Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Medieninformatik

Bachelorarbeit

von

Jonas **Schimmer**

API-First bei der Überführung verschiedener Systeme eines Industrie 4.0 Forschungsprojektes in eine neue MEVN-Webanwendung

API-First in the transfer of different systems of an Industry 4.0 research project into a new MEVN web application

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Studiengang Medieninformatik

Bachelorarbeit

von

Jonas **Schimmer**

API-First bei der Überführung verschiedener Systeme eines Industrie 4.0 Forschungsprojektes in eine neue MEVN-Webanwendung

API-First in the transfer of different systems of an Industry 4.0 research project into a new MEVN web application

Bearbeitungszeitraum: von 4. Dezember 2020
bis 3. Mai 2021

1. Prüfer: Prof. Dr. Dieter Meiller

2. Prüfer: M. Eng. Veit Stephan

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Ostbayerische Technische Hochschule
Amberg-Weiden

Bestätigung gemäß § 12 APO

Name und Vorname
der Studentin/des Studenten: **Schimmer, Jonas**

Studiengang: **Medieninformatik**

Ich bestätige, dass ich die Bachelorarbeit mit dem Titel:

API-First bei der Überführung verschiedener Systeme eines Industrie 4.0 Forschungsprojektes in eine neue MEVN-Webanwendung

selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: 30. April 2021

Unterschrift: 

Ostbayerische Technische Hochschule Amberg-Weiden
Fakultät Elektrotechnik, Medien und Informatik

Ostbayerische Technische Hochschule
Amberg-Weiden

Bachelorarbeit Zusammenfassung

Studentin/Student (Name, Vorname): **Schimmer, Jonas**
Studiengang: Medieninformatik
Aufgabensteller, Professor: Prof. Dr. Dieter Meiller
Durchgeführt in (Firma/Behörde/Hochschule): /
Betreuer in Firma/Behörde: /
Ausgabedatum: 4. Dezember 2020 Abgabedatum: 30. April 2021

Titel:

API-First bei der Überführung verschiedener Systeme eines Industrie 4.0 Forschungsprojektes in eine neue MEVN-Webanwendung

---

Zusammenfassung:

Etablierung eines API-First Ansatzes bei der Überführung und Zusammenfassung mehrerer Systeme in eine neue Full-Stack Webanwendung. Anschließende Evaluierung der Auswirkungen des API-First Ansatzes auf das Softwareprojekt.

Schlüsselwörter: API-First, API, REST, MEVN

Inhaltsverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
REST	Representational State Transfer
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
CRUD	Create, Read, Update, Delete
MEVN	MongoDB, Express.js, Vue.js, Node.js
OAS	OpenAPI Specification
RAML	RESTful API Modeling Language
NoSQL	Not only SQL

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Kapitel 1

Einführung

Im Rahmen des ISAC@OTH-AW Industrie 4.0 Forschungsprojektes wird eine Webanwendung entwickelt, welche ein Expertensystem zur Bewertung von Fertigungsverfahren bereitstellen soll und sich an kleine und mittelständische Unternehmen richtet (Gerlang, 2019a, 2019b). Über die Jahre wurden mehrere Anwendungsteile getrennt voneinander entwickelt, wobei jeweils unterschiedliche Technologien Einsatz gefunden haben. Aufgrund dieser Teilung und der vorhandenen Programmfehler ist die fortführende Entwicklung dieser Anwendung, vor allem für neue Mitarbeiter, stark erschwert.

Im Zuge der Modernisierung der Anwendung sollen die Anwendungsteile in ein neues System überführt werden, wobei ein einheitlicher Softwarestack eingeführt werden soll und die Programmierschnittstelle (engl. Application Programming Interface, kurz API) des Systems im Vordergrund stehen soll. Demnach wird die Implementierung mit einem API-First Ansatz durchgeführt und im Rahmen der Bachelorarbeit folgender Fragestellung nachgegangen:

Wie wird der API-First Ansatz im Zuge des Überführungsprozesses angewandt und welche Auswirkungen hat die Etablierung des Ansatzes auf das Softwareprojekt?

Diese Fragestellung kann wie folgt untergliedert werden:

- Wie wird der Überführungsprozess aufgebaut und wie gliedert sich der API-First Ansatz in diesen Prozess ein? Was gibt es dabei im Bezug auf diesen Entwicklungsansatz zu beachten?
- Welche Auswirkungen hat der API-First Ansatz auf die Überführung und auf das Projekt?

Ziel der Arbeit ist es, die neue Implementierung der Anwendung so durchzuführen, dass die fortführende Entwicklung und die spätere Auslieferung des Systems vereinfacht wird. Die Einführung eines einheitlichen Softwarestacks und des API-First Ansatzes soll zu einer besseren Wartung und Weiterentwicklung des Projektes führen. Darüber hinaus soll der API-First Ansatz dabei helfen, Funktionen der Anwendung für andere Produkte bereitzustellen und dabei den Lernaufwand für neue

API-Konsumenten zu verringern.

Zu Beginn der Arbeit wird die Ausgangslage geschildert. Die allgemeinen Anforderungen des ISAC-Projektes werden dargestellt und die bisherigen Entwicklungsstände der Projektteile werden beschrieben. Anschließend werden theoretische Grundlagen erklärt, wobei das Prinzip des API-First Ansatzes und dessen wichtigste Bausteine aufgezeigt werden. Danach folgt der Hauptteil, welcher die Entwicklung des Systems mit API-First Ansatz beinhaltet. Zu Beginn dieses Teils wird sich mit der Etablierung eines einheitlichen Softwarestacks beschäftigt. Im Anschluss wird die Überführungsstrategie aufgestellt und auf jeden Anwendungsteil angewandt. Zuerst wird jeweils die bestehende Anwendung und deren Funktionalität analysiert und den neuen Anforderungen gegenübergestellt. Ebenso wird das bestehende Datenmodell analysiert. Anschließend folgt der Entwurf der API. Hierbei entsteht die API-Spezifikation, welche ein zentrales Element beim API-First Ansatz ist. Daraufhin folgt der Entwurf des neuen Datenmodells. Zum Ende der Arbeit wird das Backend implementiert und mittels Contract-Testing getestet, woraufhin das Frontend implementiert wird. Abschließend wird evaluiert, welchen Einfluss API-First auf das Projekt und dessen Wartbarkeit hat. Hierfür wird eine Analyse anhand von Qualitätskriterien für Softwareprojekte nach der Norm ISO/IEC 25010 durchgeführt. Daraufhin folgt im Fazit die Zusammenfassung der Forschungsergebnisse.

Kapitel 2

Grundlagen

Im folgenden Kapitel werden die zugrunde liegenden Rahmenbedingungen der Bachelorarbeit beschrieben. Zu Beginn wird die Ausgangslage des Projektes beschrieben, woraufhin die aktuellen Entwicklungsstände der Projektteile aufgezeigt werden. Im Anschluss werden die theoretischen Grundlagen erklärt.

2.1 Ausgangslage

Im folgenden Abschnitt wird die Ausgangslage geschildert. Zuerst werden die Rahmenbedingungen des Forschungsprojektes aufgezeigt. Hierbei werden die Anforderungen des Projektes sowie die aktuellen Entwicklungsstände der Anwendungsteile beschrieben. Im Anschluss wird dargestellt, aus welchen Gründen und mit welchen Zielen die Modernisierung der Anwendung in die Wege geleitet wurde.

2.1.1 Beschreibung des Projektes

Das Forschungsinteresse ist im Industrie 4.0 Forschungsprojekt ISAC@OTH-AW entstanden. Die dortige Arbeit zielt darauf ab, kleinen und mittelständischen Unternehmen die Vorzüge der Industrie 4.0 zugänglich zu machen (Gerlang, 2019a). Das Projekt ist in vier Teilprojekte aufgeteilt, welche sich verschiedenen Herausforderungen der Industrie 4.0 stellen. Im Teilprojekt 1 wird ein Expertensystem zur Bewertung von Fertigungsverfahren entwickelt (Gerlang, 2019b). Konkret gibt es hierbei eine Untergliederung in die drei Teilbereiche Subtraktive Fertigung, Additive Fertigung und Qualitätsprüfteile.

Im Bereich Subtraktive Fertigung soll die Effizienz und Wirtschaftlichkeit von Fräs- und Bohrprozessen bewertet werden, um mehrere Prozesse miteinander vergleichen zu können und um sich im konkreten Anwendungsfall für den am besten passenden Prozess entscheiden zu können. Hierfür soll der Benutzer Angaben zum Werkzeug, Material und weiteren Rahmenbedingungen tätigen, woraufhin die Anwendung die Benutzerangaben verarbeitet und daraus Kennzahlen für den Vergleich ableitet.

Im Bereich Additive Fertigung sollen ähnliche Berechnungen für additive Fertigungsverfahren durchgeführt werden. Durch Angaben wie Druckvolumen, Drucker und Material sollen Prognosen für die Herstellungskosten eines Bauteils aufgestellt werden. Der Benutzer kann diese Berechnungen für beliebige Kombinationen von Drucker und Material durchführen, um die für sich passenden Rahmenbedingungen für den Herstellungsprozess zu finden.

Im dritten und letzten Bereich des Projektes werden die von der Hochschule gedruckten Qualitätsprüfteile verwaltet. Das Qualitätsprüfteil ist ein druckbares, mit verschiedenen Merkmalen ausgestattetes, genormtes 3D-Druckteil, mit dessen Hilfe die Kostenstruktur von 3D-Druckern ermittelt und verglichen werden kann. Diese Prüfteile werden von Firmen in Auftrag gegeben und an der Hochschule gelagert. Hierfür soll ein Einlagerungssystem entwickelt werden, das den Prüfteilen einen eindeutigen Standort zuweist sowie den verwendeten Drucker und den Auftraggeber festhält.

2.1.2 Bisherige Entwicklungsstände der Projektteile

In den letzten zwei Jahren haben mehrere Projektmitarbeiter und studentische Hilfskräfte an dem Projekt gearbeitet. In diesem Zeitraum wurden einzelne Anwendungsteile getrennt voneinander entwickelt.

Der Anwendungsteil zur Berechnung der Bauteilkosten im Bereich Additive Fertigung wurde exemplarisch mit dem Content-Management-System TYPO3 erstellt. Hierbei gab es erste Versuche, das Volumen eines Druckteils automatisch durch den Import der dazugehörigen STL-Datei zu bestimmen und mit diesem im Anschluss die Herstellungskosten zu berechnen. Da die Langzeitunterstützung für die verwendete TYPO3-Version 8 eingestellt wurde, kann an dieser Stelle ohne aufwändiges Refactoring nicht weitergearbeitet werden.

Das Einlagerungssystem wurde mit dem Framework CakePHP und einer MySQL-Datenbank entwickelt. Es bietet die Möglichkeit, die Objekte Firmen, Drucker, Lagerorte, Boxen und Qualitätsprüfteile zu verwalten und miteinander zu verknüpfen. Das Einlagerungssystem weist im Vergleich zu den anderen Anwendungsteilen keine Programmfehler auf.

Parallel dazu gab es einen ersten Versuch zur Entwicklung des Expertensystems zur Bewertung der Zerspanungsprozesse. Hierbei wurden erste Implementierungsversuche mit Vanilla PHP durchgeführt und später mit dem Einlagerungssystem in CakePHP zusammengeführt. Bei der aktuellen Benutzung des Systems treten mehrere Fehler auf, die dazu führen, dass die gewünschte Funktionalität der Anwendung nicht vollständig gewährleistet ist.

Neben diesen Programmfehlern stellen die unterschiedlichen Softwaretechnologien, die in den Anwendungsteilen verwendet wurden, ein großes Problem dar. Die Einarbeitung in das Projekt wird für neue Mitarbeiter erschwert, da diese sich gegebenenfalls mit mehreren Softwarestacks vertraut machen müssen. Zusätzlich gewinnt das Projekt durch diese Aufteilung an verzichtbarer Komplexität, was dessen Wartung beeinträch-

tigen kann. Der ohne Anpassungen nicht weiter verwendbare TYPO3-Anwendungsteil spielt hierbei eine große Rolle. Durch den ausgelaufenen Support und die eingeschränkten Update-Möglichkeiten sowie teilweise anderen Anforderungen bietet sich hier eine neue Implementierung an. Die anderen beiden Anwendungsteile wurden mit dem Framework CakePHP entwickelt. Da der Bereich Zerspanungsprozesse zuvor mit Vanilla PHP geschrieben wurde und erst später in das CakePHP-Projekt des Prüfteileinlagerungssystem integriert worden ist, entstand hierbei eine unübersichtliche Projektstruktur. Hinzu kommt eine schlechte Wartbarkeit des Quellcodes, welche beispielsweise durch auskommentierte Code-Blöcke und nicht aussagekräftige Variablennamen verursacht wird. Die genannten Probleme beeinträchtigen die Wartbarkeit des gesamten Projektes auf lange Sicht.

Darüber hinaus soll die Bewertung von Zerspanungsprozessen mit einem anderen Ansatz durchgeführt werden. Bisher hat die Hochschule eigene Versuche im Labor durchgeführt, um eine genaue Bestimmung der Kennzahlen bei bestimmten Rahmenbedingungen vorzunehmen. Dies wurde experimentell erprobt. Es hat sich gezeigt, dass der Hochschule für eine breite Masse an benötigten Versuchen zur Bewertung verschiedener Fertigungsverfahren nicht genügend Kapazitäten zur Verfügung stehen. Demnach sollen weniger die eigenen Versuche die Grundlage für die Bewertung darstellen, sondern vielmehr die gebündelten Daten, die mit Hilfe von Herstellerangaben gesammelt werden.

Aus diesen Gründen wurde sich im Rahmen des Refactoring für eine Überführung der Projektteile in eine einzige Anwendung mit einheitlichem Softwarestack entschieden, wobei der Großteil des vorhandenen Quellcodes neu geschrieben werden muss.

2.2 Theoretische Grundlagen

Im folgenden Abschnitt werden die theoretischen Grundlagen erläutert. Zu Beginn wird das Konzept des Representational State Transfer (REST) definiert, woraufhin die theoretischen Grundlagen der API-Spezifikation und des API-First Ansatzes erläutert werden.

2.2.1 Definition einer REST-API

APIs ermöglichen die Kommunikation mit einem System. Solch ein Softwaresystem hinter einer API kann als Black-Box aufgefasst werden, deren genaue Funktionsweise für den Nutzer verborgen bleibt, um den Aufwand für die Nutzung der gekapselten Funktionalität zu verringern (Nitze, 2018, S. 1).

Eine API, welche die Prinzipien des Programmierparadigmas REST erfüllt, wird REST-API oder RESTful-API genannt. REST ist nach Fielding (2000, S. 76) ein Architekturkonzept für verteilte Hypermedia-Systeme. Hierbei sollen folgende Prinzipien erfüllt sein, welche flexibel implementiert werden können:

- **Client-Server:** Hierbei wird verlangt, dass Client und Server getrennt voneinander und individuell änderbar sind. Änderungen am Server sollen keine Auswir-

kungen auf den Client haben und vice versa.

- **Zustandslosigkeit:** Die Kommunikation zwischen Client und Server muss zustandslos sein. Jede Anfrage muss alle nötigen Informationen beinhalten, um die Anfrage richtig zu interpretieren und soll hierfür keinen serverseitig gespeicherten Kontext benötigen.
- **Cache:** Die Client-Server-Kommunikation soll durch Caching optimiert werden.
- **Einheitliche Schnittstelle:** Die Informationsübertragung geschieht in einer standardisierten Form. Informationen werden als Ressourcen abstrahiert und können über die jeweilige Repräsentation modifiziert werden. Ressourcen werden durch Uniform Resource Identifier (URI) identifiziert. Neben den Anfragen des Clients sind auch die Antworten vom Server selbstbeschreibend und können ohne zusätzlichen Kontext darstellen, wie der Server die Informationen bzw. die Anfrage des Clients verarbeitet hat.
- **Schichtsystem:** Aufstellen eines mehrstufigen Systems, bei dem die Komponenten in hierarchisch angeordneten Schichten organisiert werden.
- **Code-On-Demand:** Hierunter wird eine optionale Richtlinie verstanden, die besagt, dass die Client-Funktionalität erweitert werden kann, indem Code vom Server runtergeladen und ausgeführt wird (Fielding, 2000, S. 76-97).

REST wird meist über das Hypertext Transfer Protocol (HTTP) realisiert. Die HTTP-Methoden bilden ein Äquivalent zu CRUD, das ein Akronym für Create, Read, Update und Delete ist und im Zusammenhang mit Datenbankoperationen benutzt wird. Die Art, wie mit einer Datenbank mit diesen Operationen interagiert wird, kann auf gleiche Weise auf die Interaktion des Benutzers mit einer API abgebildet werden. So wird GET benutzt, sobald der Client Informationen vom Server abfragt, ohne dabei Daten zu ändern. Hierbei finden meist Query-Strings Einsatz, um die Ausgabe zu ändern. POST wird verwendet, um ein Objekt zu erstellen, wobei die zu erstellenden Daten meist per HTTP-Body übertragen werden. Dies gilt auch für PUT, welches für die Änderung einer Ressource steht. Bei einer DELETE Anfrage wird die Anweisung gegeben, ein oder mehrere Objekte zu löschen (Stowe, 2015, S. 91-98).

Die Antworten der API werden mit HTTP-Status-Codes versehen. Diese sind den meisten Entwicklern bekannt und bieten eine einheitliche und standardisierte Klassifikation der Antworten der API (Stowe, 2015, S. 102). Fängt dieser beispielsweise mit einer fünf an, wird damit ein Serverfehler ausgedrückt, bei einer vier wiederum ein Clientfehler. Mit einer vorangestellten zwei wird signalisiert, dass kein Fehler aufgetreten ist und die Anfrage planmäßig vom Server bearbeitet werden konnte (Vasudevan, o. J.).

2.2.2 Theoretische Grundlagen der API-Spezifikation

Die API-Spezifikation ist das zentrale Element beim API-First Ansatz. Sie liefert ein breites Verständnis von der Funktionalität der API und wie diese sich verhält (Wagner, o. J.-a). Sie listet alle wichtigen Endpunkte mit Operationen sowie die nötigen Anfrage- und Antwortschemas auf, wobei kein Einblick gegeben wird, wie die API implementiert

wurde. Wenn die API gut entworfen und beschrieben ist, kann ein Konsument die API mit einem minimalen Lernaufwand und Implementationsverständnis verwenden (Krzyk, 2020).

Es gibt mehrere Standards für die API-Spezifikation. OpenAPI Specification (OAS), RESTful API Modeling Language (RAML) und API-Blueprint sind Beispiele für solche Standards, mit denen REST-Schnittstellen beschrieben werden können (Heidenreich, Hahn, Stoikovitch & Ralphson, 2019; Nemec, 2019; Ratovsky & Miller, 2017).

Im folgenden Listing 2.1 wird beispielhaft der Anfang einer API-Spezifikation nach OAS gezeigt. Enthalten sind die Version des Spezifikationsdokuments, allgemeine Informationen über die API, Server und eine Auflistung aller Endpunkte der API mit den jeweiligen Anfrage- und Antwortschemas.

```yaml
openapi: 3.0.0
info:
  title: Example API
  description: #...
  contact:
    name: Jonas Schimmer
  version: 1.0.0
servers: #...
tags: #...
paths:
  /exampleItem:
    get:
      summary: #...
      responses:
        "200":
          description: OK
          content:
            application/json:
              schema:
                type: object
                properties:
                  itemName:
                    description: #...
                    type: string
                    example: Example Name
    post:
      summary: #...
      requestBody: #...
      responses:
        "201": #...
```

Listing 2.1: Ausschnitt einer beispielhaften API-Spezifikation nach OAS

Für alle drei Standards gibt es Anwendungen, mit denen Entwicklungsschritte durch automatische Generierungen auf Basis der API-Spezifikation optimiert werden können. So können mit Werkzeugen, wie oas-generator oder Swagger Codegen, Code-Gerüste generiert werden (Cheng, Tam, Tumanischvili & Ratovsky, 2020; Fernandez, Molina, Fresno & Lozada, 2018). Weiterhin besteht die Möglichkeit, auf Basis der Spezifikation Code-Gerüste für Unit-Tests zu generieren, wofür OpenAPI Test Templates oder andere Werkzeuge verwendet werden können (Dietz & Uzlopak, 2019). Mit diesen Tests allein wird jedoch nicht sichergestellt, dass die Implementierung der API der festgelegten Spezifikation entspricht. Der Prism Validation Proxy ist ein mögliches Werkzeug, um solch ein Contract-Testing durchzuführen. Hierbei werden Anfragen an den Proxy gestellt, welcher diese mit der hinterlegten API-Spezifikation vergleicht und einen Fehler wirft, wenn die Anfrage oder die vom Server gelieferte Antwort nicht mit der API-Spezifikation übereinstimmt (Chianese, Sturgeon & Maciaszek, 2020).

Mit Prism oder anderen Tools kann zudem ein Mock-up der API auf Basis der Spezifikation erstellt werden. Anfragen können dann an den Mocking-Service gestellt werden, der mit Beispieldaten aus der Spezifikation antwortet (Chianese, Sturgeon, Conrad & Maciaszek, 2020).

2.2.3 Prinzip des API-First Ansatzes

Um den API-First Ansatz zu verstehen, ist ein Blick auf den gegensätzlichen Code-First Ansatz hilfreich. Bei diesem Ansatz wird die gewünschte Funktionalität zuerst implementiert und im Anschluss wird eine Schnittstelle darüber entwickelt. Die bereits erfolgte Implementierung führt also den Entwurf und die Entwicklung der Schnittstelle, was einen schlechten API-Entwurf zur Folge haben kann. Bestehende Datenstrukturen der Anwendung schränken ein, wie Informationen über die neue Schnittstelle bereitgestellt werden können. Ist die gewünschte Schnittstelle nicht mit den vorhandenen Daten realisierbar, müssen Einsparungen beim API-Entwurf gemacht werden (Lin, 2018).

Bei der Definition von API-First gibt es mehrere Ansätze. Laut Wagner (o. J.-b) geht es beim API-First Ansatz darum, dass die API das zentrale Element im Projekt repräsentiert. Es soll viel Aufwand in den Entwurf investiert werden, um eine konsistente und wiederverwendbare API zu kreieren. Das kann durch Einsatz einer API-Beschreibungssprache wie OAS gewährleistet werden, die beschreibt, wie sich die API verhalten soll (Wagner, o. J.-b).

Lin (2018) beschreibt API-First als Ansatz, bei dem der Entwurf und die Entwicklung der API vor der Implementierung stattfinden. Es wird zuerst die Schnittstelle für die Anwendung kreiert und sobald diese existiert, wird sie als Basis zur Entwicklung der restlichen Anwendung verwendet. Statt mit der Implementierung zu starten, soll der Entwicklungsprozess mit Planung, Entwurf, Mock-ups und Tests beginnen. Mit dem Entwurf kann anschließend ein frühes Feedback eingefordert werden und so können gravierende Fehler im Konzept früh erkannt und behoben werden. Sobald der Entwurf gefestigt ist, kann er die Rolle eines Contracts einnehmen, gegen welchen alle Teammitglieder mit der Hilfe von Mock-ups parallel arbeiten können. Soll später ein

neues Feature hinzugefügt werden, so wird der gleiche Prozess durchlaufen. Zuerst kommt der Entwurf des neuen Features, woraufhin Feedback erhalten werden kann und erst danach wird das Feature implementiert. Die Definition von API-First kann noch weiter ausgeführt werden, wobei zwischen API-First Development und API-First Design unterschieden wird:

- **API-First Development:** Wenn eine neue Funktionalität entwickelt wird, dann wird diese als Erstes über eine Schnittstelle bereitgestellt. Die Teammitglieder, welche den restlichen Teil der Anwendung entwickeln, sind die ersten Konsumenten der API. Mit diesem Ansatz wird sichergestellt, dass die gesamte Funktionalität des Systems immer über eine API bereitgestellt wird, sodass auch andere Anwendungen diese Funktionen nutzen können.
- **API-First Design:** Dieser Ansatz geht einen Schritt weiter und sagt aus, dass Planung und Entwurf der API vor der Implementierung erfolgen sollen. Hierbei soll der Frage nachgegangen werden, welche Funktionalität die API haben und welche Daten sie bereitstellen soll. Es bedeutet nicht, dass der Entwurf bis ins letzte Detail durchführt wird, um dann keine Zeit mehr für die Implementierung zu haben. Änderungen sollen möglich sein, was durch einen guten Entwurf gewährleistet wird (Lin, 2018).

Der Ansatz soll einerseits positive Auswirkungen auf den Entwicklungsprozess haben. Levin (2019) verdeutlicht, dass er den Entwicklungsprozess an manchen Stellen beschleunigt. So können beispielsweise mit dem Spezifikationsdokument eine Dokumentation sowie Code-Gerüste und Testfälle generiert werden, um die Entwicklungszeit zu verringern. Krzyk (2020) hebt hervor, dass durch die erstellten Mock-ups paralleles Arbeiten ermöglicht wird. Das Frontend-Team kann zu gleicher Zeit wie das Backend-Team in die Implementierungsphase starten, da die Schnittstelle klar definiert ist und Anfragen der Client-Applikation zu Testzwecken an das Mock-up gestellt werden können. Somit muss nicht auf die Fertigstellung des Backends gewartet werden (Krzyk, 2020). Weiterhin bietet das zentrale Dokument der API-Spezifikation eine bessere Orientierung für Entwickler sowie Anwender und bietet eine konkrete Diskussionsgrundlage für die Weiterentwicklung (Nitze, 2018, S. 4).

Andererseits ist mit diesem Ansatz ein hoher Entwurfsaufwand verbunden, welcher in manchen Fällen, beispielsweise bei experimentellen Projekten mit stetig wechselnden Anforderungen, ein Nachteil sein kann (Krzyk, 2020).

Zusammenfassend lässt sich sagen, dass API-First einen Entwicklungsansatz beschreibt, bei dem die API an erster und oberster Stelle stehen soll. Design-First, Contract-First, API-First Design oder vereinfachter Weise nur API-First beschreiben alle einen Ansatz, bei dem der Entwurf der API ein erhöhtes Gewicht hat und vor der Implementierung stattfindet. Das beim Entwurf entstehende Spezifikationsdokument nimmt für den weiteren Entwicklungsprozess die Rolle eines Contracts ein.

Kapitel 3

Entwicklung der Full-Stack Webanwendung mit API-First Ansatz

Im folgenden Kapitel wird die Entwicklung der Webanwendung beschrieben. Zuerst wird die Einführung eines einheitlichen Softwarestacks behandelt. Im Anschluss daran wird die Überführungsstrategie aufgestellt und auf jeden Anwendungsteil angewendet.

3.1 Einführung eines einheitlichen Softwarestacks

Für die Entwicklung des Anwendungsteils Additive Fertigung wurde das Content-Management-System TYPO3 verwendet. Da die Langzeitunterstützung für die verwendete TYPO3 Version 8 beendet wurde und ein Update auf eine aktuelle Version viele Änderungen am PHP-Code nach sich ziehen würde, bietet sich der verwendete Softwarestack nicht für die weitere Entwicklung an. Die anderen beiden Anwendungsteile Subtraktive Fertigung und Prüfteileinlagerungssystem wurden mit dem Framework CakePHP und einer MySQL-Datenbank entwickelt, wobei der Bereich Subtraktive Fertigung hierbei erst nachträglich von Vanilla PHP nach CakePHP überführt worden ist.

Da der Großteil des Projektes bereits mit CakePHP und MySQL realisiert wurde, würde sich dieser Softwarestack für die Weiterentwicklung anbieten. Da hier jedoch, wie bereits angemerkt, viel Refactoring-Arbeit nötig wäre und neu aufgekommene Anforderungen eine großflächige Überarbeitung des Quellcodes benötigen würden, besteht die Möglichkeit, einen neuen Softwarestack einzuführen.

Um gleiche Technologien wie im Projektumfeld zu nutzen, wird der MEVN-Stack eingeführt. MEVN ist ein Akronyom und steht für MongoDB, Express.js, Vue.js und Node.js. Diese Begriffe bezeichnen die wichtigen Komponenten des Softwarestacks:

- **MongoDB:** Hierbei handelt es sich um die verwendete Datenbank. MongoDB ist eine dokumentenorientierte Datenbank, welche die Daten in flexiblen Dokumenten speichert, wobei die Datenstruktur leicht veränderbar bleibt (MongoDB, Inc., o. J.).

- **Express.js:** Express.js ist ein Webframework für Node.js, welches nützliche Funktionen für die Entwicklung bereitstellt und eine einfache Programmierung einer API ermöglicht (OpenJS Foundation, o. J.).
- **Vue.js:** Vue.js ist ein clientseitiges Webframework für die Entwicklung von Benutzeroberflächen (You, o. J.).
- **Node.js:** Node.js ist eine ereignisgesteuerte Laufzeitumgebung für die Entwicklung von Netzwerk-Anwendungen, welche die serverseitige Ausführung von JavaScript ermöglicht (Rogers, Hemberger, Senkpiel & You, 2020).

Folglich ergibt sich eine Client-Server-Architektur, wie es in Abbildung 3.1 zu sehen ist. Clientseitig befindet sich die Vue.js-Anwendung, die über die REST-API mit dem Server kommuniziert. Serverseitig bildet der Node.js-Server die zentrale Komponente der Anwendung, welche die Geschäftslogik beinhaltet und eine Verbindung zu der MongoDB-Datenbank hat. Hierduch ergibt sich eine klare Separation of Concerns und der Client sowie der Server können unabhängig voneinander angepasst werden, ohne dass diese Änderungen zwangsweise Auswirkungen auf die andere Seite haben (Doglio, 2018, S. 26). Zusätzlich wird die Möglichkeit geschaffen, dass andere Clients bzw. Anwendungen, beispielsweise aus dem Projektumfeld, auf die Funktionalitäten des Projektes über die REST-API zugreifen können.

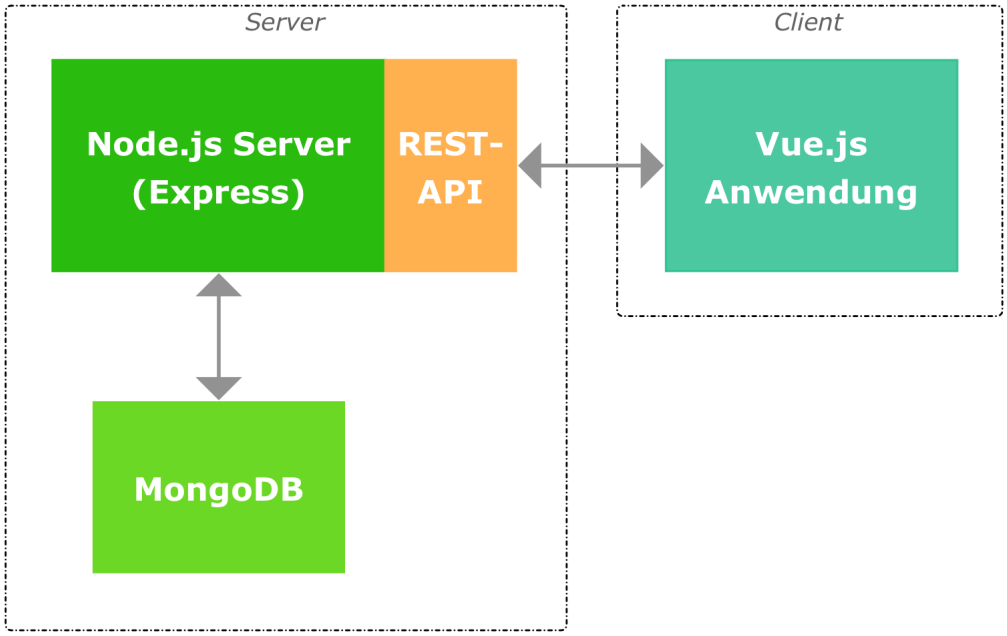


Abbildung 3.1: Architektur der MEVN-Anwendung

3.2 Aufstellen der Überführungsstrategie

Um eine Überführungsstrategie mit integriertem API-First Ansatz aufstellen zu können, soll zuerst untersucht werden, wie ein Entwicklungsprozess mit API-First Ansatz in der Literatur beschrieben wird. Hier gibt es wie bei der Definition von API-First mehrere Ansätze, welche zum Großteil gleiche Ansichten beschreiben.

Nach Nitze (2018, S. 2-4) sind folgende Schritte für den Entwicklungsprozess relevant:

1. Zuerst sollen existierende und gewünschte Funktionalitäten aufgenommen und miteinander abgeglichen werden.
2. Im Anschluss soll die Funktionalität der API beschrieben werden, wobei eine Spezifikationssprache verwendet wird. Hierbei werden die möglichen Anfragen mit Parametern und Antworten aufgelistet. Damit lässt sich dann eine interaktive Schnittstellenbeschreibung erzeugen.
3. Mit Hilfe der Spezifikation kann jetzt Code generiert werden (Nitze, 2018, S. 2-4).

Einen etwas ausführlicher beschriebenen Entwicklungsprozess stellt Riggings (2015) mit folgenden fünf Schritten dar:

1. Zu Beginn soll geplant werden. Es soll sich der Zweck der Anwendung verdeutlicht werden.
2. Anschließend folgt der API-Entwurf und dessen Validierung. Hierbei können durch Mock-ups Use-Cases praktisch durchlaufen werden, um zu prüfen, wie die API genutzt wird.
3. Wenn der API-Entwurf bis auf Weiteres abgeschlossen ist, werden an der API-Spezifikation vorerst keine Änderungen vorgenommen und eine Dokumentation kann generiert werden.
4. Mit der API-Spezifikation können anschließend Tests generiert werden.
5. Der letzte Schritt ist die Implementierung der API (Riggings, 2015).

Bei beiden Abläufen wird zu Beginn geplant, wobei die nötigen Funktionalitäten des Systems zusammengetragen werden. Vor der Implementierung findet, wie vom API-First Ansatz verlangt, bei beiden Abläufen der Entwurf der API statt. Auf Basis der API-Spezifikation können dann Entwicklungsschritte durch automatische Generierungen optimiert und beschleunigt werden.

Bezüglich des API-Entwurfs lässt sich zusätzlich festhalten, dass dieser als eine Art Grüne-Wiese-Übung durchgeführt werden sollte, wobei anfänglich der Fokus nicht auf das alte System und dessen Funktionen zu legen ist, sondern mehr auf die Funktionalität, die das neue System bereitstellen soll. Alte Abhängigkeiten sollten also zunächst ignoriert werden, um den Fokus des Entwurfs darauf zu lenken, was den Konsumenten der API am meisten dient (Riggings, 2015).

Das Erstellen der API-Spezifikation kann außerdem als iterativer Prozess umgesetzt werden. Anknüpfend an die Konzeption der API wird Feedback eingefordert und

der Entwurf wird validiert. Daraufhin müssen entweder Nachbesserungen getätigt werden und die Entwurfsphase wird weitergeführt, oder der Entwurf ist in Ordnung und kann vorerst abgeschlossen werden. Mock-ups können in der Feedbackphase bei Bedarf unterstützend eingesetzt werden, da die API simuliert und somit greifbarer wird, wodurch sie besser bewertet werden kann (Stowe, 2015, S. 30).

Auf die eigene Überführungsstrategie lassen sich einige Elemente der beschriebenen Prozesse übernehmen. Um nach dem API-First Ansatz zu arbeiten, ist zwingend erforderlich, dass der Entwurf der API und das Erstellen der API-Spezifikation früh im Prozess und vor der Implementierung stattfinden. Die Möglichkeit, auf Basis der API-Spezifikation und ggf. Mock-ups frühes Feedback zu erhalten, kann mittels eines iterativen Entwurfsprozesses ausgenutzt werden. Um zu Beginn des Überführungsprozesses die neue Anwendung planen zu können, muss dieser mit einer Analyse der bestehenden Anwendungen beginnen. Geänderte Anforderungen sollen an dieser Stelle aufgeführt werden. Schließlich kann der API-Entwurf durchgeführt werden, woraufhin die API implementiert wird. Da die API-Spezifikation beim API-First Ansatz die Rolle eines Contracts einnimmt, muss gegen Ende des Überführungsprozesses mittels Contract-Tests geprüft werden, ob dieser eingehalten wird. Diese Überlegungen führen zu der in Abbildung 3.2 dargestellten Überführungsstrategie, deren einzelne Schritte im kommenden Abschnitt 3.3 genauer beschrieben und praktisch durchlaufen werden. Für jeden Anwendungsteil wird dieser Prozess von Anfang bis Ende durchgeführt.

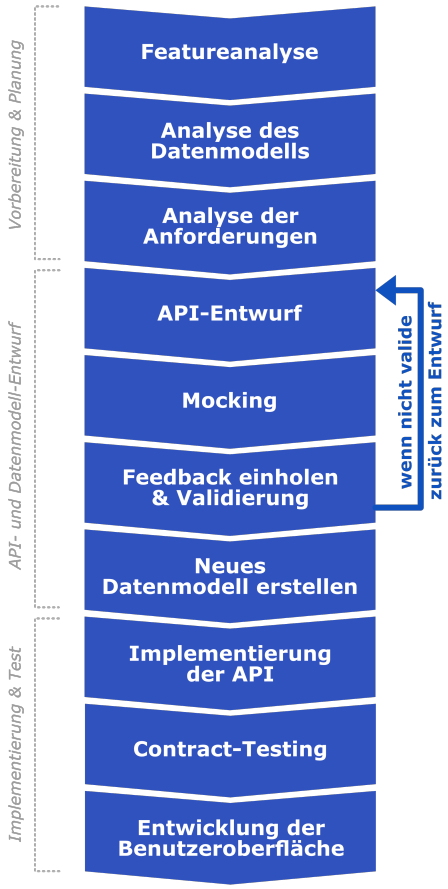


Abbildung 3.2: Modell der aufgestellten Überführungsstrategie

3.3 Beschreibung der Überführungsschritte

Im folgenden Abschnitt werden die einzelnen Schritte des Überführungsprozesses genauer beschrieben. Es wird jeweils dargestellt, wieso der jeweilige Schritt zu dem Zeitpunkt im Prozess durchgeführt wird. Zusätzlich werden beispielhaft die praktischen Ergebnisse sowie die anschließenden Erkenntnisse beschrieben.

3.3.1 Featureanalyse der bestehenden Anwendung

Zu Beginn der Überführung muss eine Analyse des bestehenden Systems und dessen Features durchgeführt werden. Somit wird sich ein Überblick über das alte System sowie ein erster Eindruck über die wichtigen Daten und deren Struktur verschafft. Dieser Schritt ist anfänglich wichtig, um den Status quo festzuhalten und davon ausgehend die weitere Entwicklung planen zu können.

Bei der Featureanalyse werden die Funktionen der Anwendung mit der Benutzeroberfläche oder anderen Schnittstellen getestet und analysiert. Hierbei wird schriftlich festgehalten, welche Features das System bereitstellt. Ebenso wird an dieser Stelle vermerkt, welche Funktionalitäten fehlerhaft oder nur zum Teil funktionieren. Ein beispielhaftes Ergebnis der Analyse ist in Tabelle 3.1 zu sehen.

Feature	**Beschreibung**	**Systemfehler**
Verwaltung Firmen	Erstellen, Lesen, Ändern, Löschen von Firmen	
Verwaltung Drucker	Erstellen, Lesen, Ändern, Löschen von Druckern	
Verwaltung Lagerorte	Erstellen, Lesen, Ändern, Löschen von Lagerorten	
Verwaltung Boxen	Erstellen, Lesen, Ändern, Löschen von Boxen. Eine Box kann in einen Lagerort eingelagert werden.	
Verwaltung Prüfteile	Erstellen, Lesen, Ändern, Löschen von Prüfteilen. Ein Prüfteil verweist auf einen Drucker und eine Firma. Es kann in eine Box eingelagert werden.	
Upload Messdateien	Die Messdateien eines Prüfteils können hochgeladen werden und einem Prüfteil zugewiesen werden. Die Dateien werden auf dem Dateisystem des Servers gespeichert.	

Tabelle 3.1: Ergebnisse der Featureanalyse des Prüfteileinlagerungssystems

Tabelle 3.1 zeigt beispielhaft die Featureanalyse des Prüfteileinlagerungssystems. In der ersten Spalte wird das jeweilige Feature benannt. In der zweiten Spalte wird eine kurze Beschreibung des Features aufgeführt. In der letzten Spalte werden Probleme und Fehler der alten Implementierung festgehalten. Die Ergebnisse der Featureanalyse des Prüfteileinlagerungssystems sind Funktionalitäten zum Verwalten von Firmen, Druckern, Lagerorten, Boxen und vor allem die Möglichkeit der Einlagerung von Qualitätsprüfteilen. Weiterhin ist der Upload von Messdateien zu einem gegebenen Prüfteil eine wichtige Funktion des Systems.

Die Analyse der restlichen Anwendungsteile kann dem Anhang B entnommen werden.

Das Ergebnis der Featureanalyse ist ein Überblick über die Funktionalitäten des bestehenden Systems. Es hat sich herausgestellt, dass beim Anwendungsteil der subtraktiven Fertigung einige Implementierungsfehler bestehen, wohingegen die anderen beiden Anwendungsteile überwiegend fehlerfrei funktionieren. Durch die Featureanalyse konnte ein guter Eindruck gewonnen werden, welche Funktionalitäten wichtig sind. Ohne Berücksichtigung der neuen Anforderungen kann an dieser Stelle eingeschätzt werden, welche Funktionalitäten in der neuen Implementierung voraussichtlich vorhanden sein müssen und welchen Umfang diese haben werden.

3.3.2 Analyse des bestehenden Datenmodells

Im Anschluss an die Featureanalyse wird eine Analyse des Datenmodells durchgeführt. Dieser Schritt ist wichtig, um einen tieferen Einblick in die bestehenden Datenstrukturen zu erhalten. Daraus resultiert ein umfangreiches Verständnis von den Daten, die auch für die neue Anwendung relevant sein können.

Zu Beginn muss das Datenmodell beschafft werden. Beim Anwendungsteil der additiven Fertigung liegt ein Datenmodell vor, bei den anderen beiden Teilen muss dieses anhand vom jeweiligen Datenbankschema rekreiert werden. Anschließend ist ein tiefer Einblick in die Datenstrukturen möglich.

Abbildung 3.3 zeigt beispielhaft das Datenmodell des Prüfteileinlagerungssystems. Es ist zu erkennen, dass die *quality_test_parts*, repräsentativ für die Qualitätsprüfteile, das zentrale Element im Datenmodell des Einlagerungssystems sind. Die Datenstruktur verweist jeweils auf die zugehörige Firma und den verwendeten Drucker. Für die Einlagerungsfunktion gibt es an dieser Stelle eine Referenz zu den *boxes*, welche einen Verweis auf die *storage_locations* haben. Diese repräsentieren einen festen Lagerort an der Hochschule und setzen sich aus dem Standort, dem Gebäude, der Raumnummer und der Schranknummer zusammen. Für die Funktionalität zum Speichern von Messdateien gibt es eine Datenstruktur *files*, die auf das jeweilige Qualitätsprüfteil verweist und den Pfad zur Datei auf dem Dateisystem des Servers speichert.

Die Datenmodelle der restlichen Anwendungsteile können dem Anhang C entnommen werden.

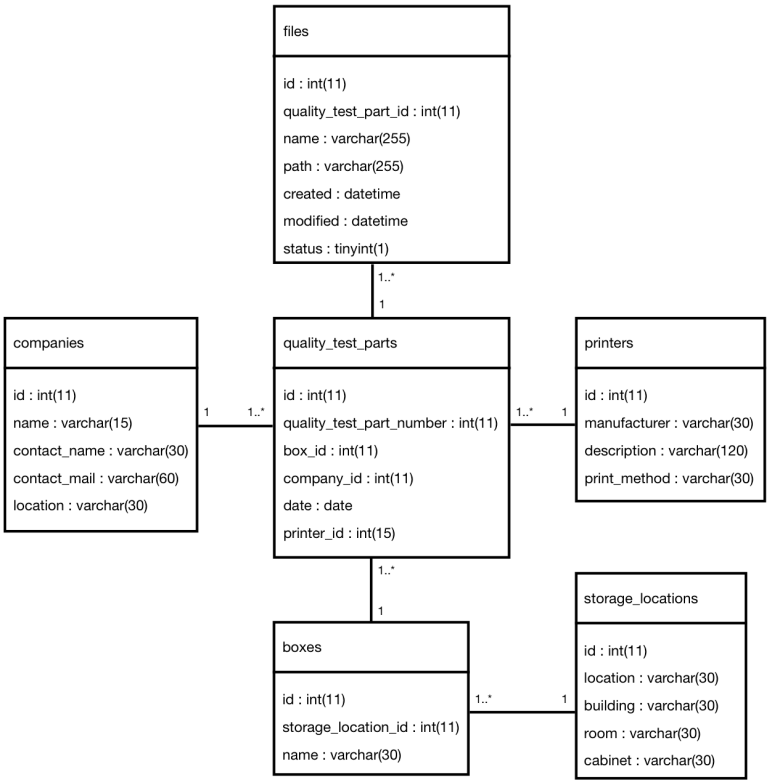


Abbildung 3.3: Altes Datenmodell des Prüfteileinlagerungssystems

Durch diesen Prozessschritt wird ein besseres Verständnis dafür geschaffen, wie die Daten strukturiert sind, um die Funktionen, welche in der Featureanalyse erfasst wurden, realisieren zu können. Die Analyse der Datenmodelle bzw. der Datenstrukturen schafft eine Inspiration für den späteren API-Entwurf, weil sie zeigt, welche Daten im bestehenden System eine Rolle spielen und somit auch für die neue Anwendung von Bedeutung sein können.

3.3.3 Analyse und Evaluierung der Anforderungen

Nach den ersten beiden Überführungsschritten besteht ein tieferes Verständnis von den Funktionen und Datenstrukturen der alten Anwendungen. Beim letzten Schritt der Vorbereitungs- und Planungsphase wird sich nun mit den Anforderungen des neuen Systems befasst. Dieser Schritt ist wichtig, weil die Funktionalitäten aufgrund neuer Anforderungen nicht unverändert vom alten System übernommen werden können. Somit müssen die Funktionen aus der Featureanalyse mit den ggf. neuen Anforderungen abgeglichen werden.

In der Tabelle 3.2 werden beispielhaft die Anforderungen des Prüfteileinlagerungssystems aufgelistet. Der Großteil der Anforderungen deckt sich mit den Funktionalitäten aus der Featureanalyse, jedoch gibt es auch Anforderungen, die nicht implementiert wurden. Beispiele hierfür sind die Suche bzw. das Filtern der verschiedenen Objekte nach Name oder Nummer sowie die Anzeige aller Qualitätsprüfteile einer Firma oder eines Druckers. Ebenso lässt sich durch diesen Abgleich erkennen, dass manche Features des alten Systems nicht in gleicher Art oder mit gleichem Umfang in der neuen

Anwendung realisiert werden müssen. Beispielsweise ist der Einlagerungsprozess im alten System so realisiert, dass ein Prüfteil in eine Box eingelagert und diese Box wiederum einem Lagerort zugewiesen wird. Zu Beginn muss ein Lagerort erstellt werden, woraufhin eine Box erstellt und dem Lagerort zugewiesen werden muss. Im Anschluss lässt sich ein neues oder bestehendes Prüfteil in diese Box einlagern. Dieser Einlagerungsprozess ist sehr umständlich und muss laut den Anforderungen nicht zwingend in der beschriebenen Weise realisiert werden.

Die restlichen Anforderungen sind im Anhang D aufgeführt.

Anforderungsgruppe	**Beschreibung**
Verwaltung Firmen	Das Erstellen, Lesen, Ändern und Löschen einer Firma soll durchführbar sein. Neben dem Firmennamen besitzt eine Firma einen Namen und eine E-Mail Adresse der Kontaktperson sowie eine Adresse, welche aus einer Straße, einer Postleitzahl und einem Ort besteht. Es soll die Möglichkeit geben, eine Firma mit ihrem Namen zu suchen.
Verwaltung Drucker	Das Erstellen, Lesen, Ändern und Löschen eines Druckers soll durchführbar sein. Ein Drucker besitzt einen Namen, einen Hersteller und eine Druckmethode. Es soll die Möglichkeit geben, einen Drucker mit seinem Namen zu suchen.
Verwaltung Prüfteile	Das Erstellen, Lesen, Ändern und Löschen eines Qualitätsprüfteils soll durchführbar sein. Ein Prüfteil besitzt eine Prüfteilnummer, ein Druckdatum sowie eine oder mehrere Messdateien, welche der Benutzer hochladen und einsehen können soll. Es soll der verwendete Drucker sowie die auftragstellende Firma hinterlegt werden. Ein Qualitätsprüfteil soll in einen Lagerort eingelagert werden können. Zusätzlich soll die Möglichkeit angeboten werden, ein Prüfteil nach Nummer oder Lagerort zu suchen sowie alle Prüfteile einer Firma oder eines Druckers auflisten zu lassen.
Verwaltung Lagerorte	Das Erstellen, Lesen und Ändern eines Lagerorts soll durchführbar sein. Ein Lagerort besteht aus dem Standort, einem Gebäude, einem Raum, einer Schranknummer und einer Boxnummer.

Tabelle 3.2: Neue Anforderungen des Prüfteileinlagerungssystems

Durch die Niederschrift der neuen Anforderungen und den Abgleich mit den Funktionen des alten Systems wird ein gutes Verständnis vom Funktionsumfang der neuen Anwendung geschaffen. Dadurch lässt sich im Anschluss der API-Entwurf durchführen. Zusätzlich wird für die spätere Implementierung ein Überblick geschaffen, an welchen Stellen die Algorithmen oder die Implementierungsweise der Funktionen des alten Systems eine mögliche Inspirationsquelle für das neue System sein können.

3.3.4 Iterativer API-Entwurf

In der Vorbereitungs- und Planungsphase wurden die bestehenden Systeme und deren Funktionen analysiert. Mit der zusätzlichen Analyse und Evaluierung der Anforderungen wurde ein tiefes Verständnis von der Funktionalität, welche das neue System bereitstellen soll, geschaffen. An dieser Stelle des Überführungsprozesses kann durch das gesammelte Funktions- und Datenverständnis die Entwurfsphase beginnen. Je nach Entwicklungsansatz kann diese unterschiedlich ablaufen. Beim klassischen Code-First Ansatz kann beispielsweise mit dem Entwurf des Datenmodells begonnen werden. Daraufhin kann die Implementierung geplant und umgesetzt werden. Beim API-First Ansatz hingegen wird der Entwurf der API höher gewichtet. Die API-Spezifikation beschreibt die Funktionalität der API und soll, damit sie als Contract

fungieren kann, so früh wie möglich im Entwicklungsprozess erstellt werden. Deshalb wird dieser Entwurf der Implementierung und auch der Erstellung des Datenmodells vorgezogen.

Mit dem Entwurf der API und der Erstellung der Schnittstellenbeschreibung ist dies die erste Stelle im Überführungsprozess, an welcher der API-First Ansatz greift. Da die API-Spezifikation einen guten Überblick über die zu erwartende Funktionalität des Systems gibt, besteht die Möglichkeit, mit diesem Spezifikationsdokument frühes Feedback zum Entwurf zu erhalten. Dies wird wie bereits erwähnt im Entwicklungsprozess ausgenutzt. Sobald der API-Entwurf abgeschlossen ist, wird dieser den Stakeholdern für Feedback vorgelegt. Wenn hierbei schwerwiegende Fehler im Entwurf festgestellt werden, muss im Überführungsprozess zurück zum API-Entwurf gesprungen werden. Hierdurch entsteht ein iterativer Entwurfsprozess der API, wobei die API-Spezifikation in jedem Schritt verbessert und evaluiert wird, bis sie den Anforderungen der zu entwickelnden Anwendung am besten entspricht. Somit wird gewährleistet, früh im Entwicklungsprozess eine solide Spezifikation der Schnittstelle zu erstellen, wobei große Fehler im Entwurf ausgeschlossen werden.

Um eine leichte Verständlichkeit der API zu gewährleisten, gibt es mehrere Konventionen für den Entwurf. Bei der Erstellung der URI der API-Endpunkte gibt es beispielsweise folgende Konventionen zu beachten:

- Eine Ressource, die eine Sammlung repräsentiert, sollte mit einem Nomen im Plural benannt werden.
- Für die Benennung der URI sollten lediglich Kleinbuchstaben verwendet werden, da die Mischung von Klein- und Großbuchstaben das Lesen und Schreiben der URI erschwert.
- Leerzeichen und andere Zeichen, welche beim Transport eine Codierung benötigen, sollen vermieden werden.
- Bezeichnungen von CRUD-Operationen sollten innerhalb der URI vermieden werden (De, 2017, S. 37).

An Stelle von CRUD-Operationen oder anderen Verben in den URI werden die HTTP-Verben im Kontext der Anfrage verwendet, um Operationen auf Ressourcen durchzuführen (Mulloy, 2012, S. 6). Weiterhin sollen die Namen der Endpunkte, die Eingabeparameter und die Antwortschemas einheitlich beschrieben sein, um die Konsistenz der API zu sichern. Dadurch wird ein geringer Lernaufwand gewährleistet und das Verstehen der API wird erleichtert (Jin, Sahni & Shevat, 2018, S. 50-51).

Unter Berücksichtigung der genannten Regeln und der zuvor aufgestellten Anforderungen lässt sich der API-Entwurf durchführen. Der Entwurf wird so durchgeführt, dass die API die verlangten Funktionalitäten der Anforderungen bereitstellen kann. Im Folgenden wird beispielhaft der Entwurfsprozess vom Prüfteileinlagerungssystem beschrieben.

Wie in Abbildung 3.4 zu sehen ist, werden auf übergeordneter Ebene der REST-API zuerst die drei Pfade für jeweils einen Anwendungsteil festgelegt. Anschließend

lassen sich beim Prüfteileinlagerungssystem die vier Ressourcen *testparts*, *othprinters*, *companies* und *storagelocations* identifizieren, welche sich aus der Analyse der neuen Anforderungen ableiten.

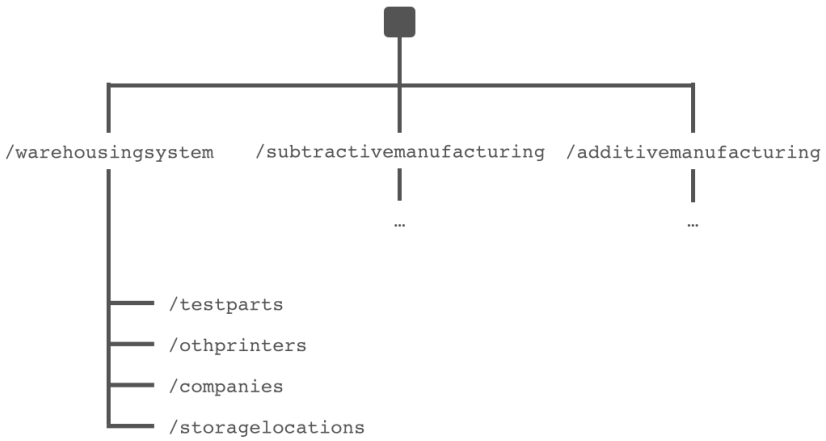


Abbildung 3.4: Übergeordnete Struktur der API

Für alle Ressourcen werden Endpunkte mit GET-, POST-, PUT- und DELETE-Operationen benötigt, weil das Erstellen, Lesen, Ändern und Löschen der Objekte laut den Anforderungen erforderlich ist. Daraus resultiert eine grobe Struktur der API-Endpunkte, die in Tabelle 3.3 zu sehen ist.

Operation	**Ressource**	**Beschreibung**
GET	/warehousingsystem/testparts	Alle Prüfteile lesen
POST	/warehousingsystem/testparts	Ein Prüfteil erstellen
GET	/warehousingsystem/testparts/id	Ein Prüfteil lesen
PUT	/warehousingsystem/testparts/id	Ein Prüfteil ändern
DELETE	/warehousingsystem/testparts/id	Ein Prüfteil löschen

Tabelle 3.3: Auflistung möglicher Endpunkte für die Qualitätsprüfteil-Ressource

Die Schemas der Datenobjekte hinter einer Ressource lassen sich aus den Anforderungen ableiten. Hierbei ist auch das alte Datenmodell nützlich, um mögliche Attribute der Objekte zu identifizieren. Abhängig von den Anforderungen müssen die Standard-Endpunkte für CRUD-Operationen erweitert werden. Beim Prüfteileinlagerungssystem gibt es beispielsweise folgende Erweiterungen:

- Die Suche nach einem Drucker, einer Firma oder einem Prüfteil soll mit einem Namen oder einer Nummer durchführbar sein. Hierfür müssen die jeweiligen GET-Endpunkte um einen entsprechenden Query-Parameter erweitert werden, um das Ergebnis der Anfrage passend einschränken zu können. In gleicher Weise wird die Suche von Qualitätsprüfteilen nach einem Lagerort umgesetzt.
- Um alle Prüfteile einer Firma bzw. eines Druckers bereitstellen zu können, müssen

die jeweiligen Antwortschemas der GET-Endpunkte um eine eingebettete Liste der Prüfteile ergänzt werden.

- Für die Verwaltung der Messdateien eines Prüfteils wird eine neue, untergeordnete Ressource *files* eingeführt.

Die abgebildeten Endpunkte werden in der API-Spezifikation festgehalten. In diesem Fall wird ein Spezifikationsdokument nach OAS erstellt, welches in der Datei `server/ISAC-TP1-API.yaml` gespeichert wird. Zu Beginn werden allgemeine Informationen aufgelistet, wie es in Listing 3.1 zu sehen ist.

```yaml
openapi: 3.0.0
info:
  title: ISAC TP1 API
  contact:
    name: ISAC
    url: https://www.isac-oth.de
  version: 1.0.0
```

Listing 3.1: Allgemeine Informationen der API und der Spezifikation

Für eine bessere Übersicht werden Tags eingeführt, wobei die Endpunkte aus den drei Anwendungsteilen in übergeordnete Gruppen eingeteilt werden. Diese sind dem Listing 3.2 zu entnehmen.

```yaml
tags:
  - name: Subtractive Manufacturing
    description: Subtraktive Fertigung
  - name: Additive Manufacturing
    description: Additive Fertigung
  - name: Warehousing System
    description: Pruefteil-Einlagerungssystem
```

Listing 3.2: Tags in der API-Spezifikation

Im Anschluss werden alle Endpunkte der API aufgelistet. Hierbei werden zuerst die Ressource und untergeordnet davon die möglichen Operationen aufgeschrieben. Es wird jeweils eine kurze Beschreibung sowie die Zuordnung zu bestimmten Tags angegeben. Anschließend werden mögliche Parameter der Anfrage aufgelistet. Zum Schluss erfolgt eine Aufzählung der möglichen Antworten der API, wobei nach HTTP-Statuscodes untergliedert wird. Das Schema des zurückzugebenden Objektes kann in einen separaten Abschnitt ausgelagert und mehrmals wiederverwendet werden. In Listing 3.3 wird ein Endpunkt zur Auflistung aller gespeicherten Firmen spezifiziert.

```yaml
paths:
  /warehousingsystem/companies:
      get:
        summary: Laedt alle Firmen
        tags:
          - Warehousing System
        parameters:
          - in: query
            name: name
            description: Filtern nach Firmenname
            schema:
              type: string
        responses:
          "200":
            description: OK
            content:
              application/json:
                schema:
                  type: array
                  items:
                    $ref: '#/components/schemas/Company'
          "500":
            description: Internal Server Error
```

Listing 3.3: Spezifikation eines Endpunktes der API

Basierend auf dem Spezifikationsdokument kann ein guter Eindruck davon erlangt werden, welche Funktionen das System bereitstellt und ob diese die gewünschten Anforderungen erfüllen können. Somit wird die API-Spezifikation nach fertiger Erstellung den Stakeholdern bzw. anderen Projektmitgliedern vorgelegt, um Feedback zu erhalten. Wird der aktuelle Entwurf nicht als valide eingestuft, muss die Spezifikation gemäß den gewünschten Änderungen angepasst und im Anschluss erneut evaluiert und validiert werden. Dieser iterative Entwurfsprozess wird so lange durchgeführt, bis die Spezifikation als valide eingestuft werden kann. Für eine greifbare Evaluierung des API-Entwurfs besteht die Möglichkeit, Mock-ups der API zu generieren. Mit den beispielhaften, simulierten Antworten des Mock-ups kann die Funktionsweise der API besser eingeschätzt werden.

Mit dem Tool SwaggerUI wird aus der API-Spezifikation eine interaktive Oberfläche generiert, welche als API-Dokumentation fungiert. Abbildung 3.5 zeigt einen Ausschnitt der Visualisierung aller Endpunkte des neuen Systems, die auf übergeordneter Ebene nach den drei angelegten Tags gruppiert werden. Mit einem Klick auf einen Endpunkt werden alle relevanten Informationen zu der Anfrage und der zu erwartenden Antwort angezeigt. Ein neuer Konsument der API kann mit Hilfe dieser interaktiven Oberfläche alle Endpunkte der API durchsuchen und somit einen besseren Eindruck von der Funktionsweise der API erlangen.

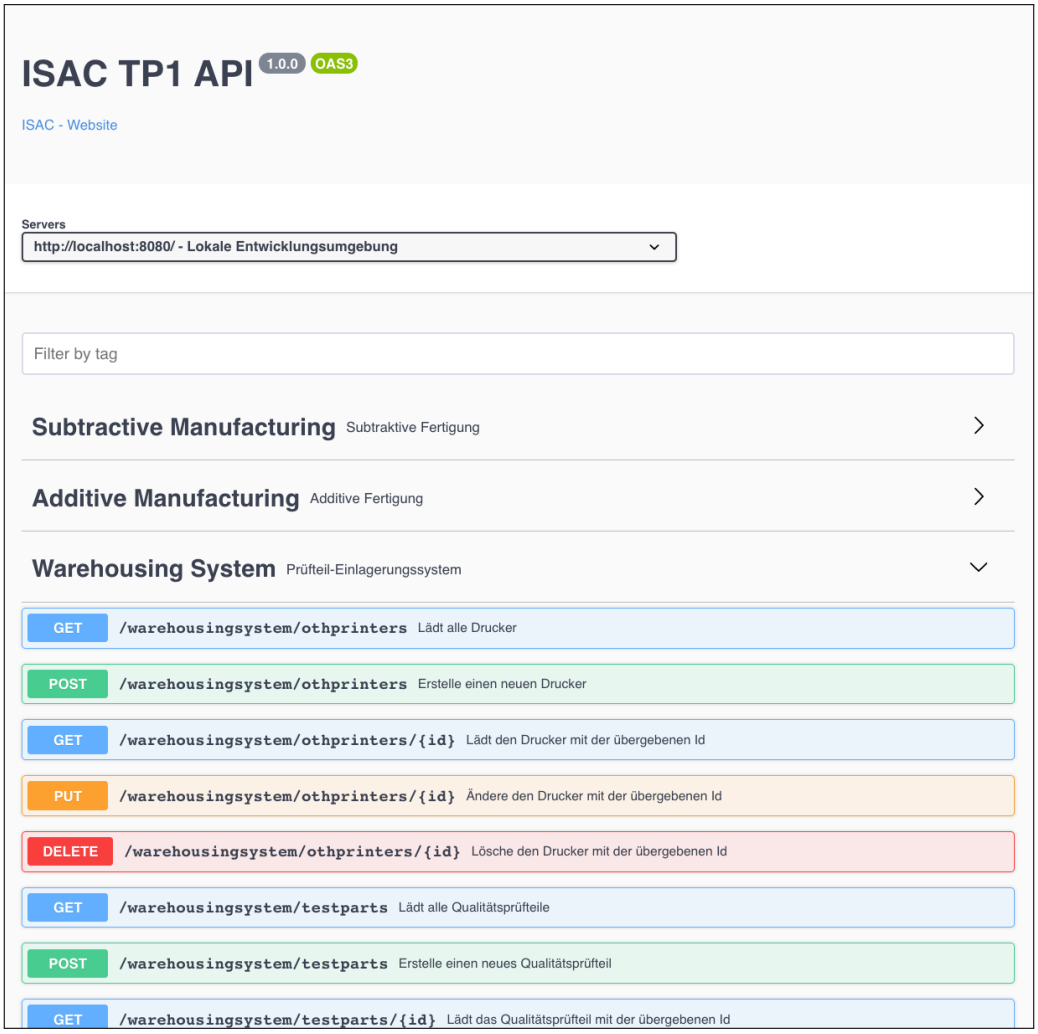


Abbildung 3.5: Mit SwaggerUI generierte, interaktive Oberfläche

3.3.5 Aufstellen des neuen Datenmodells

Im anschließenden Schritt des Überführungsprozesses folgt das Aufstellen des neuen Datenmodells. Hierbei führt der API-Entwurf den Entwurf des Datenmodells, weil dieser so durchgeführt werden muss, dass die Antworten, welche in der API-Spezifikation definiert wurden, in dem gewünschten Schema realisiert werden können. Neben der Beachtung der API-Spezifikation ist ein zusätzlicher Blick auf das alte Datenmodell hilfreich, da es aufgrund von teilweise gleichbleibenden Anforderungen zu gleichbleibenden Datenobjekten und Attributen kommen kann.

Die Datenmodelle werden somit nach den folgenden Schritten aufgestellt:

1. Die groben Datenobjekte lassen sich mit Blick auf die Ressourcen des API-Entwurfs und das alte Datenmodell bestimmen.
2. Es muss verifiziert werden, dass die in der Spezifikation definierten Antwortschemas mit den Datenstrukturen realisiert werden können.
3. Für die Beziehung zwischen den Datenobjekten muss die Entscheidung getroffen

werden, ob referenziert oder eingebettet wird.

4. Zum Schluss wird das Datenmodell überprüft und einzelne Entwurfsprobleme werden gelöst. Gegebenenfalls müssen neue Datenobjekte eingeführt werden.

Beim Prüfteileinlagerungssystem lassen sich anhand der API-Spezifikation und des alten Datenmodells die vier Datenobjekte Qualitätsprüfteil, Firma, Drucker und Lagerort identifizieren. Die Standardattribute wie beispielsweise der Name einer Firma oder die Prüfteilnummer eines Qualitätsprüfteils lassen sich hierbei übernehmen. Anschließend ist zu klären, wie die Beziehung zwischen Qualitätsprüfteil und Drucker bzw. Firma realisiert wird. Im alten Datenmodell, welches in Abbildung 3.3 zu sehen ist, wurde diese mit einem Fremdschlüssel innerhalb des Datenobjektes des Qualitätsprüfteils gelöst. Es besteht die einfache Möglichkeit, die Tabellen mit den Referenzen unverändert zu übernehmen. Allerdings sind die Operationen zum Zusammenführen der Datensätze bei Not only SQL (NoSQL) Systemen nicht im gleichen Maße optimiert, wie es bei relationalen Datenbanken der Fall ist. Dementsprechend wird in der Literatur zum Ansatz der Denormalisierung geraten, welcher zu Duplikaten in den Daten führen kann, jedoch schnellere Abfragezeiten gewährleistet (Dai, 2019, S. 2-3).

Auch nach Boeker (2010, S. 49) ist der Ansatz, Referenzen durch Embedding zu ersetzen, vorzuziehen. Erzeugen die eingebetteten Informationen jedoch vollständige Redundanz, sollten sie ausgelagert werden. Sind die Informationen nur teilweise redundant, so ist es legitim, diese einzubetten (Boeker, 2010, S. 107).

Unter Berücksichtigung dieser Regeln werden die Firma und der Drucker nicht in das Datenobjekt des Qualitätsprüfteils eingebettet und werden mit einem Fremdschlüssel referenziert, weil sonst Redundanz entstehen kann. Zusätzlich gibt es Anwendungsfälle, bei denen auf alle Drucker bzw. Firmen zugegriffen werden soll. Dabei kann es zu aufwändigen Abfragen kommen, wenn diese Datenobjekte eingebettet und nicht ausgelagert sind.

Abschließend gilt es festzulegen, wie der Lagerort eines Prüfteils in der Datenbank gespeichert werden soll. Um einen einfacheren Einlagerungsprozess implementieren zu können, werden die Datenobjekte Box und Lagerort aus dem alten Datenmodell zusammengefasst. Weiterhin stellt sich die Frage, ob dieser vollständige Lagerort, bestehend aus dem Standort, dem Gebäude, der Schranknummer und der Boxnummer, in das Prüfteilobjekt eingebettet oder wieder als separates Datenobjekt ausgelagert werden soll. Da in eine Box nur ein Prüfteil eingelagert werden kann und somit ein im Prüfteilobjekt eingebetteter Lagerort nur teilweise redundante Informationen erzeugen kann, wird hier das Einbetten bevorzugt.

Abbildung 3.6 zeigt das neue Datenmodell. Ein weiterer Unterschied neben der Umstrukturierung des Lagerorts ist die Speicherung der Messdateien in der Datenbank, welche mit einem Fremdschlüssel im Prüfteilobjekt referenziert werden.

Die Datenmodelle der restlichen Anwendungsteile werden im Anhang E aufgeführt.

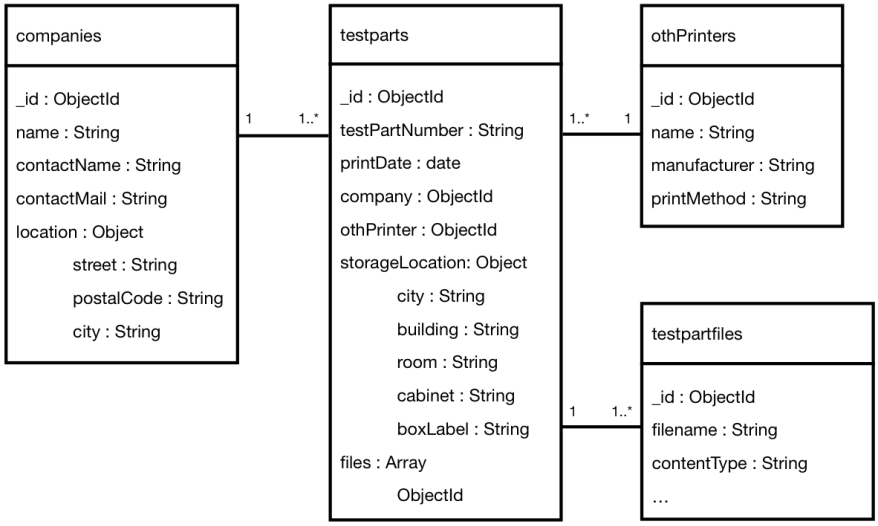


Abbildung 3.6: Neues Datenmodell des Prüfteileinlagerungssystems

3.3.6 Implementierung des Backends

Nach der Erstellung des Datenmodells kann die Implementierungsphase gestartet werden. Diese beginnt mit der Entwicklung des Backends. Die MongoDB-Datenbank wird angelegt und der Node.js-Server, welcher die REST-API bereitstellt, wird implementiert.

Der API-First Ansatz bringt hierbei mehrere Möglichkeiten mit sich, den Entwicklungsprozess zu beschleunigen. Zum einen bietet die API-Spezifikation eine gute Orientierungsmöglichkeit, an welche sich während der Implementierung gehalten werden muss. Zum anderen kann die Entwicklung durch automatische Generierungen beschleunigt werden. Das Tool oas-generator ermöglicht beispielsweise eine automatische Generierung einer umfassenden Projekt- bzw. Ordnerstruktur für einen Node.js-Server basierend auf der API-Spezifikation (Fernandez et al., 2018). Für Java, PHP oder andere Technologien bietet das Tool Swagger Codegen die Möglichkeit, ein grobes Code-Gerüst für das Projekt zu generieren. Hiermit können auch Client-Bibliotheken für mehrere Programmiersprachen generiert werden (Cheng et al., 2020). Aufgrund der mangelnden Konfigurationsmöglichkeiten dieser Generatoren wird hier von einer Benutzung abgesehen. Für die gewünschte Ordnerstruktur müssen viele Anpassungen getätigt werden und das nachträgliche Hinzufügen eines Endpunktes ist umständlich. In diesem Fall ist eine neue und vollständige Generierung nötig, wobei beispielsweise bereits implementierte Controller-Dateien überschrieben werden.

Eine andere Möglichkeit, die Entwicklungsphase zu beschleunigen, ist die Generierung von Tests, welche sich bereits zu Beginn der Implementierungsphase auf Basis der API-Spezifikation erzeugen lassen. Hierfür wird das Tool OpenAPI Test Templates verwendet, mit dem Code-Gerüste für Tests von allen Endpunkten erstellt werden.

Abbildung 3.7 zeigt die Ordnerstruktur des Node.js-Servers. In diesem Ordner befinden sich die Dateien zur Verwaltung der eingebundenen Module und die Datei zum Starten des Servers. Ebenso befindet sich hier das Spezifikationsdokument der API. Innerhalb

des Ordners `app/` werden in `config/` die Konfigurationsdateien wie beispielsweise der Verbindungsstring der Datenbank abgelegt. Im Ordner `controller/` befinden sich die Controller-Dateien. In `model/` werden die Objekte, welche die Datenstrukturen aus dem Datenmodell repräsentieren, abgelegt und in `routes/` werden die Endpunkte der bereitgestellten API festgelegt. Die Dateien in diesen drei Verzeichnissen werden jeweils nach Anwendungsteil in einem Ordner gruppiert. Die Tests werden in `test/` abgelegt.

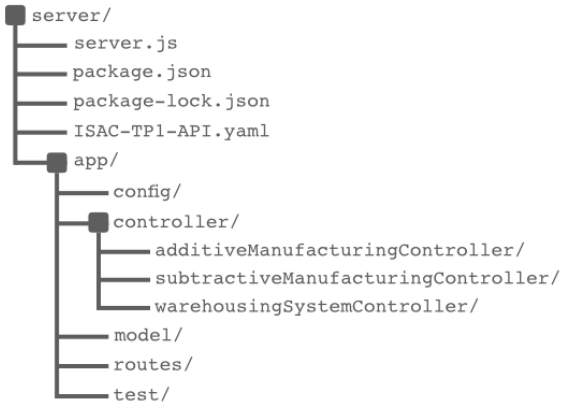


Abbildung 3.7: Ordnerstruktur des Node.js-Servers

Zuerst werden die Model-Dateien anhand des aufgestellten Datenmodells erzeugt. Anschließend werden die Endpunkte der API angelegt und die jeweiligen Controller-Funktionen implementiert. Hierbei können manche Algorithmen aus den alten Systemen wiederverwendet werden. So werden beispielsweise die Formeln und Rechenschritte zur Berechnung der Kennzahlen im Rahmen der subtraktiven Fertigung übernommen, die in der Vergangenheit bereits aufgestellt wurden.

Der komplette Quellcode des Servers und der Client-Applikation wird im Anhang A zur Verfügung gestellt.

3.3.7 Contract-Testing der API-Implementierung

Das Contract-Testing der API-Implementierung ist ein wichtiger Schritt im Überführungsprozess. Hierbei wird verifiziert, dass die Implementierung der API mit der zuvor entworfenen Spezifikation übereinstimmt. Somit wird überprüft, ob der Contract, welcher zu Beginn aufgestellt wurde, eingehalten wird.

Um das Contract-Testing durchzuführen, wird der Prism Validation Proxy verwendet. Dieser wird mit der entworfenen API-Spezifikation initialisiert. Anfragen an den Proxy werden automatisch an den Server weitergeleitet und die Antworten des Servers laufen wiederum über den Proxy. Hierbei wird überprüft, ob die Antwortschemas der API mit den Vorgaben aus der eingelesenen API-Spezifikation übereinstimmen. Ist dies nicht der Fall, wird ein Fehler geworfen, wie es in Abbildung 3.8 zu sehen ist.

Die Validierung der Endpunkte kann jeweils nach der Implementierung manuell durchgeführt werden. Um während des gesamten Entwicklungsprozesses durchgehend zu

prüfen, ob die Implementierung der API noch mit der Spezifikation übereinstimmt, kann der Prozess automatisiert werden, indem bei den API-Tests die Anfragen über den Proxy getätigt werden. Eine andere Möglichkeit ist, bei der anschließenden Entwicklung des Frontends den Prism Proxy als Server anzugeben, an den die Anfragen des Clients geschickt werden. Während der Entwicklung und dem Test der Client-Applikation kann somit nebenbei die Validität der API-Implementierung sichergestellt werden.

```
[10:58:20] › [CLI] ▶  start     Prism is listening on http://127.0.0.1:4010
[10:58:50] › [HTTP SERVER] get /warehousingsystem/othprinters ℹ  info      Request received
[10:58:50] ›     [PROXY] ℹ  info      Forwarding "get" request to http://localhost:8080/warehousingsystem/othprinters...
[10:58:50] ›     [PROXY] ℹ  info      The upstream call to /warehousingsystem/othprinters has returned 200
[10:59:08] › [HTTP SERVER] get /warehousingsystem/companies ℹ  info      Request received
[10:59:08] ›     [PROXY] ℹ  info      Forwarding "get" request to http://localhost:8080/warehousingsystem/companies...
[10:59:08] ›     [PROXY] ℹ  info      The upstream call to /warehousingsystem/companies has returned 200
[10:59:08] ›     [VALIDATOR] ✖  error     Violation: response.body.[1].contactName should be string
[10:59:08] ›     [VALIDATOR] ✖  error     Violation: response.body.[1].contactMail should be string
[10:59:08] ›     [VALIDATOR] ✖  error     Violation: response.body.[2].contactMail should be string
```

Abbildung 3.8: Validierung der eingehenden Anfragen durch Prism

3.3.8 Implementierung des Frontends

Abschließend wird das Frontend, bestehend aus der Vue.js-Applikation, implementiert. Der Entwurf und die Implementierung der Benutzeroberfläche erfolgen hierbei unter Beachtung der aufgestellten Anforderungen.

Abbildung 3.9 zeigt die Verzeichnisstruktur der Frontend-Anwendung, welche mit Vue CLI generiert und anschließend angepasst wurde. Im Ordner `public/` werden eingebundene Bilder und die `index.html`-Datei abgelegt. Innerhalb des Ordners `src/` werden neben der zentralen Vue-Komponente die Assets und andere Komponenten gespeichert. Komponenten, die vollständige Seiten der Anwendung repräsentieren, werden im Verzeichnis `pages/` abgelegt. In `services/` werden die Funktionen zum Ausführen der HTTP-Anfragen, welche an den Server geschickt werden, verwaltet.

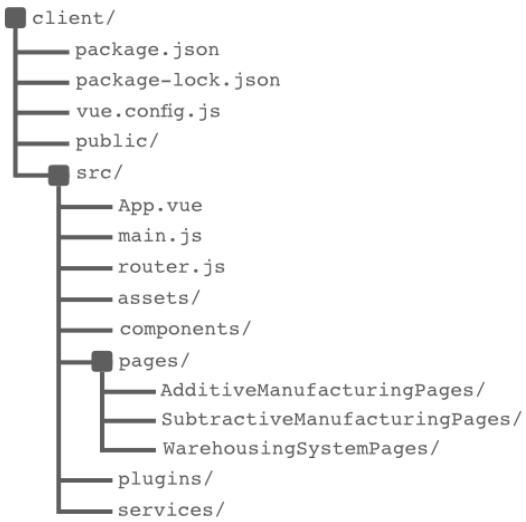


Abbildung 3.9: Ordnerstruktur der Vue.js-Applikation

Abbildung 3.10 zeigt beispielsweise die Benutzeroberfläche des Systems beim Anlegen eines neuen Qualitätsprüfteils. Hierbei ist der vereinfachte Einlagerungsprozess für Prüfteile zu sehen. Anstatt auf einer separaten Seite den Lagerort und wiederum auf einer anderen Seite die Box erstellen zu müssen, ist hier die Einlagerung ohne vorheriges Anlegen des Lagerorts möglich. Der Benutzer wird bei der Einlagerung des Prüfteils schrittweise durch die Komponenten des Lagerorts geführt, wobei eine Auswahl aus den kummulierten Lagerortkomponenten, welche bei bereits eingelagerten Prüfteilobjekten hinterlegt sind, möglich ist. Wenn hierbei kein passender Lagerort zur Auswahl steht, kann dieser manuell eingegeben werden. Er wird dann im Prüfteilobjekt gespeichert und steht somit für kommende Einlagerungen zur Auswahl.

Im Anhang F wird die Benutzeroberfläche bei Navigation zu anderen Anwendungsteilen gezeigt.

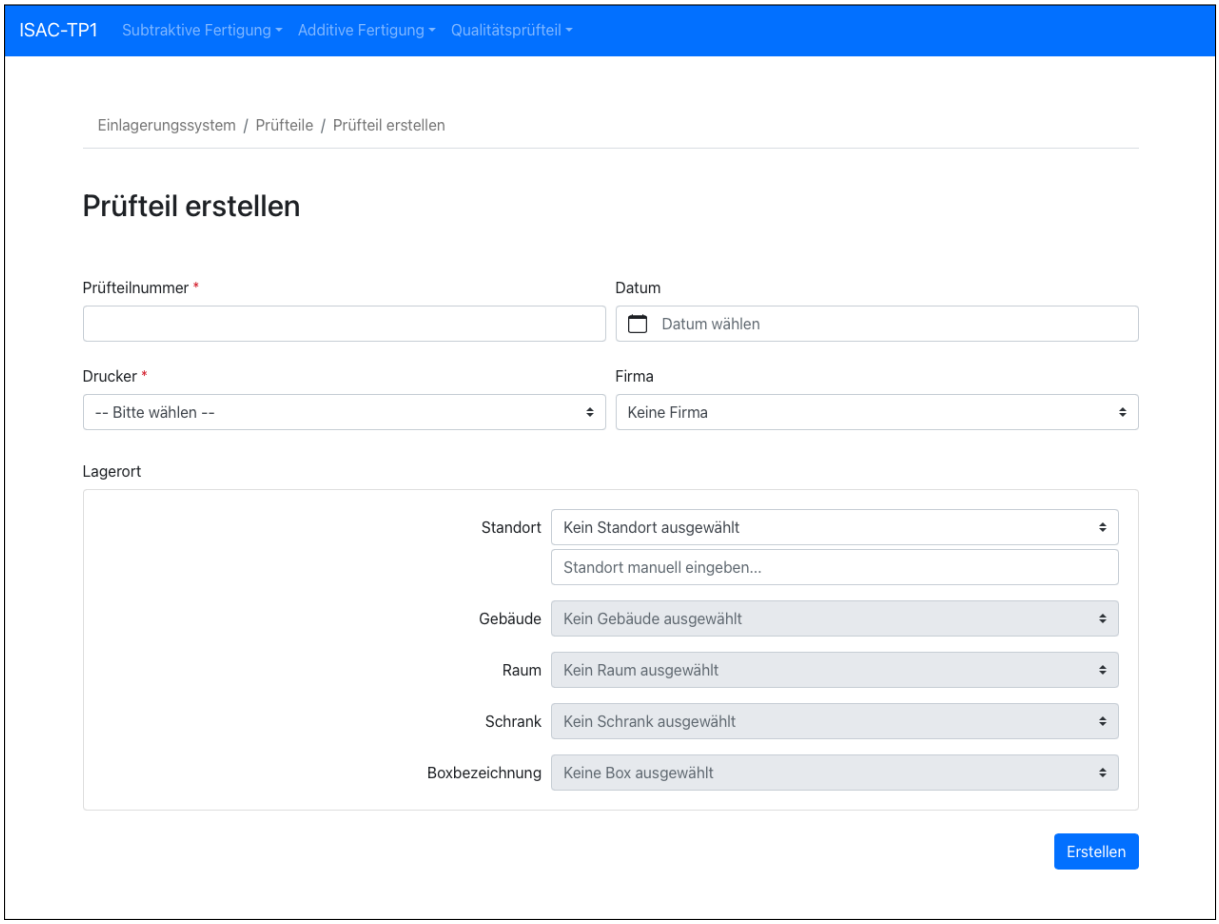


Abbildung 3.10: Formular zum Erstellen eines Qualitätsprüfteils

Kapitel 4

Evaluierung anhand des Qualitätsmodells nach ISO/IEC 25010

In diesem Kapitel wird eine Evaluierung des entwickelten Systems durchgeführt, um die Auswirkungen von API-First zu veranschaulichen. Hierfür wird ein Modell zur Bewertung von Softwarequalität verwendet, das zu Beginn des Kapitels beschrieben wird. Im Anschluss wird auf Basis dieses Modells die Evaluierung durchgeführt.

4.1 Beschreibung des Qualitätsmodells

ISO/IEC JTC 1/SC 7 Software and systems engineering (2011, S. 1) definiert im Standard ISO/IEC 25010:2011 das zu verwendende Qualitätsmodell. In dieser Norm werden zwei Qualitätsmodelle definiert. Beide stellen Leitkriterien auf, anhand welcher eine Software oder ein System bezüglich der Benutzungs- oder Produktqualität bewertet werden kann. Diese Leitkriterien sind wiederum in untergeordnete Kriterien aufgeteilt. Mit dem ersten Modell kann die Qualität der Benutzung einer Software bewertet werden. Beim zweiten Modell steht die allgemeine Produktqualität der Software im Vordergrund (ISO/IEC JTC 1/SC 7 Software and systems engineering, 2011, S. 1).

Das zweite Modell wird im Rahmen der Arbeit genutzt, um das entwickelte System anhand der Leitkriterien zu untersuchen und zu bewerten. Es soll überprüft werden, an welchen Stellen der API-First Ansatz die Bewertung positiv beeinflusst.

Die folgenden Leitkriterien werden in dem Qualitätsmodell aufgestellt:

1. **Funktionalität:** Der Grad, zu dem ein System Funktionen bereitstellt, welche die angegebenen und implizierten Anforderungen erfüllen, wenn es unter bestimmten Bedingungen verwendet wird.
2. **Leistungsfähigkeit:** Der Grad der Leistungsfähigkeit eines Systems im Verhältnis zur Menge der verwendeten Ressourcen unter angegebenen Bedingungen.
3. **Kompatibilität:** Der Grad, zu dem ein System oder eine Komponente Informationen mit anderen Systemen oder Komponenten austauschen kann.

4. **Benutzbarkeit:** Der Grad, zu dem ein System von bestimmten Anwendern benutzt werden kann, um bestimmte Ziele mit Effektivität, Effizienz und Zufriedenheit in einem bestimmten Nutzungskontext zu erreichen.
5. **Zuverlässigkeit:** Der Grad, zu dem ein System oder eine Komponente bestimmte Funktionen unter bestimmten Bedingungen für eine bestimmte Zeitspanne ausführt.
6. **Sicherheit:** Der Grad, zu dem ein System Daten und Informationen schützt, sodass Personen oder andere Systeme einen Datenzugriff nach ihren Berechtigungen haben.
7. **Wartbarkeit:** Der Grad der Effektivität und Effizienz, mit dem ein System von den vorgesehenen Instandhaltern verändert werden kann.
8. **Portabilität:** Der Grad der Effektivität und Effizienz, mit dem ein System oder eine Komponente von einer Hardware-, Software- oder anderen Betriebs- und Nutzungsumgebung in eine andere übertragen werden kann (ISO/IEC JTC 1/SC 7 Software and systems engineering, 2011, S. 10-16).

Diese Leitkriterien lassen sich in folgende, untergeordnete Kriterien aufteilen, wie es in Abbildung 4.1 gezeigt wird.

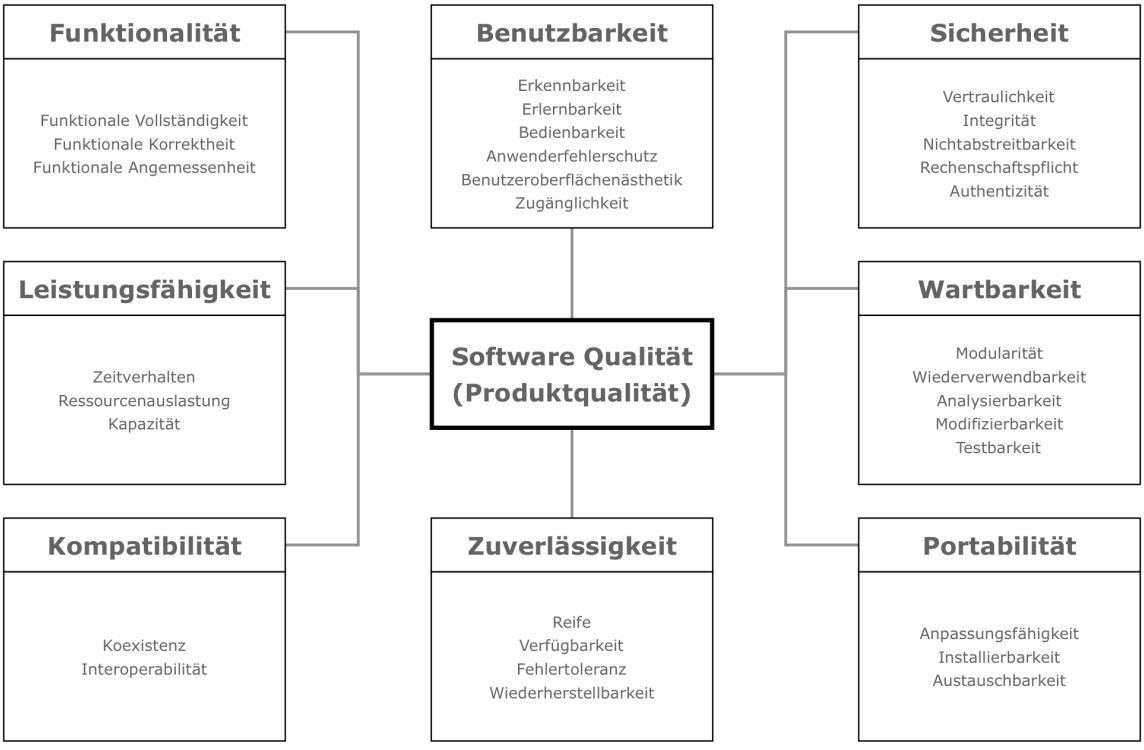


Abbildung 4.1: Qualitätskriterien für Produktqualität nach ISO/IEC JTC 1/SC 7 Software and systems engineering (2011, S. 10-16)

4.2 Anwendung des Qualitätsmodells auf das System

Das beschriebene Qualitätsmodell listet Kriterien in Form eines Art Leitfadens auf, welche für die Gewährleistung der Qualität eines Softwareprodukts zu berücksichtigen sind. Mit dieser Evaluierung soll untersucht werden, an welchen Stellen der API-First Ansatz einen positiven Einfluss auf diese Qualitätskriterien hat und somit die Softwarequalität verbessert.

Für die Evaluierung werden die Qualitätskriterien in einen Bewertungsbogen eingegliedert, wobei die Umsetzung des jeweiligen Kriteriums im System bewertet wird. Es werden die möglichen Werte *negativ*, *neutral* und *positiv* eingeführt, um die Bewertung durchzuführen. *Negativ* bedeutet, dass das jeweilige Kriterium nicht in einem zufriedenstellenden Maße umgesetzt ist. *Positiv* ist die Bewertung nach dem jeweiligen Kriterium, wenn das System entsprechende Merkmale im Hinblick auf die Forderungen des Qualitätskriteriums aufweist. *Neutral* bedeutet, dass die Umsetzung des Kriteriums nicht genau bestimmt werden kann bzw. weder *negativ* noch *positiv* einzuschätzen ist.

Der Bewertungsbogen wird auf das alte System, bestehend aus den drei Anwendungsteilen sowie auf das neue System angewendet. Hierbei werden die Client-Applikation und der Server zusammen betrachtet. Das komplette Ergebnis ist dem Anhang G zu entnehmen. Tabelle 4.1 zeigt beispielhaft die Struktur des Bewertungsbogens anhand einer Zeile. In der ersten Spalte ist das jeweilige Qualitätskriterium aufgeführt, dessen Beschreibung in der zweiten Spalte von links gegeben wird. Daneben befinden sich die drei Spalten für die Bewertung. Abschließend folgt eine Spalte, in welcher Begründungen oder Kommentare aufgeführt werden können.

Kriterium	**Beschreibung**	**Negativ**	**Neutral**	**Positiv**	**Bemerkung**
functional completeness	degree to which the set of functions covers all the specified tasks and user objectives			X	Die spezifizierten Funktionen sind umgesetzt. Das wird durch Contract-Testing verifiziert. Hierbei wird überprüft, ob die Implementierung der API mit der Spezifikation, welche die Soll-Funktionalität des Systems widerspiegelt und als Contract fungiert, übereinstimmt.

Tabelle 4.1: Struktur des Bewertungsbogens

Das neu entwickelte System erhält im Vergleich zum alten System bessere Bewertungen. Das liegt zum Teil an der Entwicklung nach dem API-First Ansatz. API-First etabliert einige Praktiken, welche auf mehrere Qualitätskriterien des Modells einen positiven Einfluss haben:

- **Funktionalität:** Beim API-First Ansatz kann die API-Spezifikation die Rolle eines Contracts einnehmen, welcher unter Berücksichtigung der Anforderungen aufgestellt wird. Durch das Contract-Testing wird am Ende der Implementierungsphase sowie zu jedem zukünftigen Zeitpunkt geprüft, ob die Implementierung der API mit der Spezifikation bzw. dem Contract übereinstimmt. Somit wird funktionale Vollständigkeit und Korrektheit gewährleistet.

- **Kompatibilität:** Durch das Verwenden des API-First Ansatzes werden Anforderungen des Kriteriums der Kompatibilität erfüllt. Nach dem API-First Development Ansatz wird vorausgesetzt, dass jede entwickelte Funktionalität immer zuerst über eine Schnittstelle bereitgestellt werden muss. Somit besteht die Möglichkeit, dass andere Systeme auf die Daten und Funktionen der eigenen Anwendung zugreifen können.
- **Wartbarkeit:** Der API-First Ansatz hat positive Effekte auf die Wartbarkeit. Wird nach API-First entwickelt, so wird für ein neues oder anzupassendes Feature zuerst der Entwurf der API durchgeführt und evaluiert, bevor das Feature implementiert wird. Mit anschließendem Contract-Testing wird gewährleistet, dass nach jeder Anpassung die Implementierung noch mit der Spezifikation übereinstimmt und der Contract nicht durch neue Änderungen gebrochen wird. Ein Nachteil davon ist, dass eine Modifikation des Systems mit mehr Aufwand verbunden ist, da der komplette Entwurfs- und Entwicklungsprozess durchlaufen werden muss. Allerdings wird durch automatische Generierungen die Möglichkeit geschaffen, an manchen Stellen Entwicklungszeit einzusparen.

Weil in dem Bewertungsbogen im Bezug auf das Kriterium der Usability eine Bewertung der graphischen Benutzeroberfläche stattgefunden hat, wurde die Benutzbarkeit der API nicht evaluiert. Auch auf dieses Kriterium hat der API-First Ansatz einen positiven Einfluss. Da der Entwurf der API früh im Entwicklungsprozess erfolgte, wurde er nicht auf Basis einer Implemenierung umgesetzt, sondern konnte nutzerorientiert durchgeführt werden. Durch einen Blick auf die API-Dokumentation, welche auf Basis der Spezifikation generiert wurde, erhält ein neuer Benutzer einen Einblick in die Funktionalität der API und allen verfügbaren Endpunkten. Somit wird er schnell mit der Funktionsweise der API vetraut und kann über moderne Dokumentations-Tools API-Anfragen ausprobieren. Dadurch wird die Lernkurve abgeflacht.

Zusammenfassend lässt sich sagen, dass der API-First Ansatz neben der Benutzbarkeit der API auch einige Kriterien des beschriebenen Qualitätsmodells positiv beeinflusst. Mit dem Ansatz werden dem Entwickler Praktiken an die Hand gegeben, die in direkter Weise die Softwarequalität des zu entwickelnden Systems verbessern.

Kapitel 5

Fazit

In der Arbeit wurde das Konzept des API-First Ansatzes dargestellt. Es wurde beispielhaft beschrieben, wie dieser sich bei der Überführung mehrerer Anwendungen in ein neues System in den Entwicklungsprozess eingliedern lässt. Weiterhin wurde untersucht, welche Auswirkungen dieser Entwicklungsansatz auf das Projekt hat.

In Kapitel 2 wurden die Grundlagen der Arbeit erklärt, wobei die Ausgangslage und die theoretischen Grundlagen des API-First Ansatzes dargelegt wurden. Anschließend wurde in Kapitel 3 die durchgeführte Entwicklung des Systems beschrieben. Zu Beginn dieses Kapitels wurde sich mit der Einführung eines einheitlichen Softwarestacks beschäftigt. Daraufhin erfolgte die Aufstellung der Überführungs- bzw. Entwicklungsstrategie, wobei untersucht wurde, wie sich die Praktiken des API-First Ansatzes in diesem Prozess unterbringen ließen. Im weiteren Verlauf dieses Kapitels wurde die praktische Umsetzung der aufgestellten Überführungsstrategie diskutiert. Praktiken im Zusammenhang mit dem API-First Ansatz wurden hierbei früh im Entwicklungsprozess angewandt. Nach einer Planungsphase wurde in der Entwurfsphase sehr viel Zeit in den Entwurf der API investiert. Als zentrales Dokument entstand hierbei die API-Spezifikation, welche die Funktionsweise der API beschreibt und alle möglichen Endpunkte mit den nötigen Anfrage- und Antwortschemas auflistet. Im Anschluss folgte die Implementierungsphase, wobei einige Entwicklungsschritte durch den API-First Ansatz beschleunigt wurden. In Kapitel 4 lag der Fokus auf der Evaluation des Systems.

Auf die Frage nach dem Einfluss des API-First Ansatzes auf das Projekt lässt sich zusammenfassend sagen, dass der Ansatz mehrere Vorteile auf Kosten von größerem Planungs- und Entwurfsaufwand hat. Nach der damit verbundenen Entwicklungsstrategie werden viele Praktiken etabliert, die eine positive Auswirkung auf die Softwarequalität nach dem in Kapitel 4 beschriebenen Qualitätsmodell haben. Da die beim frühen API-Entwurf entstandene API-Spezifikation einen guten Überblick über die gesamte Funktionsweise des Systems gibt, besteht die Möglichkeit, frühes Feedback zu erhalten, um früh im Entwicklungsprozess große Fehler in dem Entwurf der API bzw. der allgemeinen Funktionsweise des Systems aufzudecken und auszubessern. Dadurch, dass der API-Entwurf nicht auf Basis einer bestehenden Implementierung

umgesetzt wird, kann er nutzerorientiert durchgeführt werden, um eine gut durchdachte API zu erzeugen, welche einheitlich, leicht verständlich und passend für die gewünschte Funktionsweise entworfen ist. Mit der erstellten API-Spezifikation lassen sich anschließende Entwicklungsschritte beschleunigen, indem beispielsweise Tests oder die Dokumentation generiert werden. Zudem ist festzuhalten, dass durch diesen Ansatz alle Funktionen des Systems stets über eine Schnittstelle bereitgestellt werden, wodurch beliebige Client-Applikationen bzw. andere Systeme die zur Verfügung gestellten Daten und Funktionen nutzen können.

Trotz der beschriebenen Vorteile konnte das Potenzial des API-First Ansatzes im Rahmen dieser Arbeit nicht vollständig ausgenutzt werden. Das durch Mock-ups ermöglichte parallele Arbeiten mehrerer Entwicklungsteams war aufgrund des Fehlens anderer Entwickler nicht möglich. Dieses parallele Arbeiten würde es ermöglichen, viel Entwicklungszeit zu sparen, womit der zeitaufwändige Entwurf der API ausgeglichen werden kann.

Zusätzlich ist festzuhalten, dass der fortgeschrittene Forschungsstand des Projektes für die Wahl des API-First Entwicklungsansatzes unabdingbar war. In einer früheren, experimentellen Phase des Forschungsprojektes, in der sich die Anforderungen bzw. der Funktionsumfang jederzeit ändern konnten, hätte die Verwendung des API-First Ansatzes einen nicht tragbaren Organisations- und Entwurfsaufwand nach sich gezogen. Da in dem hier dargestellten Fall das Forschungsprojekt inhaltlich sehr fortgeschritten war und die Anforderungen feststanden, war eine Entwicklung nach dem API-First Ansatz vertretbar.

Abschließend lässt sich sagen, dass der API-First Ansatz vor allem in Projekten mit einem größeren Umfang und mehreren externen Abhängigkeiten sowie verschiedenen Entwicklungsteams genutzt werden sollte. In solch einem Fall lohnt sich der erhöhte Aufwand für den Entwurf und das sonstige Potenzial des API-First Ansatzes kann komplett ausgeschöpft werden. Ebenso wichtig ist, dass der Projektumfang und die Anforderungen zu Beginn feststehen und sich im weiteren Verlauf der Entwicklung so wenig wie möglich ändern, weil anderenfalls diese Anforderungsänderungen schwerwiegende und zeitlich aufwändige Anpassungen des API-Entwurfs nach sich ziehen können.

Literaturverzeichnis

Boeker, M. (2010). *MongoDB: sag ja zu NoSQL*. Frankfurt am Main: Entwickler.Press.

Cheng, W., Tam, T., Tumanischvili, F. & Ratovsky, R. (2020, 29. Dezember). *Swagger Codegen.* Zugriff am 21.02.2021 auf `https://github.com/swagger-api/swagger-codegen/blob/master/README.md`

Chianese, V., Sturgeon, P., Conrad, C. & Maciaszek, K. (2020, 24. September). *HTTP Mocking.* Zugriff am 21.02.2021 auf `https://github.com/stoplightio/prism/blob/master/docs/guides/01-mocking.md`

Chianese, V., Sturgeon, P. & Maciaszek, K. (2020, 17. September). *Validation Proxy.* Zugriff am 21.02.2021 auf `https://github.com/stoplightio/prism/blob/master/docs/guides/03-validation-proxy.md`

Dai, J. (2019, März). SQL to NoSQL : What to do and How. *IOP Conference Series: Earth and Environmental Science*, *234*, 012080. doi: 10.1088/1755-1315/234/1/012080

De, B. (2017). *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization*. Apress. doi: 10.1007/978-1-4842-1305-6

Dietz, N. & Uzlopak. (2019, 28. Mai). *OpenAPI Test Templates (oatts).* Zugriff am 21.02.2021 auf `https://github.com/google/oatts/blob/master/README.md`

Doglio, F. (2018). *Scaling Your Node.js Apps: Progress Your Personal Projects to Production-Ready*. Apress. doi: 10.1007/978-1-4842-3991-9

Fernandez, P., Molina, P. J., Fresno, R. & Lozada, I. P. (2018, 28. November). *oas-generator.* Zugriff am 21.02.2021 auf `https://github.com/isa-group/oas-generator/blob/master/README.md`

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures* (Unveröffentlichte Dissertation). University of California, Irvine.

Gerlang, B. (2019a). *Industry Software Application Center an der Ostbayerischen Technischen Hochschule Amberg-Weiden.* Zugriff am 26.02.2021 auf `https://www.isac-oth.de/`

Gerlang, B. (2019b). *Teilprojekte.* Zugriff am 26.02.2021 auf `https://www.isac-oth.de/arbeitspakete/`

Heidenreich, C., Hahn, K., Stoikovitch, J. & Ralphson, M. (2019, 6. Juni). *RAML Version 1.0: RESTful API Modeling Language.* Zugriff am 21.02.2021 auf `https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md`

ISO/IEC JTC 1/SC 7 Software and systems engineering. (2011). *ISO/IEC 25010:2011: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. Schweiz.

Jin, B., Sahni, S. & Shevat, A. (2018). *Designing Web APIs: Building APIs That Developers Love* (1. Aufl.). Kalifornien: O'Reilly Media, Inc.

Krzyk, M. (2020, 5. Juni). *Surprisingly Simple Tools to Help You Smash API-First Approach.* Zugriff am 10.12.2020 auf https://dzone.com/articles/surprisingly-simple-tools-to-help-you-smash-api-fi

Levin, G. (2019, 4. Juli). *OpenAPI Spec: Documentation and Beyond.* Zugriff am 12.11.2020 auf http://blog.restcase.com/openapi-documentation-and-beyond/

Lin, J. (2018, 19. September). *API-first software development for modern organizations.* Zugriff am 29.12.2020 auf https://medium.com/better-practices/api-first-software-development-for-modern-organizations-fdbfba9a66d3

MongoDB, Inc. (o. J.). *What Is MongoDB?* Zugriff am 26.02.2021 auf https://www.mongodb.com/what-is-mongodb

Mulloy, B. (2012). *Web API Design - Crafting Interfaces that Developers Love*. Apigee.

Nemec, Z. (2019, 12. September). *API Blueprint.* Zugriff am 21.02.2021 auf https://github.com/apiaryio/api-blueprint/blob/master/API%20Blueprint%20Specification.md

Nitze, A. (2018, 19. April). Contract-first Development – Entwurf und Implementierung nachhaltiger API-Angebote. In A. Schmietendorf & A. Nitze (Hrsg.), *API-First/API-Management - Open APIs als Treiber der Digitalisierung* (Bd. 18, S. 1–6). Hamburg: Shaker Verlag.

OpenJS Foundation. (o. J.). *Express.* Zugriff am 26.02.2021 auf https://expressjs.com/

Ratovsky, R. & Miller, D. (2017, 26. Juli). *OpenAPI Specification.* Zugriff am 21.02.2021 auf https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md

Riggings, J. (2015, 10. Juli). *How To Design Great APIs With API-First Design.* Zugriff am 13.01.2021 auf https://www.programmableweb.com/news/how-to-design-great-apis-api-first-design-and-raml/how-to/2015/07/10

Rogers, M., Hemberger, F., Senkpiel, J. & You, C. (2020, 8. Januar). *About Node.js.* Zugriff am 26.02.2021 auf https://github.com/nodejs/nodejs.org/blob/master/locale/en/about/index.md

Stowe, M. (2015). *Undisturbed REST: A Guide to Designing the Perfect API*. Lulu.com.

Vasudevan, K. (o. J.). *Best Practices in API Design.* Zugriff am 23.12.2020 auf https://swagger.io/resources/articles/best-practices-in-api-design/

Wagner, J. (o. J.-a). *Understanding the Differences Between API Documentation, Specifications, and Definitions.* Zugriff am 02.02.2021 auf https://swagger.io/resources/articles/difference-between-api-documentation-specification/

Wagner, J. (o. J.-b). *Understanding the API-First Approach to Building Products.* Zugriff am 01.11.2020 auf https://swagger.io/resources/articles/adopting-an-api-first-approach/

You, E. (o. J.). *Introduction — Vue.js.* Zugriff am 26.02.2021 auf https://vuejs.org/v2/guide/

Anhang A

Digitaler Anhang

A.1 Inhalt digitaler Anhang

Bachelorarbeit-Schimmer.pdf	Die Bachelorarbeit in digitaler Form
isac-tp1/	Das Softwareprojekt
l client/	Die Vue.js-Anwendung
l db/	Ein MongoDB-Dump
l server/	Der Node.js-Server

A.2 Download digitaler Anhang

ba.jschimmer.de

Anhang B

Ergebnisse der Featureanalyse

B.1 Featureanalyse Subtraktive Fertigung

Feature	Beschreibung	Systemfehler
Verwaltung Prozesse	Erstellen, Lesen, Ändern von Prozessen	
Verwaltung Ansprechpartner	Erstellen, Lesen, Ändern von Ansprechpartnern	
Verwaltung Hersteller	Erstellen, Lesen, Ändern von Herstellern. Kategorisieren nach Maschinen-/Werkzeug-/Werkstoffhersteller	
Verwaltung Maschinen	Erstellen, Lesen, Ändern von Maschinen mit Angabe der Eckdaten. Verweis auf Maschinenhersteller	
Verwaltung Werkzeuge	Erstellen, Lesen, Ändern von Werkzeugen mit Angabe der Eckdaten. Verweis auf Werkzeughersteller. Vergleich der Listenpreise in einem Balkendiagramm	Materialeignung kann ausgewählt werden, wird aber nicht übernommen und zusätzlich falsch angezeigt. Unbekannter Fehler beim Versuch, ein Werkzeug zu erstellen.
Verwaltung Werkstoffe	Erstellen, Lesen, Ändern von Werkstoffen. Verweis auf Werkstoffhersteller	
Verwaltung Versuchsdaten	Erstellen neuer Versuchsdaten zu einem durchgeführten Prozess. Angabe von Maschine, Werkzeug, Schnittparametern und weiteren Prozessdaten	Eingetragene Versuchsdaten sind nicht einsehbar. Der Verweis auf Ansprechpartner und Bearbeitungsstrategie ist fehlerhaft.
Datenbankabfrage	Angabe der Randbedingungen des durchzuführenden Prozesses durch Benutzer. Berechnung der KPI1-KPI5, Darstellung der Kosten und des Werkzeugbedarfs	Berechnungen funktionieren nicht mit allen Materialien. Mehrere kleine Fehler nach dem Durchführen der Datenbankabfrage.

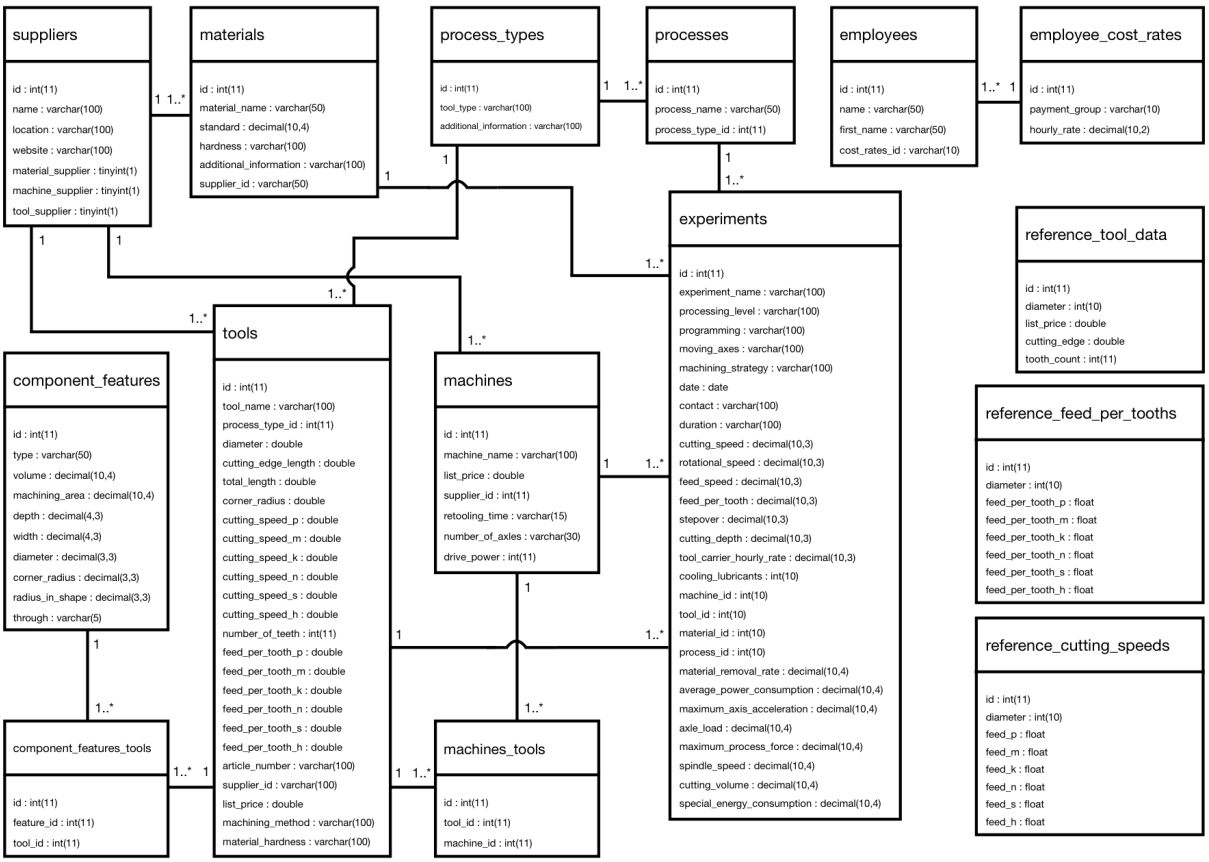
B.2 Featureanalyse Additive Fertigung

Feature	Beschreibung	Systemfehler
Upload und Analyse einer STL-Datei	Analyse einer STL-Datei nach Volumen, Hüllquader und weiteren Daten	Dateigröße ist begrenzt. Dateien mit gängigen Dateigrößen können nicht analysiert werden.
Drucker- und Materialauswahl	Auswahl eines Druckers anhand eines Hüllquaders und anschließende Auswahl eines Materials	
Berechnung der Zwischenergebnisse	Berechnung der Hüllquaderausnutzung des Bauteilvolumens und der geschätzten Bauzeit mit einem ausgewählten Drucker	
Berechnung der Endergebnisse	Nach Bestätigung der Kalkulationsparameter (Druckerkosten, Personaleinsatz, Ressourcenkosten) werden folgende Kosten berechnet: Gesamtkosten, Aufschlüsselung der Kosten in Maschinenkosten, Ressourcenkosten, Personalkosten	

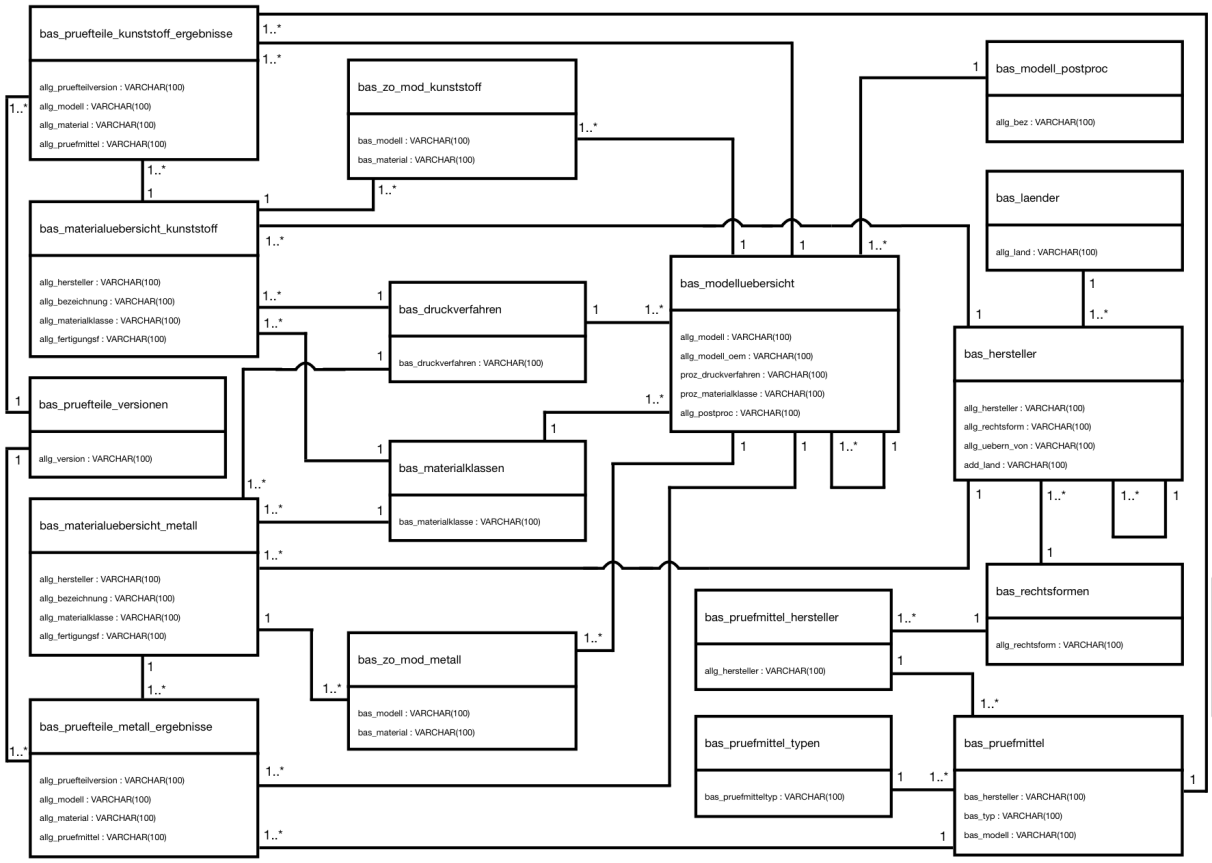
Anhang C

Datenmodelle des alten Systems

C.1 Datenmodell Subtraktive Fertigung



C.2 Datenmodell Additive Fertigung



Anhang D

Ergebnisse der Anforderungsanalyse

Ausgegraute Anforderungsteile beinhalten zukünftige Features, die mit dem aktuellen Datenbestand noch nicht umgesetzt werden können.

D.1 Anforderungsanalyse Subtraktive Fertigung

Anforderungsgruppe	**Beschreibung**
Berechnung der optimalen Schnittparameter	Das System soll für einen gegebenen Zerspanungsprozess die optimalen Schnittparameter auf Basis der KPI-Berechnung bestimmen. Die KPI repräsentieren hierbei verschiedene Anwendungsfälle, wie bspw. kosteneffizientester Prozess oder kürzester Prozess. Der Benutzer macht Angaben zum durchzuführenden Zerspanungsprozess, wie dem Werkzeug, dem Werkstoff und der Bearbeitungsphase sowie der verwendete Maschine. Anschließend sollen die KPI für mehrere Schnittparameter-Sätze berechnet werden. Abhängig von der Benutzerangabe, welcher KPI beim Prozess optimal sein soll, wird der jeweilige Schnittparameter-Satz, der in den Berechnungen den besten KPI-Wert erzielt, vorgeschlagen.
Werkzeug vorschlagen	Der Benutzer soll Angaben zum durchzuführenden Zerspanungsprozess machen können. Die Berechnung der Schnittparameter soll anschließend für mehrere, zum Prozess passende, Werkzeuge durchgeführt werden. Das System soll auf Basis der KPI-Priorisierung des Benutzers ein oder mehrere Werkzeuge mit den jeweiligen Schnittparametern vorschlagen.
Berechnung mehrerer Bearbeitungsschritte	Das System soll dem Benutzer für beliebige und aneinandergereihte Bearbeitungsschritte die optimalen Schnittparamter berechnen können, wobei der Benutzer bei jedem Bearbeitungsschritt das zu bearbeitende Feature angeben kann.
Verwaltung der eigenen Werkstatt	Der Benutzer soll eine digitale Repräsentation seiner Werkstatt in der Anwendung verwalten können. Zur Auswahl stehende Maschinen sollen dem Bestand hinzugefügt und ggf. konfiguriert werden können.

D.2 Anforderungsanalyse Additive Fertigung

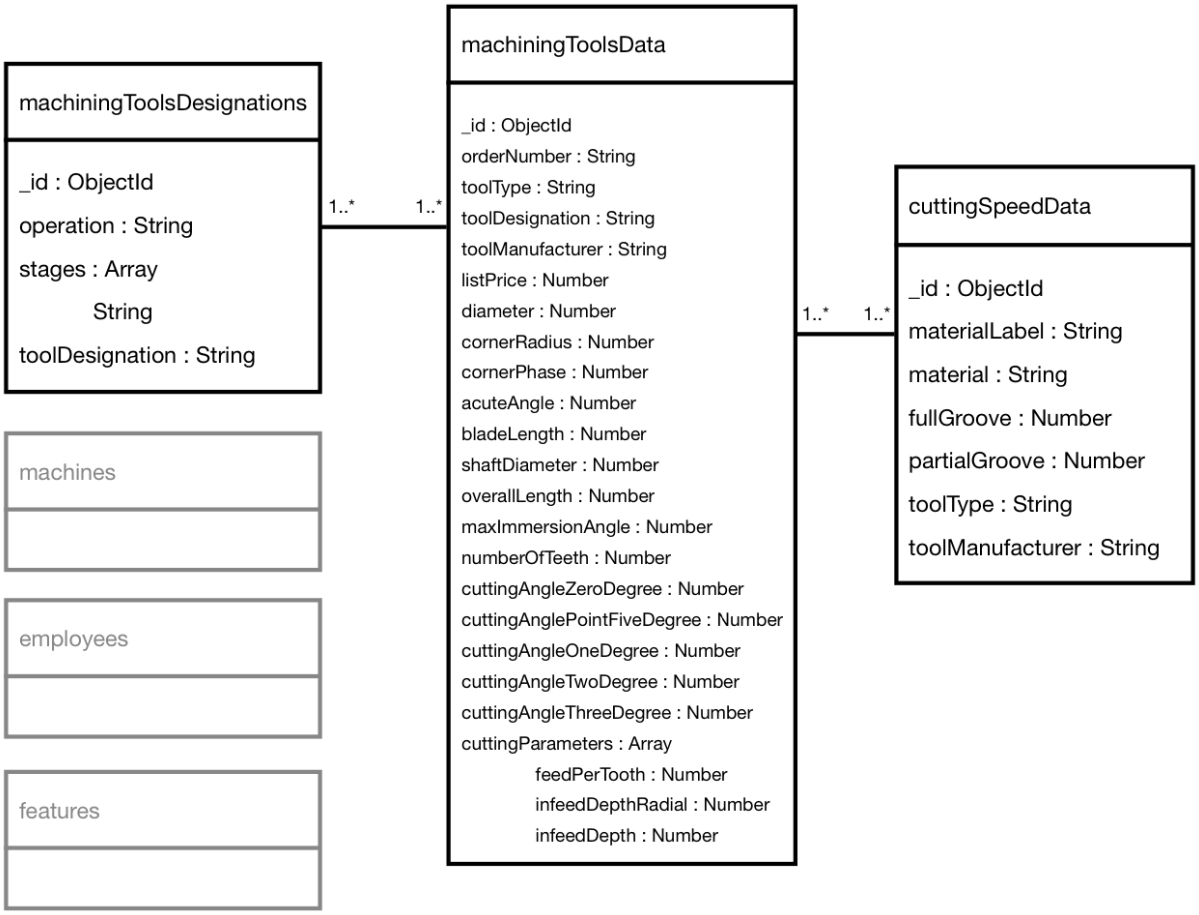
Anforderungsgruppe	Beschreibung
Upload und Analyse einer STL-Datei	Es soll möglich sein, eine STL-Datei hochzuladen, um diese automatisch nach Bauvolumen, Hüllquader und anderen Kennwerten analysieren zu lassen.
Berechnung der Bauteilkosten	Der Benutzer soll mit Angabe von Material und Drucker sowie Bauvolumen die zu erwartenden Herstellungskosten berechnen können. Die Herstellungskosten werden in Material-, Maschinen- und Personalkosten unterteilt. Weiterhin soll dem Benutzer Auskunft über die zu erwartende Druckqualität gegeben werden.
Berechnung der Bauteilkosten: Vergleich mehrerer Drucker	Die Berechnung der Bauteilkosten soll für mehrere Drucker abhängig von der Materialauswahl durchgeführt werden. Dem Benutzer sollen abhängig vom gewünschten Prozess (schnellster, günstigster, wenig Nachbearbeitung) ein oder mehrere Drucker vorgeschlagen werden.
Verwaltung der eigenen Werkstatt	Auf Basis der Dimensionen des berechneten Hüllquaders eines Druckteils sowie der Dimensionen des zu verwendeten Druckers soll vom System die optimale Ausrichtung des Druckteils berechnet werden können.

Anhang E

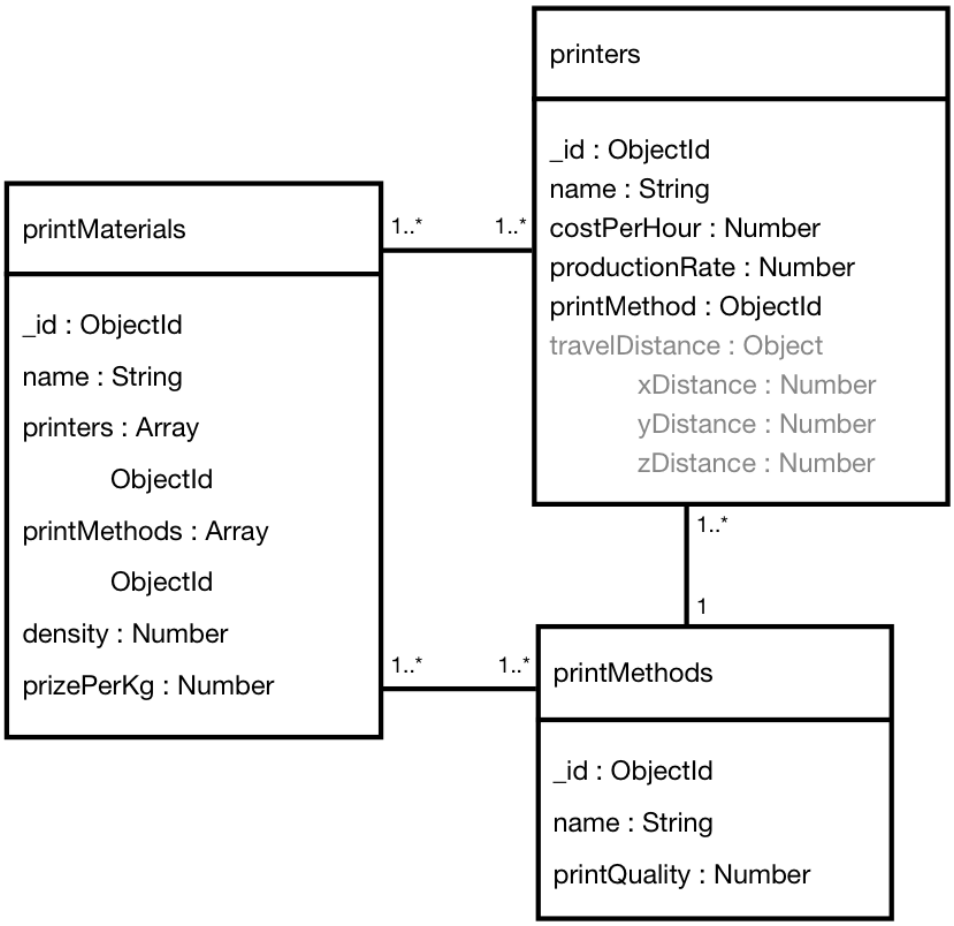
Datenmodelle des neuen Systems

Ausgegraute Datenstrukturen werden vorerst nicht berücksichtigt, da die hierfür benötigten Daten noch nicht vorhanden sind. Sie werden erst für zukünftige Features benötigt.

E.1 Datenmodell Subtraktive Fertigung



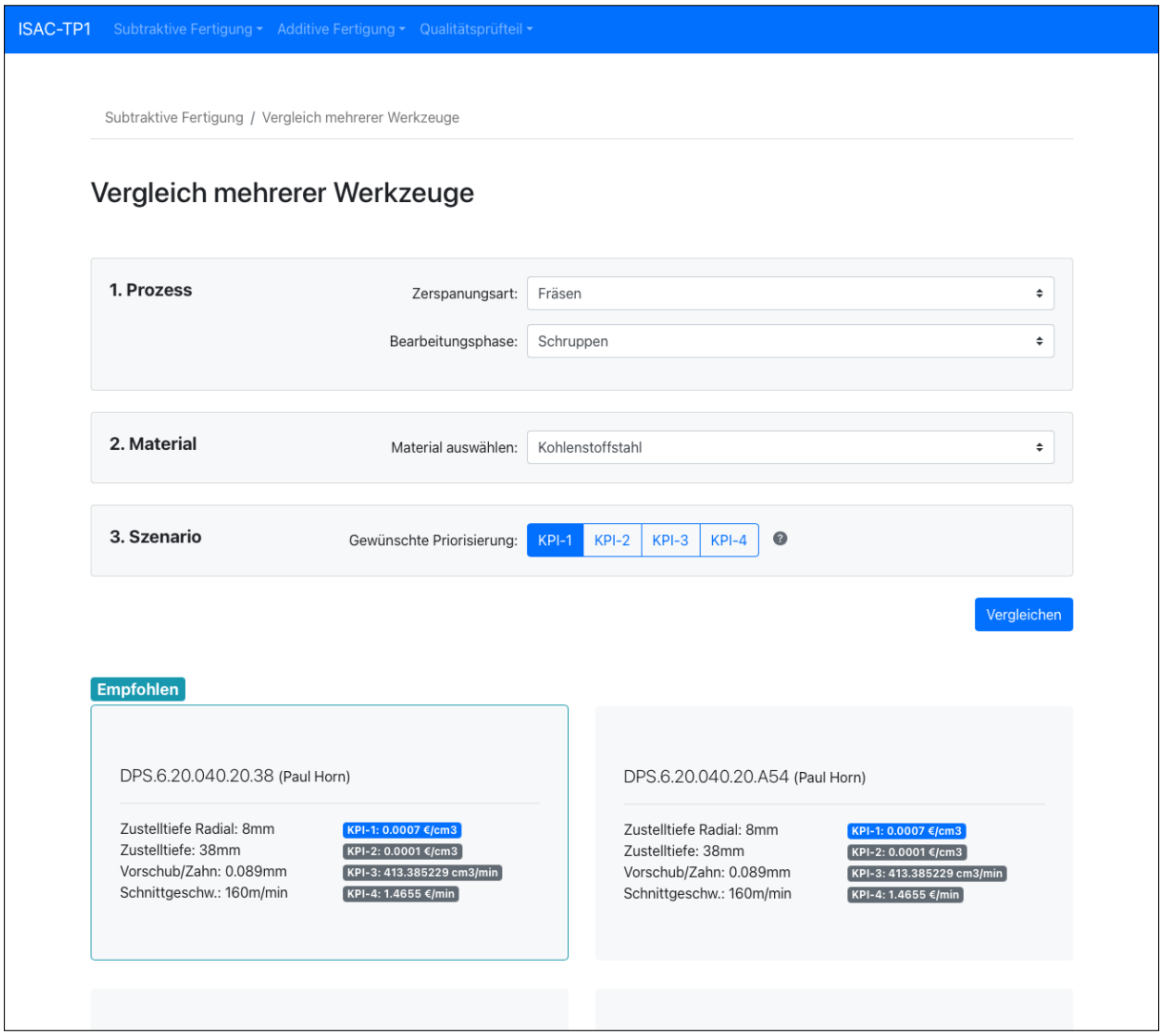
E.2 Datenmodell Additive Fertigung



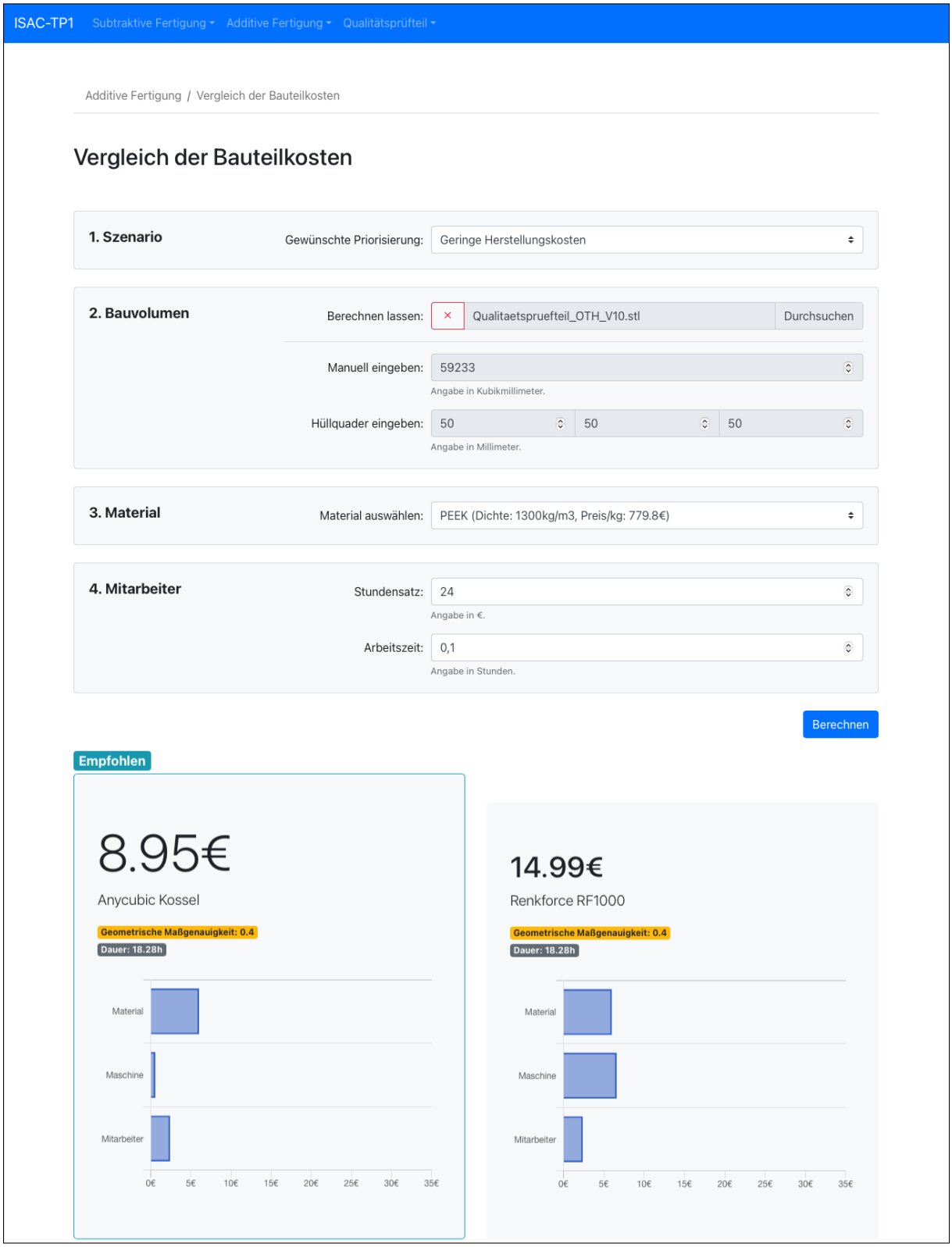
Anhang F

Screenshots der Anwendung

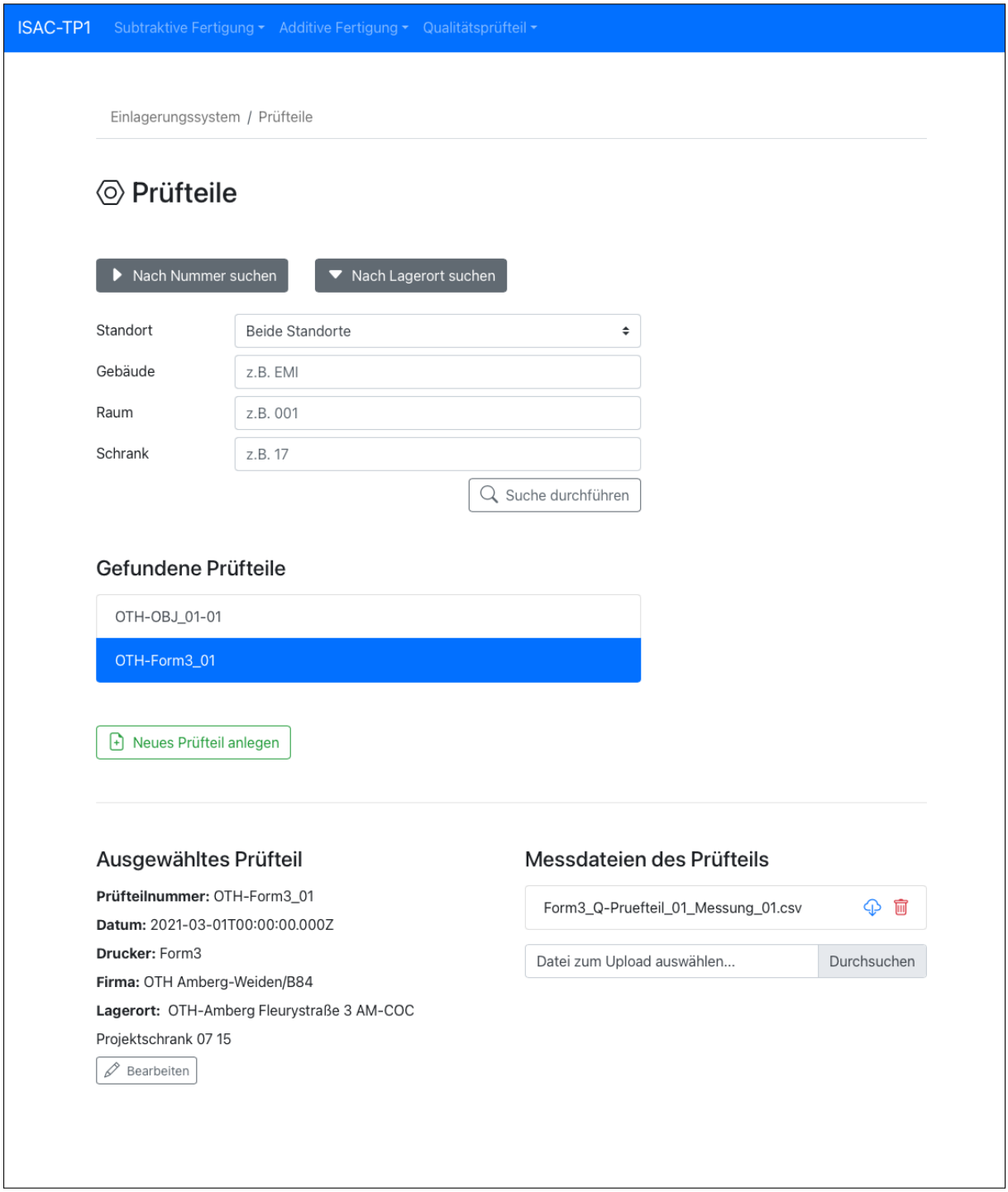
F.1 Screenshot Subtraktive Fertigung



F.2 Screenshot Additive Fertigung



F.3 Screenshot Prüfteileinlagerungssystem



Anhang G

Ergebnisse der Evaluierung

G.1 Bewertungskatalog altes System

Die Kriterien des Bewertungskataloges wurden nach ISO/IEC JTC 1/SC 7 Software and systems engineering (2011, S. 10-16) aufgestellt.

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
1. functional suitability	***degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions***				
functional completeness	degree to which the set of functions covers all the specified tasks and user objectives	X			Siehe Featureanalyse
functional correctness	degree to which a product or system provides the correct results with the needed degree of precision	X			Siehe Featureanalyse
functional appropriateness	degree to which the functions facilitate the accomplishment of specified tasks and objectives		X		
2. performance efficiency	***performance relative to the amount of resources used under stated conditions***				
time behaviour	degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements		X		
resource utilization	degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements		X		

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
capacity	degree to which the maximum limits of a product or system parameter meet requirements		X		
3. compatibility	***degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment***				
co-existence	degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product		X		
interoperability	degree to which two or more systems, products or components can exchange information and use the information that has been exchanged	X			Es sind keine Schnittstellen vorhanden, über welche Daten ausgetauscht werden können.
4. usability	***degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use***				
appropriateness recognizability	degree to which users can recognize whether a product or system is appropriate for their needs		X		
learnability	degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use		X		
operability	degree to which a product or system has attributes that make it easy to operate and control		X		
user error protection	degree to which a system protects users against making errors	X			Der Benutzer muss teilweise versuchen, ein Formular abzuschicken, um zu erfahren, ob er alle nötigen Eingabefelder ausgefüllt hat.

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
user interface aesthetics	degree to which a user interface enables pleasing and satisfying interaction for the user		X		Dieses Kriterium ist bei der Anwendung der subtraktiven Fertigung/Prüfteileinlagerungssystem erfüllt, jedoch besitzt die Anwendung der additiven Fertigung eine veraltete Benutzeroberfläche.
accessibility	degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use		X		
5. reliability	***degree to which a system, product or component performs specified functions under specified conditions for a specified period of time***				
maturity	degree to which a system, product or component meets needs for reliability under normal operation		X		
availability	degree to which a system, product or component is operational and accessible when required for use		X		
fault tolerance	degree to which a system, product or component operates as intended despite the presence of hardware or software faults		X		
recoverability	degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system		X		
6. security	***degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization***				Es wurden keine Vorkehrungen zur Authentifizierung und Autorisierung getätigt.
confidentiality	degree to which a product or system ensures that data are accessible only to those authorized to have access	X			
integrity	degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data	X			

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
non-repudiation	degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later	X			
accountability	degree to which the actions of an entity can be traced uniquely to the entity	X			
authenticity	degree to which the identity of a subject or resource can be proved to be the one claimed	X			
7. maintainability	***degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers***				
modularity	degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components			X	Modularität ist durch das MVC-Pattern bei der CakePHP-Anwendung gegeben. (Bei dem System der additiven Fertigung nicht)
reusability	degree to which an asset can be used in more than one system, or in building other assets		X		
analysability	degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified	X			Schlechte Wartbarkeit des Codes: Auskommentierte Code-Blöcke, keine aussagekräftigen Variablennamen, ...
modifiability	degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality		X		
testability	degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met		X		

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
8. portability	***degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another***				
adaptability	degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments		X		
installability	degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment		X		
replaceability	degree to which a product can replace another specified software product for the same purpose in the same environment		X		

G.2 Bewertungskatalog neues System

Die Kriterien des Bewertungskataloges wurden nach ISO/IEC JTC 1/SC 7 Software and systems engineering (2011, S. 10-16) aufgestellt.

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
1. functional suitability	***degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions***				
functional completeness	degree to which the set of functions covers all the specified tasks and user objectives			X	Die spezifizierten Funktionen sind umgesetzt. Das wird durch Contract-Testing verifiziert. Hierbei wird überprüft, ob die Implementierung der API mit der Spezifikation, welche die Soll-Funktionalität des Systems widerspiegelt und als Contract fungiert, übereinstimmt.
functional correctness	degree to which a product or system provides the correct results with the needed degree of precision			X	Die funktionale Korrektheit ist gegeben, welche durch Tests verifiziert wurde.
functional appropriateness	degree to which the functions facilitate the accomplishment of specified tasks and objectives		X		
2. performance efficiency	***performance relative to the amount of resources used under stated conditions***				
time behaviour	degree to which the response and processing times and throughput rates of a product or system, when performing its functions, meet requirements		X		
resource utilization	degree to which the amounts and types of resources used by a product or system, when performing its functions, meet requirements		X		
capacity	degree to which the maximum limits of a product or system parameter meet requirements		X		

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
3. compatibility	***degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment***				
co-existence	degree to which a product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product		X		
interoperability	degree to which two or more systems, products or components can exchange information and use the information that has been exchanged			X	Jede Funktionalität der Anwendung ist über eine REST-API bereitgestellt. Somit besteht die Möglichkeit, dass andere Systeme auf die Daten und Funktionen der Anwendungen zugreifen können. Das wurde durch den API-First (Development) Ansatz gewährleistet, nach welchem eine neue Funktionalität stets über eine Schnittstelle bereitgestellt werden soll.
4. usability	***degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use***				Das System besitzt mehrere Schnittstellen, mit denen ein Benutzer interagieren kann: die graphische Benutzeroberfläche der Client-Applikation und die REST-API des Servers. Die aufgeführten Kriterien der Usability können auf die graphische Benutzeroberfläche und nur teilweise auf die API angewandt werden. Demnach muss die Usability der API separat betrachtet werden. In diesem Abschnitt wird nun lediglich die graphische Benutzeroberfläche evaluiert.
appropriateness recognizability	degree to which users can recognize whether a product or system is appropriate for their needs		X		
learnability	degree to which a product or system can be used by specified users to achieve specified goals of learning to use the product or system with effectiveness, efficiency, freedom from risk and satisfaction in a specified context of use			X	Das System gibt dem Benutzer mittels Tooltips und anderen Nachrichten Tipps oder andere Informationen zur Benutzung der Anwendung.

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
operability	degree to which a product or system has attributes that make it easy to operate and control		X		
user error protection	degree to which a system protects users against making errors			X	Durch eine deutliche Anzeige, welche Eingabefelder der Benutzer zwingend ausfüllen muss sowie Hinweise, welche Einheiten oder Formate bei diesen Feldern verwendet werden, ist der Benutzer zu einem gewissen Grad vor Eingabefehlern beim Ausfüllen eines Formulars geschützt.
user interface aesthetics	degree to which a user interface enables pleasing and satisfying interaction for the user			X	Die Benutzeroberfläche weist moderne Oberflächenelemente mit einheitlichem Farbschema sowie Icons auf.
accessibility	degree to which a product or system can be used by people with the widest range of characteristics and capabilities to achieve a specified goal in a specified context of use		X		
5. reliability	***degree to which a system, product or component performs specified functions under specified conditions for a specified period of time***				
maturity	degree to which a system, product or component meets needs for reliability under normal operation		X		
availability	degree to which a system, product or component is operational and accessible when required for use		X		
fault tolerance	degree to which a system, product or component operates as intended despite the presence of hardware or software faults		X		
recoverability	degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system		X		
6. security	***degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization***				Es wurden keine Vorkehrungen zur Authentifizierung und Autorisierung getätigt.

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
confidentiality	degree to which a product or system ensures that data are accessible only to those authorized to have access	X			
integrity	degree to which a system, product or component prevents unauthorized access to, or modification of, computer programs or data	X			
non-repudiation	degree to which actions or events can be proven to have taken place, so that the events or actions cannot be repudiated later	X			
accountability	degree to which the actions of an entity can be traced uniquely to the entity	X			
authenticity	degree to which the identity of a subject or resource can be proved to be the one claimed	X			
7. maintainability	***degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers***				
modularity	degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components			X	Dieses Kriterium ist durch die Client-Server Architektur erfüllt. Es ist beispielsweise möglich, die Client-Applikation ohne Anpassung des Servers auszutauschen.
reusability	degree to which an asset can be used in more than one system, or in building other assets		X		
analysability	degree of effectiveness and efficiency with which it is possible to assess the impact on a product or system of an intended change to one or more of its parts, or to diagnose a product for deficiencies or causes of failures, or to identify parts to be modified			X	Dieses Kriterium ist erfüllt. Aufgrund von einheitlichem Quellcode mit vielen Kommentaren und einer übersichtlichen Ordnerstruktur ist eine leichte Analysierbarkeit gegeben.

Kriterium	Beschreibung	Negativ	Neutral	Positiv	Bemerkung
modifiability	degree to which a product or system can be effectively and efficiently modified without introducing defects or degrading existing product quality			X	Durch die Trennung der Geschäftslogik und der Präsentationsschicht sowie der leichten Wartbarkeit des Quellcodes und der geringen Abhängigkeit zwischen Komponenten innerhalb des Systems ist eine leichte Modifizierbarkeit gegeben. Wird weiterhin mit API-First gearbeitet, so muss bei Änderungen der beschriebene Entwicklungsprozess der Arbeit erneut durchlaufen werden. Es muss zuerst der API-Entwurf für das neue oder anzupassende Feature durchgeführt werden, bevor die Implementierungsphase gestartet werden kann. Somit ist eine Modifikation des Systems mit mehr Aufwand verbunden, jedoch kann im Nachgang mit Contract-Testing und anderen Tests sichergestellt werden, dass keine Fehler in anderen Bereichen des Systems verursacht wurden.
testability	degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met			X	Dieses Kriterium ist erfüllt. Die REST-API stellt ein gutes Test-Interface dar, da sie die komplette Funktionalität des Systems repräsentiert. Auf Basis der API-Spezifikation können jederzeit Code-Gerüste für Tests generiert werden.
8. portability	***degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another***				
adaptability	degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments		X		
installability	degree of effectiveness and efficiency with which a product or system can be successfully installed and/or uninstalled in a specified environment		X		
replaceability	degree to which a product can replace another specified software product for the same purpose in the same environment		X		