Entwicklung einer Web-Anwendung zur Projektantragsverwaltung mit JSON Datenbank

Bericht zur betrieblichen Projektarbeit im Zeitraum:

06.04.2016 – 24.04.2016

von

**Timo Menhorn**

zur Erlangung des Abschlusses als

**Fachinformatiker Anwendungsentwicklung**



Ausbildungsberuf: Fachinformatiker Anwendungsentwicklung

Ausbildungsbetrieb: Donat IT GmbH

Verfasser: Timo Menhorn

Prüfung: Sommer 2020

Prüflingsnummer: 23056

Projektbetreuer: Jan Brinkmann

# Verzeichnisse

## Inhaltsverzeichnis

**Inhaltsverzeichnis**

[1 Verzeichnisse 2](#_Toc40261976)

[1.1 Inhaltsverzeichnis 2](#_Toc40261977)

[1.2 Abbildungsverzeichnis 3](#_Toc40261978)

[1.3 Glossar 4](#_Toc40261979)

[1.3.1 Begriffsklärung 4](#_Toc40261980)

[1.3.2 Reactive Forms Begriffe 8](#_Toc40261981)

[2 Einleitung 9](#_Toc40261982)

[2.1 Projektumfeld 9](#_Toc40261983)

[2.2 Projektziele 9](#_Toc40261984)

[2.3 Projektbegründung 10](#_Toc40261985)

[2.4 Projektschnittstellen 10](#_Toc40261986)

[2.5 Projektabgrenzung 10](#_Toc40261987)

[2.6 Zielplattform 11](#_Toc40261988)

[3 Projektplanung 12](#_Toc40261989)

[3.1 Zeitplanung 12](#_Toc40261990)

[3.2 Ressourcenplanung 12](#_Toc40261991)

[3.3 Personalplanung 12](#_Toc40261992)

[3.4 Sachmittelplanung 12](#_Toc40261993)

[4 Analysephase 13](#_Toc40261994)

[4.1 Ist- Analyse 13](#_Toc40261995)

[4.2 Soll- Analyse 13](#_Toc40261996)

[4.3 Wirtschaftlichkeitsanalyse 14](#_Toc40261997)

[4.3.1 Make-or-Buy Entscheidung 14](#_Toc40261998)

[4.3.2 Projektkosten 14](#_Toc40261999)

[4.3.3 Nicht-monetäre Vorteile 15](#_Toc40262000)

[4.4 Lastenheft 15](#_Toc40262001)

[5 Projektphasen 16](#_Toc40262002)

[6 Angular 17](#_Toc40262003)

[6.1 Abgrenzung von Angular zu AngularJS 18](#_Toc40262004)

[7 Alternativen zum Framework Angular 19](#_Toc40262005)

[7.1 React 20](#_Toc40262006)

[7.2 VueJS 21](#_Toc40262007)

[7.3 Angular Versionen 22](#_Toc40262008)

[7.4 Übersicht Frameworks und Entscheidung 23](#_Toc40262009)

[8 Kostenschätzung 24](#_Toc40262010)

[9 Realisierung 25](#_Toc40262011)

[10 Fazit 33](#_Toc40262012)

[11 Quellen 34](#_Toc40262013)

[11.1 Webquellen 34](#_Toc40262014)

[11.2 Buchquellen 34](#_Toc40262015)

[12 Selbsterklärung 35](#_Toc40262016)

[13 Screenshots des Prototypens 36](#_Toc40262017)

[14 Screenshots der Anwendung 38](#_Toc40262018)

## Abbildungsverzeichnis

[Abbildung 1: Klassendiagramm Zusammenhang der Formfields 14](#_Toc40100858)

[Abbildung 2: Funktionsweise von Angular 17](#_Toc40100859)

[Abbildung 3: Hello World Beispiel Angular 18](#_Toc40100860)

[Abbildung 4: Star History der Frameworks im Vergleich 19](#_Toc40100861)

[Abbildung 5: Jobangebote für Entwickler auf gängigen Plattformen 20](#_Toc40100862)

[Abbildung 6: Hello World Beispiel React 21](#_Toc40100863)

[Abbildung 7: Hello World Beispiel VueJS 21](#_Toc40100864)

[Abbildung 8: Ablaufdiagramm Genehmigungsprozess 23](#_Toc40100865)

[Abbildung 9: JSON Beispiel für Home 24](#_Toc40100866)

[Abbildung 10: HTML Beispiel für Home 24](file:///C:\Users\menhorti\Documents\Ausbildung\Abschlussarbeit\projectdoku_v2.docx#_Toc40100867)

[Abbildung 11: JSON Beispiel für Create 25](#_Toc40100868)

[Abbildung 12: HTML Beispiel für Create 26](#_Toc40100869)

[Abbildung 13: Initialisierung des Formulars 27](#_Toc40100870)

[Abbildung 14: Validation Messages der FormControls 27](#_Toc40100871)

[Abbildung 15: Zwischenspeicher für Fehlermeldungen zur Laufzeit 28](#_Toc40100872)

[Abbildung 16: Loggen von Value Changes 29](#_Toc40100873)

[Abbildung 17: Usermanagement 30](#_Toc40100874)

[Abbildung 18: Entwurf Login 34](#_Toc40100875)

[Abbildung 19: Entwurf Home 34](#_Toc40100876)

[Abbildung 20: Entwurf List 35](#_Toc40100877)

[Abbildung 21: Entwurf Usermanagement 35](#_Toc40100878)

[Abbildung 22: Anwendung Login 36](#_Toc40100879)

[Abbildung 23: Anwendung Home 36](#_Toc40100880)

[Abbildung 24: Anwendung List 37](#_Toc40100881)

[Abbildung 25: Anwendung Create 37](#_Toc40100882)

[Abbildung 26: Anwendung Usermanagement 38](#_Toc40100883)

## Glossar

### Begriffsklärung

**Architekturkonzepte**

Eine Softwarearchitektur ist einer der Architekturtypen in der Informatik und beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb eines Softwaresystems.

**Authentifizierung**

Die Authentifizierung sorgt im Computer- und Internetumfeld dafür, dass die Identität eines Benutzers gegenüber einem System nachgewiesen und verifiziert werden kann.

**Backend**

Als Backend wird der Teil eines IT-Systems bezeichnet, der sich mit der Datenverarbeitung im Hintergrund beschäftigt.

**Bindings**

Verknüpfung zwischen Elementen im View und Datenstrukturen im Model.

**Client-Server-Architektur**

Das Client-Server-Modell beschreibt eine Möglichkeit, Aufgaben und Dienstleistungen innerhalb eines [Netzwerkes](https://de.wikipedia.org/wiki/Rechnernetz) zu verteilen. Die Aufgaben werden von [Programmen](https://de.wikipedia.org/wiki/Computerprogramm) erledigt, die in [Clients](https://de.wikipedia.org/wiki/Client) und [Server](https://de.wikipedia.org/wiki/Server_(Software)) unterteilt werden.

**Controller**

Die Steuerung verwaltet die Präsentation und das Modell.

**CSS**

CSS (Cascading Style Sheets) ist eine Formatierungssprache für HTML-, SVG- und XML-Dokumente.

**Desktop-Applikation**

Als Anwendungssoftware werden [Computerprogramme](https://de.wikipedia.org/wiki/Computerprogramm) bezeichnet, die genutzt werden, um eine nützliche oder gewünschte nicht [systemtechnische](https://de.wikipedia.org/wiki/Systemsoftware) Funktionalität zu bearbeiten oder zu unterstützen.

**Direktives**

Strukturelle Template Direktiven - verändern den DOM, d.h. fügen Elemente hinzu oder entfernen sie.

Attribut-Direktiven - verändern nur das Aussehen oder Verhalten von DOM-Elementen.

**Framework**

Ein Framework ist ein Programmiergerüst, das in der Softwaretechnik, insbesondere im Rahmen der objektorientierten Softwareentwicklung sowie bei komponentenbasierten Entwicklungsansätzen, verwendet wird.

**Frontend**

Frontend ist ein Begriff aus der Informationstechnik, welcher die sichtbare Oberfläche für Nutzer bezeichnet.

**Hashwertbildung**

Kryptografische Hash-Funktionen generieren aus beliebig langen Datensätzen eine Zeichenkette mit einer festen Länge. Ein Datensatz kann ein Wort, ein Satz, ein längerer Text oder auch eine ganze Datei sein.

**Interpolation**

Template-Strings können Platzhalter beinhalten. Die Ausdrücke in den Platzhaltern und der Text innerhalb der Back-Ticks können an eine Funktion übergeben werden.

**JSON**

Die JavaScript Object Notation ist ein kompaktes Datenformat in einer einfach lesbaren Textform und dient dem Zweck des Datenaustausches zwischen Anwendungen.

**MVVM**

Model View ViewModel ist ein Entwurfsmuster und eine Variante des Model-View-Controller-Musters. Es dient zur Trennung von Darstellung und Logik der Benutzerschnittstelle. Gegenüber dem MVC-Muster kann die Testbarkeit verbessert und der Implementierungsaufwand reduziert werden, da keine separaten Controller-Instanzen erforderlich sind.

**Node.js**

Node.js ist eine serverseitige Plattform in der Softwareentwicklung zum Betrieb von Netzwerkanwendungen. Insbesondere lassen sich Webserver damit realisieren.

**NPM**

npm (ehemals Node Package Manager) ist ein Paketmanager für die JavaScript-Laufzeitumgebung Node.js.

**Open-Source-Software**

Als Open Source wird [Software](https://de.wikipedia.org/wiki/Software) bezeichnet, deren [Quelltext](https://de.wikipedia.org/wiki/Quelltext) öffentlich und von Dritten eingesehen, geändert und genutzt werden kann. Open-Source-Software kann meistens kostenlos genutzt werden.

**PDF**

Das Portable Document Format ist ein plattformunabhängiges Dateiformat, das 1993 vom Unternehmen Adobe Inc. entwickelt und veröffentlicht wurde und aktuell von der PDF Association weiterentwickelt wird.

**PostgreSQL**

PostgreSQL, ist ein freies, objektrelationales Datenbankmanagementsystem. Seine Entwicklung begann in den 1980er Jahren, seit 1997 wird die Software von einer Open-Source-Community weiterentwickelt.

**Rendern**

Software-Rendering bezeichnet Bildsynthese ohne spezialisierte Hardware, d. h. nur durch die CPU ohne Unterstützung durch eine Grafikkarte oder Vergleichbares. Die Grafikkarte leitet dabei unbeteiligt die von der CPU berechneten Daten an den Monitor weiter.

**REST API**

REST steht für REpresentational State Transfer, API für Application Programming Interface. Gemeint ist damit ein Programmierschnittstelle, die sich an den Paradigmen und Verhalten des World Wide Web orientiert und einen Ansatz für die Kommunikation zwischen Client und Server in Netzwerken beschreibt.

**Scopes**

Beim Hinzufügen von Eigenschaften zum $scope-Objekt im Controller erhält die Ansicht (HTML) Zugriff auf diese Eigenschaften.

**Template**

Templates sind Schablonen, die Programmcode-Sequenzen in allgemein wiederverwendbarer Form bereitstellen.

**TypeScript**

TypeScript ist eine von Microsoft entwickelte Programmiersprache, die auf den Vorschlägen zum ECMAScript-6-Standard basiert. Sprachkonstrukte von TypeScript, wie Klassen, Vererbung, Module und anonyme Funktionen wurden auch in ECMAScript 6 übernommen.

**User-Interface-Komponenten**

Grafische Benutzeroberfläche oder auch grafische Benutzerschnittstelle bezeichnet eine Form von Benutzerschnittstelle eines Computers. Sie hat die Aufgabe, Anwendungssoftware auf einem Rechner mittels grafischer Symbole, Steuerelemente oder auch Widgets genannt, bedienbar zu machen.

**Usermanagement**

Die Benutzerverwaltung ist eine wichtige Aufgabe des Administrators. Er richtet Benutzer und Benutzerkennungen ein und vergibt oder entzieht Zugriffsberechtigungen für IT-Systeme oder Anwendungen.

**Webanwendung**

Eine Webanwendung ist ein Anwendungsprogramm nach dem Client-Server-Modell. Anders als klassische Desktopanwendungen werden Webanwendungen nicht lokal auf dem Rechner des Benutzers installiert.

**Webbrowser**

Webbrowser oder allgemein auch Browser sind spezielle Computerprogramme zur Darstellung von Webseiten im World Wide Web oder allgemein von Dokumenten und Daten.

### Reactive Forms Begriffe

**Form Array**

Ein Form Array verfolgt den Wert und Validierungsstatus eines Arrays von FormControls, FormGroups und FormArrays. Der Status eines FormArray wird mit dem reduce-Verfahren berechnet. Wenn ein FormControl innerhalb des FormArray invalid ist, ist das ganze FormArray invalid.

**Form Control**

Ein Form Control entspricht einem einzelnen Feld des Formulars und verfolgt den Wert und Validierungsstatus.

**Form Group**

Eine Form Group aggregiert Werte von allen unterliegenden FormControls und mündet sie in ein Objekt, das den Namen des FormControls als Schlüssel verwendet. Ein Root-Element eines reactive form ist immer eine FormGroup.

**Dirty**

Die Eigenschaft dirty eines Form Controls wird auf true gesetzt, sobald der Nutzer die Value ändert.

**Invalid**

Die Eigenschaft valid eines Form Controls wird auf true gesetzt, wenn das Control keine Validatoren besitzt oder alle Validatoren erfüllt sind.

**Touched**

Die Eigenschaft touched eines Form Controls wird auf true gesetzt, sobald der User erstmalig auf das Control geklickt hat.

**Validatoren**

Reactive Forms haben den Vorteil, dass die Strukturen und alle Validierungen an einem Platz definiert werden und feldübergreifende Validierungen so sehr einfach umsetzbar sind.

# Einleitung

## Projektumfeld

Das Projekt wird im Rahmen einer Umstrukturierung abgewickelt und in den Örtlichkeiten der Donat IT GmbH umgesetzt. Verantwortlich für den gesamten Projektverlauf ist der Auszubildende.

Vor 35 Jahren begann die Geschichte der DONAT Group. Durch permanente Weiterentwicklung hat sich die Unternehmensgruppe seither zu einem der größten IT-Unternehmen der Region Ingolstadt etabliert.

Beginnend als IT-Dienstleister der Automotive-Branche stellt die Donat IT ihre Kernkompetenzen künftig auch weiteren Branchen bundesweit zur Verfügung. Von Prozessberatung, agile sowie klassische individuelle Software-Entwicklung und App-Entwicklung bis zu Mainframe Technologien und Life Cycle Management gemäß ITIL-Service-Set bietet die Donat IT alles aus einer Hand.

Damit das Projekt den zeitlichen Rahmen nicht übersteigt, wird auf eine imitierte REST API zurückgegriffen. Im weiteren Zuge der Ausfertigung werden die Daten auf eine SQL-Datenbank transferiert werden, damit auch größere Datenbestände im Verlauf des Projekts nicht zu Problemen führen und ein Mehrfachzugriff ermöglicht wird.

## Projektziele

Die Webapplikation\* basiert auf einem Angular 9 Frontend\*. Die Daten, die durch die Usereingaben und Aktionen generiert werden, werden in einer JSON\* Datenbank abgespeichert.

Die Komponenten sollen sich größtenteils dynamisch generieren, um eine einfache Wartbarkeit bei sowohl kleinen als auch größeren Änderungen zu garantieren. Die Daten hierfür kommen jeweils aus einer Export Klasse im JSON\* Format. Die Validierung der Nutzereingaben kommt ebenfalls aus dieser Klasse.

Die Login Komponente soll überprüfen, ob der Nutzer bereits angelegt ist und im Erfolgsfall das Passwort überprüfen. Wenn der Nutzer sich neu registriert, soll das Passwort gehasht\* übertragen und in der Datenbank abgelegt werden.

Auf der Home Komponente soll ein Antragsteller die Auswahl haben, ob er einen Antrag anlegen oder eine Übersicht der gestellten Anträge haben will.

Wenn er einen Antrag anlegen will, bekommt er eine Liste mit möglichen Projektthemen und landet anschließend in der Komponente zum Anlegen. Wenn er auf die Übersichtskomponente will, bekommt er eine Liste mit möglichen Projektstatus, woraufhin nur die Anträge in diesem Status angezeigt werden.

Zum Usermanagement\*, soll es eine Komponente geben, auf der ein Admin Rollen vergeben, User ändern, hinzufügen oder löschen kann.

Darüber hinaus muss es eine Möglichkeit geben um Screenshots aufzunehmen. Diese sollen als PDF\* heruntergeladen werden um eine einfache Handhabung zu garantieren.

## Projektbegründung

Derzeit werden Projektanträge durch die Bereichsleiter mithilfe von Excel Dateien erstellt und abgespeichert. Je nach Größe und Umfang des Projektes bedarf es noch weiterer Genehmigungen, die durch Versenden des Excel Dokuments eingeholt werden.

Diese Anträge durchlaufen einen gewissen Genehmigungsprozess, welcher jedoch kaum zu überblicken ist, daher ist eine Statusschaltung gewünscht. Diese soll anzeigen, ob der Antrag bereits genehmigt oder abgelehnt ist und je nach aktuellem Status nur bestimmte Weiterschaltungsmöglichkeiten zulassen.

Die Lösung soll zeitsparend, modern und übersichtlich konstruiert werden.

## Projektschnittstellen

Das Projekt ist unabhängig von technischen Schnittstellen, da die Datenbank aktuell ein NPM\* Modul ist und die REST API\* simuliert wird. Wenn POST-, PUT-, PATCH- oder DELETE-Requests gestellt werden, werden Änderungen mithilfe von lowdb (Kleine JSON-Datenbank für Node, Electron und den Browser) automatisch in der Datei db.json gespeichert.

In einer späteren Version des Projektes werden die Daten zur Verwaltung der Nutzer, der Statusschaltung und angelegten Anträgen auf eine PostgreSQL\*-Datenbank transferiert, damit es auch bei größeren Datenbeständen zu keinen Problemen kommt. Dadurch wird auch ein Mehrfachzugriff möglich, welcher sich für eine Applikation mit einer großen Anzahl an Nutzern mit verschiedenen Rechten durchaus anbietet. Gleichzeitig wird dadurch auch die Kompatibilität mit mobilen Endgeräten gewährleistet, da SQL Interpreter auf jedem Gerät verfügbar sind. Auf die Datenbank soll dann mit einem Java Backend\* zugegriffen werden, da dies firmenintern genutzt wird.

Der Einstieg in das Projekt erfolgte durch ein Meeting zur Klärung der Statusschaltung und die Abnahme der Webapplikation\* bildete den Ausstieg.

## Projektabgrenzung

Um den Mitarbeitern den Umstieg auf das Onlinetool zu erleichtern, soll ausgewählten Mitarbeitern dieser Prototyp als vorläufige Version zur Verfügung gestellt werden.

Dieser Prototyp beinhaltet einen funktionierenden Login um Zugriff auf das System zu erlangen. Der Nutzer kann bereits Projektanträge erstellen oder die bestehenden Anträge einsehen sowie nach einem Status filtern. Zudem hat der Admin eine Usermanagement\* Oberfläche, auf der er User ändern, hinzufügen oder löschen kann.

## Zielplattform

Das Projektantragsverwaltungstool wurde als Webapplikation\* erstellt. Der Vorteil hiervon ist, dass auf Seiten des Clients\* lediglich ein Webbrowser\* nötig ist. Die Benutzerfreundlichkeit ist verbessert, da die Software nicht extra installiert werden muss. Außerdem kann durch die Client-Server-Architektur\* von mehreren Geräten aus gleichzeitig auf die Software zugegriffen werden. Auch die Wartung wird vereinfacht, da das Wartungspersonal über das Internet einfach auf die Webapplikation\* zugreifen und z.B. eine neue Version hochladen, die anschließend sofort vom User genutzt werden kann.

# Projektplanung

## Zeitplanung

Dem Autor stand ein Durchführungszeitraum von 70 Stunden zur Verfügung. Eine grobe Zeitplanung findet sich nachfolgend.

## Ressourcenplanung

Für die Erstellung des Projekts wird ein Schreibtisch Arbeitsplatz mit einem Laptop (HP Pro-Book 650 G4 mit Windows 10 x64) sowie ein Internetzugang benötigt.

## Personalplanung

Der Ausbilder stand bei organisatorischen Fragen zur Verfügung. Die Stundensätze sind Pauschalsätze, die sich unter anderem aus dem Bruttostundenlohn, den Sozialaufwendungen des Arbeitgebers und der Ressourcennutzung zusammensetzen.

## Sachmittelplanung

Um die anfallenden Kosten möglichst gering zu halten, wurde bei der Auswahl der Software darauf geachtet, dass diese kostenfrei zur Verfügung stehen, z.B. als Open-Source-Software\* oder durch bereits vorhandene Lizenzen.

# Analysephase

## Ist- Analyse

Derzeit werden Projektanträge durch die Bereichsleiter mithilfe von Excel Dateien erstellt und abgespeichert. Je nach Größe und Umfang des Projektes bedarf es noch weiterer Genehmigungen, die durch Versenden des Excel Dokuments eingeholt werden.

Im Moment wird das firmenweit eingesetzte Tool Excel als Grundlage zur Erstellung der Anträge genutzt. Jedoch sind innerhalb dieser Excelvorlage kaum Einschränkungen durch definierte Felder (nahezu durchgängig erstellte Freitextfelder) gegeben, was zu einer hohen Fehleranfälligkeit führt.

Zudem kommt es immer wieder zu Fehlern, wenn ein Antrag weiterversendet wird, da oft die falschen Ansprechpartner ausgewählt werden und die Anträge so bei den falschen Genehmigungsebenen landen.

Zudem konnte ein Antragsteller bisher nur auf Nachfrage beim Vorgesetzten, den aktuellen Status seines Antrags erfragen und hatte dies nie übersichtlich dargestellt.

## Soll- Analyse

Es soll eine Webapplikation\* entwickelt werden, bei der sich Nutzer authentifizieren\* oder neu registrieren müssen um anschließend auf ein Dashboard zu gelangen. Der Standardnutzer sieht auf diesem Dashboard zwei Kacheln.

Wenn er die erste Kachel auswählt, werden ihm darunter verschiedene Projektgebiete angezeigt, aus denen er ein Themengebiet für einen neuen Antrag auswählen kann. Wenn er ein Themengebiet ausgewählt hat, gelangt er so auf die Ansicht zum Anlegen von Anträgen.

Die zweite Kachel zeigt eine Vorauswahl über mögliche Antragsstatus. Wenn er einen davon auswählt, gelangt er auf die Übersicht der von ihm gestellten Anträge, die sich im ausgewählten Status befinden.

Der Admin sieht in dieser Ansicht noch eine dritte Kachel, über welche er zum Usermanagement\* gelangt.

Auf der Ansicht zum Anlegen werden beim Initialisieren die Eingabefelder samt Validatoren\* und zusätzliche CSS\* Attribute geladen, der Antragsteller wird mit dem Usernamen und die Antragsart mit dem ausgewählten Projektthema vorbefüllt. Wenn die nötigen Eingaben gemacht wurden, kann der User den Antrag absenden.

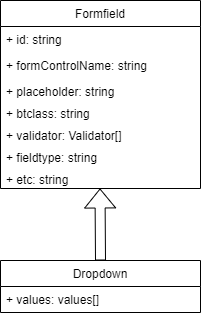


Abbildung : Klassendiagramm Zusammenhang der Formfields

Auf der Übersichtsseite kann der Nutzer Anträge auswählen, um diese nochmal in einer geschützten Ansicht detailliert zu sehen.

Der Genehmiger kann auf dieser Ansicht Anträge weiterschalten oder ablehnen, wenn dieser Antrag nicht genehmigt wird.

Im Usermanagement\* kann ein Admin die User verwalten. Hier kann er neue User anlegen, löschen oder bestehende User ändern. Der Admin soll hier volle Kontrolle haben. Username, Passwort oder Rollen sollen änderbar sein.

## Wirtschaftlichkeitsanalyse

### Make-or-Buy Entscheidung

Da es sich hierbei um ein kleines internes Projekt handelt, wurde entschieden das Projekt intern entwickeln zu lassen, da der Wissenstransfer zu externen Firmen zu lange dauern und umständlich wäre. Zudem ist es ein Übungsprojekt, um Angular Reactive Forms Funktionen für andere Großprojekte zu entwickeln, auszuarbeiten und weiter zu entwickeln.

### Projektkosten

Bei dem Projekt müssen Kosten für das Personal, sowie für die Sachmittel aufgebracht werden.

### Nicht-monetäre Vorteile

Das Projekt dient der Ausarbeitung von Angular Funktionen zum dynamischen Generieren von Eingabefeldern und Formularen um zu testen, welche Daten in einer JSON\* mitgeliefert werden müssen, um ein Formular vollständig dynamisch und basierend auf Daten generieren zu können.

## Lastenheft

Das Lastenheft wurde bereits im Voraus des Projekts vom Autor gemeinsam mit Verwaltungsmitarbeitern des Unternehmens erstellt. Dadurch wurden die Anforderungen schon früh klar definiert, weswegen die weitere Planung schneller und präziser gemacht werden konnte.

# Projektphasen

Zur Umsetzung des Teilprojekts standen dem Auszubildenden eine Zeitspanne von 70 Stunden zur Verfügung. Diese wurde während der Planung des Projekts festgelegt, bevor an dem Projekt zu arbeiten begonnen wurde. Die Einteilung sieht wie folgt aus. Es wird die Zeit berücksichtigt, in der das Programm, sowie seine In- und Outputs analysiert werden. Außerdem wird auf verschiedene Möglichkeiten eingegangen, wie der Ersatz durchgeführt werden könnte und eine Kostenschätzung erstellt. Im Folgenden wird die Zeiteinteilung der verschiedenen Phasen des Projektes sowie der Projektdokumentation graphisch dargestellt.

|  |  |  |
| --- | --- | --- |
| Projektphase | Inhalt | Dauer |
| Analyse | Ermittlung IST-Zustand | 2 |
|  | Ermittlung SOLL-Zustand | 2 |
|  | Analyse des Genehmigungsprozess | 1 |
| Konzeption | Ausarbeitung des Genehmigungsprozess | 1 |
|  | Erstellen eines Dummy Prototypen\* | 4 |
| Realisierung | Login mit Passwort Hashing\* | 8 |
|  | UI zum Anlegen von Anträgen | 7 |
|  | UI zum Anzeigen und Filtern von Anträgen | 5 |
|  | UI für Usermanagement\* | 10 |
|  | Erstellung der Statusschaltung | 4 |
| Sonstiges | Projektbesprechungen und Wissenstransfer | 1 |
|  | Projektdokumentation | 25 |
| Gesamt |  | 70 |

# Angular



Abbildung : Funktionsweise von Angular

Die clientseitige Programmierung\* ist in der letzten Zeit extrem komplex geworden. Dazu sind die Anforderungen an eine Webapplikation\* mittlerweile genauso hoch wie an eine Desktop-Applikation\* – wenn nicht sogar höher. Der Wunsch nach einem Framework\*, das es dem Entwickler ermöglicht, all diese Anforderungen im Web abzudecken, wird somit größer.

Angular ist ein clientseitiges Framework\*, mit dem es möglich ist, Webapplikationen\* zu erstellen. Es hilft dem Entwickler dabei, bekannte sowie neue Architekturkonzepte\* auf den Client zu bringen und komplexe Anwendungen zu entwickeln. Dabei kann die Arbeit nicht nur auf dem Server ausgerichtet werden, sondern via Javascript auch auf dem Client. Die Applikation kann durch Trennung der Zuständigkeiten moderne Architekturkonzepte\* erfüllen und ist damit wartbarer und testbarer.

Angular wurde mit Typescript\* entwickelt und ist auf eine komponentenbasierte Architektur\* ausgerichtet. Es vereint moderne Architekturansätze mit der nötigen Flexibilität, um komplexe Anforderungen an Webapplikationen\* zu meistern. Durch die Verwendung von Typescript\* hat Angular ähnliche Typensicherheit, wie Entwickler das von anderen Sprachen wie C# oder Java kennen. Trotzdem ist Angular nicht an Typescript\* gebunden und ermöglicht es dem Entwickler, auch Javascript mit ES5 oder ES2015/ES6 zu verwenden.

## Abgrenzung von Angular zu AngularJS

* Angular kennt keine „scopes“\* oder Controller\*, sondern verwendet eine Hierarchie von Komponenten als zentrales Architekturkonzept.
* Angular hat eine einfachere Syntax für Ausdrücke: Mit „[ ]“ werden Bindings\* für Eigenschaften und mit „( )“ werden Bindings\* für Events erzeugt.
* Mobile-First-Ansatz: Die Anforderungen von mobilen Plattformen haben besondere Priorität.
* Modularität: Module können Funktionalitäten auslagern, sodass Code leichtgewichtiger und schneller wird.
* Nur moderne Browser werden unterstützt, wodurch weniger Workarounds wegen Browser-Kompatibilitätsproblemen benötigt werden.
* Verbesserte Dependency Injection: Bindings\* ermöglichen es, dass Abhängigkeiten benannt werden können.
* Direktives\* dienen dem Zweck, das Aussehen und Verhalten eines Tags dynamisch zu verändern.
* Dynamisches Laden
* Asynchrone Kompilierung von Templates
* Einfacheres Routing
* Kontroller\* und $scope\* werden durch Komponenten und Direktiven\* ersetzt. Eine Komponente ist eine Direktive\* mit einem Template\*.

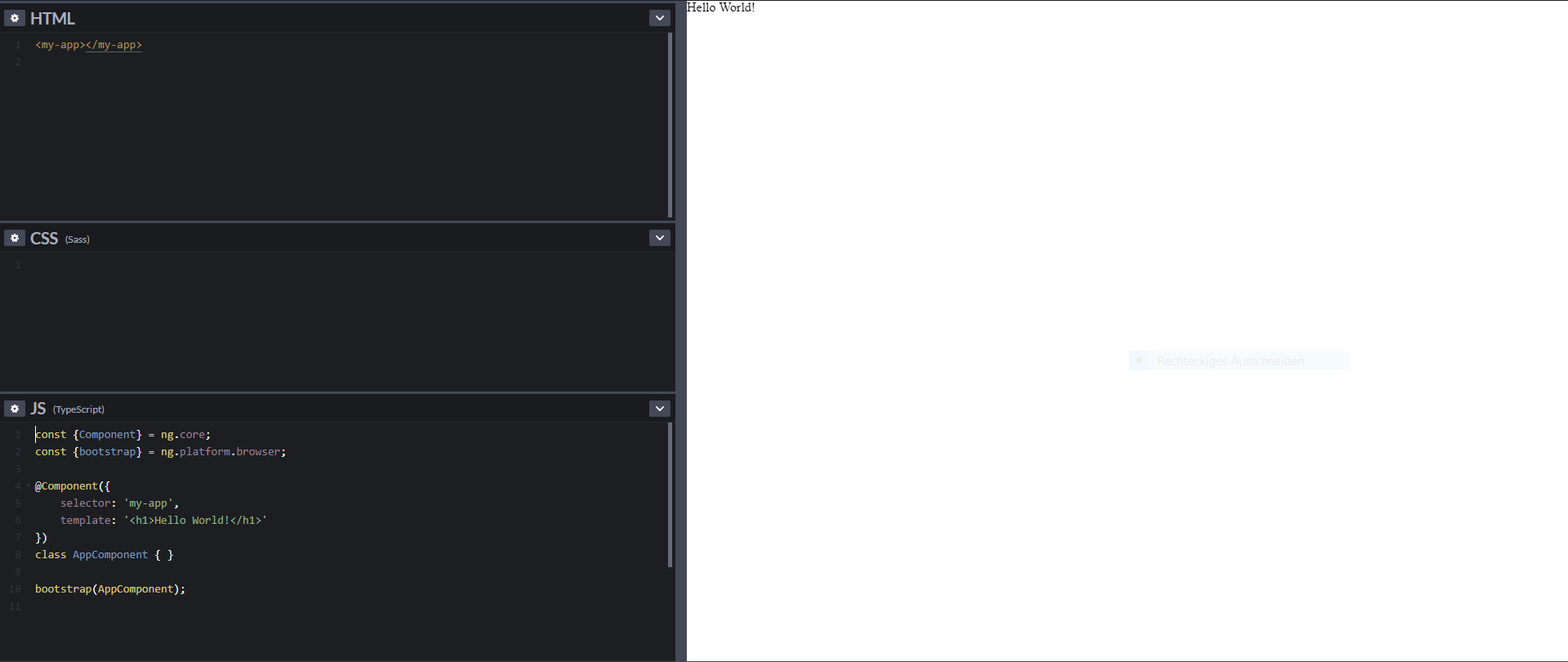


Abbildung : Hello World Beispiel Angular

# Alternativen zum Framework Angular

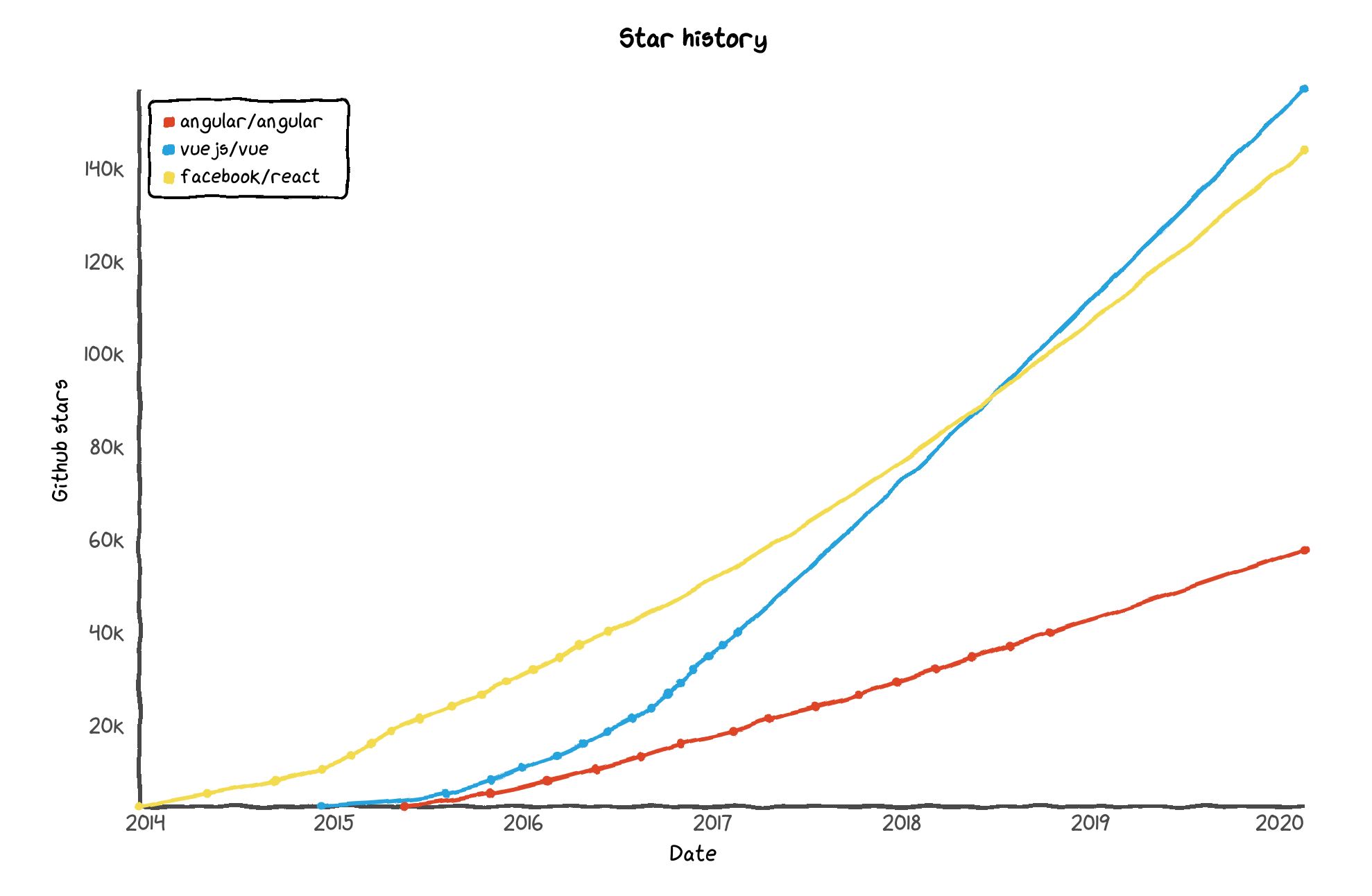


Abbildung : Star Historie der Frameworks im Vergleich

Durch die Sterne Vergabe auf Github lässt sich die Beliebtheit schön vergleichen. Bei Angular ist es eine stetige und eher linear ansteigende Beliebtheit erkennbar, wo hingehend man bei VueJS und React eher von einem exponentiellen Wachstum reden kann.

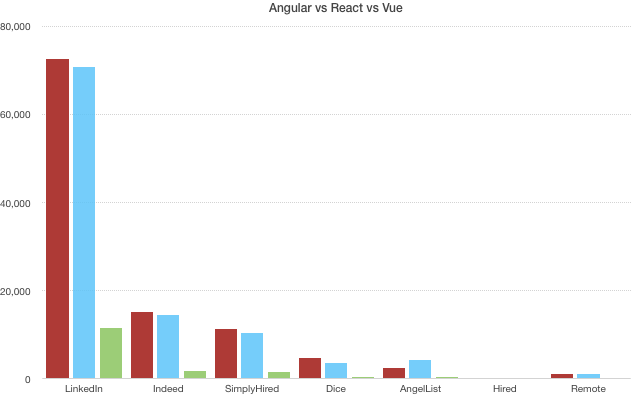


Abbildung : Jobangebote für Entwickler auf gängigen Plattformen

Die aktuellen Jobangebote für Entwickler sehen ähnlich aus, wobei es erstaunlich ist, das VueJS bisher kaum angefragt wird, da es einfach zu lernen ist und ähnlich umfangreiche Funktionalität bietet, wie Angular. Dieser Trend kommt vermutlich daher, dass die größeren Unternehmen eher Frameworks\* mit einer vollständigen und transparenten Dokumentation nutzen möchten, um es den Entwicklern einfacher zu machen.

## React

React ist eine JavaScript-Softwarebibliothek, die ein Grundgerüst für die Ausgabe von User-Interface-Komponenten\* von Webseiten zur Verfügung stellt (Webframework\*). Komponenten werden in React hierarchisch aufgebaut und können in dessen Syntax als selbst definierte HTML-Tags repräsentiert werden. Das Modell von React verspricht durch die Konzepte des unidirektionalen Datenflusses und des „Virtual DOM“ den einfachen, aber trotzdem performanten Aufbau auch komplexer Anwendungen. React bildet typischerweise die Basis für Single-Page-Webanwendungen, kann jedoch auch mittels Node.js\* serverseitig (vor-)gerendert\* werden.

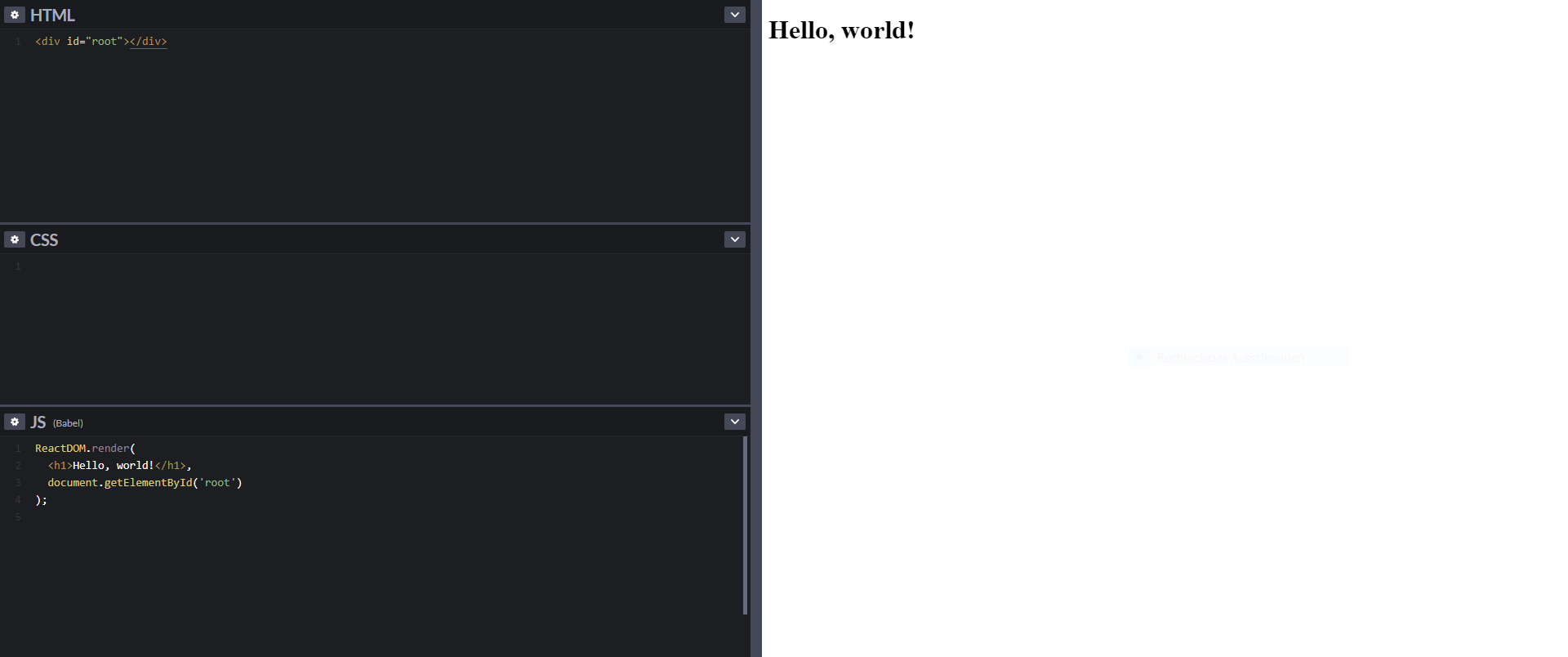


Abbildung : Hello World Beispiel React

## VueJS

Vue.js ist ein clientseitiges [JavaScript](https://de.wikipedia.org/wiki/JavaScript)-[Webframework](https://de.wikipedia.org/wiki/Webframework)\* zum Erstellen von [Single-Page-Webanwendungen](https://de.wikipedia.org/wiki/Single-Page-Webanwendung) nach dem [MVVM](https://de.wikipedia.org/wiki/Model_View_ViewModel" \o "Model View ViewModel)-Muster\*, es kann allerdings auch in Multipage Webseiten für einzelne Abschnitte verwendet werden. Ab Version 2.0 unterstützt es auch serverseitiges Rendern\*.

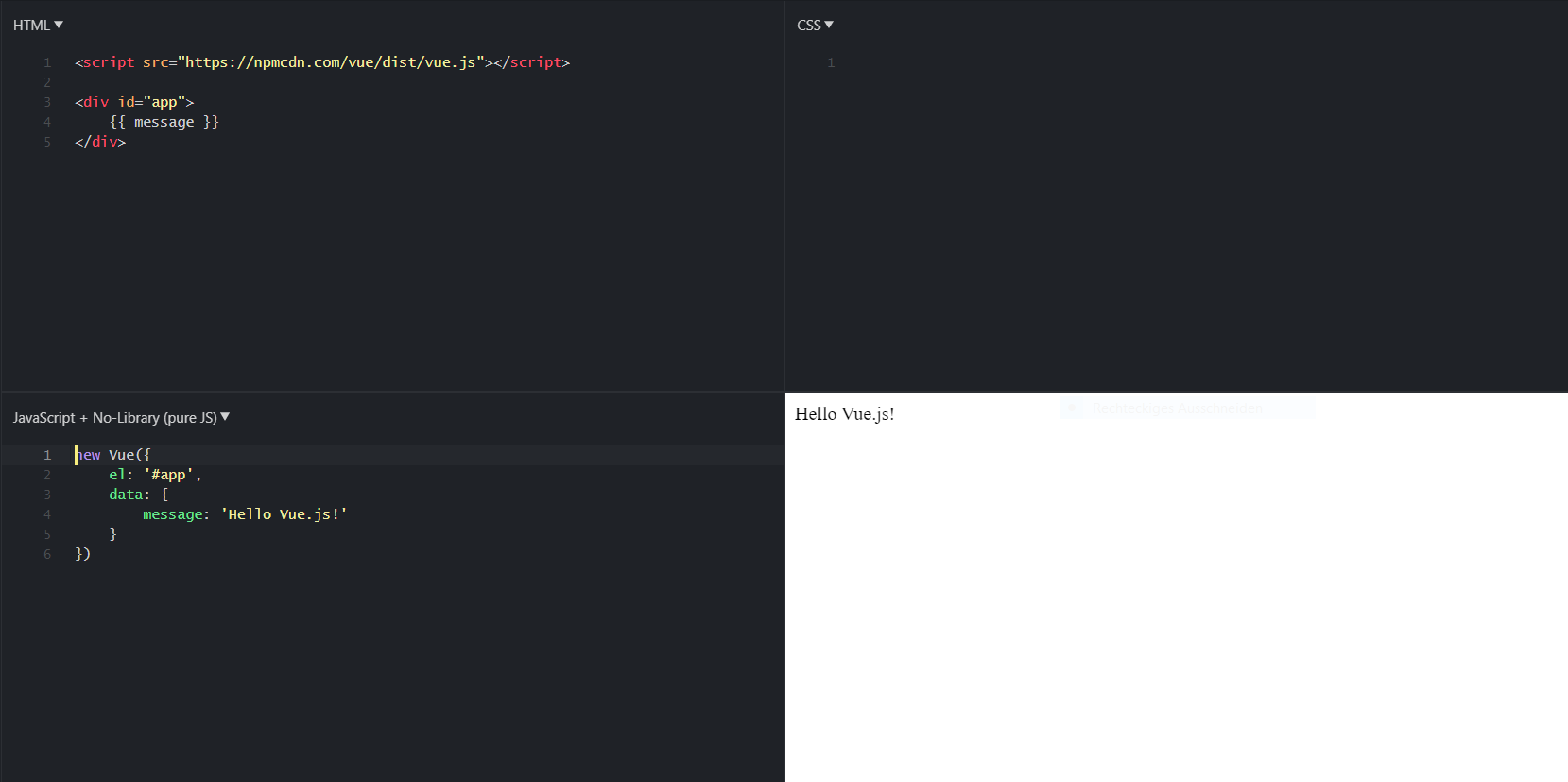


Abbildung : Hello World Beispiel VueJS

## Angular Versionen

**Angular 2**

Angular 2 ist eine komplett überarbeitete Version von AngularJS, da es komplett in Typescript geschrieben wurde. Zudem wurde es unter dem Mobile First Ansatz entwickelt. Es werden mehr Programmiersprachen unterstützt, wie z.B. ES5, ES6, Typescript oder Dart.

**Angular 3**

Angular 3 wurde übersprungen, da sich @angular/router bereits in einer 3.X Version befand.

**Angular 4**

Generierter Code für die Komponenten ist jetzt um bis zu 60% reduziert worden, wodurch die Anwendung schneller kompiliert. Animationen wurden aus der @angular/core entfernt und in ein externes BrowserAnimationsModule ausgelagert. Zudem wurden weitere Validierungspattern eingeführt.

**Angular 5**

Die Internationalisierung von Zahlen und Daten wurde deutlich erweitert. Ebenfalls gab es weitere Optimierungen während des kompilieren.

**Angular 6**

Das Component Develoment Kit sowie ng update und ng add wurden eingeführt. Zudem wurde die Performance der Animationen weiterentwickelt.

**Angular 7**

Angular Material hat Drag und Drop sowie Virtual Scrolling implementiert.

**Angular 8**

Einige zusätzliche Befehle für die CLI wurden eingeführt. Unter anderem ng build, ng test und ng deploy.

**Angular 9**

Die Dependency Injection und Internationalisierung wurden überarbeitet. Mit Angular Ivy wurde zusätzlich ein neuer Renderer eingeführt.

## Übersicht Frameworks und Entscheidung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Angular | | React | | VueJS | |
| PRO | CON | PRO | CON | PRO | CON |
| ausgereift | Core 500kB | flexibel | Fragmentierte Libs | minimalistisch | Libs/Features |
| Komplettpaket | Konfiguration | Core 100kB | JSX | Core 20kB | Testing |
| @angular/cli | Erfahrung | CLI | ES2015+ | HTML+ES5+CSS | Debugging |
| Architektur | TS Overhead | performant | Konfiguration | CLI | Architektur |
| Code Qualität | re-render cycles | ausgereift | DOM Konsistenz | Doku | unausgereift |
| @angular/forms |  | Web+Native |  | performant |  |
| Doku |  | erweiterbar |  | sicherer State |  |
| Testing |  | viele Daten |  | Web+Native |  |
| Zukunft |  | Zukunft |  | skalierbar |  |

Da die Anwendung später weiterentwickelt werden soll und mehrere Entwickler an der Anwendung weiterarbeiten, ist die Entscheidung auf das firmenweite Framework Angular gefallen. Zudem sind der Umfang, die benötigten Schnittstellen und Module des endgültigen Projekts noch nicht klar.

Mit dem Projekt sollten Funktionen für zukünftige Projekte ausgearbeitet werden. Durch die Kunden wurde das Framework Angular vorgegeben.

Zudem hat der Auszubildende in seiner Lehrzeit viel Erfahrungen mit diesem Framework gesammelt, wodurch sich ein gewisser Erfahrungsschatz angesammelt hat.

# Kostenschätzung

Nach der Konzeption ist es die Aufgabe eine Schätzung abzugeben, welche Kosten und in welcher Höhe auf die Firma zukommen. Dabei werden die Projektkosten berechnet und mit einem Verwaltungs- und Gewinnzuschlagssatz verrechnet. In diesem Fall sind diese von der Verwaltung im Vorfeld berechnet worden und dem Projektverantwortlichen nur der daraus resultierende Preis angegeben. Dieser wird in Euro pro am Projekt geleisteter Arbeitsstunde gemessen. Der Preis beläuft sich somit bei einem Entwickler auf 90 € pro Stunde und bei einem Auszubildenden auf 15 €.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Auszubildender (15€) | Entwickler (90€) | Preis |
| Analyse | 4 | 1 | 150 |
| Konzeption | 10 |  | 150 |
| Realisierung | 33 |  | 495 |
| Sonstiges | 23 |  | 345 |
| Gesamt |  |  | 1140 |

# Realisierung

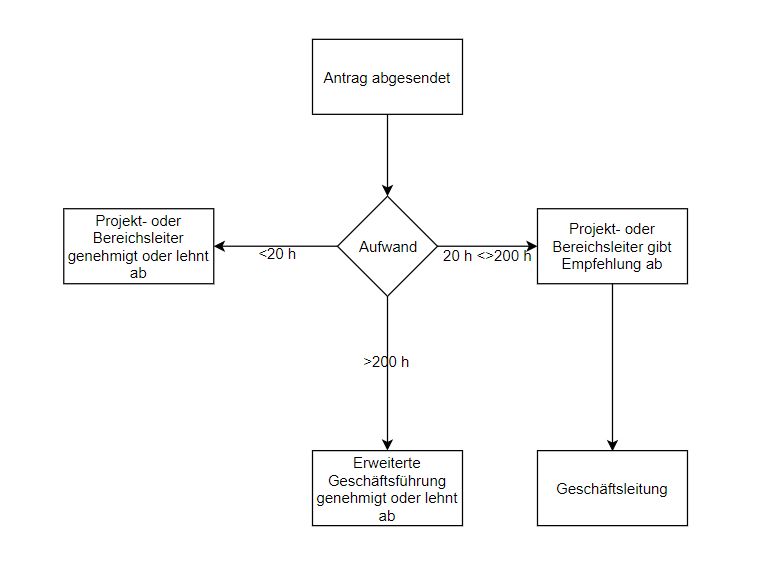


Abbildung : Ablaufdiagramm Genehmigungsprozess

Da das dynamische generieren der Material Komponenten bei diesem Projekt sehr wichtig war, wurde während der Realisierung stark darauf geachtet Komponenten wiederverwendbar zu schreiben.

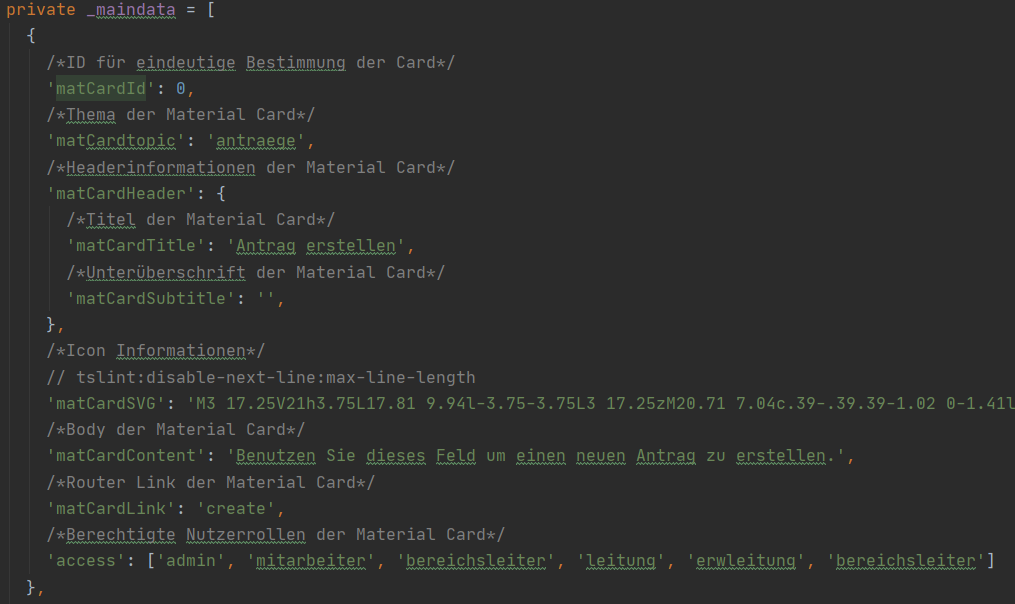


Abbildung : JSON Beispiel für Home

Abbildung : HTML Beispiel für Home

Abbildung : HTML Beispiel für Home

Die Daten für die Komponenten kommen jeweils aus einer Typescript\* Datei als Export Klasse, damit Komponenten Attribute und Eigenschaften an einem zentralen Ort liegen.

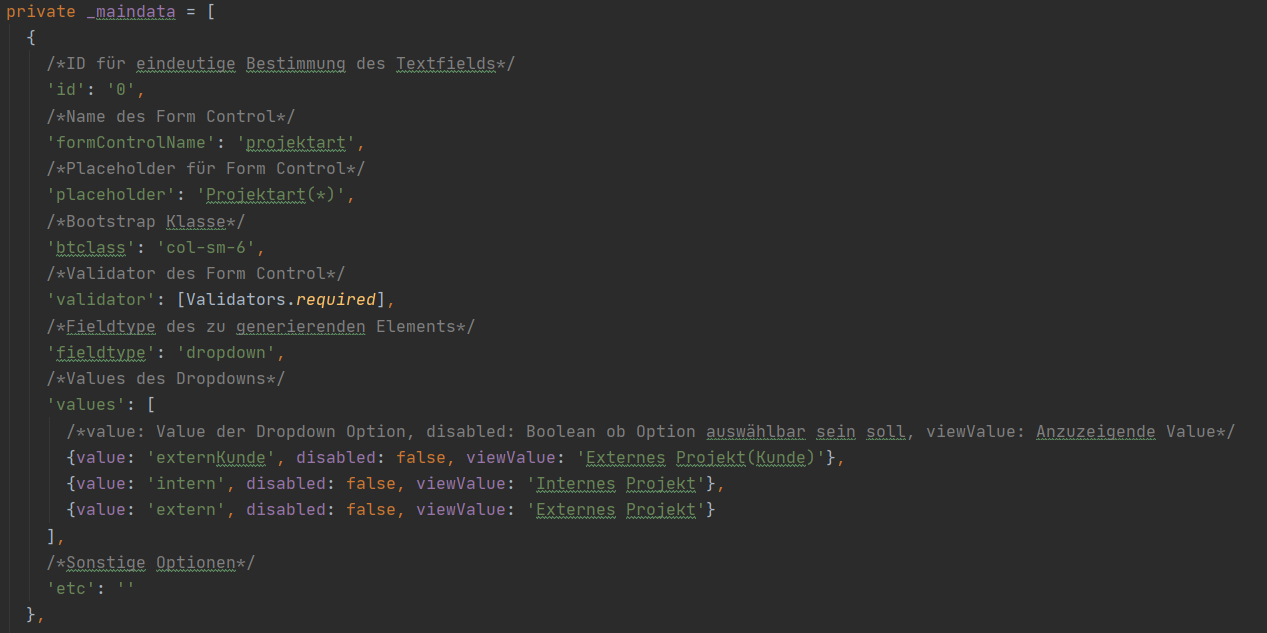


Abbildung : JSON Beispiel für Create

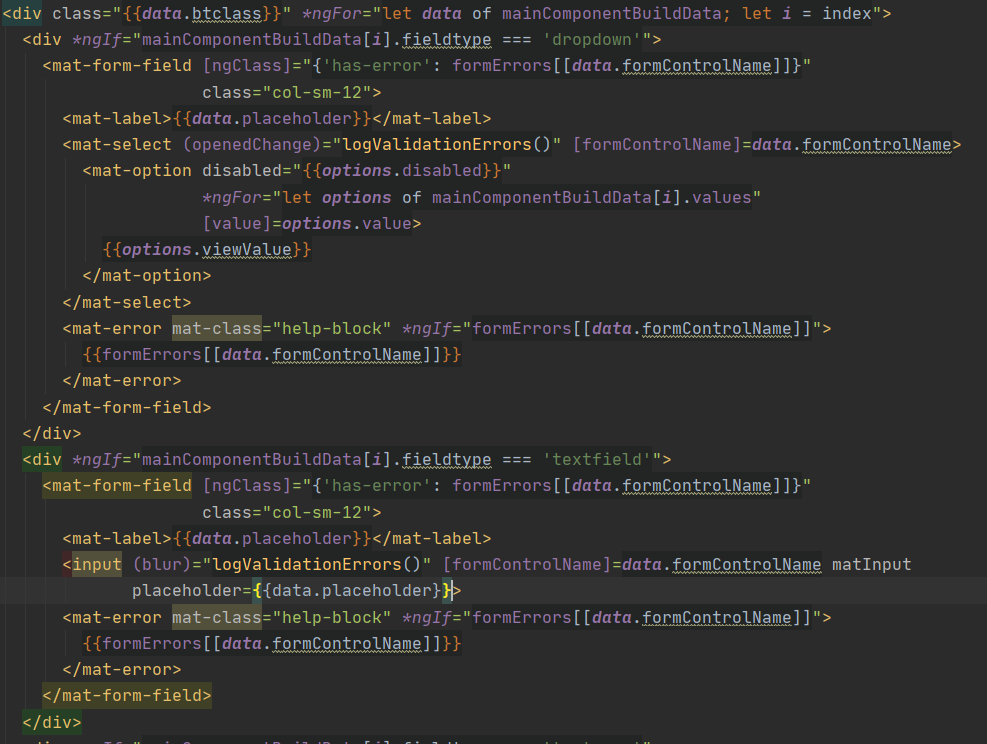


Abbildung : HTML Beispiel für Create

Dies ist ein Beispiel für die Generierung der HTML Komponenten. Es zeigt die Generierung von Dropdowns und Textfeldern. Besonderes Augenmerk gilt den Formerrors\* im mat-error Tag. Hier musste die doppelte Interpolation\* genutzt werden, da sowohl formErrrors, als auch der formControlName vollständig dynamisch aus einer Klasse generiert werden.

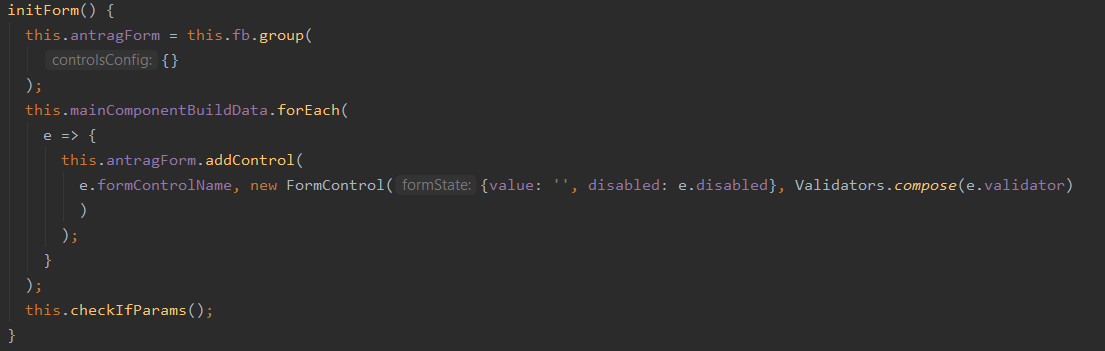


Abbildung : Initialisierung des Formulars

In dieser Funktion werden die Form Controls\* im Antragsformular hinzugefügt. Für jeden Datensatz, der aus der Export Klasse kommt, wird ein Form Control\* angelegt und die Value, dem Attribut disabled und den Validatoren\* angelegt.

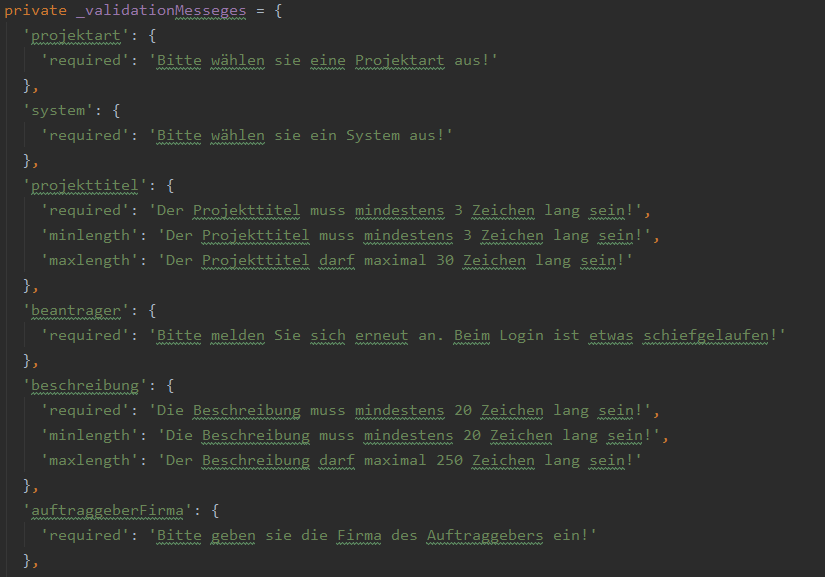


Abbildung : Validation Messages der FormControls

Dieses Objekt beinhaltet die Form Controls\* und ihre Validatoren\*, welche ebenfalls beim initialisieren der Komponenten übergeben werden. Zu jeder Art Validator\* gibt es dann eine Nachricht, die bei Form Errors\* des Controls\* angezeigt werden. Die Überprüfung, welche Meldungen angezeigt werden, findet in der Funktion logValidationErrors statt.



Abbildung : Zwischenspeicher für Fehlermeldungen zur Laufzeit

Dieses Objekt dient dem Zwischenspeichern der Form Error\* Validation\* Messages. Für jedes Form Control\* werden hier aus dem Objekt die entsprechenden Values der zutreffenden Keys aus dem Objekt Validation\* Messages zwischengespeichert und wenn nötig die Strings mit einem Leerzeichen separiert zwischengespeichert und in der HTML für das entsprechende Form Control\* angezeigt.



Abbildung : Loggen von Value Changes

An diese Funktion wird eine Form Group\* übergeben, als default Wert wird hier die komplette Form Group\* übergeben. Diese Funktion wird bei jeder Änderung eines Controls\* im Formular aufgerufen.

In dieser Funktion wird über die Controls\* iteriert und anschließend geprüft, ob diese weitere Groups\* enthalten. Wenn das der Fall ist, wird die Form Group wieder an die Funktion übergeben. Wenn das nicht der Fall ist, wird die Value im Form Errors\* Object an dieser Stelle auf einen Leerstring Gesetzt. Anschließend wird geprüft, ob das Control\* invalid\* und touched\* oder dirty\* ist. Wenn das Form Control\* daraufhin Errors\* hat, wird das Array auf den Key des Errors durchsucht. Wenn es vorhanden ist, wird die Value des Keys in das Form Errors\* Objekt geladen. Wenn es mehrere gibt, werden sie mit einem Leerzeichen getrennt.



Abbildung : Usermanagement

Beim Initialisieren wird ein Form Array\* erstellt. In dieses Form Array\* wird für jeden Nutzer eine Form Group\* mit Form Controls\* gepusht.

Selbiges geschieht beim Hinzufügen von neuen Usern.

# Fazit

Durch die Abschlussarbeit konnten wichtige Funktionen zum dynamischen Generieren von Formularen und Komponenten erprobt und weiterentwickelt werden. Die dynamische Zuweisung und Gruppierung von CSS-Attributen mittels Attribute Binding muss dennoch etwas weiter ausgebaut werden um in vollem Umfang genutzt werden zu können.

Das Projekt konnte im geforderten Umfang fertiggestellt werden. Um die dynamische Generierung der Inhalte nach und nach ausbauen zu können, wurde mit der Home Komponente angefangen. Hier gestaltete sich der Aufbau relativ leicht, da die Material Cards sehr ähnlich sind und sich der Inhalt nicht sonderlich unterscheidet.

Eine besondere Herausforderung war die Ansicht zum Anlegen von Anträgen, da die Material Komponenten andere Attribute haben. Hier fiel die Entscheidung auf eine Abfrage, welche ein HTML Element mit den jeweiligen Attributen bereitstellt. Zudem mussten die Form Controls\* nach der Initialisierung angelegt werden.

# Quellen

## Webquellen

<https://angular.io/>

<https://angular.io/guide/reactive-forms>

<https://de.wikipedia.org/wiki/React>

<https://de.wikipedia.org/wiki/Vue.js>

<https://medium.com/@scarlett8285/why-is-reactjs-the-best-fit-for-your-social-networking-app-15646d6ad082>

<https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190>

<https://de.wikipedia.org/wiki/Angular#Unterschiede_zwischen_Angular_und_AngularJS>

<https://www.ngdevelop.tech/angular/history/>

<https://app.diagrams.net/>

<https://react-etc.net/entry/angular-4-coming-in-2017-backwards-compatible-angular-2>

<https://www.infoworld.com/article/3225511/angular-5-javascript-framework-delayed.html>

<https://blog.angular.io/version-6-of-angular-now-available-cc56b0efa7a4>

<https://blog.angular.io/version-7-of-angular-cli-prompts-virtual-scroll-drag-and-drop-and-more-c594e22e7b8c>

<https://blog.angular.io/a-plan-for-version-8-0-and-ivy-b3318dfc19f7>

## Buchquellen

*Angular: Grundlagen, fortgeschrittene Themen und Best Practices* von Ferdinand Malcher, Johannes Hoppe, Danny Koppenhagen

*EssentialsAngular: The Essential Guide to Learn Angular* von Dhananjay Kumar

# Selbsterklärung

Ich versichere durch meine Unterschrift, dass ich das zugrunde liegende Projekt und diesen Bericht zur betrieblichen Projektarbeit (Projektbericht) selbstständig und ohne fremde Hilfe angefertigt habe. Alle aufgeführten Preise sind netto und entsprechen nicht der Realität. Sie sind aus wettbewerbstechnischen Gründen fiktiv angegeben. Die Arbeit hat in dieser Form keiner anderen Prüfungsinstitution vorgelegen.

Ingolstadt, 20.04.2016   
Ort, Datum

**Erklärung des Ausbildungsbetriebes**

Das Projekt wurde, wie in der Dokumentation dargestellt, in unserem Unternehmen durchgeführt.

Ingolstadt, 20.04.2016   
Ort, Datum

# Screenshots des Prototypens

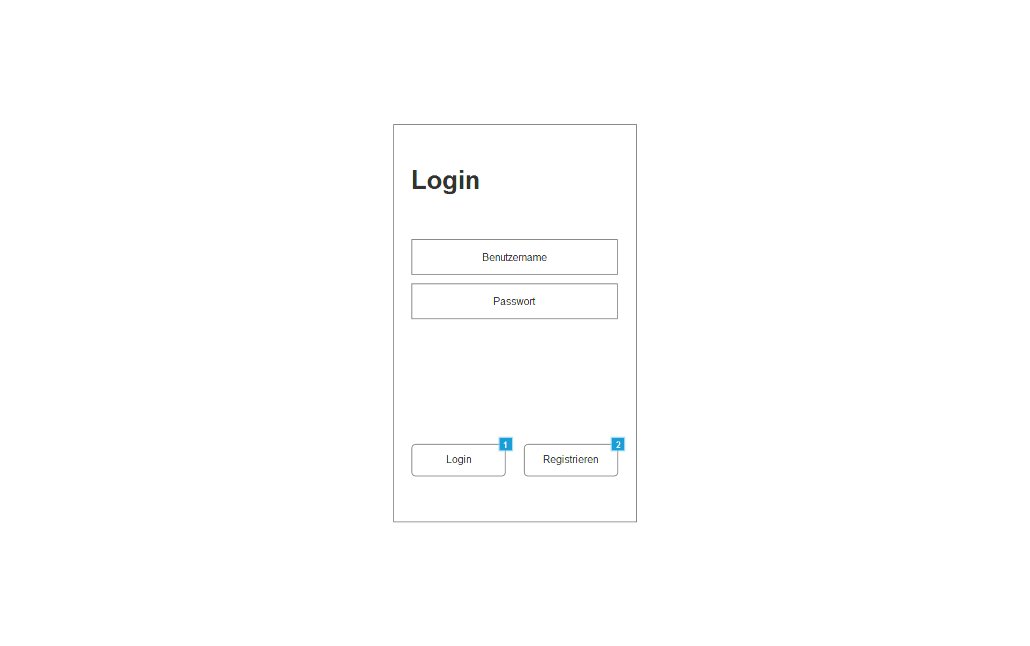


Abbildung : Entwurf Login

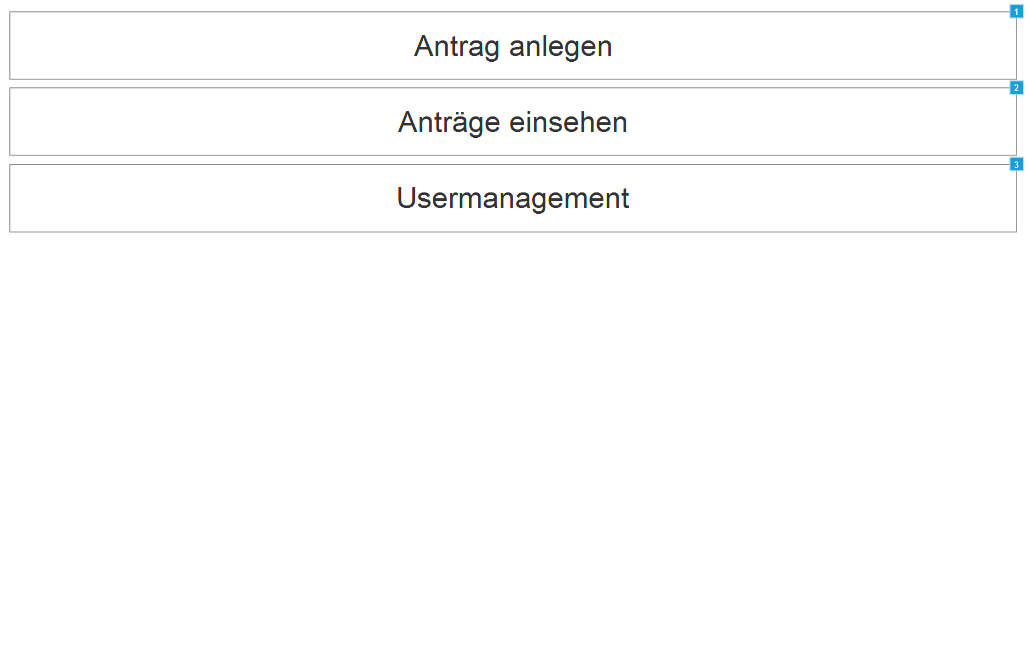


Abbildung : Entwurf Home

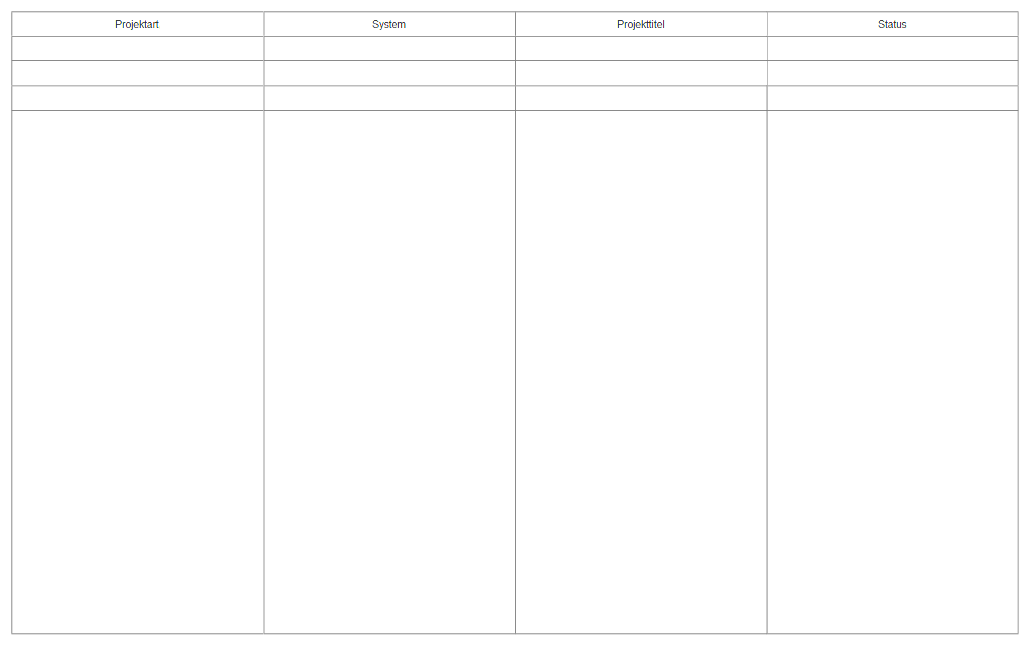


Abbildung : Entwurf List



Abbildung : Entwurf Usermanagement

# Screenshots der Anwendung

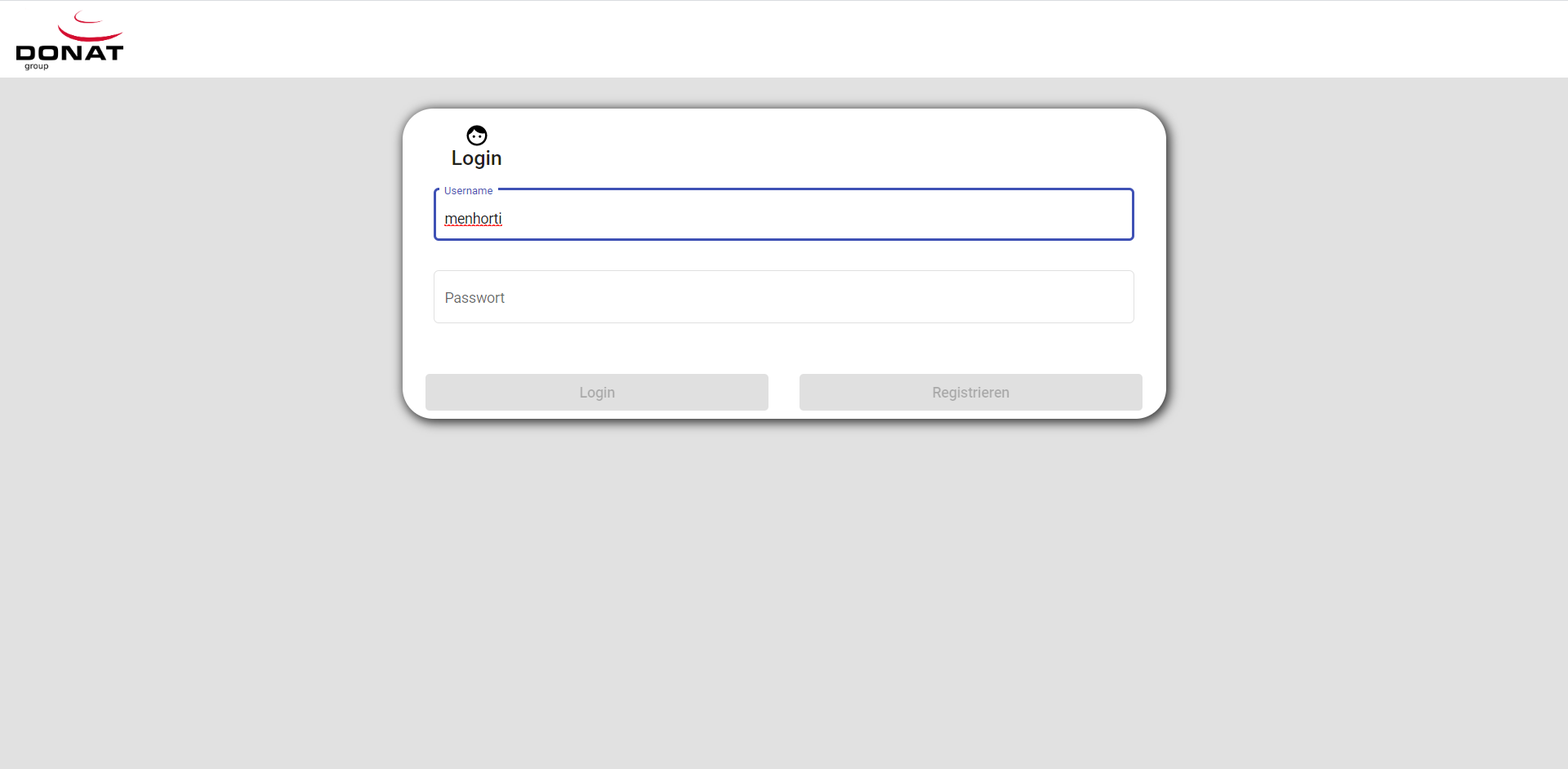


Abbildung : Anwendung Login

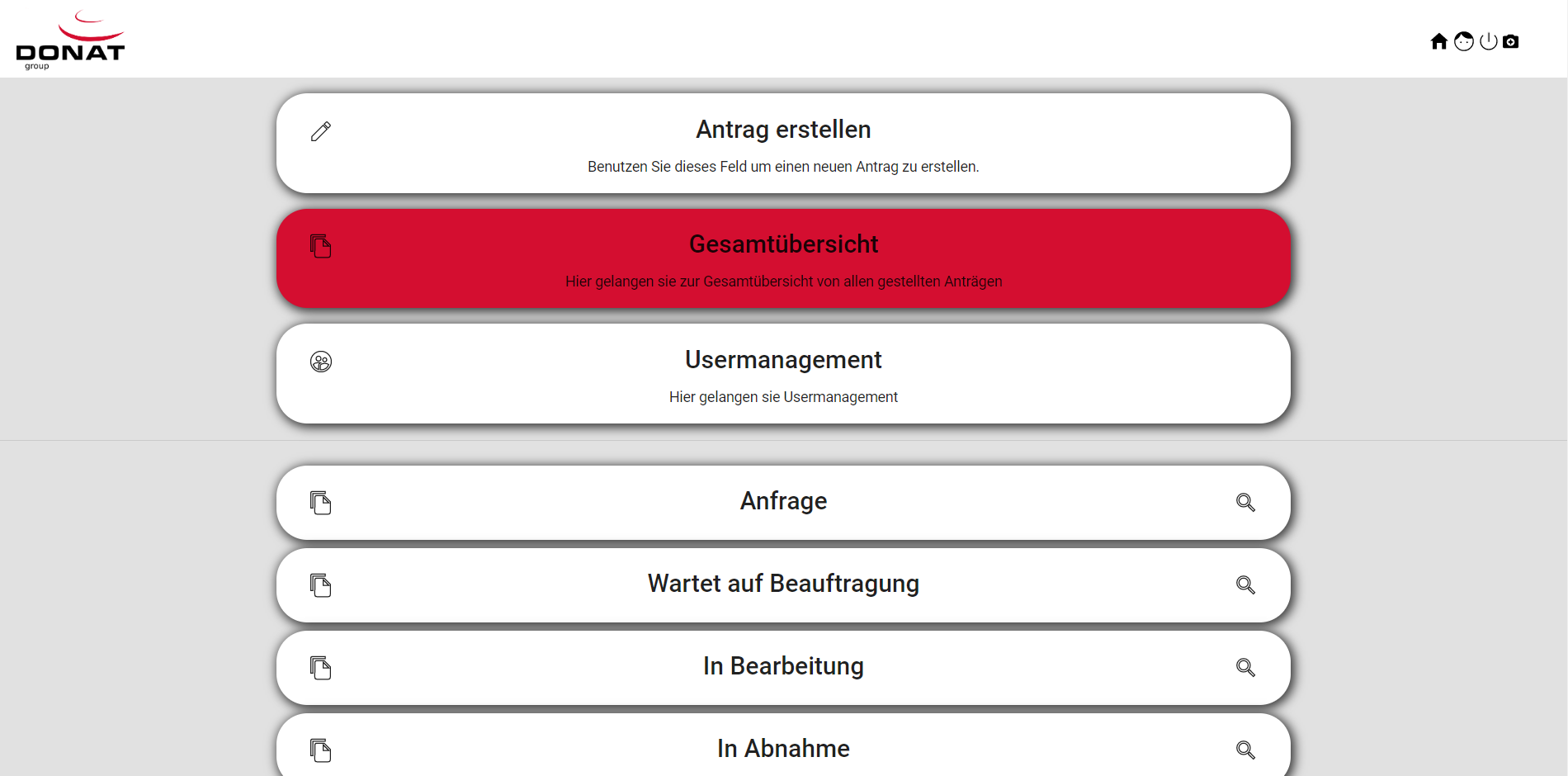


Abbildung : Anwendung Home

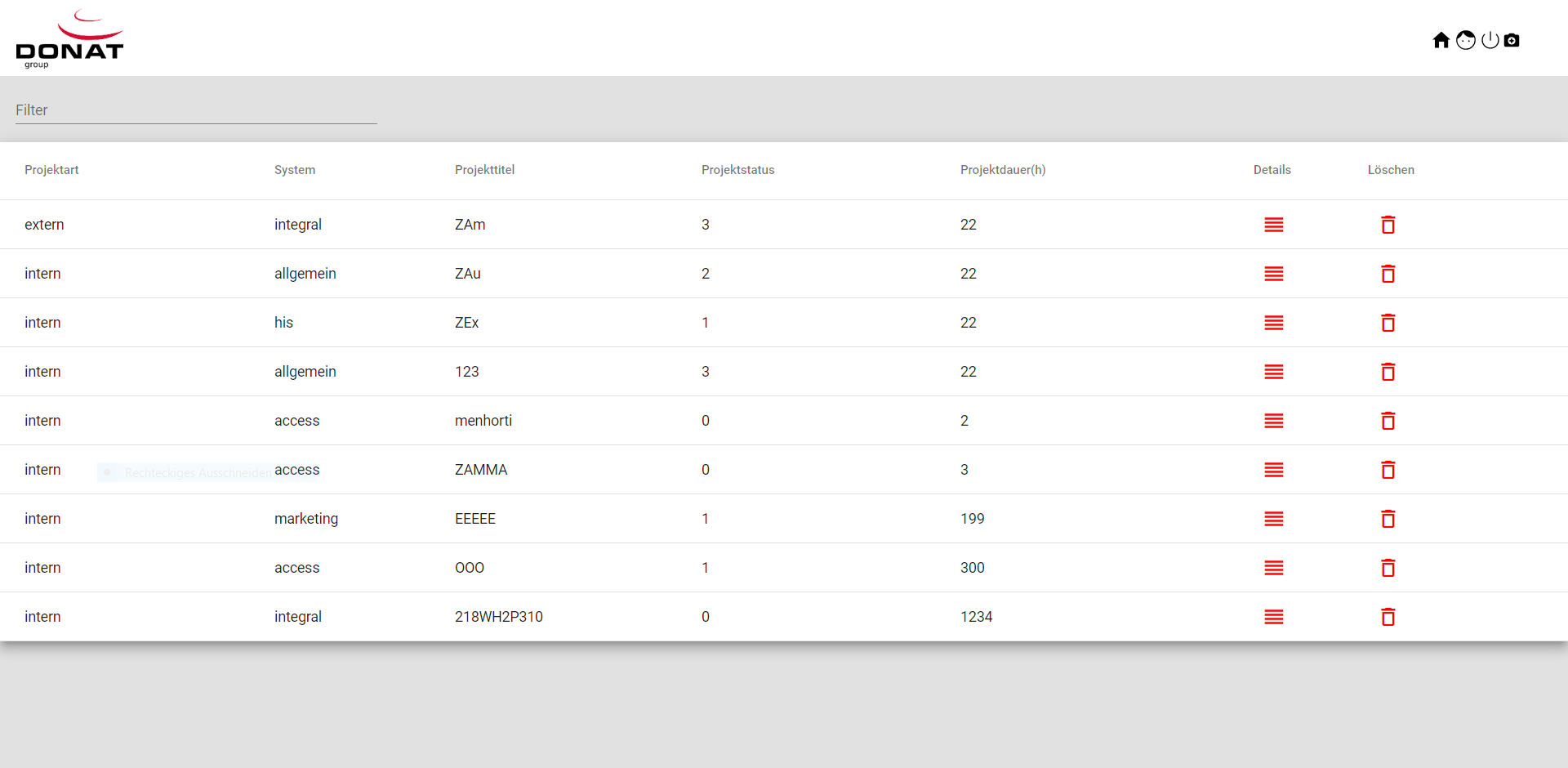


Abbildung : Anwendung List

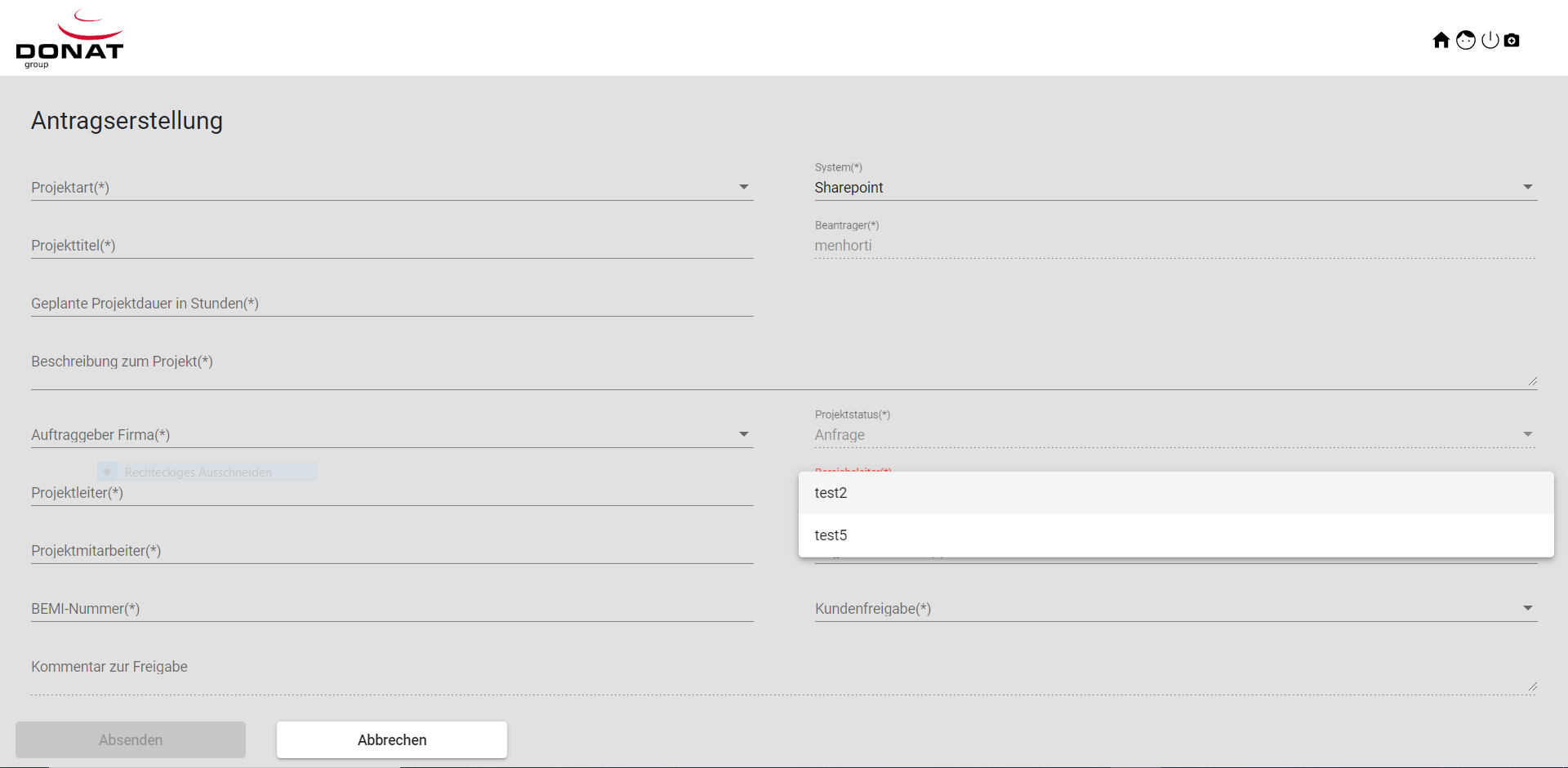


Abbildung : Anwendung Create

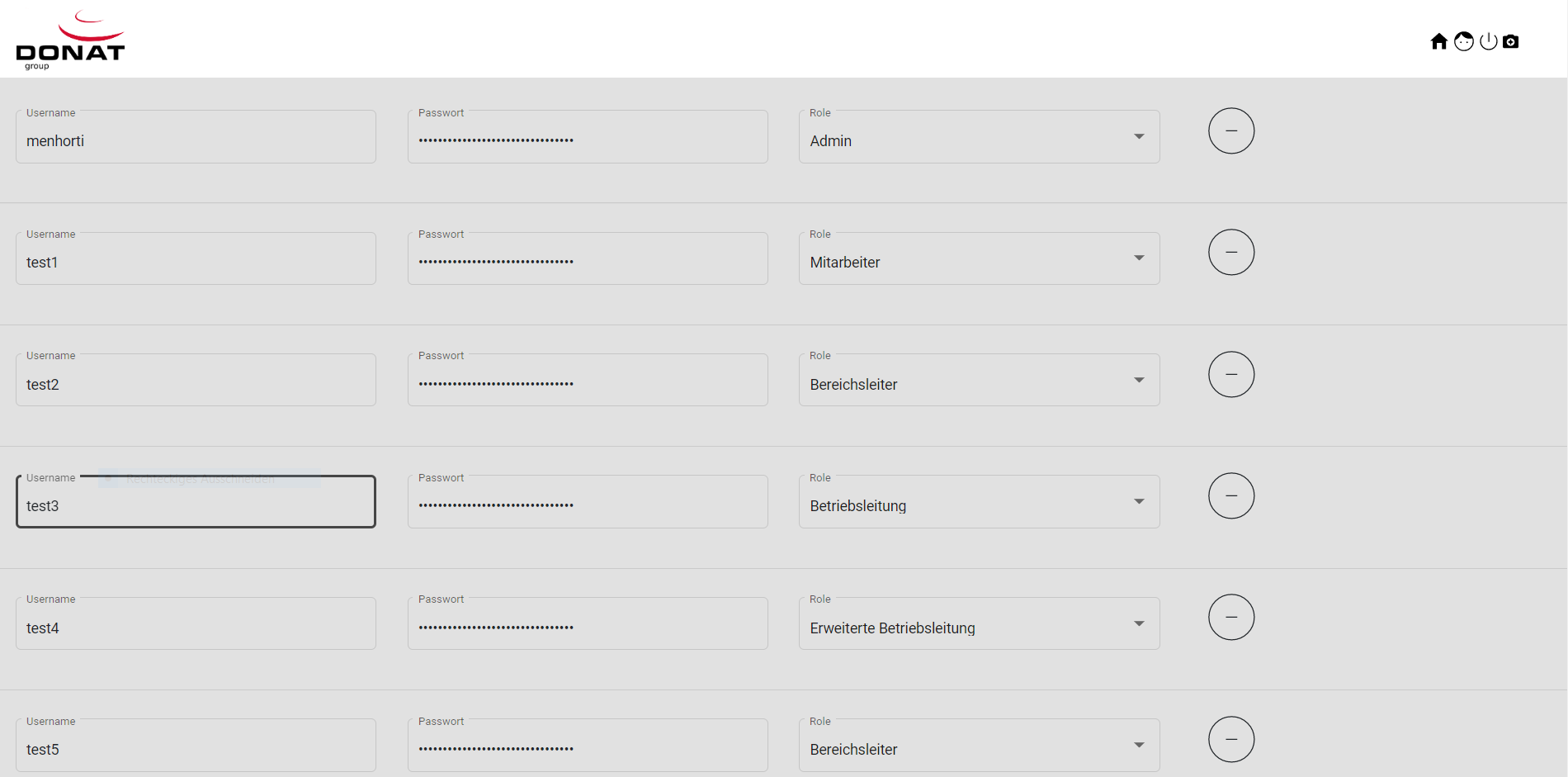


Abbildung : Anwendung Usermanagement