Modularisierung und Zentralisierung von Angular-Komponenten auf Basis des Atomic Design

T1000

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Jonas Scherer

07.09.2021

Bearbeitungszeitraum	12 Wochen
Matrikelnummer, Kurs	6499975, TINF20B
Dualer Partner	DaimlerTSS, Stuttgart
Betreuer*in des Dualen Partners	Herr Sven Dahlmann

Inhaltverzeichnis

Abkürzungsverzeichnis

Git

Repo

SSH

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Betroffen dieser Zentralisierung und Modularisierung sind drei voneinander autark agierende Userinterfaces, welche in einem gemeinsamen GitHub Repository liegen. Aufgrund des Sourcecode Umfangs dieser drei Userinterfaces, ihrer Kopplung und ihrer monolithischen Softwarestruktur, sollen sich stark ähnelnde Komponenten dieser zentralisiert werden, sodass eine Reduktion von Sourcecode sowie eine einheitliche User Experience erzielt werden kann. Zur Kategorisierung der einzelnen Komponenten soll das Atomic Design Pattern angewendet werden, um eine klar definierte Reihenfolge der zu Zentralisierten Komponenten aufstellen zu können. Ohne diese Reihenfolge kann dieses Vorhaben zu mehr Schaden als Nutzen führen.

Da die Userinterfaces auf dem Framework Angular und der Programmiersprache Typescript basieren, soll der technologische Fokus auf diese gelegt werden. Die Module erfordern nicht nur eine Separierung im Code, sondern auch auf der Ebene der Source Code Verwaltung. Hier gilt es ein zentrales Repository bereitzustellen, aus dem die Komponenten konsumiert werden. Mit dieser Erweiterung muss außerdem eine Integration der Separierung in den Build-Workflow des Endsystems durchgeführt werden.

Die relevanten Komponenten werden konkret angepasst, modularisiert und zentralisiert.

2. Vor- und Nachteile einer Zentralisierung

Die Zentralisierung von Softwarecode bringt eine Menge Vorteile mit sich. Nicht nur die Möglichkeit, gleiche Funktionalität bei weniger Sourcecode beizubehalten, sondern kann durch diese auch über mehrere Userinterfaces hinweg ein einheitlicher Style eingehalten und so eine einheitliche User Experience gewahrt werden. Außerdem ist man in der Lage Änderungen in einer einzelnen Komponente vorzunehmen, welche sich direkt auf alle Userinterfaces auswirken, welche diese Komponente eingebunden haben. Mit dieser Vorgehensweise, kann man sich von der Monolithischen Struktur eines Softwarecodes hervorragend distanzieren.

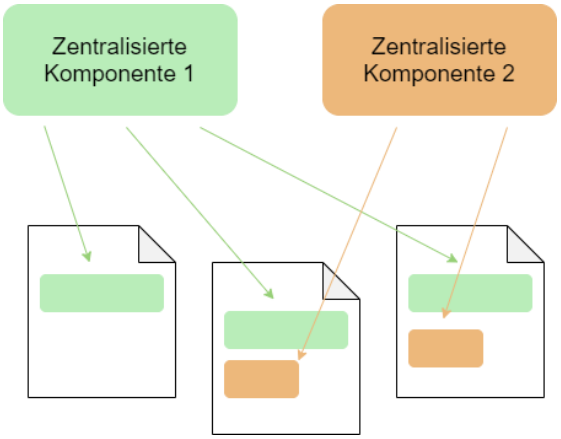

Abbildung 1: Visualisierung der Zentralisierung

Das Auslagern von Softwarecode kann aber auch ein Nachteil sein. So schön es klingt, dass nur eine Änderung in einer zentralisierten Komponente vorgenommen werden muss und dies sich auf alle Userinterfaces auswirkt, genauso gefährlich ist dies auch. Durch eine kleine Änderung in einer Datei können unzählige Fehler auftreten.
Weitere stark überwiegende Nachteile birgt die Zentralisierung nicht. Hat man einmal die richtigen Tools, wie hier GitHub, Angular, Typescript, bereits existierende Userinterfaces und ein Buildsystem, so steht der Zentralisierung nichts mehr im Wege.

3 Grundlagen, Stand der Technik

3.1 Atomic Design Pattern

Das Atomic Design Pattern nach Brad Frost, beschreibt die Zerlegung von komplexem Softwarecode in kleine, für jeden Softwareengineer nachvollziehbare Elemente. So wird aus einem enormen Code für ein Userinterface ein deutlich schlankerer, welcher einzelne Komponenten aufruft. Diese wiederrum können auch wieder zerlegt werden.

Das von Brad Frost entwickelte Pattern findet seinen Ursprung in der Chemie. Er beschreibt die Hinführung und den Gedankengang, welcher zur Entwicklung dieses Design Patterns beitrugen, in seinem Buch „Atomic Design“, welches 2016 erschienen ist, genauer. Faszinierend ist aber nicht die Geschichte, sondern die Theorie sowie die Anwendung des Atomic Designs.

Dieses Design unterteilt Softwarecode in fünf Kategorien, nämlich in:

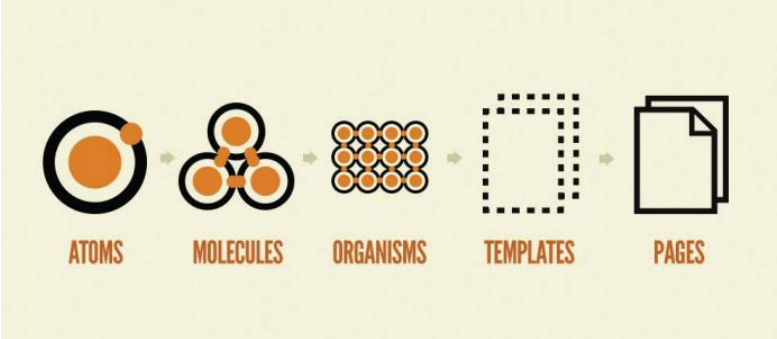

Abbildung 2: Atomic Design Pattern

3.1.1 Atome

Selbstverständlich gibt es in der Chemie kleinere Einheiten als ein Atom, beispielsweise Neutronen oder Protonen. Das Atom aber, ist die kleinste funktionelle Einheit, und somit stellt es auch im Atomic Design, die kleinstmögliche Einheit da.

Ein Atom, nach Brad Frost, enthält grundlegende HTML Elemente wie beispielweise einen Knopf zum Schließen eines Fensters oder ein Label in welches etwas hineingeschrieben wird.[1)] Zu Atomen zählen aber auch die verschiedenen Styles eines jeweiligen Elements, zum Beispiel die Farbe und Größe eines Knopfes.

[1)] Vgl. Brad Frost (2016), S. 42.

3.1.2 Moleküle

In der Chemie ist ein Molekül ein Zusammenschluss aus mindestens zwei Atomen, was auch für das Atomic Design Pattern gilt. Das Beispiel Molekül in Abbildung 2 enthält einen Knopf zum Schließen des Fensters und ein Label, in welchem der Fülltext steht. Somit besteht dieses Molekül aus zwei Atomen.

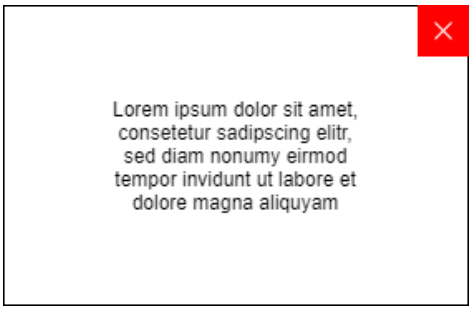

Abbildung 3: Beispiel Molekül

Werden zwei verschiedene Atome kombiniert, so erlangen diese einen Sinn.[2)] Schon jetzt zeigt sich deutlich, inwiefern dieses Prinzip im späteren Verlauf von Nutzen sein kann

3.1.3 Organismen

Ein Organismus ist ein schon deutlich komplexeres Softwareelement, denn sie können aus verschiedensten Zusammensetzungen von Atomen und Molekülen bestehen. Auch können sie das gleiche Atom oder Molekül beliebig oft beinhalten. Ein perfektes Beispiel, dessen sich auch Brad Frost in seinem Buch „Atomic Design“ bedient, ist die Kopfzeile, bzw. der sogenannte Header einer Website.

Abbildung 4: Beispiel Header

Der Beispiel Header in Abbildung 3 ist ein Beispiel für eine sehr einfache Kopfzeile einer Website. Dort befinden sich, von links betrachtet, drei Knöpfe und rechts davon ein Molekül. Dieses Molekül besteht aus einem Eingabefeld und einem Knopf. Somit ist hier eine steigende Komplexität festzustellen

[2)] Vgl. Frost B. (2016), S. 44 f.

3.1.4 Templates

Templates beziehungsweise Vorlagen kommen nicht mehr in der Chemie vor, sie wurden von Brad Frost in diese Reihe gegliedert. Sie beschreiben den grundlegenden Aufbau einer Webseite zum Beispiel welche Komponenten an welche Stelle eingefügt werden. Sie beachten aber nicht, welchen Inhalt eine Komponente besitzt.

„You can create good experiences without knowing the content. What you can't do is create good experiences without knowing your content structure"[3)], ein Zitat von Mark Boulton, welches die Wichtigkeit eins guten Template ausdrückt.

3.1.5 Pages

Pages beziehungsweise Seiten wurden ebenfalls von Brad Frost selbst in die Reihe des Atomic Design Patterns gegliedert. Sie werden deklariert als spezifische Instanz einer Vorlage, welche mit Inhalt gefüllt wird. Sie zeigt dann dem Entwickler, wie die Webseite final aussehen wird. Mit genau dieser Webseite interagieren dann die Endnutzer.[4)]

[3)] Boulton M. (2012)
[4)] Vgl. Frost B. (2016), S. 52

3.2 GitHub

„GitHub ist ein webbasierter kollaborativer Hosting-Dienst für […]“[5] Softwareprojekte. Durch diesen Dienst wird mehr als 56 Millionen Entwicklern und mehr als drei Millionen Organisationen die Möglichkeit gegeben, gemeinsam und ortsunabhängig an einem oder mehreren Projekten zu arbeiten.[6] Außerdem wird man als Entwickler nicht von der Arbeit anderer eingeschränkt, da jeder vorerst lokal auf seinem Endgerät arbeitet.

Die Kombination aus Repositorys und der Verwaltung von Version verschiedenster Dateien macht GitHub erst so Interessant für Unternehmen, welche mehrere Mitarbeiten an einem Projekt beschäftigen. Ein Repository ist sozusagen ein Ordner, auf welchen ein Mitarbeiter eines Projekts Zugriff hat. Dieses kann er sich klonen und hat den neusten Stand dieses Repos dann lokal auf seinem Endgerät und kann dort arbeiten ohne andere Entwickler dieses Projekts zu beeinflussen. Ist er fertig mit seiner Arbeit, so kann er diese in das Repo in der Cloud wieder ‚pushen‘, womit jeder Entwickler dann Zugriff auf die Änderungen hat. Jedes Repository hat einen sogenannten Masterbranch, in welchem das Hauptprojekt verläuft. Jeder Entwickler kann von diesem Branch, einen eigenen Brach abzweigen und auf diesem beispielsweise ein neues Feature entwickeln. Ist dieses Feature fertig, so kann er seinen Featurebranch pushen. Möchten seine Mitarbeiter dieses Feature im Projekt haben, so können sie diesen Featurebranch wieder zurück in das Projekt führen und alle Änderungen werden auf den Masterbranch übernommen.

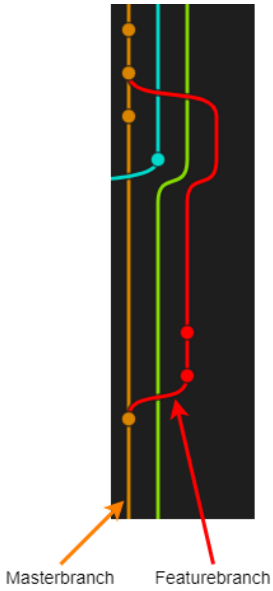

Abbildung 5: Gitbranching

In diesem Projekt führt an einer Versionsverwaltung kein Weg vorbei da die Zentralisierung von Komponenten an schon bestehenden oder noch mitten in der Entwicklung steckenden Userinterfaces durchgeführt wird und somit mehrere Entwickler zeitgleich an diesen arbeiten.

[5] Lederer A. (2021), S. 2
[6] GitHub (2021)

3.3 Angular

Der Nachfolger Angular von AngularJS ist ein Open Source Front-End-Webapplikationsframework, welches von Google LLC 2016 entwickelt wurde und auf Typescript basiert. Die Besonderheit von Angular ist der Aufbau einer Webapplikation in einzelnen Komponenten welche eine festgelegte Hierarchie befolgen.

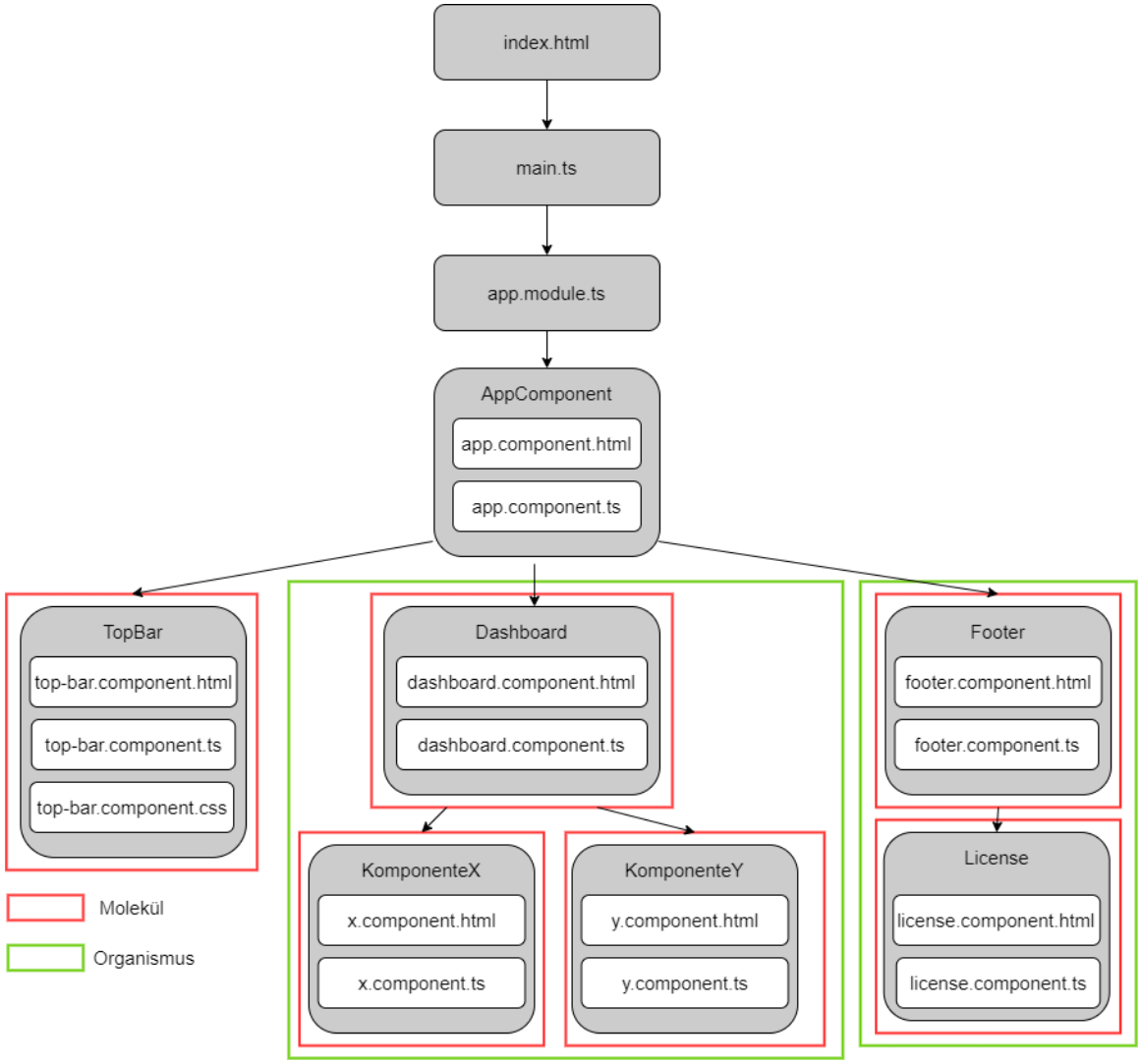

Abbildung 6: Beispielhafte Angular Struktur

Die index.html stellt das HTML-Grundgerüst dar, auf welchem die main.ts aufsetzt, die Startdatei für das Bootstrapping. In der darauffolgenden Typescript Datei app.module.ts, dem zentralen Angular-Modul, müssen alle Komponenten welche das Projekt enthält und benutzt importiert und deklariert werden, um diese nutzen zu können. Ab diesem Punkt befindet man sich schon mitten in der Webapplikation, denn in der AppComponent befindet sich das Template (app.component.html) sowie die

Root-Komponente (app.component.ts) der Webapplikation. In Abbildung 6 ist anschaulich, dass aus diesen Dateien im Anschluss alle zu verwendenden Angular Komponenten aufgerufen werden, ob direkt, oder indirekt. Hier kann es sich bei einzelnen Komponenten um Moleküle oder gar Organismen handeln.[7)]

Atome befinden sich normalerweise innerhalb der einzelnen Komponenten beziehungsweise Molekülen. Enthält die TopBar beispielsweise nur einen Knopf und einen Text, so ist sie ein Molekül. Nutzt eine Komponente aber eine andere, so ist diese Einheit ein Organismus, sie ist deutlich komplexer als ein einfaches Molekül. Stylings der einzelnen Atome innerhalb der Moleküle können sich entweder in der jeweiligen HTLM-Datei oder in einer weiteren CSS-Datei befinden.

Um eine Webapplikation lokal auf einer Adresse, beispielweise ***localhost:4200/xy*** zu testen, bietet Angular-CLI den Befehl ***ng serve***, welcher im jeweiligen Projektordner ausgeführt werden muss. Dies bietet die Möglichkeit in Echtzeit seinen Code zu kompilieren und zu testen, was im Bereich der Front-End-Entwicklung zu einem angenehmeren Workflow beiträgt.

[7)] Vgl. Malcher et al. 2020, S. 5ff

3.4 Typescript

Die noch junge Programmiersprache Typescript wurde 2012 von Microsoft entwickelt und ist eine objektorientiere Programmiersprache wie beispielsweise Java. Sie bietet daher auch verschiedenste bekannte Sprachkonstrukte wie Klassen, Vererbungen und Interfaces. Typescript ist jedoch eine Obermenge der 1996 entwickelten Sprache JavaScript. Im Vergleich zu dieser hat Typescript einige Vorteile, vor allem aber die starke Typisierung von Variablen, welche JavaScript nicht berücksichtigt. Dies führt zu deutlich weniger Fehlern und gestaltet das Verstehen des Codes signifikant einfacher. Letztendlich übersetzt der Typescript Compiler aber dennoch den Typescript Code in JavaScript Code, so ist jeder JavaScript Code auch automatisch korrekter Typescript Code.

Die wohl größten Vorteile von Typescript in Kombination mit Angular sind Modularität, Konsistenz sowie das zur Verfügung stehen verschiedenster Sprachkonstrukte. Angular wurde, wie im vorherigen Kapitel schon erwähnt, mit einer Komponentenstruktur entwickelt, welche einer programmierbaren Hierarchie entspricht. Diese Modularität ist mit Typescript sehr gut umzusetzen und führt zu einer klaren Struktur und einer guten Verständlichkeit des Programmcodes. Dank eben dieser Kombination zweier Technologien ist es auch mögliche eine hohe Konsistenz im Softwarecode zu erzielen, da eine Angular Komponente in Typescript immer den gleichen Aufbau haben muss.[8)]

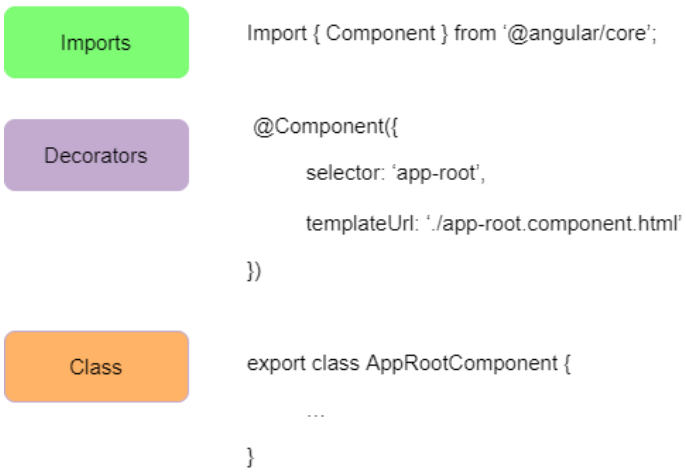

Abbildung 7: Aufbau einer Angular Komponente

[8)] Vgl. Wahlin D. (2019)

4. Bereitstellen eines zentralen Repositorys

Es gilt ein neues Repository in dem bevorzugten Version Control System GitHub, oder auch Source Code Verwaltung, einzurichten. Es stehen zur Auswahl das Erstellen eines neuen Repos lokal auf dem eigenen Endgerät, welches man dann in GitHub pushen muss, oder eben das Erstellen eines Repos in GitHub direkt, welches anschließend auf das Endgerät zu pullen ist. In diesem Fall wird Variante zwei präferiert und zusätzlich soll ein SSH-Key generiert werden, um sich nicht bei jedem push oder pull Authentifizieren zu müssen. Das zu nutzende Betriebssystem ist Linux beziehungsweise Ubuntu 20.04, weshalb die meisten Arbeitsschritte im Terminal stattfinden.

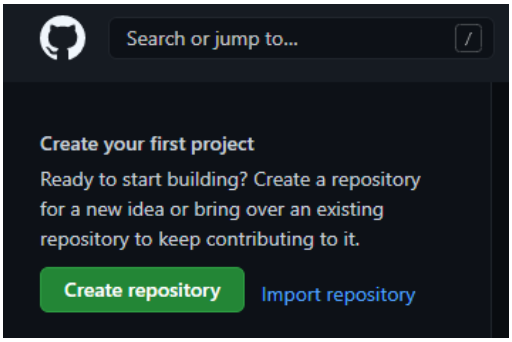

Abbildung 8: GitHub Create Repo

Zu allererst muss das Repository in GitHub online erstellt werden, um auf das lokale Endgerät geklont zu werden. Hierzu bietet Git die Funktion ***Create new repository***. Dort kann man anschließend einen Namen eintragen und die Sichtbarkeit von public auf private abändern, wenn gewünscht. Nun könnte man schon das bereits existierende Git Repository klonen und damit arbeiten, man müsste sich aber wie zuvor erwähnt bei jedem pull oder push authentifizieren. Dies wäre ein klonen über die HTTPS-Adresse des Repos, präferiert wird aber das klonen über das SSH Netzwerkprotokoll, wozu erst ein SSH Key eingerichtet werden muss. Dieser trägt dann zur Verknüpfung des GitHub Kontos mit dem Nutzer des Endgerätes bei. Durch den Befehl ***studienarbeit ~ $ ssh -keygen -t ed25519 -C ‚email@studienarbeit.com‘*** wird dieser generiert und durch eingeben eines Passwortes in ***/home/USER/.ssh/id_ed25519*** gespeichert. In dem Ordner ***.ssh*** befinden sich dann mehrere Dateien, interessant für dieses Vorhaben ist aber lediglich ***id_ed25519.pub***. In dieser Datei befindet sich nämlich der öffentliche SSH-Key welcher nun in GitHub unter den ***Settings***, ***SSH and GPG keys*** eingetragen werden muss. Nun ist der Nutzer des Endgerätes mit dem Nutzer von GitHub verknüpft und das Repo kann durch den Befehl ***studienarbeit ~ $ git clone git@github.com:USER/DHC.git*** geklont werden. Somit wurde das zentrale Repository erfolgreich erstellt.

Literaturverzeichnis

Anke Lederer (2021): GitHub Eine praktische Einführung. Von den ersten Schritten bis zu eigenen GitHub Actions. 1. Aufl.: dpunkt.verlag GmbH.

Brad Frost (2016): Atomic Design.

Dan Wahlin (2019): Die 5 wesentlichen Vorteile von Angular und TypeScript. Hg. v. t2informatik GmbH. o.O. Online verfügbar unter https://t2informatik.de/blog/softwareentwicklung/die-5-wesentlichen-vorteile-von-angular-und-typescript/, zuletzt geprüft am 22.04.2021.

GitHub, Inc (2021): Where the world builds software. o.O. Online verfügbar unter https://github.com/, zuletzt geprüft am 22.04.2021.

Malcher, Ferdinand; Hoppe, Johannes; Koppenhagen, Danny (2020): Angular. Grundlagen, fortgeschrittene Themen und Best Practices- inkl. RxJS, NgRx und PWA. 3. Aufl. Heidelberg: dpunkt.verlag GmbH.

Mark Boulton (2012): Structure First. Content Always. o.O. Online verfügbar unter https://markboulton.co.uk/journal/structure-first-content-always/, zuletzt aktualisiert am 06.02.2012, zuletzt geprüft am 20.04.2021.