Auswertung und Visualisierung von Messdaten des Antriebprüfstandes im Bereich ÖBB Drehgestellaufarbeitung Linz

DIPLOMARBEIT

verfasst im Rahmen der

Reife- und Diplomprüfung

an der

Höheren Abteilung für Informatik

Eingereicht von:
Fabian Tischler
Philip Trinkl

Betreuer:
Matthias Braun

Projektpartner:
Dipl. Ing. Dr. techn. Christian Maier, ÖBB

Leonding, April 2022

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Weise keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Die vorliegende Diplomarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Leonding, April 2022 F. Tischler & P. Trinkl

Zur Verbesserung der Lesbarkeit wurde in diesem Dokument auf eine geschlechtsneutrale Ausdrucksweise verzichtet. Alle verwendeten Formulierungen richten sich jedoch an beide Geschlechter.

Abstract

Task

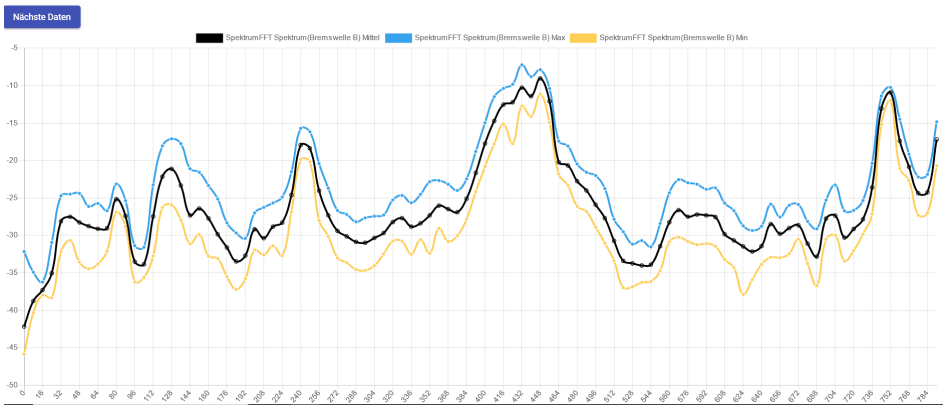
The task was to program a new test program for the train drives of the ÖBB Technical Service Linz. Because this task turned out to be too extensive for this work, it was agreed upon that an evaluation program for the errors of the currently used program was to be made. This program should filter and display errors according to various criteria, which should increase error detection in productive operation.

Implementation

The implementation was done in Csharp and Angular as we already had experience with these programming languages. We get the data which is used in this project from QTX-Files that we recieved from the ÖBB-TS Linz. This data is then evaluated in the backend and displayed in the frontend.

Result

An application with the basic functionalities was created. These functionalities can be expanded in the future by, for example, more criteria or a connection to the program which is used in productive operation.



Zusammenfassung

Aufgabenstellung

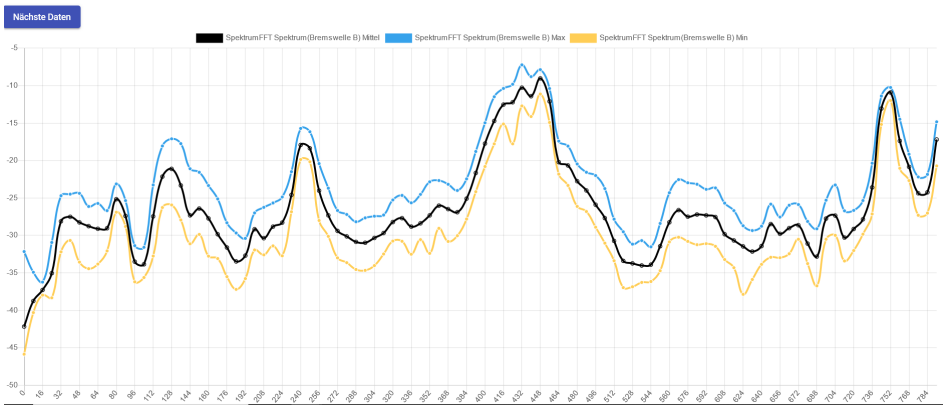
Aufgabe war es, ein neues Prüfprogramm für die Zugantriebe des ÖBB Technischen Service Linz zu programmieren. Da sich diese Aufgabe für diese Arbeit als zu umfangreich herausgestellt hat, wurde sich darauf geeinigt, ein Auswertungsprogramm für die Fehler des derzeit eingesetzten Prüfprogramms zu programmieren. Dieses Programm soll die Fehler nach verschiedenen Kriterien filtern und darstellen können, was die Fehlererkennung im Produktivbetrieb erhöhen soll.

Realisierung

Die Implementierung wurde in Csharp und Angular durchgeführt, da wir bereits Erfahrung mit diesen Programmiersprachen hatten. Die Daten, welche in diesem Projekt benutzt werden, werden aus QTX-Dateien, die wir von dem ÖBB-TS Linz zur Verfügung gestellt bekommen haben, eingelesen und danach im Backend ausgewertet. Im Frontend werden die ausgewerteten Daten angezeigt.

Ergebnis

Es wurde eine Applikation erstellt, die grundsätzlich die vereinbarten Funktionalitäten enthält. Diese Funktionalitäten können in Zukunft durch beispielsweise mehr Kriterien oder einer Anbindung an das im Produktivbetrieb benutzte Prüfprogramm erweitert werden.



Inhaltsverzeichnis

1 Einleitung

Die Österreichischen Bundesbahnen (kurz ÖBB) sind ein Bahnunternehmen, welches österreichweit agiert. Das Technische Service Werk in Linz beschäftigt sich mit leichter und schwerer Instandhaltung, Lackierung, Komponentenaufarbeitung, Unfallreperaturen, Engineering-Leistungen, Umbauten beziehungsweise Modifikationen und der Überprüfung von Zugsicherungs- und Zugfunksystemen. Diese Arbeit beschäftigt sich konkret mit der Komponentenaufarbeitung, genauer mit der Motorenaufarbeitung/-Getriebeaufarbeitung. Im TS-Werk Linz werden neue oder reparierte Antriebe, bevor sie in eine Lok eingebaut werden, gründlich auf Fehler getestet, jedoch werden die auftretenden Fehler nur zur Laufzeit angezeigt und dann in unübersichtlichen Dateien abgespeichert. Das macht die spätere Fehlerauswertung und Prognose für zukünftige Antriebe schwer und umständlich. Hier kommt ASUQZ ins Spiel. ASUQZ bedeutet Antriebsprüfstands-Software zur Untersuchung und Qualitätssicherung von Zugteilen und ermöglicht es, die Fehler nach verschiedenen Kriterien leicht und schnell abzurufen sowie graphisch darzustellen.

2 Pflichtenheft

2.1 Ausgangssituation

Zurzeit gibt es drei voneinander abhängige Programme, die die Aufnahme und Auswertung der Daten übernehmen. Die derzeit eingesetzten Programme können nur von einigen wenigen Fachkräften benutzt werden und sind unübersichtlich.

2.2 Zielsetzung

Zielsetzung ist eine Vereinfachung der Bedienung der Antriebsprüfstände. Des Weiteren eine Verbesserung der Datenauswertung mithilfe von Mustererkennung. Außerdem eine Visualisierung der ausgewerteten Daten. Abschließend soll auch die Fehlererkennung und -beseitigung erhöht werden.

2.3 Funktionale Anforderungen

2.3.1 Analyse der vorhandenen Daten

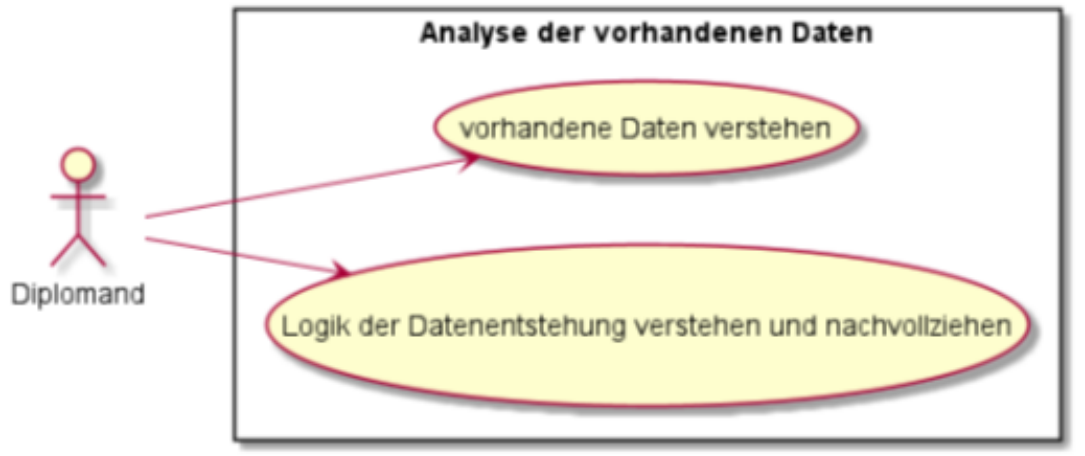


Abbildung 1: Analyse der vorhandenen Daten

- Der Diplomand soll die vorhandenen Daten analysieren, das beinhaltet welche Daten vorhanden sind, wie groß die Datenmenge ist, ob die Daten auswertbar sind und was man in den Daten erkennt.
- Der Diplomand soll die Logik der Datenentstehung verstehen und nachvollziehen können, das beinhaltet welche Signale vom System übermittelt werden, in welcher Qualität diese Signale kommen, wie diese Signale in benutzbare Daten umgewandelt werden und wie die Schnittstelle aussieht und arbeitet.

2.3.2 Antriebsprüfstands-Software zur Untersuchung und Qualitätssicherung von Zugteilen

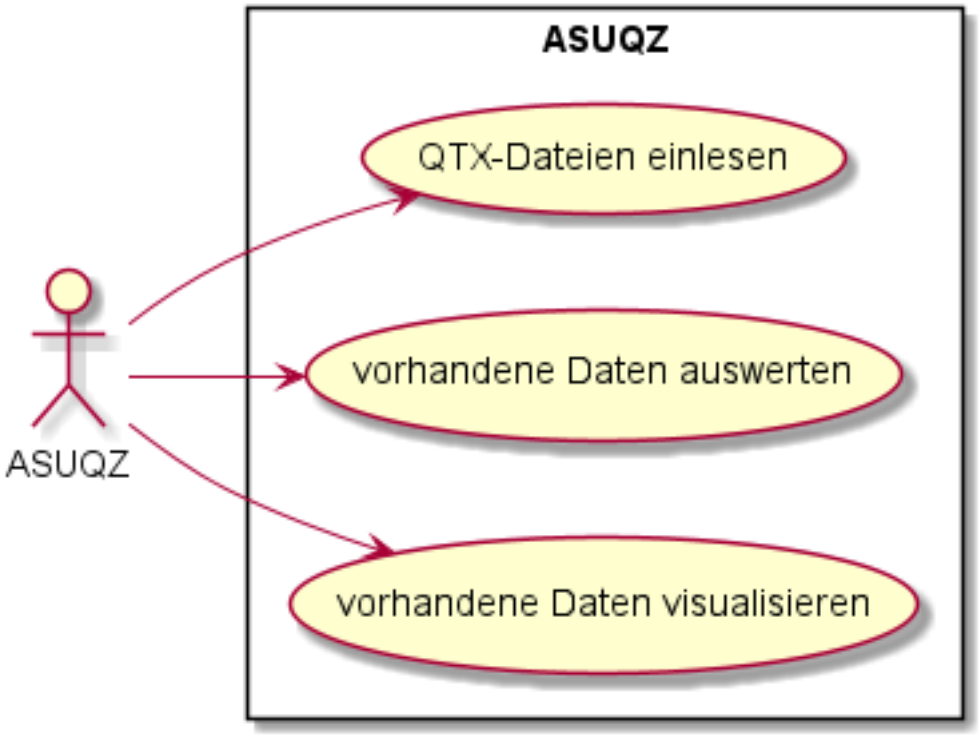


Abbildung 2: ASUQZ Anforderungen

- Der Prototyp soll die vorhandenen QTX-Dateien, welche von dem derzeit benutzten Auswertungsprogramm der ÖBB produziert werden, einlesen können.
- Der Prototyp soll die vorhanden Daten nach Fehlern durchsuchen und ausgewerteten.
- Der Prototyp soll die ausgewerteten Daten graphisch darstellen.

2.4 Nicht-funktionale Anforderungen

- Der Prototyp soll in Deutsch realisiert section
- Die Auswertung soll formatiert und leicht lesbar sein
- Der Prototyp soll in einer angemessenen Zeitspanne funktionieren und reagieren
- Die vorhandenen und im Laufe des Projekts erhobenen Daten sollen nicht an Dritte weitergegeben werden
- Der Prototyp soll eine gute Systemarchitektur aufweisen und infolgedessen gut wartbar sein

2.5 Programm

2.5.1 Allgemeine Beschreibung

Ein Mitarbeiter kann sich die Fehler zurückliegender Testdurchläufe nach verschiedenen Kriterien gefiltert und sortiert anzeigen lassen. Des Weiteren kann er sich die verschiedenen Testdurchläufe graphisch anzeigen lassen.

2.5.2 Mockups

Beim Starten des Programms gelangt man zum Auswahlbereich der Files. Hier kann der Mitarbeiter die gewünschte QTX-Datei laden [3].

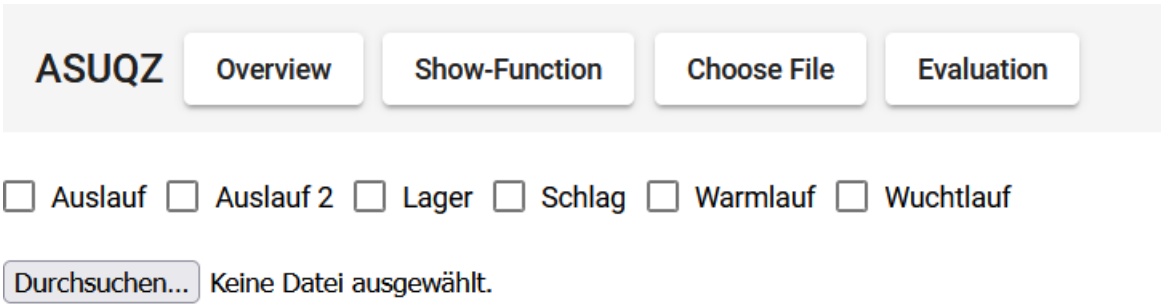


Abbildung 3: Datei auswählen

Wurde eine Datei ausgewählt, wird der Mitarbeiter zur grafischen Auswertung weitergeleitet. Hier kann der Mitarbeiter alle verschiedenen Messungen dieser Datei ansehen [4].

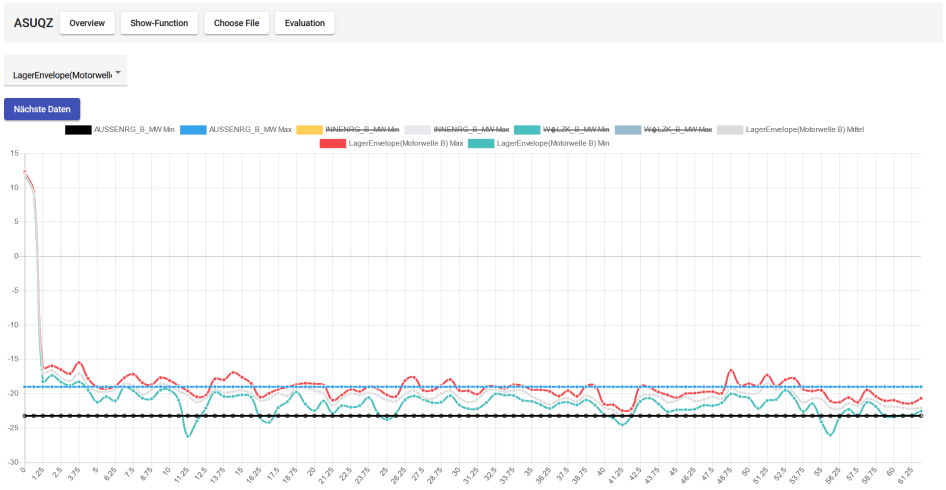
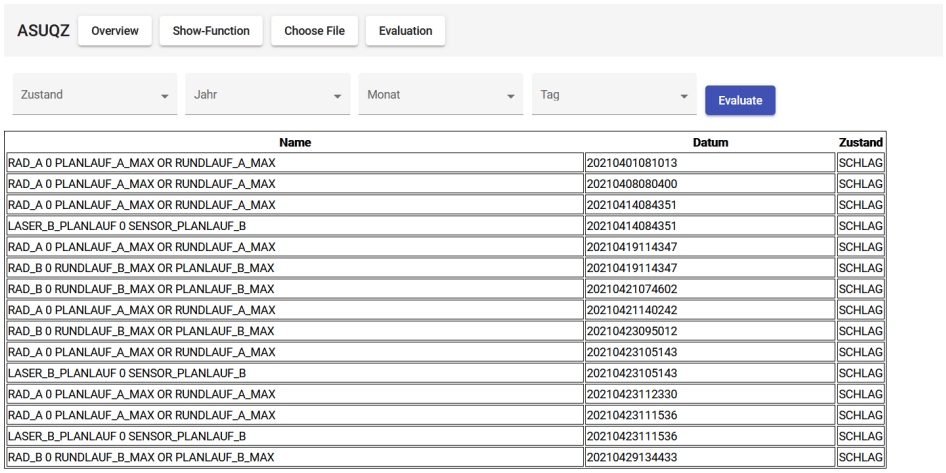


Abbildung 4: Visualisierung

Für das Auswerten der Fehler werden die Kriterien in dieser [5] Maske ausgewählt.



Name	Datum	Zustand
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210401081013	SCHLAG
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210408080400	SCHLAG
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210414084351	SCHLAG
LASER_B_PLANLAUF 0 SENSOR_PLANLAUF_B	20210414084351	SCHLAG
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210419114347	SCHLAG
RAD_B 0 RUNDLAUF_B_MAX OR PLANLAUF_B_MAX	20210419114347	SCHLAG
RAD_B 0 RUNDLAUF_B_MAX OR PLANLAUF_B_MAX	20210421074602	SCHLAG
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210421140242	SCHLAG
RAD_B 0 RUNDLAUF_B_MAX OR PLANLAUF_B_MAX	20210423095012	SCHLAG
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210423105143	SCHLAG
LASER_B_PLANLAUF 0 SENSOR_PLANLAUF_B	20210423105143	SCHLAG
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210423112330	SCHLAG
RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX	20210423111536	SCHLAG
LASER_B_PLANLAUF 0 SENSOR_PLANLAUF_B	20210423111536	SCHLAG
RAD_B 0 RUNDLAUF_B_MAX OR PLANLAUF_B_MAX	20210429134433	SCHLAG

Abbildung 5: Auswertung

3 Technologien Frontend

3.1 Auswahl der Frontend-Technologie

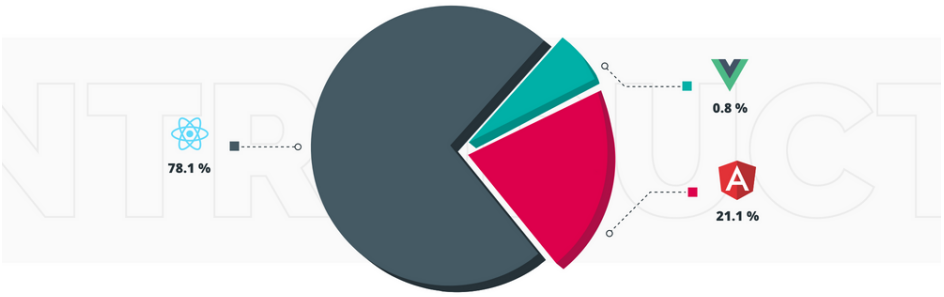


Abbildung 6: Marktanteil

Da der Auftrag in einer Progressive-Web-App besteht, war klar, dass eine JavaScript-Anwendung die Anforderungen der ÖBB am besten abdecken würde. In dieser Grafik werden die Marktanteile der drei am häufigsten benutzten JavaScript-Frameworks für Webanwendungen dargestellt. Im folgenden gehen wir weiter auf diese drei Frameworks ein [1].

3.1.1 ReactJS

ReactJS ist eine Open-Source JavaScript Bibliothek, welche von Facebook zur Verfügung gestellt wird.

Vorteile

- leicht zu lernen
- hohe Flexibilität
- viele Updates durch den hohen Marktanteil

Nachteile

- wenig offizielle Dokumentation
- "zu viel Auswahl"

3.1.2 Angular

Angular ist ein auf TypeScript [3.2.2] basierendes Open-Source Framework von Google zur Erstellung von Webanwendungen. [2] [3]

Vorteile

- einheitliche Struktur, woraus sich sauberer und gut lesbarer Code ableitet
- Vielzahl von Bibliotheken
- Modularisiert
- Aufgebaut als Single-Page-Anwendung

Nachteile

- schlechte Unterstützung in alten Browsern
- Nicht gut skalierbar
- höhere Test- und Buildzeiten

3.1.3 Vue.js

Vue.js ist ein clientseitiges JavaScript-Framework zur Erstellung von anpassungsfähigen Single-Page-Anwendungen.

Vorteile

- Detaillierte Dokumentation
- Hohe Anpassungsfähigkeit
- Sehr gut skalierbar
- Winzige Größe

Nachteile

- Große Teile der Dokumentation noch in Chinesisch
- Wenig Wissensaustausch bedingt durch den niedrigen Marktanteil

Aufgrund gründlicher Eavluierung der oben genannten Vor- und Nachteile und gemeinsamer Rücksprache mit dem ÖBB Technischen Service Linz wurde sich für diese Arbeit für Angular entschieden.

3.2 Angular

Wie oben beschrieben ist Angular ein auf TypeScript basierendes Open-Source Framework, welches von Google entwickelt wird. Angular beinhaltet neben der reinen Entwicklungs-API auch Codegeneratoren und vordefinierte Architektur-Konzepte.

3.2.1 Aufbau des Angular-Frameworks

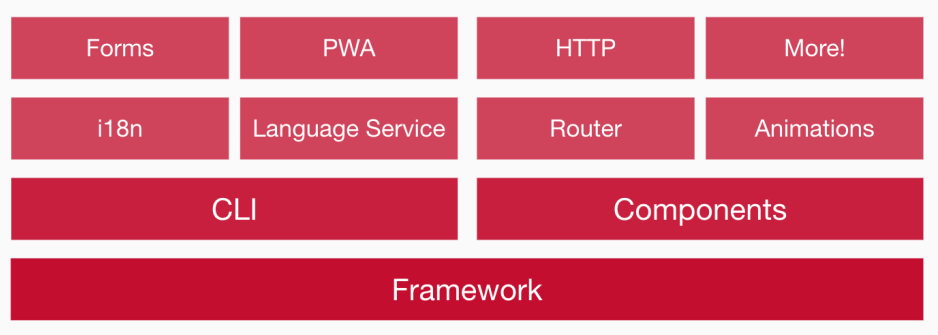


Abbildung 7: Aufbau des Angular-Frameworks

Die Basis von Angular bildet das Core-Framework. In diesem Framework sind die Grundkonzepte für moderne Webentwicklung implementiert. Darunter stehen die Angular-CLI, die Angular Komponenten und die Angular Services und darunter stehen wiederum kleinere, einzelne Module, die optional in die Applikation eingebunden werden können. Unter diese Module fallen beispielsweise das Progressive Web App Modul, was in diesem Projekt benutzt wird um die Offlinefähigkeiten der Anwendungen zur Verfügung zu stellen oder das Router Modul, welches benutzt wird um zwischen den verschiedenen Komponenten zu wechseln.

Die Angular CLI wird benutzt, um die benötigten Strukturen der Applikation zu generieren. Mit dem Befehl `ng new "name"` wird beispielsweise ein neues Angular-

Projekt erstellt, mit `ng generate component "name"` wird eine neue Komponente in dem eben generierten Projekt erstellt.

Angular Komponenten sind die Anzeigeelemente einer Anwendung. Für jede Funktion, die in der Anwendung benötigt wird, wird eine eigene Komponente erstellt, welche wiederum aus einer TypeScript-Datei, einer HTML-Datei und einem Stylesheet bestehen. Sollte es notwendig sein, lassen sich diese Komponenten untereinander auch leicht verschachteln. [4]

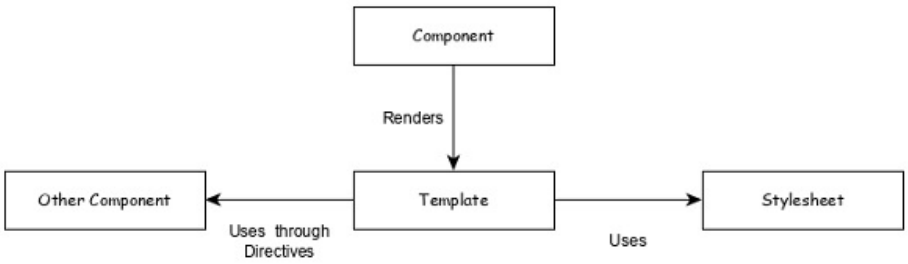


Abbildung 8: Aufbau einer Komponente

Die Komponente, beziehungsweise die TypeScript-Datei, rendert die HTML-Datei, die wiederum den Stylesheet benutzt. Andere Komponenten können durch importieren in der TypeScript-Datei benutzt werden.

Angular Services werden benutzt, um Logik und Daten, die nicht an einzelne Komponenten gebunden sind, herauszunehmen um Codeverdopplung zu vermeiden. Beispielsweise werden Daten, die in mehreren Komponenten benötigt werden, in einem Service gespeichert und in den Komponenten werden diese Daten dann von dem Service aus aufgerufen. [5]

3.2.2 TypeScript

TypeScript ist, im Gegensatz zu JavaScript, statisch typisiert. Daher werden Typfehler schon in der Entwicklungszeit aufgezeigt. TypeScript ist ein Superset von JavaScript, was aufgeteilt wird in die JavaScript Syntax, die Typisierung und das Laufzeitverhalten.

Syntax

Da, wie oben gesagt TypeScript ein Superset von JavaScript ist, ist auch jede Syntax eines JavaScript-Codes legal in TypeScript-Code. Code wie zum Beispiel `let`

`name = (Max` beinhaltet durch die fehlende Klammer einen Syntaxfehler, was aber in JavaScript-Syntax legal ist und daher auch in TypeScript funktioniert. Dadurch kann jeder JavaScript-Code einfach in eine TypeScript-Datei kopiert werden und es funktioniert.

Typisierung

TypeScript ist, wie oben genannt, stark typisiert, das heißt es werden verschiedene Regeln hinzugefügt, welche Werte man wann und bei welchen Objekten nutzen kann. Würde man zum Beispiel Code wie `x = 10 / 'hallo'` ausführen, würde es einen Fehler werfen, da auf der rechten Seite einer mathematische Funktion kein String stehen darf. Die Stärke dieser Typisierung lässt sich mit verschiedenen Einstellung anpassen. Sobald der Compiler den Code fertig nach Typen gecheckt hat, werden diese Typen beim Kompilationsvorgang gelöscht, um wieder zu einfachen JavaScript-Code zu kommen.

Laufzeitverhalten

TypeScript bewahrt das Laufzeitverhalten von JavaScript, das heißt wenn Code von JavaScript zu TypeScript geändert wird, funktioniert der ausgeführte Code komplett gleich.[6]

Funktionen von TypeScript:

- TypeScript wird zu normalen JavaScript konvertiert, da TypeScript-Code nicht von Browsern interpretiert werden kann. Durch diese Transpilieren genannte Konvertierung kann TypeScript-Code von Browsern angezeigt werden.
- JavaScript kann durch das ändern der Dateiendung von .js auf .ts direkt auf TypeScript konvertiert werden.
- TypeScript kann in jeder Umgebung, Browser, oder Betriebssystem benutzt werden
- JavaScript-Bibliotheken können ohne Probleme in TypeScript-Programmen benutzt werden [7]

3.3 Visual Studio Code

Visual Studio Code ist die IDE, welche für die Frontendentwicklung dieser Anwendung eingesetzt wird. Es ist ein mächtiger Code Editor welcher auf dem Desktop von Windows, Linux oder macOS läuft. In diesem Editor integriert ist Support für beispielsweise TypeScript und JavaScript, was aber über Extensions auf so gut wie jede häufig benutzte Programmiersprache (CSharp, C++, Java, ...) erweitert werden kann.

Die eingebaute IntelliSense stellt einem Funktionen für Autokomplettierung und Syntaxhervorhebung zur Verfügung, debuggen kann direkt in dem Editor durchgeführt werden. Git-Befehle sind ebenfalls in Visual Studio Code integriert, was die Versionskontrolle und Zusammenarbeit mit dem Projektteam erleichtert.

Außerdem kann Visual Studio Code durch verschiedene Extensions komplett angepasst werden. Die wichtigsten darunter sind zum Beispiel die oben genannten Extensions für andere Programmiersprachen, Extensions für Docker und Extensions für andere Themes. [8]

4 Umsetzung Frontend

4.1 Datenmodell

4.1.1 OEBBError

Ein Fehler beschreibt den Zustand einer Funktion, sollte diese einen gewissen Grenzwert über- oder unterschreiten.

Listing 1: OEBB-Error

```
export class OEBBError{
  constructor(
    public name: String,
    public date: String,
    public zustand: String
  ){}
}
```

4.1.2 OEBBFunktion

Eine Funktion ist jeweils eine spezifische Messung, welche in den QTX-Dateien abgespeichert ist.

Listing 2: OEBB-Funktion

```
export class OEBBFunktion{
  constructor(
    public id: number,
    public beschreibung: string,
    public headersize: number,
    public entries: number,
    public echtEntries: number,
    public numRead: number,
    public messergebnisse: OEBBMessergebnisse[]
  ){}
}
```

4.1.3 OEBBKriterien

Die Kriterien sind über den Zustand verschiedenen Funktionen zugeordnet und bilden die Werte, welche nicht über- oder unterschritten werden dürfen.

Listing 3: OEBB-Kriterien

```
export class OEBBKriterien{
  constructor(
    public name: string,
    public angesprochen: number,
    public minWert: number,
    public maxWert: number,
    public istWert: number,
    public resultCode: number,
    public maskMin: number,
    public maskMax: number,
    public messCount: number
  ){}
}
```

4.1.4 OEBBMessergebenisse

Die Messergebnisse sind die spezifischen Messungen, welche jeweils zu einer Funktion zugeordnet sind.

Listing 4: OEBB-Messergebnisse

```
export class OEBBMessergebnisse{
  constructor(
    public id: number,
    public number: number,
    public yWert: number,
    public mittel: number,
    public max: number,
    public min: number
  ){}
}
```

4.2 Service

Hier werden die Funktionen und Kriterien der verschiedenen QTX-Dateien gespeichert, um sie dem Rest der Anwendung zur Verfügung zu stellen. Über die Hilfsmethoden `setFunctions` und `setKriterien` werden die Daten im Service gespeichert und

mit den Hilfsmethoden `getFunctions` und `getKriterien` können die Daten wieder abgerufen werden.

Listing 5: OEBB-Service

```
export class OebbService {
  private functions: OEBBFunktion[];
  private kriterien: OEBBKriterien[];

  constructor() {
    this.functions = [];
    this.kriterien = [];
  }
  setFunctions(data: OEBBFunktion[]){
    this.functions = data;
  }
  getFunctions(){
    return this.functions;
  }
  setKriterien(data: OEBBKriterien[]){
    this.kriterien = data;
  }
  getKriterien(){
    return this.kriterien;
  }
}
```

4.3 Auswählen der Datei

Hier wird die Datei ausgewählt, welche in der Visualisierungs-Komponenete angezeigt werden soll. Zuerst wurde versucht, die Datei über einen file-input auszuwählen und den Namen dieser Datei dann über einen HTTP-Request an das Backend zu schicken, wo die Daten dieser Datei ausgewertet und in Funktionen und Kriterien aufgeteilt zurückgeschickt werden.

Listing 6: Fileinput

```
    <input type="file" class="file-input"
       (change)="onFileSelected($event)">
```

Dieser Fileinput öffnet ein Explorer-Fenster, wo dann die Datei ausgewählt werden kann.

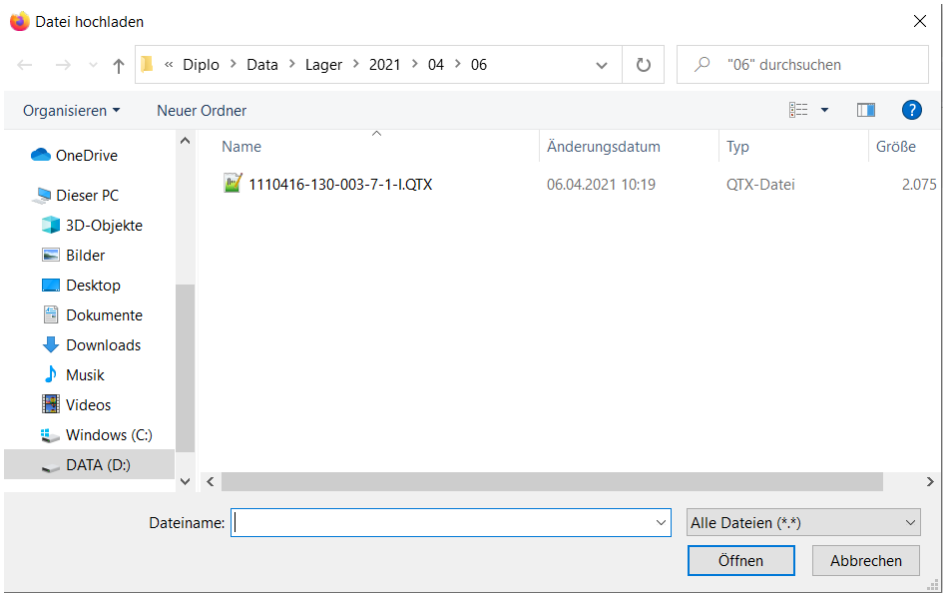


Abbildung 9: File-Explorer

Da die Dateien, mit denen gearbeitet wird, häufig über die verschiedenen Zustände hinweg oft die gleichen Namen haben und das nicht aus den Dateien selbst herauslesbar ist, wurde als Lösung eine Funktion implementiert, mit welcher vor dem Fileinput ein Zustand ausgewählt wird.

Mögliche Zustände:

- Auslauf / Auslauf 2: Hier wird getestet, wie sich der Antrieb beim Auslaufen lassen verhält
- Lager: Hier wird das Radwellenlager, das Bremswellenlager und das Motorwellenlager des Antriebs getestet
- Schlag: Hier wird gemessen, ob der Antrieb rund läuft
- Warmlauf: Ein Testlauf nachdem der Antrieb warmgelaufen ist
- Wuchtlauf: Hier wird der Antrieb auf die Unwucht gemessen

Listing 7: Checkboxen zum Auswählen des Zustandes

```
1 <section class="example-section">
2   <mat-checkbox class="example-margin" [disabled]="disabled"
      (change)="onAuslaufSelected()">Auslauf</mat-checkbox>  
3   <mat-checkbox class="example-margin" [disabled]="disabled"
      (change)="onAuslauf2Selected()">Auslauf
      2</mat-checkbox>  
4   <mat-checkbox class="example-margin" [disabled]="disabled"
      (change)="onLagerSelected()">Lager</mat-checkbox>  
```

```
5    <mat-checkbox class="example-margin" [disabled]="disabled"
        (change)="onSchlagSelected()">Schlag</mat-checkbox>  
6    <mat-checkbox class="example-margin" [disabled]="disabled"
        (change)="onWarmSelected()">Warmlauf</mat-checkbox>  
7    <mat-checkbox class="example-margin" [disabled]="disabled"
        (change)="onWuchtSelected()">Wuchtlauf</mat-checkbox>
8  </section>
```

Danach wurde der Request erweitert, um neben dem Dateinamen auch den ausgewählten Zustand als Parameter mitzuschicken.

Listing 8: Http-Request für Funktionen und Kriterien

```
1      const file: File = event.target.files[0];
2
3      const f : OEBBFunktion[] = await this.httpClient
4        .get<OEBBFunktion[]>('https://localhost:5001/' +
            this.zustand + '/' + file.name)
5        .toPromise();
6      const k : OEBBKriterien[] = await this.httpClient
7        .get<OEBBKriterien[]>('https://localhost:5001/criteria/'
            + this.zustand + '/' + file.name)
8        .toPromise();
9
10     this.service.setFunctions(f);
11     this.service.setKriterien(k);
```

Die Funktionen und Kriterien, die der HTTP-Request zurückliefert, werden in dem Service gespeichert, sodass im Rest der Anwendung darauf zugegriffen werden kann [4.2].

In den verschiedenen `onZustandSelected` Methoden wird jeweils der ausgewählte Zustand gesetzt und die Checkboxen deaktiviert, um immer nur einen Zustand auswählen zu können.

Listing 9: Eine der onZustandSelect-Methoden

```
1      onAuslaufSelected(): void {
2      this.disabled = true;
3      this.zustand = 'Auslauf';
4    }
```

Die Weiterleitung auf die Visualisierungskomponente wird über ein Switch-Case-Statement durchgeführt. Je nach Anzahl der Kriterien, die vom Backend zurückgeschickt werden, wird der Benutzer auf die richtige Komponente weitergeleitet. Die Weiterleitung selbst passiert über einen Router, welcher die verschiedenen Komponenten miteinander verbindet.

Listing 10: Switch-Case für die Weiterleitung zur Visualisierung

```
switch(k.length){
  case 27:
    this.router.navigate(['show-function']);
    break;
  case 2:
    this.router.navigate(['show-function-auslauf']);
    break;
  case 15:
    this.router.navigate(['show-function-schlaglauf']);
    break;
  case 8:
    this.router.navigate(['show-function-warmlauf']);
    break;
  case 7:
    this.router.navigate(['show-function-wuchtlauf']);
    break;
}
```

Die fertige Komponente für das Auswählen der Datei sieht folgendermaßen aus (10):

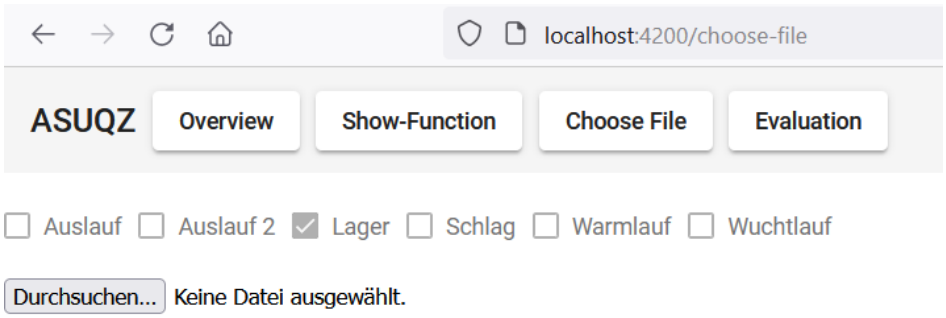


Abbildung 10: Fertige Komponente zum Dateien auswählen

Hier wurde bereits der Zustand "Lager"ausgewählt, folgendermaßen wird nun eine Lagerdatei aus dem Explorer ausgewählt und der Benutzer wird auf die Visualisierungskomponente des Lagerzustandes weitergeleitet.

4.4 Visualisierungskomponente

Hier werden die Funktionen der Datei visualisiert, welche zuvor in der Auswahlskomponente [4.3] ausgewählt wurde. Zur Visualisierung selbst wird die JavaScript-Bibliothek ng2-charts benutzt, welche auf der chart.js-Bibliothek basiert.

4.4.1 Initialisierung

Beim Initialisieren der Komponente werden zuerst die Funktionen und Kriterien aus dem Service geladen, um sie innerhalb der Komponente nur einmal aus dem Service abrufen zu müssen.

Listing 11: Initialisierung der Komponente

```
async ngOnInit(): Promise<void> {
  this.functions = this.service.getFunctions();
  this.kriterien = this.service.getKriterien();
}
```

Danach werden die Grundoptionen des Diagramms gesetzt, wo später die Daten dargestellt werden.

Listing 12: Grundoptionen des Diagramms

```
dataPoints: ChartDataSets[] = [];
labels: Label[] = [];
options = {
  responsive: false,
  scales:  {
  }
};
colors: Color[] = [
  {
    borderColor: 'black',
    backgroundColor: undefined
  },
];
legend = true;
plugins = [];
type: ChartType = 'line';
```

Die dataPoints sind die Punkte, wodurch das Diagramm aufgebaut wird. Da zum Initialisierungszeitpunkt noch keine Daten vorhanden sind, bleiben diese vorerst leer, ebenso wie die labels, welche die Namen der verschiedenen Graphen darstellen. Beide werden später in einer eigenen Methode befüllt. Mit `options responsive: false` wird das Verhalten der Größenänderung bei verschiedenen Fenstergrößen ausgestellt. Da diese Anwendung nur auf dem immer gleichen Rechner laufen muss, ist diese Option in diesem Fall unnötig. Mit `borderColour: 'black'` wird die Farbe des Graphengerüsts auf schwarz gestellt und mit `backgroundColour: 'undefined'` wird die Farbe, mit der die Graphen normalerweise hinterlegt werden, ausgestellt. `legend = true` bedeutet, dass die DataSets, welche in dem Graph benutzt werden, angezeigt werden. Mit `type:`

`ChartType = 'line'` wird dem Diagramm mitgeteilt, welche Art von Diagramm es ist, in diesem Fall ein Linien-Diagramm.

Der HTML-Code des Diagramms sieht folgendermaßen aus:

Listing 13: HTML für das Diagramm

```
<div id="one" class="chartjs-block">
  <canvas baseChart height="600" width="1500"
      [datasets]="dataPoints"
      [labels]="labels"
      [options]="options"
      [colors]="colors"
      [legend]="legend"
      [chartType]="type"
      [plugins]="plugins">
  </canvas>
</div>
```

4.4.2 Auswählen der Funktion

In einem Dropdown-Menü kann die gewünschte Funktion ausgewählt werden, welche der Benutzer sich ansehen möchte. Sobald eine Funktion ausgewählt wurde, wird die Methode `onSelect` aufgerufen, welche die Visualisierung durchführt.

Listing 14: Dropdown-Menü für die Funktionen

```
<div id="select">
  <mat-form-field appearance="fill" (change)="onSelect()">
    <select matNativeControl [(ngModel)]="selectedFunction"
       name="f">
      <option value="" selected></option>
      <option *ngFor="let f of functions"
         [value]="f.beschreibung">{{f.beschreibung}}</option>
    </select>
  </mat-form-field>
</div>
```

4.4.3 Darstellen der Funktion

Wie bereits beschrieben wird zur Darstellung der Daten ng2-charts benutzt und die Visualisierung wird in der Methode `onSelect` durchgeführt, welche in drei Teile aufgeteilt werden kann.

Reset und Kriterien

Zuerst werden alle Daten und Kriterien gelöscht, um sicherzustellen, dass keine falschen Daten oder Kriterien von einem vorherigen Aufruf der Methode noch vorhanden sind.

Listing 15: Zurücksetzen der Daten

```
1       this.disabled = false;
2       this.dataMittel = [];
3       this.dataMax = [];
4       this.dataMin = [];
5       this.labels = [];
6       this.kriteriumOneMin = [];
7       this.kriteriumOneMax = [];
8       this.kriterium = [];
9       this.kriteriumTwoMin = [];
10      this.kriteriumTwoMax = [];
11      this.kriteriumThreeMin = [];
12      this.kriteriumThreeMax = [];
13      this.pushCount = 50;
```

Danach werden die richtigen Kriterien aus der Liste der Kriterien, die von dem Backend kommt, herausgelesen. Da in einer Datei mehrere Funktionen sind und diese nicht alle die gleichen Kriterien haben, sind in der Datei alle Kriterien zusammengefasst gespeichert. Die richtigen Kriterien werden über den Namen der Funktion herausgefiltert und in ein extriges Array von Kriterien gespeichert. Sollte der Name der ausgewählten Funktion auf keinen der Fälle zutreffen, werden in der Graphik keine Kriterien angezeigt. Dies wird zum Beispiel bei den Spektrums-Funktionen genutzt, da diese keine Kriterien benötigen (12).

Listing 16: Filtern der Kriterien

```
1       switch(f.beschreibung){
2         case 'LagerEnvelope(Bremswelle A)':
3           this.kriterium = [this.kriterien[1],
              this.kriterien[9], this.kriterien[19]];
4           break;
5         case 'LagerEnvelope(Bremswelle B)':
6           this.kriterium = [this.kriterien[5],
              this.kriterien[13], this.kriterien[23]];
7           break;
8         case 'LagerEnvelope(Motorwelle A)':
9           this.kriterium = [this.kriterien[2],
              this.kriterien[10], this.kriterien[20]];
10          break;
11        case 'LagerEnvelope(Motorwelle B)':
```

```
12          this.kriterium = [this.kriterien[6],
                this.kriterien[14], this.kriterien[24]];
13          break;
14        case 'LagerEnvelope(Radwelle A)':
15          this.kriterium = [this.kriterien[3],
                this.kriterien[11], this.kriterien[21]];
16          break;
17        case 'LagerEnvelope(Radwelle B)':
18          this.kriterium = [this.kriterien[7],
                this.kriterien[15], this.kriterien[25]];
19          break;
20        default:
21          this.kriterium = [];
22          break;
23      }
```

Befüllen der Daten

Listing 17: Befüllen der Daten

```
1       if(this.kriterium.length > 0){
2         for(let i = 0; i < 100; i++){
3           this.dataMittel.push(f.messergebnisse[i].mittel);
4           this.dataMax.push(f.messergebnisse[i].max);
5           this.dataMin.push(f.messergebnisse[i].min);
6           this.labels.push(f.messergebnisse[i].yWert.toString());
7           this.kriteriumOneMin.push(this.kriterium[0].minWert);
8           this.kriteriumOneMax.push(this.kriterium[0].maxWert);
9           this.kriteriumTwoMin.push(this.kriterium[1].minWert);
10          this.kriteriumTwoMax.push(this.kriterium[1].maxWert);
11          this.kriteriumThreeMin.push(this.kriterium[2].minWert);
12          this.kriteriumThreeMax.push(this.kriterium[2].maxWert);
13        }
14      }
15      else{
16        for(let i = 0; i < 100; i++){
17          console.log(f.messergebnisse)
18          this.dataMittel.push(f.messergebnisse[i].mittel);
19          this.dataMax.push(f.messergebnisse[i].max);
20          this.dataMin.push(f.messergebnisse[i].min);
21          this.labels.push(f.messergebnisse[i].yWert.toString());
22        }
23      }
```

Hier werden die jeweils ersten hundert Messergebnisse für den Mittelwert, den Maximalwert und den Minimalwert in die dafür vorgesehenen Hilfsarrays gefüllt. Sollte es sich bei der ausgewählten Funktion um eine jener Funktionen handeln, welche Kriterien benötigt, werden diese ebenfalls in die dafür vorgesehenen Hilfsarrays gefüllt.

Zeichnen der Graphik

Um die Graphik zeichnen zu können, müssen hier die dataPoints befüllt werden.

Listing 18: Befüllen der DataPoints

```
1       if(this.kriterium.length > 0){
2         this.dataPoints = [
3           {fill: false, data: this.kriteriumOneMin, label:
                this.kriterium[0].name + " Min"},
4           {fill: false, data: this.kriteriumOneMax, label:
                this.kriterium[0].name + " Max"},
5           {fill: false, data: this.kriteriumTwoMin, label:
                this.kriterium[1].name + " Min"},
6           {fill: false, data: this.kriteriumTwoMax, label:
                this.kriterium[1].name + " Max"},
7           {fill: false, data: this.kriteriumThreeMin, label:
                this.kriterium[2].name + " Min"},
8           {fill: false, data: this.kriteriumThreeMax, label:
                this.kriterium[2].name + " Max"},
9           {fill: false, data: this.dataMittel, label:
                f.beschreibung + " Mittel"},
10          {fill: false, data: this.dataMax, label:
                f.beschreibung + " Max"},
11          {fill: false, data: this.dataMin, label:
                f.beschreibung + " Min"}
12        ]
13      }
14      else{
15        this.dataPoints = [
16          {fill: false, data: this.dataMittel, label:
                f.beschreibung + " Mittel"},
17          {fill: false, data: this.dataMax, label:
                f.beschreibung + " Max"},
18          {fill: false, data: this.dataMin, label:
                f.beschreibung + " Min"}
19        ]
20      }
```

Die dataPoints sind ein Array von ChartDataSets und bestehen jeweils aus fill, data und label. Fill wird benutzt, um dem Graph zu sagen, ob er den Bereich unter dem Graphen einfärben soll, in diesem Fall wird dieser Bereich nicht eingefärbt. Data sind die jeweiligen Werte, aus denen der Graph gebildet wird und Label ist der Name des jeweiligen Graphen. Auch hier wird wieder unterschieden zwischen einer Funktion mit Kriterien und einer Funktion ohne Kriterien. Sollte die ausgewählte Funktion Kriterien besitzten, werden diese ebenfalls in dem Diagramm eingezeichnet.

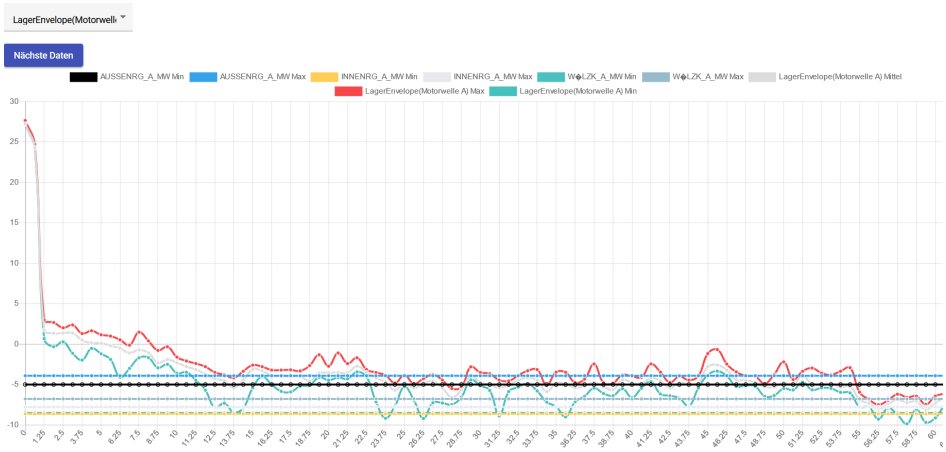


Abbildung 11: Beispiel einer Funktion mit Kriterien

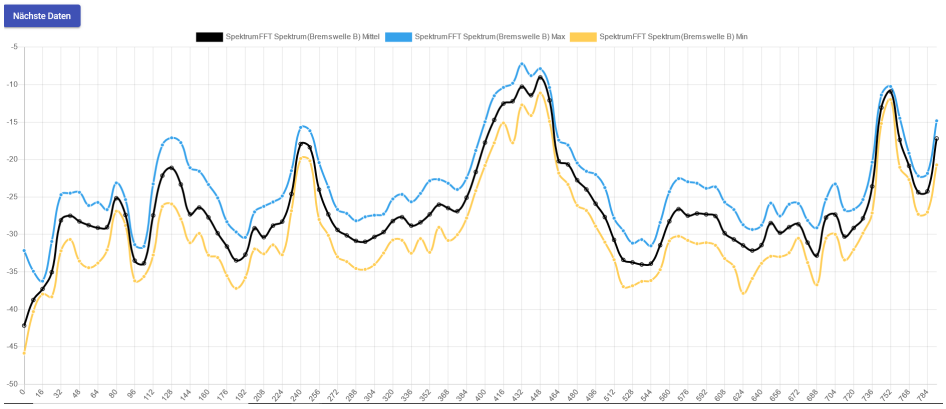


Abbildung 12: Beispiel einer Funktion ohne Kriterien

4.4.4 Push

Viele der Funktionen, die dargestellt werden, haben zu viele Messergebnisse und daher auch dataPoints, um sie übersichtlich in einem Diagramm darstellen zu können. Daher werden zuerst, wie oben beschrieben, nur die ersten hundert Werte geladen und über die folgende Methode push werden immer die nächsten 50 Werte angezeigt.

Listing 19: Push-Methode

```
1   public push(): void {
2     this.dataMittel = [];
3     this.dataMax = [];
4     this.dataMin = [];
5     this.labels = [];
6     let f: OEBBFunktion = new
          OEBBFunktion(0,'',0,0,0,0,this.messergebnisse);
7     this.functions.forEach(element =>
          {if(element.beschreibung === this.selectedFunction){f
          = element;}});
```

```
8       if(f.messergebnisse[this.pushCount+50] === undefined){
9         this.disabled = true;
10      }
11      for(let i = this.pushCount - 50; i < this.pushCount + 50;
            i++){
12        this.dataMittel.push(f.messergebnisse[i].mittel);
13        this.dataMax.push(f.messergebnisse[i].max);
14        this.dataMin.push(f.messergebnisse[i].min);
15        this.labels.push(f.messergebnisse[i].yWert.toString());
16      }
17
18      if(this.kriterium.length > 0){
19        this.dataPoints = [
20          {fill: false, data: this.kriteriumOneMin, label:
                this.kriterium[0].name + " Min"},
21          {fill: false, data: this.kriteriumOneMax, label:
                this.kriterium[0].name + " Max"},
22          {fill: false, data: this.kriteriumTwoMin, label:
                this.kriterium[1].name + " Min"},
23          {fill: false, data: this.kriteriumTwoMax, label:
                this.kriterium[1].name + " Max"},
24          {fill: false, data: this.kriteriumThreeMin, label:
                this.kriterium[2].name + " Min"},
25          {fill: false, data: this.kriteriumThreeMax, label:
                this.kriterium[2].name + " Max"},
26          {fill: false, data: this.dataMittel, label:
                f.beschreibung + " Mittel"},
27          {fill: false, data: this.dataMax, label:
                f.beschreibung + " Max"},
28          {fill: false, data: this.dataMin, label:
                f.beschreibung + " Min"}
29        ]
30      } else{
31        this.dataPoints = [
32          {fill: false, data: this.dataMittel, label:
                f.beschreibung + " Mittel"},
33          {fill: false, data: this.dataMax, label:
                f.beschreibung + " Max"},
34          {fill: false, data: this.dataMin, label:
                f.beschreibung + " Min"}
35        ]
36      }
37      this.pushCount += 50;
38    }
```

Zuerst werden die dataPoints zurückgesetzt, um Platz für die neuen Werte zu schaffen. Da die Kriterien jeweils statische Werte sind, müssen diese nicht zurückgesetzt werden. Danach wird überprüft, ob es nach den derzeitigen Werten noch Werte gibt, sollte dies nicht der Fall sein, wird der Button, mit dem die Daten weitergeschaltet werden, deak-

tiviert. Dann werden die neuen Werte wieder in ihre jeweiligen Hilfsarrays gespeichert, ähnlich wie beim ersten Befüllen der Daten. Der pushCount ist hierbei ein Laufwert, über den die derzeitigen und nächsten Daten verwaltet werden. Hiernach werden die Daten wieder neu in die dataPoints gefüllt und der Graph neu gezeichnet.

4.5 Auswertungskomponente

Hier können alle Fehler nach Zustand, Jahr, Monat und Tag gefiltert und ausgegeben werden.

4.5.1 Auswahl der Kriterien

Die Kriterien, welche wie oben beschrieben aus Zustand, Jahr, Monat und Tag bestehen, werden jeweils einzeln aus den dafür vorgesehenen Dropdown-Menüs ausgewählt (13) und über Objekt-Binding in die Komponente übertragen. Das `[(ngModel)]="selectedZustand"` referenziert auf das gleichnamige Objekt in der Typescript-Datei.

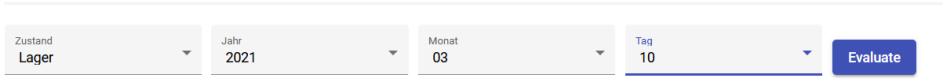


Abbildung 13: Kriterienauswahl

Listing 20: HTML-Code des Dropdown-Menüs des Zustandes

```
<mat-form-field appearance="fill">
  <mat-label>Zustand</mat-label>
  <select matNativeControl [(ngModel)]="selectedZustand"
     name="z">
    <option value="" selected></option>
    <option *ngFor="let z of zustand"
       [value]="z">{{z}}</option>
  </select>
</mat-form-field>
```

Listing 21: Binding-Objekte in der Typescript-Datei

```
selectedYear = null;
selectedMonth = null;
selectedDay = null;
```

Es kann eine beliebige Kombination aus Kriterien ausgewählt werden, beispielsweise Zustand und Jahr, nur der Zustand oder alle vier Kriterien gleichzeitig. Sofern zu den

ausgewählten Kriterien Fehler vorhanden sind, werden diese in einer Tabelle angezeigt, sollten keine Fehler vorhanden sein, wird eine Fehlernachricht ausgegeben.

Name	Datum	Zustand
RADWELLENLAGER_B 0 INNENRG_B_RW OR AUSSENRG_B_RW OR W�LZK_B_RW	20210408104339	LAGER
RADWELLENLAGER_A 0 INNENRG_A_RW OR AUSSENRG_A_RW OR W�LZK_A_RW	20210414084837	LAGER
RADWELLENLAGER_B 0 INNENRG_B_RW OR AUSSENRG_B_RW OR W�LZK_B_RW	20210420082623	LAGER
RADWELLENLAGER_A 0 INNENRG_A_RW OR AUSSENRG_A_RW OR W�LZK_A_RW	20210420143953	LAGER
RADWELLENLAGER_A 0 INNENRG_A_RW OR AUSSENRG_A_RW OR W�LZK_A_RW	20210421134516	LAGER
RADWELLENLAGER_A 0 INNENRG_A_RW OR AUSSENRG_A_RW OR W�LZK_A_RW	20210421135934	LAGER
BREMSWELLENLAGER_B 0 INNENRG_B_BW OR AUSSENRG_B_BW OR W�LZK_B_BW	20210422134256	LAGER
BREMSWELLENLAGER_B 0 INNENRG_B_BW OR AUSSENRG_B_BW OR W�LZK_B_BW	20210422141805	LAGER
RADWELLENLAGER_A 0 INNENRG_A_RW OR AUSSENRG_A_RW OR W�LZK_A_RW	20210422141805	LAGER
KEINE_MOTORDREHZAHL 1 ANTRIEBDZ_NOK	20210426111154	LAGER
KEINE_RADDREHZAHL 1 RADDZ_NOK	20210427131306	LAGER
KEINE_MOTORDREHZAHL 1 ANTRIEBDZ_NOK	20210427131306	LAGER
KEINE_RADDREHZAHL 1 RADDZ_NOK	20210428112607	LAGER
KEINE_MOTORDREHZAHL 1 ANTRIEBDZ_NOK	20210428112607	LAGER

Abbildung 14: Fehler gefiltert nach Kriterien

Name	Datum	Zustand
Es gibt keine Fehler für 2021 10 06		

Abbildung 15: Keine Fehler für die ausgewählten Kriterien

Die Tabelle, in der die Fehler ausgegeben werden, ist eine einfache HTML-Tabelle ohne TypeScript oder JavaScript. Über Objekt-Binding wird ein Objekt mit dem Namen errors in die Tabelle übergeben, welches mit einer einfachen ngFor durchiteriert und jeder einzelne Fehler im Objekt angezeigt wird.

Listing 22: HTML-Code der Fehlertabelle

```
<table>
  <tr>
    <th>Name</th>
    <th>Datum</th>
    <th>Zustand</th>
  </tr>
  <tr *ngFor="let e of errors">
    <td>{{e.name}}</td>
    <td>{{e.date}}</td>
    <td>{{e.zustand}}</td>
  </tr>
</table>
```

4.5.2 Mocking der Auswertungskomponente

Für Testzwecke wurden die gesamten Daten der Auswertungskomponente zuerst gemockt, um Fehler im Produktivbetrieb zu minimieren. Das Mocking selbst läuft über den ErrorMockingService, in welchem einige Fehler hartgecodet wurden.

Listing 23: Error-Mocking-Service

```
export class ErrorMockingService {
  private errors: OEBBError[];

  constructor() {
    this.errors = [
      {name: 'RADWELLENLAGER_B  0 INNENRG_B_RW OR
         AUSSENRG_B_RW OR WAELZK_B_RW', date: '2021-04-20',
         zustand: 'Lager'},
      {name: 'RADWELLENLAGER_B  0 INNENRG_B_RW OR
         AUSSENRG_B_RW OR WAELZK_B_RW', date: '2021-05-17',
         zustand: 'Lager'},
      {name: 'UNWUCHT_A_RW  0 UNWUCHT_VA', date:
         '2021-07-12', zustand: 'Auslauf'},
      {name: 'MOTORWELLENLAGER_B  0 INNENRG_B_MW OR
         AUSSENRG_B_MW OR WAELZK_B_MW', date: '2021-04-19',
         zustand: 'Auslauf 2'},
      {name: 'RAD_A 0 PLANLAUF_A_MAX OR RUNDLAUF_A_MAX',
         date: '2021-03-01', zustand: 'Schlag'},
      {name: 'BREMSWELLENLAGER_B  0 INNENRG_B_BW OR
         AUSSENRG_B_BW OR WAELZK_B_BW', date: '2021-06-09',
         zustand: 'Warmlauf'},
      {name: 'BREMSWELLENLAGER_A  0 INNENRG_A_BW OR
         AUSSENRG_A_BW OR WAELZK_A_BW', date: '2021-05-25',
         zustand: 'Wuchtlauf'},
    ];
  }

  getErrors(zustandParam: String, dateParam?: String){
    if(dateParam == undefined){
      return this.errors.filter(x => x.zustand ==
         zustandParam);
    }
    return this.errors.filter(x => x.zustand == zustandParam
       && x.date == dateParam)
  }
}
```

Zuerst wird ein Array aus OEBBErrors angelegt, welches im Konstruktor befüllt wird. Die Funktionsnamen und Zustände wurden hierfür aus dem Produktivbetrieb ausgewählt, die Daten wurden zufällig zugeteilt. In der Funktion `getErrors` wird dieses Array zuerst über die mitgelieferten Parameter gefiltert und danach werden die gefilterten Fehler in die Auswertungskomponente zurückgegeben.

Listing 24: GetEvaluationMocking-Methode

```
  getEvaluationMocking(){
    this.errors = [];
```

```
3       let date = undefined;
4       if(this.selectedDay != null && this.selectedMonth != null
           && this.selectedYear != null){
5         date = this.selectedYear! + this.selectedMonth! + '-' +
             this.selectedDay;
6       }
7       this.errors =
           this.mockingService.getErrors(this.selectedZustand,
           date);
8       this.selectedDay = null;
9       this.selectedMonth = null;
10      this.selectedYear = null;
11    }
```

In der Methode `getEvaluationMocking` werden zuerst die errors zurückgesetzt, um die Tabelle zu leeren. Danach werden die ausgewählten Teile des Datums zu einem gesamten String zusammengesetzt. Mit diesem und dem ausgewählten Zustand wird dann der ErrorMockingService aufgerufen und die zurückkommenden Fehler werden in das Error-Objekt gespeichert, welches über Objekt-Binding in der Tabelle angezeigt wird. Danach werden die Dropdown-Menüs, welche für das Filtern zuständig sind, wieder zurückgesetzt.

4.5.3 Befüllen der Auswertungskomponente über das Backend

Um die Fehler anzeigen zu können, welche real im Betrieb vorkommen, müssen diese zuerst im Backend ausgewertet werden. Diese ausgewerteten Fehler werden dann über einen Http-Client mit einer einfachen GET-Methode in das Frontend übertragen. Zuerst wird das error-Objekt, welches über Objekt-Binding mit der Tabelle verbunden ist, zurückgesetzt, um, ähnlich wie beim Mocken der Komponente, die Tabelle zu leeren. Danach wird eine von vier GET-Methoden aufgerufen, je nachdem welche der vier Kriterien ausgewählt wurden.

Listing 25: GET-Methoden zum Zugriff auf das Backend

```
1       if(this.selectedYear == null){
2         let x = await
             this.httpClient.get<OEBBError[]>('https://localhost:5001/'
3          + 'api/Function/GetErrors/errors/' +
              this.selectedZustand)
4         .toPromise()
5         .catch((err: HttpErrorResponse) => {
6           return [];
7         });
8         this.errors = x!;
```

```
9       }
10      else if(this.selectedMonth == null){
11        let x = await
             this.httpClient.get<OEBBError[]>('https://localhost:5001/'
12         + 'api/Function/GetErrors/errors/' +
              this.selectedZustand
13         + '/' + this.selectedYear)
14         .toPromise()
15         .catch((err: HttpErrorResponse) => {
16           return []
17         });
18         this.errors = x!;
19      }
20      else if(this.selectedDay == null){
21        let x = await
             this.httpClient.get<OEBBError[]>('https://localhost:5001/'
22         + 'api/Function/GetErrors/errors/' +
              this.selectedZustand
23         + '/' + this.selectedYear + '/' + this.selectedMonth)
24         .toPromise()
25         .catch((err: HttpErrorResponse) => {
26           return [];
27         });
28         this.errors = x!;
29      }
30      else{
31        let x = await
             this.httpClient.get<OEBBError[]>('https://localhost:5001/'
32        + 'api/Function/GetErrors/errors/' +
             this.selectedZustand
33        + '/' + this.selectedYear + '/' + this.selectedMonth +
             '/' + this.selectedDay)
34        .toPromise()
35        .catch((err: HttpErrorResponse) => {
36          return [];
37        });
38        this.errors = x!;
39      }
```

Sollten von diesem HTTP-GET keine Fehler zurückkommen, wird in das error-Objekt ein einzelner Fehler hineingespeichert, welcher aus der Nachricht "Es gibt keine Fehler für Zustand Jahr Monat Tag ", einem leeren Datum und einem leeren Zustand besteht. Danach werden die Dropdown-Menüs, welche für das Filtern zuständig sind, wieder zurückgesetzt.

Listing 26: Keine Fehler und Zurücksetzen des Filter-Menüs

```
1       if(this.errors.length == 0){
```

```
2         this.errors = [{name: 'Es gibt keine Fehler fuer ' +
              this.selectedZustand + ' ' + this.selectedYear + ' '
3         + this.selectedMonth + ' ' + this.selectedDay, date:
              "", zustand: ""}]
4       }
5
6       this.selectedDay = null;
7       this.selectedMonth = null;
8       this.selectedYear = null;
9       this.selectedZustand = "";
```

4.6 Overviewkomponente

In der Overviewkomponente wird die Prüfungsübersicht des alten Prüfprogramms des ÖBB Technischen Service Linz gemockt. Da das Erstellen eines komplett neuen Programms für die Prüfung und den Prüfungsablauf den Leistungsumfang dieser Arbeit überschritten hätte, wurde sich darauf geeinigt, nur diese Übersicht zu mocken.

4.6.1 Graphen

In der Overviewkomponente werden sechs verschiedene Graphen zu sechs verschiedenen Funktionen angezeigt, welche jeweils zu einer Prüfung gehören.

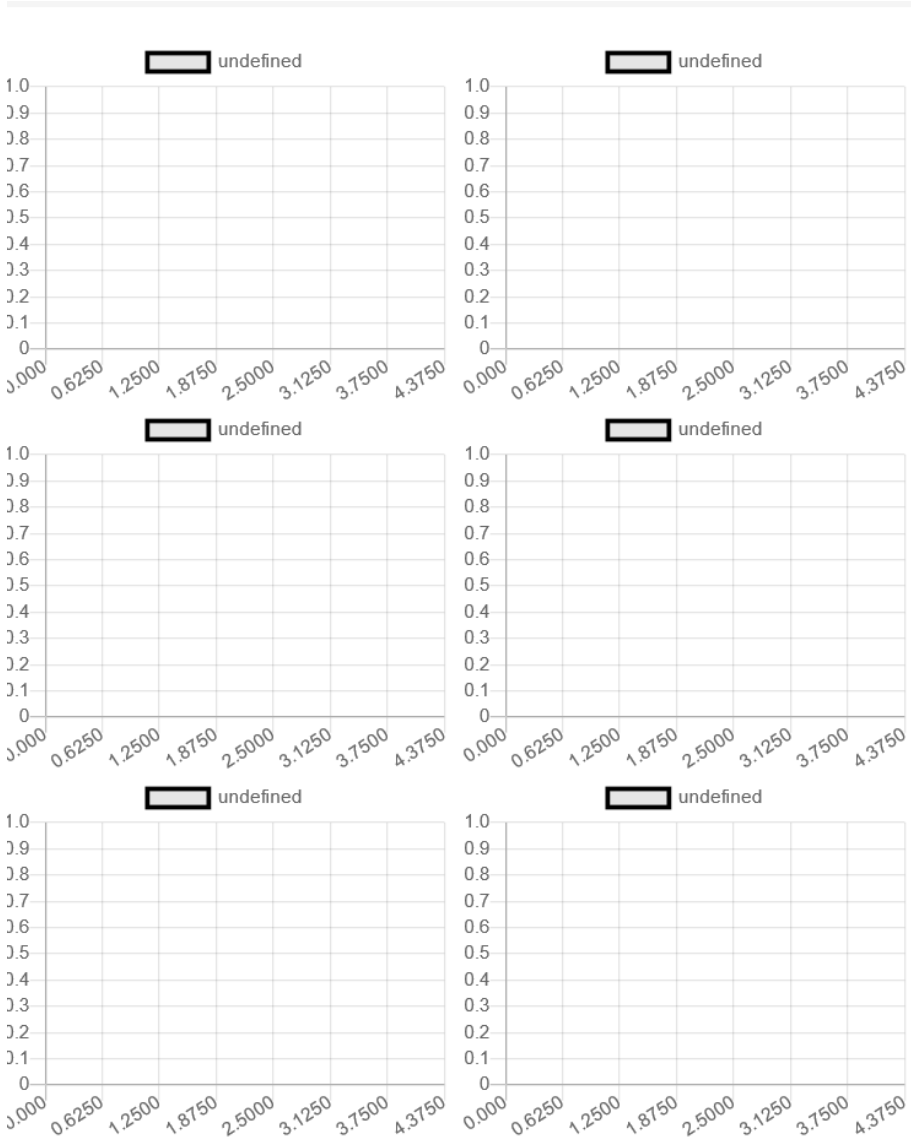


Abbildung 16: Funktionsgraphen ohne Initialisierung

Diese sechs Graphen werden zuerst ohne Daten nur mit ihren Labels erstellt und dann über die Methode `loadChartData` befüllt. LoadChartData wird über einen Button aufgerufen und mit einem weiteren Button können diese Graphen wieder geleert werden. DataPoints und Labels wurden bereits in der Visualiserungskomponente [4.4] erläutert.

Listing 27: Erstellen der Funktionsgraphen

```
1    dataPointsBWA: ChartDataSets[] = [];
```

```
2   BWALabels: Label[] = [ '0.000', '0.6250', '1.2500',
       '1.8750', '2.5000', '3.1250', '3.7500', '4.3750'];
3   dataPointsBWB: ChartDataSets[] = [];
4   BWBLabels: Label[] = [ '0.000', '0.6250', '1.2500',
       '1.8750', '2.5000', '3.1250', '3.7500', '4.3750'];
5   dataPointsMWA: ChartDataSets[] = [];
6   MWALabels: Label[] = [ '0.000', '0.6250', '1.2500',
       '1.8750', '2.5000', '3.1250', '3.7500', '4.3750'];
7   dataPointsMWB: ChartDataSets[] = [];
8   MWBLabels: Label[] = [ '0.000', '0.6250', '1.2500',
       '1.8750', '2.5000', '3.1250', '3.7500', '4.3750'];
9   dataPointsRWA: ChartDataSets[] = [];
10  RWALabels: Label[] = [ '0.000', '0.6250', '1.2500',
       '1.8750', '2.5000', '3.1250', '3.7500', '4.3750'];
11  dataPointsRWB: ChartDataSets[] = [];
12  RWBLabels: Label[] = [ '0.000', '0.6250', '1.2500',
       '1.8750', '2.5000', '3.1250', '3.7500', '4.3750'];
```

Listing 28: Befüllen und Leeren der Funktionsgraphen

```
1   loadChartData(){
2     this.dataPointsBWA = [{data: [-30.0431, -29.0684,
         -28.7432, -28.2064, -27.8274, -27.9592, -25.7714,
         -28.1736], label: 'Lager Bremswelle A'}]
3     this.dataPointsBWB = [{data: [-31.0431, -28.0684,
         -30.7432, -30.1064, -28.8274, -31.9592, -24.7714,
         -30.1736], label: 'Lager Bremswelle B'}]
4     this.dataPointsMWA = [{data: [-27.0431, -28.0684,
         -30.7432, -27.1064, -30.8274, -29.9592, -24.7714,
         -31.1736], label: 'Lager Motorwelle A'}]
5     this.dataPointsMWB = [{data: [-31.0431, -28.0684,
         -30.7432, -30.1064, -27.8274, -24.9592, -24.7714,
         -25.1736], label: 'Lager Motorwelle B'}]
6     this.dataPointsRWA = [{data: [-28.0431, -28.0684,
         -31.7432, -27.1064, -24.8274, -29.9592, -24.7714,
         -30.1736], label: 'Lager Radwelle A'}]
7     this.dataPointsRWB = [{data: [-31.0431, -28.0684,
         -30.7432, -30.1064, -27.8274, -30.9592, -28.7714,
         -25.1736], label: 'Lager Radwelle B'}]
8   }
9
10
11  clearCharts(){
12    this.dataPointsBWA = [{data: [], label: ''}]
13    this.dataPointsBWB = [{data: [], label: ''}]
14    this.dataPointsMWA = [{data: [], label: ''}]
15    this.dataPointsMWB = [{data: [], label: ''}]
16    this.dataPointsRWA = [{data: [], label: ''}]
17    this.dataPointsRWB = [{data: [], label: ''}]
18  }
```

Listing 29: HTML-Code eines Funktionsgraphen

```
<div id="one" class="chartjs-block">
  <canvas baseChart height="240" width="300"
      [datasets]="dataPointsBWA"
      [labels]="BWALabels"
      [options]="options"
      [colors]="colors"
      [legend]="legend"
      [chartType]="type"
      [plugins]="plugins">
  </canvas>
</div>
```

Der zuständige Prüfer kann über ein Dropdown-Menü ausgewählt werden, welches über Objekt-Binding mit dem selectedPruefer-Objekt der Typescript-Datei verbunden ist.

Listing 30: HTML-Code des Dropdown-Menüs für den Prüfer

```
  <mat-form-field appearance="fill">
    <select matNativeControl [(ngModel)]="selectedPruefer"
       name="pruefer">
      <option value="" selected></option>
      <option *ngFor="let pruefer of pruefers"
         [value]="pruefer.p_nachname + ' ' +
         pruefer.p_vorname">
        {{pruefer.p_nachname}}
      </option>
    </select>
  </mat-form-field>
  <br/>
  Pruefer:
    <ng-template *ngIf="selectedPruefer.p_nachname === ''then
       one; else two"></ng-template>
    <ng-template #one></ng-template>
    <ng-template #two>{{selectedPruefer}}</ng-template>
```

Der Button Speichern ruft die Methode `openDialog` auf, welche einen Dialog öffnet, in welchem ausgewählt werden kann, ob diese Prüfung frei gegeben werden kann oder nicht und ob es zusätzliche Kriterien gibt, auf welche man achten muss.

Listing 31: openDialog-Methode

```
  openDialog(): void{
    const dialogRef = this.dialog.open(DialogOverview, {
      width: '300',
      height: '500',
      data: {pruefer: this.selectedPruefer}
```

```
6       });
7
8       dialogRef.afterClosed().subscribe(result => {
9         console.log('Pruefung wurde gespeichert');
10      });
11    }
```

Mit `this.dialog.open` wird das Dialog-Fenster geöffnet, `width` und `heigth` geben die Dimensionen des Fensters an. Nachdem dieses Dialog-Fenster geschlossen wird, wird in der Konsole die Nachricht "Pruefung wurde gespeichert"ausgegeben.

Listing 32: TypeScript für DialogOverview

```
1  export class DialogOverview  {
2    panelOpenState = false;
3    kriterium = '';
4    kriterien: String[] = [];
5
6    constructor(
7      public dialogRef: MatDialogRef<DialogOverview>,
8      @Inject(MAT_DIALOG_DATA) public data: DialogData) {}
9
10   onNoClick(): void {
11     this.dialogRef.close();
12   }
13   save(): void{
14     this.kriterien = [];
15     this.dialogRef.close();
16   }
17   pushKriterium(): void{
18     this.kriterien.push(this.kriterium);
19     this.kriterium = '';
20   }
21 }
```

Listing 33: HTML-Code für DialogOverview

```
1  <h1 mat-dialog-title>Pruefer: {{data.pruefer}}</h1>
2  <div mat-dialog-content>
3    <mat-checkbox
       class="example-margin">Geprueft</mat-checkbox> 
4    <mat-checkbox class="example-margin">Geprueft und
       Frei</mat-checkbox>
5  </div>
6  <br/>
7  <div>
8    <mat-accordion>
9      <mat-expansion-panel hideToggle>
10       <mat-expansion-panel-header>
11         <mat-panel-title>
```

```
12            Zusaetzliche Kriterien/Fehler
13          </mat-panel-title>
14        </mat-expansion-panel-header>
15        <mat-list>
16          <mat-list-item id="listitems" *ngFor="let item of
               kriterien" >
17            <h3 matLine>{{item}}</h3>
18          </mat-list-item>
19        </mat-list>
20        <mat-form-field class="example-full-width"
             appearance="fill">
21          <input matInput value="" [(ngModel)]="kriterium">
22        </mat-form-field>
23        <button mat-icon-button (click)="pushKriterium()">
24          <mat-icon>add</mat-icon>
25        </button>
26      </mat-expansion-panel>
27    </mat-accordion>
28  </div>
29  <br/>
30  <div mat-dialog-actions>
31    <button mat-button (click)="onNoClick()">Zurueck</button>
32    <button mat-button (click)="save()">Speichern</button>
33  </div>
```

Über eine Checkbox kann der Status der Prüfung ausgewählt werden, geprüft oder geprüft und frei. Sollte es zusätzliche Kriterien geben, können diese in einem einfachen Textfeld eingegeben werden und über klicken auf den Plus-Button wird die Methode `pushKriterium` aufgerufen. Diese Methode fügt mit `this.kriterien.push` das eingegebene Kriterium in ein vorher angelegtes Array von Kriterien hinzu und leert das Textfeld dann wieder, um ein neues Kriterium angeben zu können. Mit einem Klick auf den Zurück-Button wird die Methode `onNoClick` aufgerufen, welche den Dialog wieder schließt und mit einem Klick auf den Speichern-Button wird die Methode `save` aufgerufen, welche den Dialog ebenfalls nur schließt, da dieses Mockup an keine Datenbank angebunden ist.

5 Zukünftige Erweiterungsmöglichkeiten

5.1 Zusätzliche Filtermöglichkeiten

Derzeit können Fehler in dieser Applikation nur über das Datum und den Zustand gefiltert werden. Um eine bessere Funktionalität der Auswertung zu gewährleisten, können noch weitere Filtermöglichkeiten eingebaut werden. Beispiele dafür wären verschiedene Kombinationen von Zuständen oder das Filtern nach spezifischen Kriterien [4.1.3].

5.2 Anbindung an das Produktivprogramm

Um eine Anbindung an das Produktivprogramm zu ermöglichen, wäre eine Schnittstelle mit einem Input-Module von Brüel und Kjaer notwendig. Durch diese Schnittstelle wäre es möglich, die Daten direkt während dem Prüfen eines Antriebes aus dem Prüflauf herauszulesen und auszuwerten. Dadurch würde das vorherige Auswerten und Abspeichern der Daten überflüssig werden, was derzeit von einem externen Programm durchgeführt wird.

Literaturverzeichnis

[1] „React vs Angular vs Vue.js — What Is the Best Choice in 2020?" article, 2021, JavaScript frameworks are developing at an extremely fast pace, meaning that today we have frequently updated versions of Angular, React.js, and another player on this market - Vue.js. Let's have a look at the demand represented in Google Trends for the last 5 years. The blue, red, and yellow lines represent Angular, React, and Vue.js respectively. Online verfügbar: https://www.techmagic.co/blog/reactjs-vs-angular-vs-vuejs-what-to-choose-in-2020/

[2] S. Eichenberger, „Webapps mit Angular entwickeln," article, 2020, Angular wird oft in einem Atemzug mit React oder Vue.js genannt. Das Google-Framework wird gerne für komplexe Webapps, Progressive Webapps oder Single-Page-Applikationen eingesetzt. Mit seinen Stärken und Schwächen starten wir eine Serie über typische Probleme und -Lösungen im UI Development mit Angular . Online verfügbar: https://zeix.com/durchdacht/2019/11/12/webapps-mit-angular-entwickeln/

[3] S. Thattil, „Vorteile und Nachteile von AngularJS," article, 2016, Was ist dieses Framework? Das Framework wurde von Misko Hevery im Jahr 2009 begründet. Die anfängliche Idee war es, Webdesigner dabei zu unterstützen, ein wenig mehr HTML in deren Code unterzubringen, so dass auch kleine statische Seiten, mehr Funktionalitäten bedienen können. Zum Beispiel eine kleine Pizzaladen-Website, welches ein Pizza-Bestellsystem über einfache HTML Tags einfügen kann. Online verfügbar: https://www.yuhiro.de/vorteile-und-nachteile-von-angularjs/

[4] „Angular 8 - Architecture," article, The core of the Angular framework architecture is Angular Component. Angular Component is the building block of every Angular application. Every angular application is made up of one more Angular Component. It is basically a plain JavaScript / Typescript class along with a HTML template and an associated name. Online verfügbar: https://www.tutorialspoint.com/angular8/angular8_architecture.htm

[5] R. Böhm, „Angular-Tutorial für Einsteiger," article, 2020, Dieses Tutorial erklärt euch die Grundlagen des Frameworks Angular. Wir behandeln hierbei Angular in der Version 2 und höher. Bewusst wird hierbei aber die Versionsnummer weggelassen, da das Framework nun semantische Versionierung benutzt. Kurz gesagt: Es ist einfach Angular. Online verfügbar: https://angular.de/artikel/angular-tutorial-deutsch/

[6] „TypeScript for the New Programmer," article, 2022, You have probably already heard that TypeScript is a "flavor" or "variant" of JavaScript. The relationship between TypeScript (TS) and JavaScript (JS) is rather unique among modern programming languages, so learning more about this relationship will help you understand how TypeScript adds to JavaScript. Online verfügbar: https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html

[7] B. K. Dubey, „Difference between TypeScript and JavaScript," article, 2022, When JavaScript was developed, the JavaScript development team introduced JavaScript

as a client-side programming language. But as people were using JavaScript, developers also realized that JavaScript could be used as a server-side programming language. However, as JavaScript was growing, JavaScript code became complex and heavy. Because of this, JavaScript wasn't even able to fulfill the requirement of an Object-Oriented Programming language. This prevented JavaScript from succeeding at the enterprise level as a server-side technology. So TypeScript was created by the development team to bridge this gap. Online verfügbar: https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/

[8] „Visual Studio Code," article, Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, Csharp, Java, Python, PHP, Go) and runtimes (such as .NET and Unity). Begin your journey with VS Code with these introductory videos. Online verfügbar: https://code.visualstudio.com/docs

Abbildungsverzeichnis

Quellcodeverzeichnis