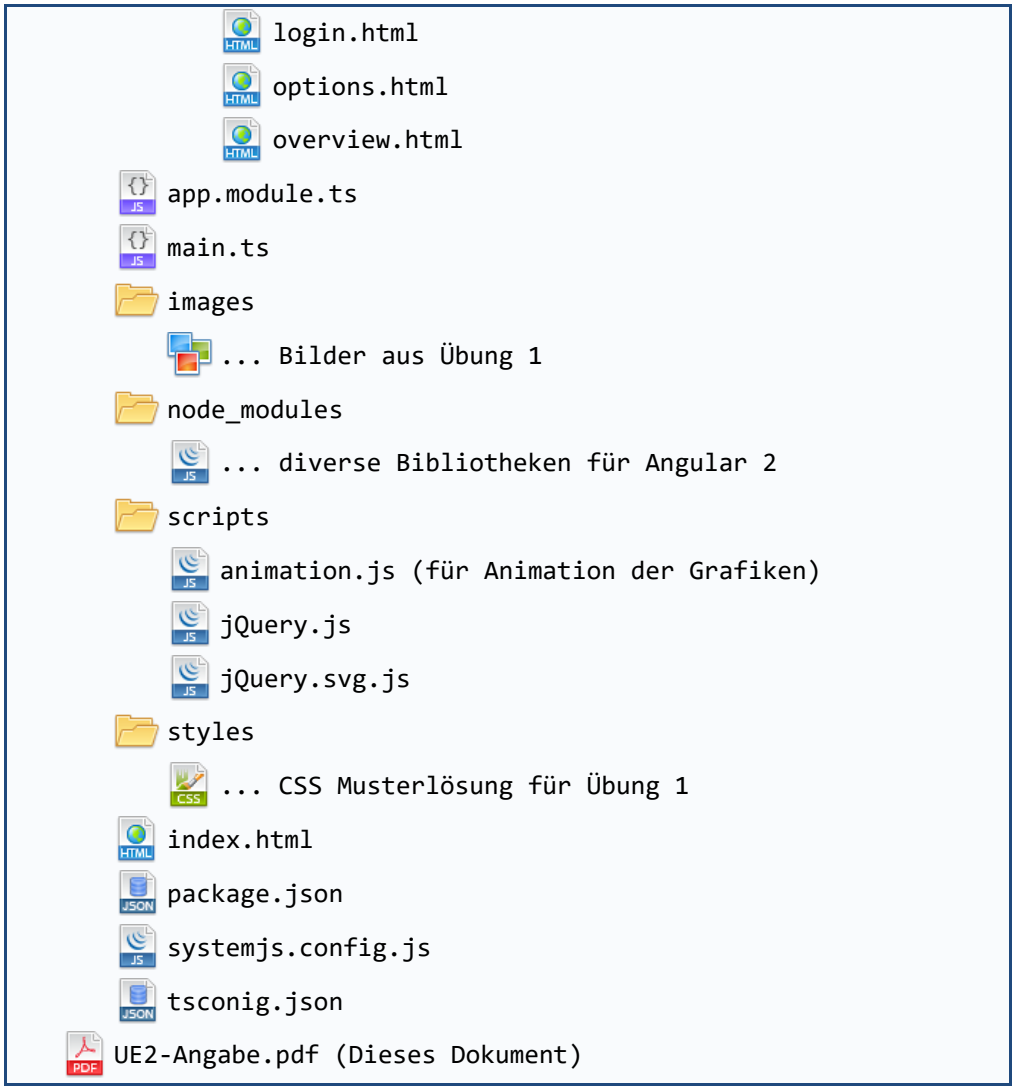
Bei BIG Smart Home handelt es sich um eine visuelle Schnittstelle zur Steuerung und Überwachung von Smart Devices eines vernetzten Haushalts. Autorisierte Personen dürfen dabei die aktuellen Zustände und Werte der Geräte einsehen, so wie auch Einfluss auf diese nehmen. Bei Neuanschaffungen können Geräte 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.

Für alle Geräte werden zusätzlich Diagramme angeboten, welche Aufschluss über den Verlauf der bisherigen Gerätezustände und Werte bieten.

Angabe

Diese Angabe umfasst folgende Dateien:





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

Implementieren Sie eine auf Angular 2 basierende client-seitige Web-Applikation, welche die BIG Smart Home Plattform realisiert. Verwenden Sie als Benutzeroberfläche den von uns zur Verfügung gestellten HTML5- und CSS-Code aus den Angaberessourcen. Es liegt an Ihnen, daraus entsprechende Angular 2 Templates zu erstellen und entsprechend anzupassen bzw. zu erweitern. Das Aussehen der Seiten soll allerdings so gut es geht erhalten bleiben. Um möglichst wenig Code zu duplizieren, können Sie Teile des HTML-Codes in neue Views auslagern. Serverseitige Funktionen sind in dieser Übung nicht zu implementieren.

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). In der Datei *details.html* sind alle möglichen Steuerungselemente für die unterschiedlichen Datentypen dargestellt (in derselben Reihenfolge wie oben). Bitte beachten Sie auch die Anmerkungen zum Modell innerhalb der zuvor genannten Dateien.

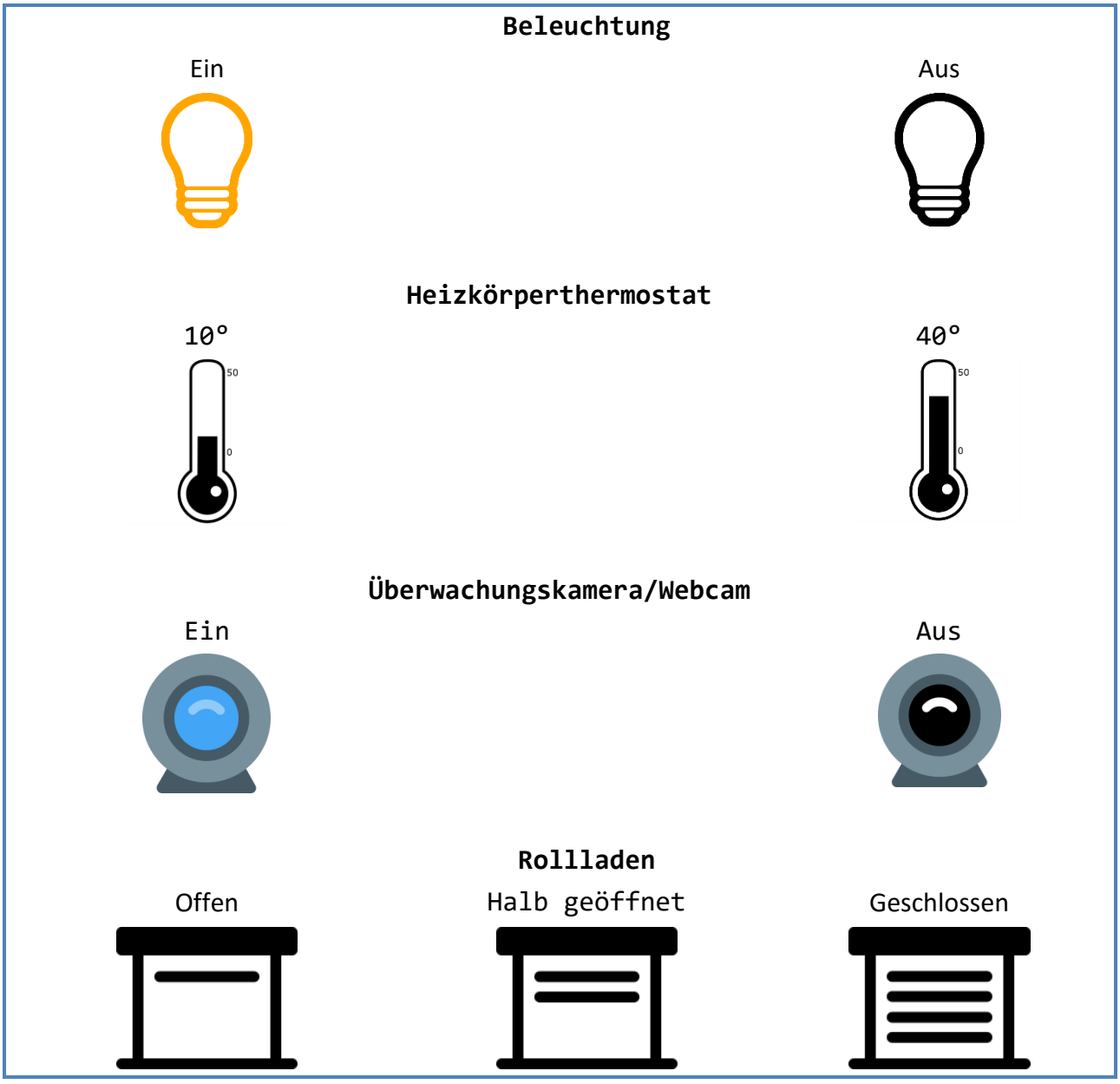
Hauptanforderungen an Ihre Implementierung

- *Seitenfluss*: Ruft der Benutzer oder die Benutzerin die Seite zum ersten Mal auf, soll das Anmeldeformular angezeigt werden. Eine Überprüfung der Benutzerdaten erfolgt in dieser Übung noch nicht, es soll jedoch sichergestellt werden, dass Eingaben getätigt wurden. Nach dem Anmelden wird eine Übersicht aller Geräte angezeigt. Durch Klicken auf ein Gerät gelangt man auf die Detailseite, welche zur Gerätesteuerung dient. Nach Betätigen der Logout-Schaltfläche soll der Benutzer/die Benutzerin auf die Log-In-Seite weitergeleitet werden. Die Optionsseite soll nach Drücken des Optionsbuttons geöffnet werden. Um auf die Übersichtsseite zurückkehren zu können, soll das BIG-Logo als „Home“-Button fungieren. Der Seitenfluss soll mittels Angular 2 Routing realisiert werden.
- *Seitenaufbau*: Die einzelnen Pages der Applikation sollen mit Angular 2 angezeigt werden. Erstellen Sie dafür selbstständig geeignete Angular 2 Components, deren Templates Sie aus den bereitgestellten HTML-Dateien extrahieren können. Versuchen Sie dabei wiederkehrende Designelemente, wie etwa die Navigation, als eigene Component zu definieren und vermeiden Sie so somit unnötige Codeduplikation.
- *Gerätedetailseite*: Auf der Detailseite eines Gerätes werden die Steuerungsmöglichkeiten für dieses angezeigt. Ein Gerät kann dabei über mehrere Steuerungselemente verfügen, welche hier allesamt dargestellt werden. Weiters wird für jedes Element ein Diagramm mit dem bisherigen Werte- bzw. Zustandsverlauf angezeigt. Verwenden Sie hier die *details.html* als Designvorlage für Ihre Angular 2 Templates.
- *Ändern des Gerätezustands*: Eine Änderung des Gerätezustands kann nur über die Detailseite erfolgen. Auf dieser soll für jedes Steuerungselement ein eigenes Formular zur Verfügung gestellt werden, welches in weiterer Folge clientseitig ausgewertet wird. Nutzen Sie zur Validierung dieser Formulare sowohl Angular 2 sowie auch HTML5 um Falscheingaben zu verhindern. Nach dem „Absenden“ eines Formulars sollen sowohl das Diagramm wie auch das Gerätelog angepasst werden. Im Log sollen die bisherigen Änderungen angezeigt werden (z.B. Änderung der Temperatur von 20 auf 25: *„6.3.2017, 10:01:30: 20 -> 25“*, vgl. *details.html*). Selbige Änderungen sollen auch auf den Diagrammen sichtbar gemacht werden. Der Verlauf des Gerätezustands soll nicht dauerhaft gespeichert werden, daher werden die Diagramme und das Gerätelog beim erneuten Öffnen der Detailseite zurückgesetzt.
- *Diagramme*: Nutzen Sie für die Darstellung der Diagramme die Angular 2 Bibliothek ng2-charts[1], welche bereits für Sie in das Projekt eingebunden wurden. Verwenden Sie für kontinuierliche Daten ein Liniendiagramm, für boolesche ein „Doughnut Chart“ und für Enums ein „Polar Area Chart“.

[1] http://valor-software.com/ng2-charts/

Verändern Sie bei der Eingabe eines neuen Wertes das Diagramm entsprechend. Bei einem Liniendiagramm fügen Sie zu diesem Zweck einen neuen Datenpunkt hinzu, in den anderen Fällen erhöhen Sie einfach die Häufigkeit des Auftretens eines Zustandes.

- *SVG-Animation*: Die bereitgestellten SVGs sollen mit Hilfe von jQuery dynamisch an den aktuellen Gerätezustand angepasst werden. In der Datei animation.js befinden sich leere Funktionen, die von Ihnen gefüllt werden sollen. In dieser Datei finden Sie weiters genauere Anweisungen zu den einzelnen Animationen. In den von uns zur Verfügung gestellten Testdaten enthalten alle Geräteinstanzen jeweils eine Referenz auf die entsprechende Funktion aus der animation.js. In Ihrem Programm müssen Sie daher nur die Funktion *Device*.draw_image() mit den entsprechenden Parametern aufrufen, wodurch die dazugehörige Funktion aus Ihrer animation.js ausgeführt wird. Da Ihre Grafiken nur beim Laden der Übersichtsseite neu gezeichnet werden müssen, bietet es ich an die *draw_image* Funktionen nach dem vollständigen Laden dieser Seite auszuführen. Die Animationen sollen wie folgt aussehen:



- *Geräte bearbeiten*: Das Bearbeiten bzw. Ändern von Gerätedaten beschränkt sich auf den Anzeigenamen eines Gerätes. Geändert werden kann dieser über die Editieren-Schaltfläche (Bleistiftsymbol) auf der Übersichtsseite. Dazu soll der Anzeigename des Gerätes durch ein Inputfeld innerhalb des DOM-Trees ersetzt werden, über welches dieser dann verändert werden kann. Das heißt, beispielsweise wird aus „Heizkörper Esszimmer“ ein *input* Feld mit value=“Heizkörper Esszimmer“. Um den neuen Wert zu übernehmen, können Sie entweder eine der Schaltflächen (Editieren bzw. Löschen) als Speichern-Button (Icon ist in Angabe unter *images* vorhanden) benutzen oder ermöglichen Sie ein Speichern durch Betätigen der Entertaste. Diese Änderungen an den Geräten sollen bis zum Neustarten oder Neuladen der Applikation vorhanden bleiben.
- *Passwort ändern*: Die eigentliche Funktion zum Ändern des Passworts ist in dieser Übung noch nicht zu implementieren, die Realisierung als clientseitiges Formular jedoch schon. Das heißt, auf der Optionsseite ist keine echte „Passwort ändern“-Funktion zu realisieren, eine Eingabe von Werten und das (client-seitige) „Absenden“ dieser jedoch schon. Dabei soll der „Änderung speichern“-Button so lange deaktiviert bleiben, bis alle benötigten Eingaben getätigt wurden (beliebiges altes Passwort, übereinstimmende neue Passwörter). Nach Betätigen des Buttons sollen einfach die Eingabefelder geleert und weiterhin die Optionsseite angezeigt werden.
- *Dynamische Inhalte*: Die Gerätedaten (beispielsweise Gerätename, aktueller Gerätezustand, der Gerätetyp, usw.) müssen dynamisch ausgegeben werden.
- *Speichern von Daten*: Da Sie in dieser Übung keine Datenbank oder Serveranbindung implementieren müssen, speichern Sie die Daten zu den Geräten in JavaScript/Typescript-Variablen. Das heißt, die Daten gehen verloren, sobald die Applikation neu geladen wird.
- Eine Überprüfung des Passworts beim Log-in muss in dieser Übung nicht implementiert werden.

Testdaten

Verwenden Sie zum Testen Ihrer Implementierung die von uns zur Verfügung gestellten Geräte in der *mock-devices.ts* Datei. Um auf diese Daten leichter zugreifen zu können, steht Ihnen der Service in *device.service.ts* zur Verfügung, welchen Sie als Provider in Ihre Applikation einbinden können, um diesen überall innerhalb Ihres Programmes nutzen zu können.

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.

Stellen Sie ebenfalls sicher, dass Ihr JavaScript Code (animation.js) in der Developer Console des Browsers Ihrer Wahl keine Fehlermeldungen ausgibt.

Entwicklungsumgebung

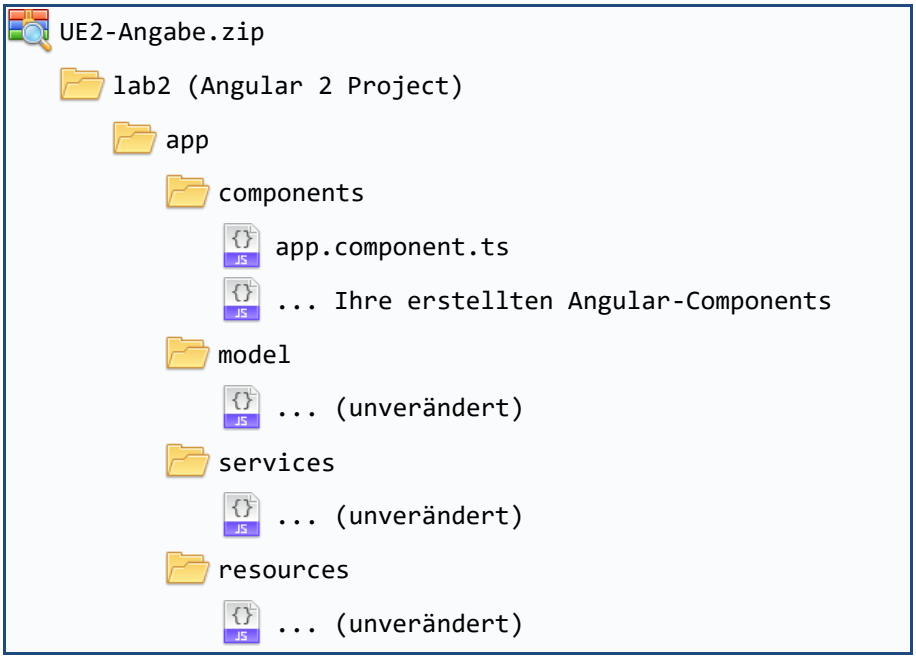
Es ist Ihnen freigestellt, welche Entwicklungsumgebung Sie für diese Übung verwenden. Beispielsweise bieten sich IntelliJ IDEA Ultimate[2] oder WebStorm[3] an, welche mit den entsprechenden Plugins eine sehr gute Basis für die Entwicklung von Angular 2 Applikationen darstellen.

npm Projekt

Das von uns zur Verfügung gestellte Projekt wird mittels npm[4] administriert, um eine leichtere Verwaltung der Bibliotheken zu ermöglichen. Gleichzeitig wird npm verwendet, um Ihre Angular 2 Applikation zu starten. Um npm verwenden zu können, installieren Sie am besten Node.js[5], welches in späteren Übungen benötigt wird und eine entsprechende Installation für npm mitbringt. Mit dem Befehl *„npm start“* können Sie in weiterer Folge Ihre Applikation starten und ausführen. 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.

Abgabemodalität

Beachten Sie die allgemeinen Abgabemodalitäten des TUWEL-Kurses[6]. 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:



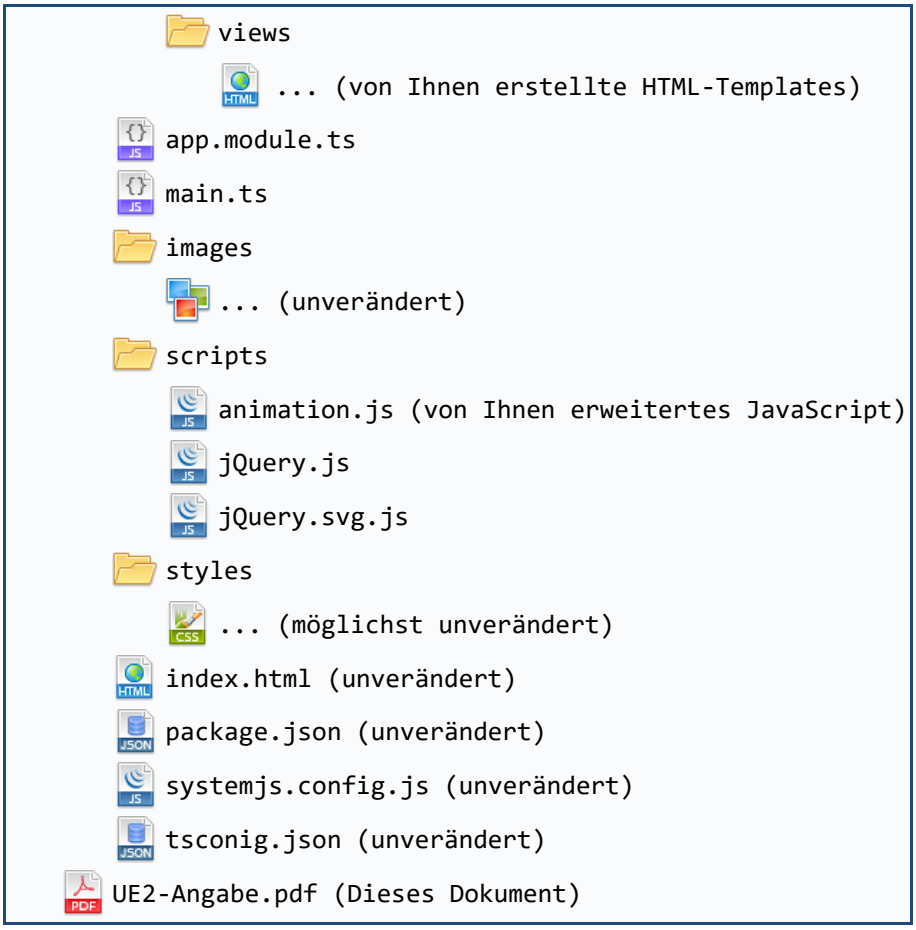
2 https://www.jetbrains.com/idea/

3 https://www.jetbrains.com/webstorm/

4 https://www.npmjs.com/

5 https://nodejs.org/en/

6 https://tuwel.tuwien.ac.at/course/view.php?id=7423



Alle Dateien müssen UTF-8 codiert sein!

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