

Bericht zur betrieblichen Projektarbeit im Zeitraum:

11.05.2020 – 27.05.2020

von

Timo Menhorn

zur Erlangung des Abschlusses als

Fachinformatiker Anwendungsentwicklung



Ausbildungsberuf: Fachinformatiker Anwendungsentwicklung

Ausbildungsbetrieb: Donat IT GmbH

Verfasser: Timo Menhorn

Prüfung: Sommer 2020

Prüflingsnummer: 20265

Projektbetreuer: Jan Brinkmann

1 Verzeichnisse

1.1 Inhaltsverzeichnis

1	Verzeichnisse	2
1.1	Inhaltsverzeichnis	2
1.2	Abbildungsverzeichnis	3
2	Einleitung.....	5
2.1	Projektumfeld	5
2.2	Projektziele	5
2.3	Projektbegründung.....	6
2.4	Projektschnittstellen	6
2.5	Projektabgrenzung.....	7
2.6	Zielplattform	7
3	Projektplanung.....	8
3.1	Zeitplanung	8
3.2	Ressourcenplanung	8
3.3	Personalplanung	8
3.4	Sachmittelplanung	8
3.5	Maßnahmen zur Qualitätssicherung.....	8
4	Analysephase	13
4.1	Ist- Analyse	13
4.2	Soll- Analyse	15
4.2.1	Login.....	15
4.2.2	Dashboard	15
4.2.3	Antrag anlegen.....	15
4.2.4	Übersichtsseite	15
4.2.5	Usermanagement	15
4.3	Wirtschaftlichkeitsanalyse	16
4.3.1	Make-or-Buy Entscheidung	16
4.3.2	Projektkosten	16
4.3.3	Nicht-monetäre Vorteile	16
5	Realisierung.....	17
6	Projektphasen.....	26

7	Angular	27
8	Alternativen zum Framework Angular	29
8.1	React	30
8.2	VueJS	31
8.3	Angular Versionen	32
8.4	Übersicht Frameworks und Entscheidung	33
9	Kostenschätzung	34
10	Fazit.....	35
11	Quellen	36
11.1	Webquellen.....	36
11.2	Buchquellen	36
12	Selbsterklärung.....	37
13	Screenshots des Prototypens	38
14	Screenshots der Anwendung	40
15	Glossar	43
15.1	Begriffsklärung.....	43
15.2	Reactive Forms Begriffe.....	48

1.2 Abbildungsverzeichnis

Abbildung 1: Testbeispiel Login: Korrektes Laden der Komponente	9
Abbildung 2: Testbeispiel Login: Login eines Users.....	9
Abbildung 3: Testbeispiel Login: Initialisieren der Form Controls	10
Abbildung 4: Testbeispiel Create: Initialisieren der Form Controls	10
Abbildung 5: Testbeispiel Create: Validierung des Eingabeformulars	11
Abbildung 6: Test Beispiel Usermanagement: Initialisierung des Form Arrays	11
Abbildung 7: Testbeispiel Gesamtübersicht: Laden des aktuellen Users	12
Abbildung 8: Excel Projektsteckbrief.....	14
Abbildung 9: Ablaufdiagramm Genehmigungsprozess	17
Abbildung 10: Klassendiagramm der Formfields.....	18
Abbildung 11: JSON Beispiel für Home	18
Abbildung 12: HTML Beispiel für Home	19

Abbildung 13: JSON Beispiel für Create.....	20
Abbildung 14: HTML Beispiel für Create.....	21
Abbildung 15: Initialisierung des Formulars	22
Abbildung 16: Validation Messages der Form Controls	22
Abbildung 17: Zwischenspeicher für Fehlermeldungen zur Laufzeit	23
Abbildung 18: Loggen von geänderten Form Controls*	24
Abbildung 19: Usermanagement	25
Abbildung 20: Funktionsweise von Angular	27
Abbildung 21: Hello World Beispiel Angular.....	28
Abbildung 22: Star History der Frameworks im Vergleich	29
Abbildung 23: Jobangebote für Entwickler auf gängigen Plattformen	30
Abbildung 24: Hello World Beispiel React.....	31
Abbildung 25: Hello World Beispiel VueJS	31
Abbildung 26: Entwurf Login.....	38
Abbildung 27: Entwurf Home	38
Abbildung 28: Entwurf List.....	39
Abbildung 29: Entwurf Usermanagement	39
Abbildung 30: Anwendung Login	40
Abbildung 31: Anwendung Home	40
Abbildung 32: Anwendung Create	41
Abbildung 33: Anwendung Usermanagement.....	42

2 Einleitung

2.1 Projektumfeld

Vor 35 Jahren begann die Geschichte der DONAT IT. Durch permanente Weiterentwicklung hat sich die Unternehmensgruppe seither als eines 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.

Das Projekt soll einen etablierten Prozess automatisiert unterstützen und später ersetzen. Es wird in den Örtlichkeiten der Donat IT GmbH umgesetzt. Verantwortlich für den gesamten Projektverlauf ist der Auszubildende.

2.2 Projektziele

Es soll eine übersichtliche Möglichkeit geben, die Projektanträge einzureichen und diese von den Verantwortlichen managen zu können. Zusätzlich soll es für Admins ein Usermanagement geben, in dem User verwaltet und Rechte vergeben werden können.

Es sollen für dieses Projekt ähnliche marktübliche Frameworks verglichen werden und das für diese Anforderungen passendste gewählt werden. Dieser Vergleich ist unter dem Punkt 7 „Alternativen zum Angular Framework“ zu finden.

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

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 ein 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 möchte.

Wenn der User einen Antrag anlegen möchte, bekommt er eine Liste mit möglichen Projektthemen und landet anschließend in der Komponente zum Anlegen. Wenn er auf die

Übersichtskomponente möchte, 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 können um eine einfache Handhabung zu garantieren.

2.3 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.

2.4 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.

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

2.5 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 welcher der Admin User ändern, hinzufügen oder löschen kann.

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

2.6 Zielplattform

Das Projektantragsverwaltungstool wird als Webapplikation* erstellt. Der Vorteil hiervon ist, dass auf Seiten des Clients* lediglich ein Webbrowser* benötigt wird. 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 zum Beispiel eine neue Version hochladen, die anschließend sofort vom User genutzt werden kann.

3 Projektplanung

3.1 Zeitplanung

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

Phase	Stunden
Analyse	4
Konzeption	10
Realisierung	33
Sonstiges	23
Gesamt	70

3.2 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.

3.3 Personalplanung

Der Ausbilder stand bei organisatorischen Fragen zur Verfügung. Die Stundensätze (Auszubildender 15€, Entwickler 90€, Abteilungsleiter 105€) sind Pauschalsätze, die sich unter anderem aus dem Bruttostundenlohn, den Sozialaufwendungen des Arbeitgebers und der Ressourcennutzung zusammensetzen. Eine Kostenverteilung findet sich nachfolgend unter dem Punkt 8 „Kostenschätzung“.

Der Abteilungsleiter fungiert hier als Anforderer und Projektleiter. Manuelle Tests wurden durch den Auszubildenden sowie den Entwickler durchgeführt. Die Projektabnahme erfolgt durch den Abteilungsleiter.

3.4 Sachmittelplanung

Um die anfallenden Kosten möglichst gering zu halten, wurde bei der Auswahl der Software auf Open-Source-Software* oder bereits vorhandene Lizenzen geachtet.

3.5 Maßnahmen zur Qualitätssicherung

Um eine möglichst hohe Qualität der Webapplikation* sicherzustellen, werden wichtige Funktionen bereits während der Implementierungsphase getestet.

Dies geschieht durch Unittests. Mithilfe der Tools Karma und Jasmine wurden die Tests automatisch ausgeführt. Dieses Vorgehen stellt eine hohe Qualität der Software sicher, da Fehler frühzeitig erkannt werden. Hierdurch wird regelmäßig geprüft, ob Änderungen an einer Stelle Probleme im restlichen Code bewirken.


```
beforeEach( action: () => {
  fixture = TestBed.createComponent(LoginComponent);
  component = fixture.componentInstance;
  fixture.detectChanges();
});

it( expectation: 'should create', assertion: () => {
  expect(component).toBeTruthy();
});
```

Abbildung 1: Testbeispiel Login: Korrektes Laden der Komponente

Beispiel Nummer	1	
Name	Komponenteninitialisation	
Rollen	Admin, Mitarbeiter, Bereichsleiter, Betriebsleitung, Erweiterte Betriebsleitung	
Vorgehen	Testschritte	Erwartetes Ergebnis
	Komponenteninitialisation	Komponenten werden standardmäßig initialisiert

```
it( expectation: 'should be valid if Userinput is registered', assertion: () => {
  component.loginForm.controls.username.patchValue( value: 'test1');
  component.loginForm.controls.password.patchValue( value: 'test1');
  component.checkUser( e: true);
  expect(localStorage.getItem( key: 'user')).toBeDefined();
});
```

Abbildung 2: Testbeispiel Login: Login eines Users

Beispiel Nummer	2	
Name	Servicetests	
Rollen	Admin, Mitarbeiter, Bereichsleiter, Betriebsleitung, Erweiterte Betriebsleitung	
Vorgehen	Testschritte	Erwartetes Ergebnis
	Login als Admin	Dashboardkomponente erscheint mit Auswahl „Usermanagement“
	Login als Standarduser	Dashboardkomponente erscheint ohne Auswahl „Usermanagement“

```
it( expectation: 'should create form controls', assertion: () => {
    component.initLoginForm();
    expect(component.loginForm.controls).toBeDefined();
});
```

Abbildung 3: Testbeispiel Login: Initialisieren der Form Controls

Beispiel Nummer	3	
Name	Initialisation der Form Controls	
Rollen	Mitarbeiter	
Vorgehen	Testschritte	Erwartetes Ergebnis
	Funktion zum Anlegen der Form Controls	Form Controls werden in der erwarteten Reihenfolge vollständig angelegt

```
it( expectation: 'should create form controls', assertion: () => {
    component.initForm();
    expect(component.antragForm.controls).toBeDefined();
});
```

Abbildung 4: Testbeispiel Create: Initialisieren der Form Controls

Beispiel Nummer	4	
Name	Initialisation der Form Controls	
Rollen	Mitarbeiter	
Vorgehen	Testschritte	Erwartetes Ergebnis
	Funktion zum Anlegen der Form Controls	Form Controls werden in der erwarteten Reihenfolge vollständig angelegt

```
it( expectation: 'Should be valid if Form Fileds are filled', assertion: () => {
  component.ngOnInit();
  component.antragForm.controls['projektart'].patchValue( value: 'externKunde');
  component.antragForm.controls['system'].patchValue( value: 'access');
  component.antragForm.controls['projekttitle'].patchValue( value: 'ZAX');
  component.antragForm.controls['beantrager'].patchValue( value: 'Timo Menhorn(IEW/23)');
  component.antragForm.controls['projektdauer'].patchValue( value: 200);
  // tslint:disable-next-line:max-line-length
  component.antragForm.controls['beschreibung'].patchValue( value: 'Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut l');
  component.antragForm.controls['auftraggeberFirma'].patchValue( value: 'kunde1');
  component.antragForm.controls['projektstatus'].patchValue( value: 0);
  component.antragForm.controls['projektleiter'].patchValue( value: 'Timo Menhorn(IEW/23)');
  component.antragForm.controls['bereichsleiter'].patchValue( value: 'test2');
  component.antragForm.controls['projektmitarbeiter'].patchValue( value: 'Timo Menhorn(IEW/23)');
  component.antragForm.controls['angebotsnummer'].patchValue( value: 54321);
  component.antragForm.controls['bemi'].patchValue( value: 54321);
  component.antragForm.controls['kundenfreigabe'].patchValue( value: '0');
  component.antragForm.controls['kommentarfreigabe'].patchValue( value: '');

  fixture.detectChanges();
  expect(component.antragForm.valid).toBeTruthy();
});
```

Abbildung 5: Testbeispiel Create: Validierung des Eingabeformulars

Beispiel Nummer	5	
Name	Validierung des Eingabeformulars	
Rollen	Mitarbeiter	
Vorgehen	Testschritte	Erwartetes Ergebnis
	Eingabe der Mindestanforderung an Daten	Formular sollte valid sein

```
it( expectation: 'should be initialiced', assertion: () => {
  component.initForm();
  expect((<FormArray>this.userForm.get('user')).controls.length).toBeGreaterThan( expected: 0);
});
```

Abbildung 6: Test Beispiel Usermanagement: Initialisierung des Form Arrays

Beispiel Nummer	6	
Name	Initialisieren des Form Arrays	
Rollen	Admin	
Vorgehen	Testschritte	Erwartetes Ergebnis
	Initialisieren des Form Array	Arraylänge ist größer 0

```
beforeEach( action: () => {  
  fixture = TestBed.createComponent(ListComponent);  
  component = fixture.componentInstance;  
  // tslint:disable-next-line:max-line-length  
  localStorage.setItem('user', '\\username\\': '\\menhori\\', '\\password\\': '\\7c97253d54d440435b2f86497377af4d\\', '\\role\\': '\\admin\\', '\\id\\': 1');  
  component.currentUser = localStorage.getItem( key: 'user');  
  fixture.detectChanges();  
});
```

Abbildung 7: Testbeispiel Gesamtübersicht: Laden des aktuellen Users

Beispiel Nummer	7		
Name	Laden des aktuellen Nutzers aus dem Local Storage		
Rollen	/		
Vorgehen	Testschritte	Erwartetes Ergebnis	
	Setzen des aktuellen Nutzers	Der aktuelle Nutzer wird mit dem entsprechenden Key geladen	

4 Analysephase

4.1 Ist- Analyse

Wenn ein internes Projekt ohne Kundenauftrag gestartet wird, muss es umfangreich beschrieben und begründet werden. Bis zu einer geplanten Projektdauer von 20 Stunden können Projektfeld- und Bereichsleiter das Projekt selbst genehmigen. Ab einer Dauer von 20 Stunden genehmigt es die Geschäftsführung anhand eines Antrags.

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, da oft die falschen Ansprechpartner ausgewählt werden und die Anträge so bei den falschen Genehmigungsebenen landen.

Außerdem konnte ein Antragsteller bisher nur auf Nachfrage beim Vorgesetzten, den aktuellen Status seines Antrags erfragen und hatte nie eine Einsicht auf eine übersichtliche Darstellung.


	A	B	C
1	<div style="display: flex; justify-content: space-between; align-items: center;"> <div>Projektsteckbrief</div>  </div>		
2			
3			
4			
5	PROJEKTAUFTRAG		
6	Projekttitel:		
7	Projektnummer:		
8	Angebotsnummer:		
9	Auftragsnummer:		
10	Datum:		
11	Projekttyp:		
12	PROJEKTBETEILIGTE/ORGANISATION		
13	Auftraggeber:		
14	Projektleiter / Productowner:		
15	Scrummaster:		
16	Projektteam / weitere Rollen:		
17			
18	PROJEKTHALTUNG UND ZIELE		
19	Projektbeschreibung:		
20			
21	Projektziel:		
22			
23	PROJEKTKONTEXT		
24	Ausgangssituation		
25	Projektrisiken		
26	PLANTERMIN		
27	Gesamtdauer		
28	Projektstart		

Abbildung 8: Excel Projektsteckbrief

4.2 Soll- Analyse

4.2.1 Login

Es soll eine Webapplikation* entwickelt werden, bei der sich Nutzer authentifizieren* oder neu registrieren müssen, um anschließend auf ein Dashboard zu gelangen.

4.2.2 Dashboard

Der Standardnutzer sieht auf diesem Dashboard zwei Kacheln.

Wenn er die erste Kachel „Projektantrag erstellen“ 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 „Laufende Anträge“ 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 dieser zum Usermanagement* gelangt.

4.2.3 Antrag anlegen

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.

4.2.4 Übersichtsseite

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.

4.2.5 Usermanagement

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.

4.3 Wirtschaftlichkeitsanalyse

4.3.1 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.

4.3.2 Projektkosten

Bei dem Projekt müssen Kosten für das Entwicklungspersonal, Softwarelizenzen, sowie für die Gerätemieten aufgebracht werden.

4.3.3 Nicht-monetäre Vorteile

Das Projekt dient sekundär 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. Der Code ist somit wiederverwendbar für andere Projekte.

5 Realisierung

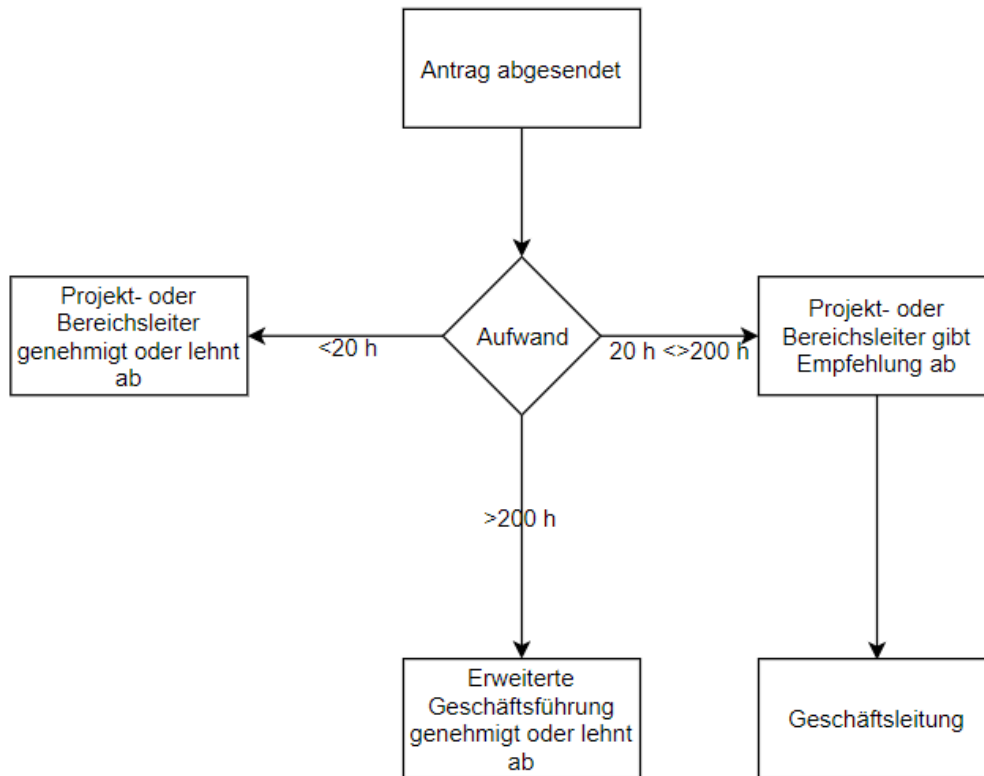


Abbildung 9: Ablaufdiagramm Genehmigungsprozess

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

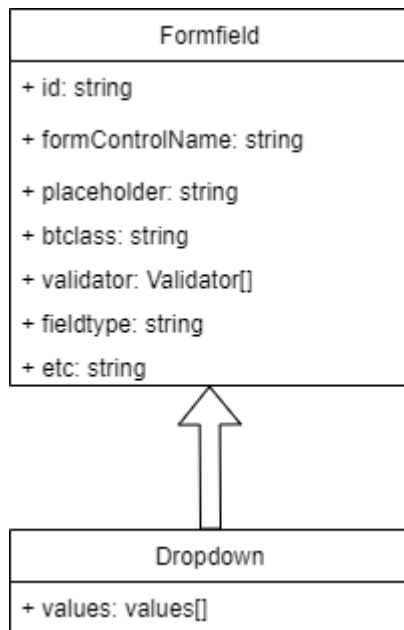


Abbildung 10: Klassendiagramm der Formfields

```
private _maindata = [
{
  /*ID für eindeutige Bestimmung der Card*/
  'matCardId': 0,
  /*Thema der Material Card*/
  'matCardtopic': 'antraege',
  /*Headerinformationen der Material Card*/
  'matCardHeader': {
    /*Titel der Material Card*/
    'matCardTitle': 'Antrag erstellen',
    /*Unterüberschrift der Material Card*/
    'matCardSubtitle': '',
  },
  /*Icon Informationen*/
  // tslint:disable-next-line:max-line-length
  'matCardSVG': 'M3 17.25V21h3.75L17.81 9.94l-3.75-3.75L3 17.25zM20.71 7.04c.39-.39.39-1.02 0-1.41',
  /*Body der Material Card*/
  'matCardContent': 'Benutzen Sie dieses Feld um einen neuen Antrag zu erstellen.',
  /*Router Link der Material Card*/
  'matCardLink': 'create',
  /*Berechtigte Nutzerrollen der Material Card*/
  'access': ['admin', 'mitarbeiter', 'bereichsleiter', 'leitung', 'erwleitung', 'bereichsleiter']
},
]
```

Abbildung 11: JSON Beispiel für Home

```
<mat-card>
  <div *ngIf="item.matCardHeader">
    <mat-card-title>
      <div class="row">
        <div class="col-sm-3">
          <svg xmlns="http://www.w3.org/2000/svg" height="24" viewBox="0 0 24 24" width="24">
            <path [attr.d]="item.matCardSVG" stroke="currentColor" stroke-width="1" fill="none"
              fill-rule="evenodd"/>
          </svg>
        </div>
        <div class="col-sm-6" style="text-align: center">
          {{item.matCardHeader.matCardTitle}}
        </div>
      </div>
    </mat-card-title>
    <mat-card-subtitle>{{item.matCardHeader.matCardSubtitle}}</mat-card-subtitle>
  </div>
  <mat-card-content>
    <div style="text-align: center">
      {{item.matCardContent}}
    </div>
  </mat-card-content>
</mat-card>
```

Abbildung 12: 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.

```
private _maindata = [
  {
    /*ID für eindeutige Bestimmung des Textfields*/
    'id': '0',
    /*Name des Form Control*/
    'formControlName': 'projektart',
    /*Placeholder für Form Control*/
    'placeholder': 'Projektart(*)',
    /*Bootstrap Klasse*/
    'btclass': 'col-sm-6',
    /*Validator des Form Control*/
    'validator': [Validators.required],
    /*Fieldtype des zu generierenden Elements*/
    'fieldtype': 'dropdown',
    /*Values des Dropdowns*/
    'values': [
      /*value: Value der Dropdown Option, disabled: Boolean ob Option auswählbar sein soll, viewValue: Anzuzeigende Value*/
      {value: 'externKunde', disabled: false, viewValue: 'Externes Projekt(Kunde)'},
      {value: 'intern', disabled: false, viewValue: 'Internes Projekt'},
      {value: 'extern', disabled: false, viewValue: 'Externes Projekt'}
    ],
    /*Sonstige Optionen*/
    'etc': ''
  },
]
```

Abbildung 13: JSON Beispiel für Create

Der Aufbau der Daten entscheidet später welche Art von Eingabefeld daraus entsteht und welche Attribute dieses haben wird.

```
<div class="{{data.btclass}}" *ngFor="let data of mainComponentBuildData; let i = index">
  <div *ngIf="mainComponentBuildData[i].fieldtype === 'dropdown'">
    <mat-form-field [ngClass]="{'has-error': formErrors[[data.formControlName]]}"
      class="col-sm-12">
      <mat-label>{{data.placeholder}}</mat-label>
      <mat-select (openedChange)="logValidationErrors()" [formControlName]=data.formControlName>
        <mat-option disabled="{{options.disabled}}"
          *ngFor="let options of mainComponentBuildData[i].values"
          [value]=options.value>
          {{options.viewValue}}
        </mat-option>
      </mat-select>
      <mat-error mat-class="help-block" *ngIf="formErrors[[data.formControlName]]">
        {{formErrors[[data.formControlName]]}}
      </mat-error>
    </mat-form-field>
  </div>
  <div *ngIf="mainComponentBuildData[i].fieldtype === 'textfield'">
    <mat-form-field [ngClass]="{'has-error': formErrors[[data.formControlName]]}"
      class="col-sm-12">
      <mat-label>{{data.placeholder}}</mat-label>
      <input (blur)="logValidationErrors()" [formControlName]=data.formControlName matInput
        placeholder="{{data.placeholder}}>
      <mat-error mat-class="help-block" *ngIf="formErrors[[data.formControlName]]">
        {{formErrors[[data.formControlName]]}}
      </mat-error>
    </mat-form-field>
  </div>
```

Abbildung 14: 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 formErrors, als auch der formControlName vollständig dynamisch aus einer Klasse stammen werden.

```
initForm() {
  this.antragForm = this.fb.group(
    controlsConfig: {}
  );
  this.mainComponentBuildData.forEach(
    e => {
      this.antragForm.addControl(
        e.formControlName, new FormControl( formState: {value: '', disabled: e.disabled}, Validators.compose(e.validator)
      )
    );
  }
);
this.checkIfParams();
}
```

Abbildung 15: 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.

```
private _validationMessages = {
  'projektart': {
    'required': 'Bitte wählen sie eine Projektart aus!'
  },
  'system': {
    'required': 'Bitte wählen sie ein System aus!'
  },
  'projekttitel': {
    'required': 'Der Projekttitel muss mindestens 3 Zeichen lang sein!',
    'minlength': 'Der Projekttitel muss mindestens 3 Zeichen lang sein!',
    'maxlength': 'Der Projekttitel darf maximal 30 Zeichen lang sein!'
  },
  'beantrager': {
    'required': 'Bitte melden Sie sich erneut an. Beim Login ist etwas schiefgelaufen!'
  },
  'beschreibung': {
    'required': 'Die Beschreibung muss mindestens 20 Zeichen lang sein!',
    'minlength': 'Die Beschreibung muss mindestens 20 Zeichen lang sein!',
    'maxlength': 'Der Beschreibung darf maximal 250 Zeichen lang sein!'
  },
  'auftraggeberfirma': {
    'required': 'Bitte geben sie die Firma des Auftraggebers ein!'
  },
}
```

Abbildung 16: Validation Messages der Form Controls

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.

```
private _formErrors = {  
    'projektart': '',  
    'system': '',  
    'projekttitel': '',  
    'beantrager': '',  
    'beschreibung': '',  
    'auftraggeberFirma': '',  
    'projektstatus': '',  
    'projektleiter': '',  
    'projektmitarbeiter': '',  
    'angebotsnummer': '',  
    'bemi': '',  
    'kundenfreigabe': '',  
    'kommentarFreigabe': ''  
};
```

Abbildung 17: 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.

```

logValidationErrors(group: FormGroup = this.antragForm): void {
  Object.keys(group.controls).forEach((key: string) => {
    const abstractControl = group.get(key);
    if (abstractControl instanceof FormGroup) {
      this.logValidationErrors(abstractControl);
    } else {
      this.formErrors[key] = '';
      if (abstractControl && !abstractControl.valid &&
        (abstractControl.touched || abstractControl.dirty)) {
        const messages = this.validationMessages[key];
        for (const errorKey in abstractControl.errors) {
          if (errorKey) {
            this.formErrors[key] += messages[errorKey] + ' ';
          }
        }
      }
    }
  });
}

```

Abbildung 18: Loggen von geänderten Form Controls*

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.


```
initForm() {
  this.userForm = this.fb.group( controlsConfig: {
    user: this.fb.array( controlsConfig: [])
  });
  this.editusergroup();
}

editusergroup() {
  for (let j = 0; j < this.userList.length; j++) {
    this.addUsergroup();
    (<FormArray>this.userForm.get('user')).controls[j].patchValue(this.userList[j]);
  }
}

addUsergroup(): void {
  (<FormArray>this.userForm.get('user')).push(this.addUsergroupGroup());
}
```

Abbildung 19: 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.

Screenshots der fertigen Anwendung sind unter dem Punkt 14 „Screenshots der Anwendung“ zu finden.

6 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 Einsatz 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

7 Angular

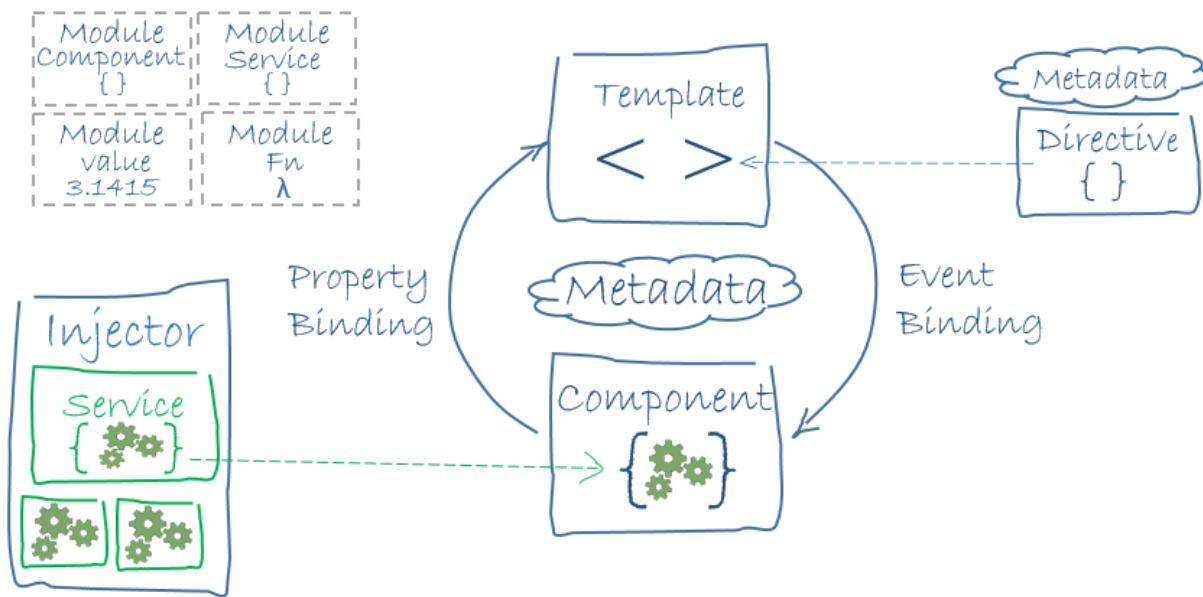


Abbildung 20: Funktionsweise von Angular

Die clientseitige Programmierung* ist in den letzten Jahren 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*, welches 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.

Vgl. WQ5

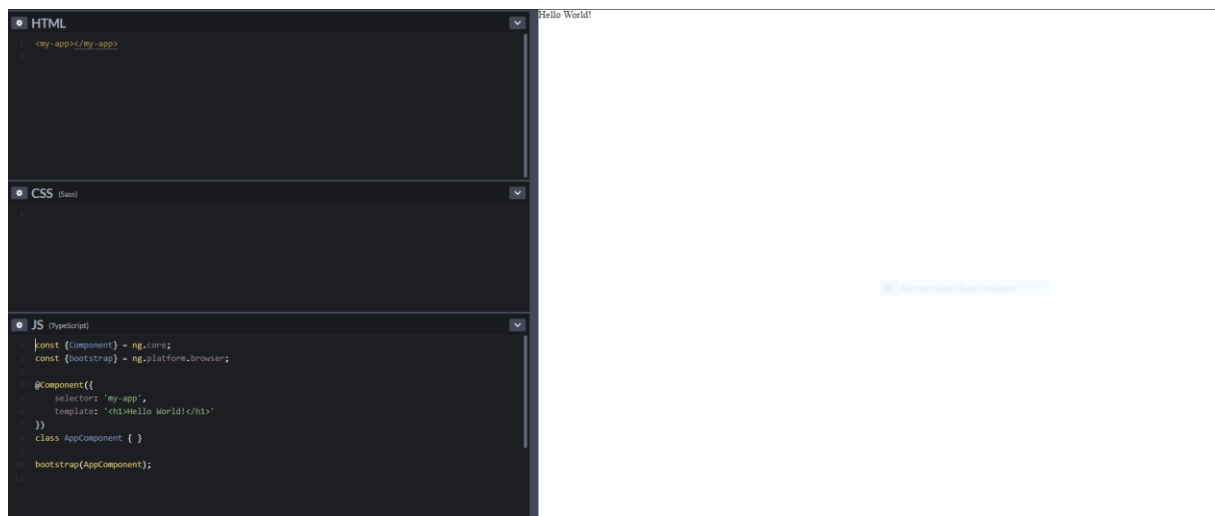


Abbildung 21: Hello World Beispiel Angular

8 Alternativen zum Framework Angular

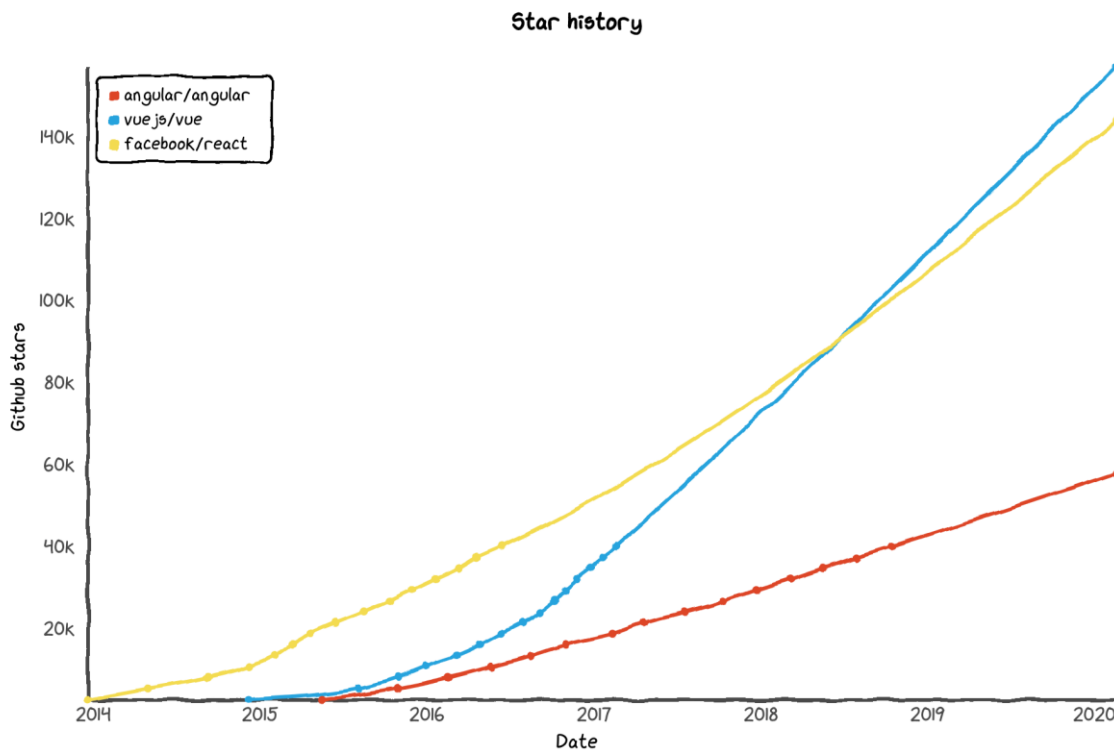


Abbildung 22: Star History der Frameworks im Vergleich

Durch die Sternevergabe auf Github lässt sich die Beliebtheit 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.

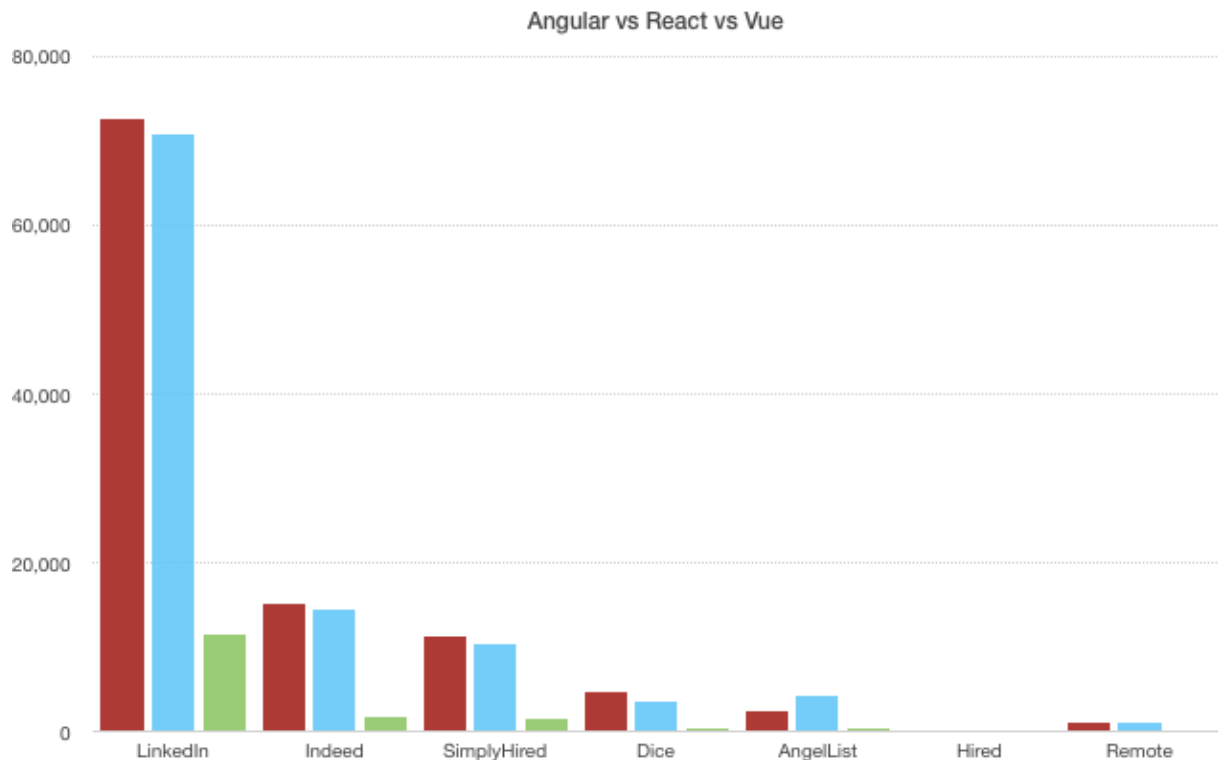


Abbildung 23: Jobangebote für Entwickler auf gängigen Plattformen

■	Angular
■	React
■	VueJS

Es ist erstaunlich das VueJS bisher kaum angefragt wird, obwohl es leichter zu lernen ist und einen ähnlichen Funktionsumfang bietet als beispielsweise Angular. Dieser Trend kommt vermutlich daher, dass die größeren Unternehmen eher Frameworks* mit einer vollständigen und transparenten Dokumentation nutzen möchten.

8.1 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.

[Vgl. WQ3](#)

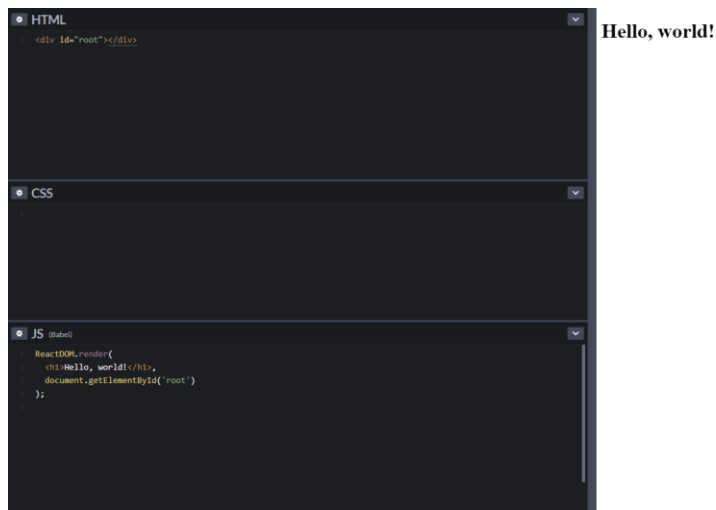


Abbildung 24: Hello World Beispiel React

8.2 VueJS

Vue.js ist ein clientseitiges JavaScript-Webframework* zum Erstellen von Single-Page-Webanwendungen nach dem MVVM-Muster*, es kann allerdings auch in Multipage Webseiten für einzelne Abschnitte verwendet werden. Ab Version 2.0 unterstützt es auch serverseitiges Rendern*.

[Vgl. WQ4](#)

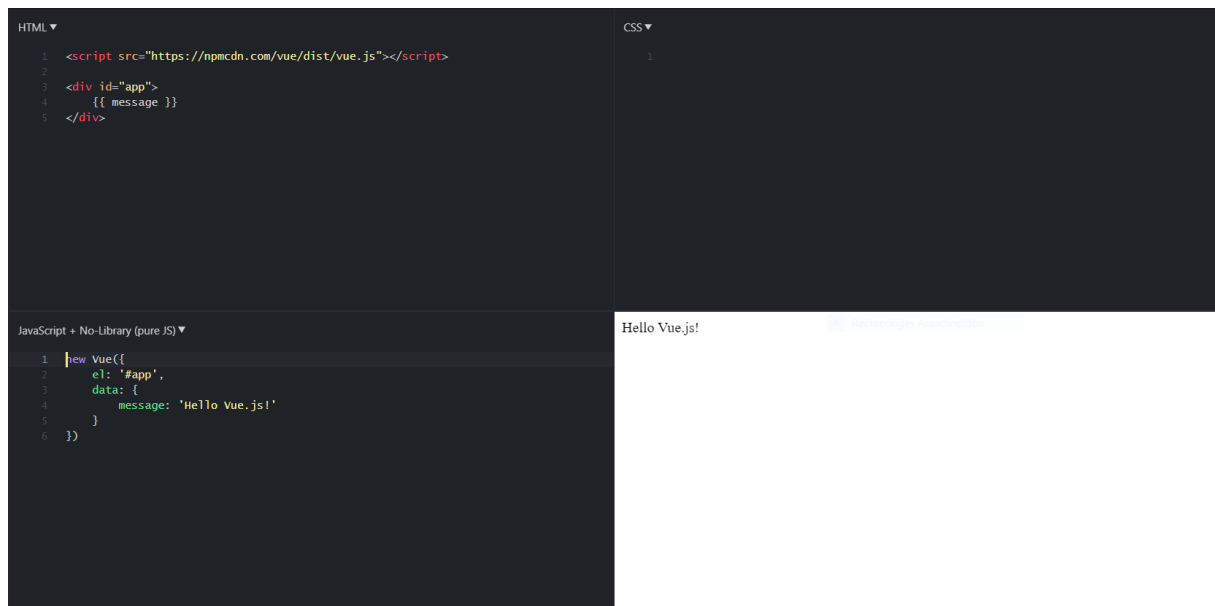


Abbildung 25: Hello World Beispiel VueJS

8.3 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 Compiler sowie Renderer* eingeführt.

8.4 Übersicht Frameworks und Entscheidung

Angular		React		VueJS	
+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 Dokumentation von Angular selbst und den Material Komponenten sehr ausführlich und umfangreich ist, fällt das Erweitern der bestehenden Anwendung entsprechend leicht. Dies war schlussendlich ausschlaggebend für die Entscheidung.

9 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, bei dem Abteilungsleiter auf 105€ pro Stunde und bei einem Auszubildenden auf 15 €. Von der Geschäftsleitung wurde für die Digitalisierung ein Budget von 1500€ zur Verfügung gestellt. Dieses Budget sollte nicht überschritten werden.

	Auszubildender (15€)	Entwickler (90€)	Abteilungsleiter (105€)	Preis
Analyse	4		1	165
Konzeption	10			150
Realisierung	33	1		585
Sonstiges	23		1	450
Gesamt p.P.	1050	90	210	
Gesamt				1350

10 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 und alle fachlichen Anforderungen wurden erfüllt. Um die dynamische Generierung der Inhalte nach und nach aufbauen 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 um die fachlichen Anforderungen erfüllen zu können.

Der Nutzer hat nun die Möglichkeit, Anträge per Webanwendung anzulegen. Sämtliche Eingabefelder verfügen über Validierungen und zugehörige Fehlermeldungen zu jeder einzelnen Art von Error. Eine Filterung nach dem aktuellen Status sowie eine Gesamtübersicht der gestellten Projektanträge ist nun möglich. Neue Mitarbeiter können die Anwendung von Anfang an nutzen und es benötigt keinen weiteren Administrationsaufwand um diese anzulegen. Bestehende Nutzer können sich anmelden und gelangen zu den eigenen Anträgen. Der Administrator kann jederzeit weitere Rechte vergeben, entziehen und Benutzerdaten ändern. Durch die neue Statusschaltung können nur noch berechnigte Personen den Antragsstatus ändern oder Bemerkungen hinzufügen bzw. ändern.

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.

11 Quellen

11.1 Webquellen

- WQ1) <https://angular.io/>
WQ2) <https://angular.io/guide/reactive-forms>
WQ3) <https://de.wikipedia.org/wiki/React>
WQ4) <https://de.wikipedia.org/wiki/Vue.js>
WQ5) <https://www.digicomp.ch/blog/2017/01/16/angular-ein-uberblick>
WQ6) <https://medium.com/@scarlett8285/why-is-reactjs-the-best-fit-for-your-social-networking-app-15646d6ad082>
WQ7) <https://gist.github.com/tkrotoff/b1caa4c3a185629299ec234d2314e190>
WQ8) https://de.wikipedia.org/wiki/Angular#Unterschiede_zwischen_Angular_und_AngularJS
WQ9) <https://www.ngdevelop.tech/angular/history/>
WQ10) <https://app.diagrams.net/>
WQ11) <https://react-etc.net/entry/angular-4-coming-in-2017-backwards-compatible-angular-2>
WQ12) <https://www.infoworld.com/article/3225511/angular-5-javascript-framework-delayed.html>
WQ13) <https://blog.angular.io/version-6-of-angular-now-available-cc56b0efa7a4>
WQ14) <https://blog.angular.io/version-7-of-angular-cli-prompts-virtual-scroll-drag-and-drop-and-more-c594e22e7b8c>
WQ15) <https://blog.angular.io/a-plan-for-version-8-0-and-ivy-b3318dfc19f7>
WQ16) <https://de.wikipedia.org/wiki/Client-Server-Modell>
WQ17) <https://www.edv-weilheim.de/anwendungen-programmieren/>
WQ18) <https://www.edv-weilheim.de/framework/>
WQ19) <https://www.json.org/json-de.html>
WQ20) https://de.wikipedia.org/wiki/Model_View_ViewModel
WQ21) https://de.wikipedia.org/wiki/Portable_Document_Format
WQ22) https://de.wikipedia.org/wiki/Software_Rendering
WQ23) <https://www.cloudcomputing-insider.de/was-ist-eine-rest-api-a-611116/>
WQ24) <https://de.wikipedia.org/wiki/TypeScript>
WQ25) <https://www.elektronik-kompodium.de/sites/net/1909041.htm>

11.2 Buchquellen

- BQ1) *Angular: Grundlagen, fortgeschrittene Themen und Best Practices* von Ferdinand Malcher, Johannes Hoppe, Danny Koppenhagen
BQ2) *EssentialsAngular: The Essential Guide to Learn Angular* von Dhananjay Kumar

12 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, 27.05.2020

Ort, Datum

Timo Menhorn

Erklärung des Ausbildungsbetriebes

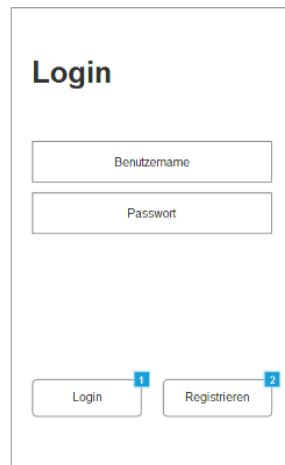
Das Projekt wurde, wie in der Dokumentation dargestellt, in unserem Unternehmen durchgeführt.

Ingolstadt, 27.05.2020

Ort, Datum

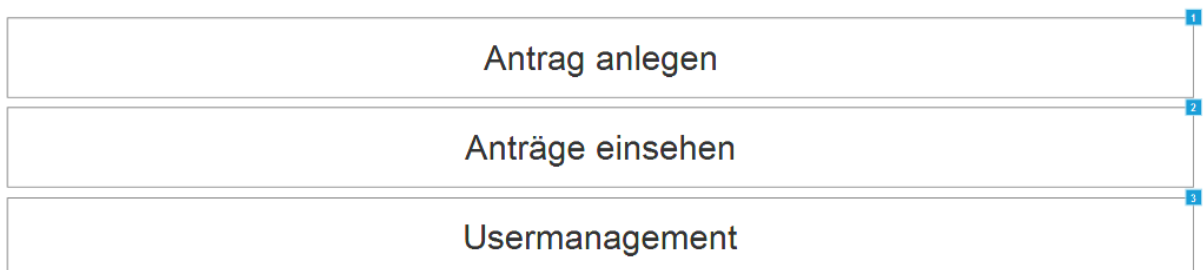
Flaster Albig

13 Screenshots des Prototypens



The image shows a login form titled "Login". It contains two input fields: "Benutzername" (Username) and "Passwort" (Password). Below these fields are two buttons: "Login" and "Registrieren" (Register). The "Login" button is marked with a small blue square containing the number 1, and the "Registrieren" button is marked with a small blue square containing the number 2.

Abbildung 26: Entwurf Login



The image shows a home page design with three horizontal buttons. The first button is labeled "Antrag anlegen" (Create application) and is marked with a small blue square containing the number 1. The second button is labeled "Anträge einsehen" (View applications) and is marked with a small blue square containing the number 2. The third button is labeled "Usermanagement" and is marked with a small blue square containing the number 3.

Abbildung 27: Entwurf Home

Entwicklung einer Web-Anwendung zur Projektantragsverwaltung mit JSON Datenbank

Projektart	System	Projekttitel	Status

Abbildung 28: Entwurf List

Username1	*****	Rolle1	-
Username1	*****	Rolle1	-
Username1	*****	Rolle1	-
Username1	*****	Rolle1	-
Username1	*****	Rolle1	-

Abbildung 29: Entwurf Usermanagement

14 Screenshots der Anwendung



Abbildung 30: Anwendung Login

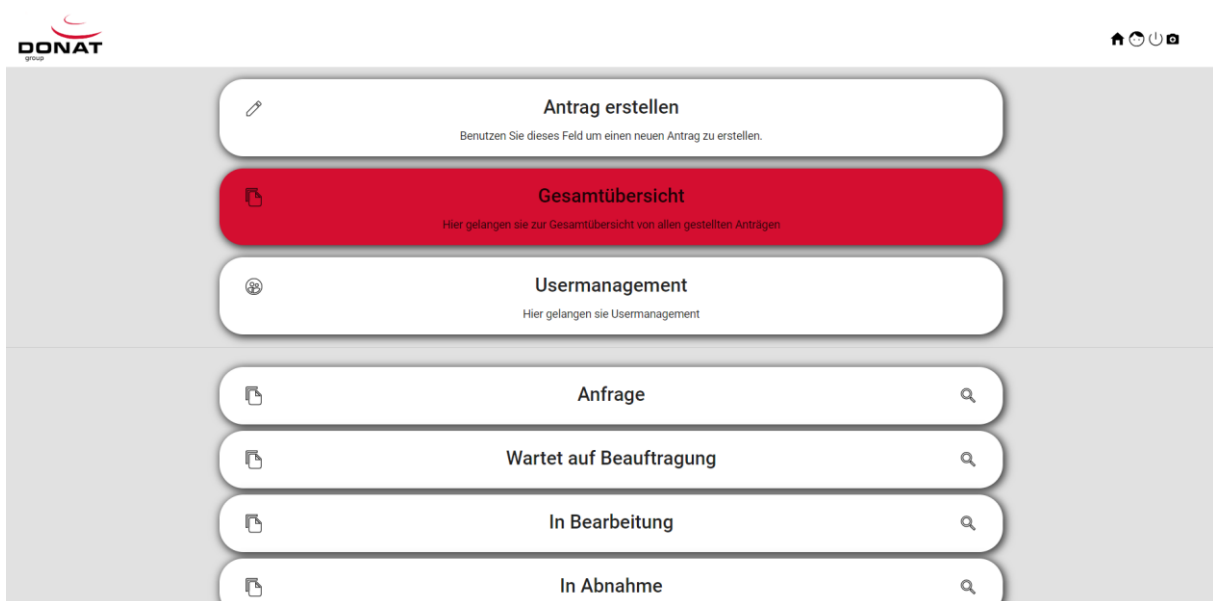




Abbildung 31: Anwendung Home



Entwicklung einer Web-Anwendung zur Projektantragsverwaltung mit JSON Datenbank



Filter

Projektart	System	Projekttitel	Projektstatus	Projektdauer(h)	Details	Löschen
extern	integral	ZAm	3	22		
intern	allgemein	ZAu	2	22		
intern	his	ZEx	1	22		
intern	allgemein	123	3	22		
intern	access	menhorti	0	2		
intern	Rechteckiges Ausschneiden	access	0	3		
intern	marketing	EEEE	1	199		
intern	access	OOO	1	300		
intern	integral	218WH2P310	0	1234		

Abbildung 25: Anwendung List



Antragserstellung

Projektart(*)

Projekttitel(*)

Geplante Projektdauer in Stunden(*)

Beschreibung zum Projekt(*)

Auftraggeber Firma(*)

Rechteckiges Ausschneiden

Projektleiter(*)

Projektmitarbeiter(*)

BEM-Nummer(*)

Kommentar zur Freigabe

System(*)
Sharepoint

Beiträger(*)
menhorti

Projektstatus(*)
Anfrage

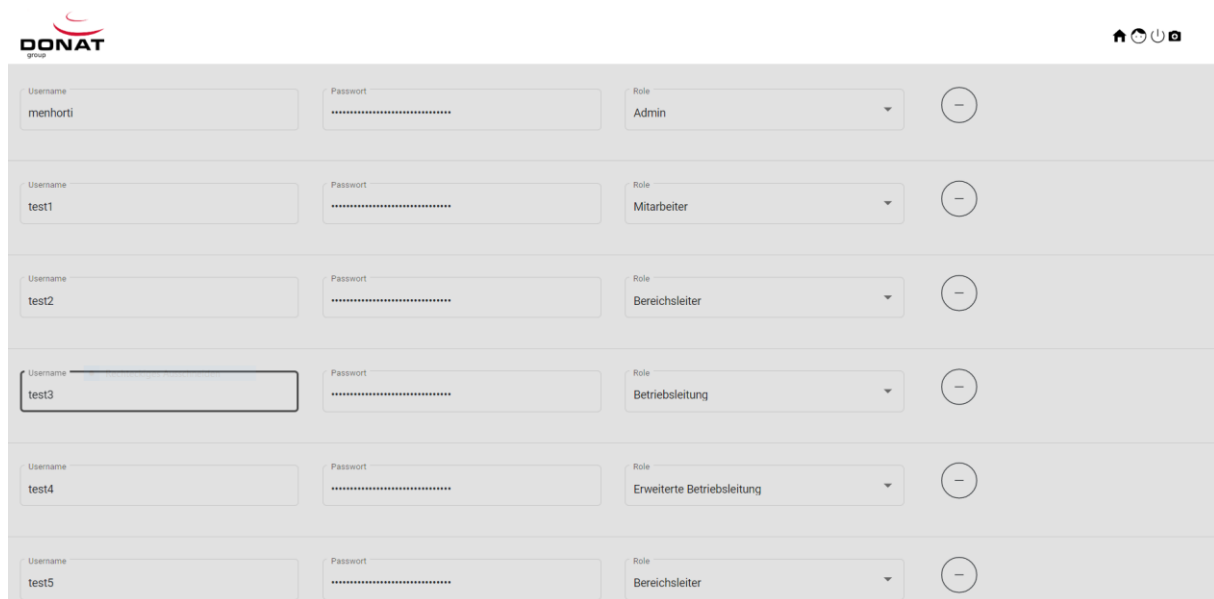
test2
test5

Kundenfreigabe(*)

Abenden

Abbrechen

Abbildung 32: Anwendung Create



The screenshot displays a web application interface for user management. At the top left is the DONAT logo, and at the top right are navigation icons (home, refresh, power, and a camera). The main content area is a table with six rows, each representing a user. Each row contains four elements: a text input for 'Username', a password input for 'Passwort', a dropdown menu for 'Role', and a circular button with a minus sign for deletion. The users listed are: menhorti (Admin), test1 (Mitarbeiter), test2 (Bereichsleiter), test3 (Betriebsleitung), test4 (Erweiterte Betriebsleitung), and test5 (Bereichsleiter). The 'test3' row is highlighted with a blue border.

Username	Passwort	Role	Action
menhorti	Admin	⊖
test1	Mitarbeiter	⊖
test2	Bereichsleiter	⊖
test3	Betriebsleitung	⊖
test4	Erweiterte Betriebsleitung	⊖
test5	Bereichsleiter	⊖

Abbildung 33: Anwendung Usermanagement

15 Glossar

15.1 Begriffsklärung

Angular Material

Von Google entwickelte Designsprache, welche materialartige, eher kantige Flächen in einem Flat Design verwendet.

Angular CLI Befehle

- **ng update:** Updated die Anwendung und ihre Bibliotheken
- **ng add:** Fügt eine neue Bibliothek hinzu
- **ng build:** Kompiliert die Anwendung
- **ng test:** Startet Unit Tests
- **ng deploy:** Deployment der Anwendung

Angular Ivy

Angulars neue Kompilation und Rendering Pipeline, welche nun die View Engine ersetzt.

Architekturkonzepte

Eine Softwarearchitektur beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb eines Softwaresystems.

Authentifizierung

Die Authentifizierung sorgt 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 in View und Datenstrukturen im Model.

Client-Server-Architektur

Das Client-Server-Modell beschreibt eine Möglichkeit, Aufgaben und Dienstleistungen innerhalb eines Netzwerkes zu verteilen. Die Aufgaben werden von Programmen erledigt, die in Clients und Server unterteilt werden.

Vgl. WQ16

Component Dev Kit (CDK)

Einige Tools für die Grundfunktionalität von Angular.

Controller

Die Steuerung (Controller) verwaltet die Präsentation und das Modell.

CSS

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

Dependency Injection

Entwurfsmuster der objektorientierten Programmierung (OOP), bei dem Abhängigkeit zwischen Objekte erst zur Laufzeit hergestellt werden.

Desktop-Applikation

Als Anwendungssoftware werden Computerprogramme bezeichnet, die genutzt werden, um eine nützliche oder gewünschte nicht systemtechnische Funktionalität zu bearbeiten oder zu unterstützen.

[*Vgl. WQ17*](#)

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.

Drag und Drop

Ziehen und Ablegen von einzelnen Elementen.

Framework

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

[*Vgl. WQ18*](#)

Frontend

Als Frontend wird die sichtbare Oberfläche für Nutzer bezeichnet.

Hashwertbildung

Kryptografische Hash-Funktionen generieren aus beliebig langen Datensätzen eine Zeichenkette mit einer festen Länge. Die Anforderungen an eine solche Hash-Funktion sind:

- **Eindeutigkeit:** Eine identische Zeichenfolge muss zum selben Hash-Wert führen.
- **Reversibilität:** Der Hash-Wert darf nicht in die ursprüngliche Zeichenfolge zurückberechnet werden können.
- **Kollisionsresistenz:** Zwei unterschiedliche Zeichenfolgen dürfen nicht den gleichen Hash-Wert ergeben.

[Vgl. WQ 25](#)

HTML

HTML bestimmt den strukturellen Aufbau einer Internetseite.

Initialisieren

Zuweisung eines Initial- oder Anfangswertes zu einem Objekt oder einer Variablen.

Internationalisierung

Optimierung eines Programms für die einfache Anpassung in andere Sprachen und Kulturkreise.

Interpolation

Template-Strings können Platzhalter beinhalten, die durch geschweifte Klammern gekennzeichnet sind ({{example}}). Durch Interpolation können Sie berechnete Zeichenfolgen in den Text zwischen HTML-Element-Tags und in Attributzuweisungen integrieren.

JSON

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

[Vgl. WQ19](#)

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.

[Vgl. WQ20](#)

Node.js

Node.js ist eine serverseitige Plattform in der Softwareentwicklung zum Betrieb von Netzwerkanwendungen.

NPM

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

Open-Source-Software

Als Open Source wird Software bezeichnet, deren Quelltext öffentlich und von Dritten eingesehen, geändert und genutzt werden kann.

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.

[Vgl. WQ21](#)

PostgreSQL

PostgreSQL ist ein freies, objektrelationales Datenbankmanagementsystem.

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.

[Vgl. WQ22](#)

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.

[Vgl. WQ23](#)

Scopes

Scopes sind die Verbindungen zwischen HTML (View) und JavaScript (Controller). Es ist ein Objekt mit den verfügbaren Eigenschaften und Methoden.

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.

[Vgl. WQ24](#)

User-Interface-Komponenten

Grafische Benutzeroberfläche oder auch grafische Benutzerschnittstelle bezeichnet eine Form von Benutzerschnittstelle eines Computers.

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.

Validierungspattern

Input Elemente können damit nach bestimmten Kriterien(Pattern) validiert werden.

Virtual Scrolling

Bei einer Webanwendung werden oft größere Datenmengen geladen. Damit die Performance weniger darunter leidet, werden beim Virtual Scrolling nur die Elemente angezeigt und geladen, die der User gerade sieht.

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.

15.2 Reactive Forms Begriffe

Form Array

Ein Form Array verfolgt den Wert und Validierungsstatus eines Arrays von FormControl, FormGroup und FormArray. 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 FormControl und mündet sie in ein Objekt, das den Namen des FormControl 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.