

Hochschule Bochum

Sommersemester 2022

Dozenten: Prof. Dr.-Ing. Carsten Köhn

Zweitprüfer: Nils Backenköhler

# **Entwicklung und Evaluierung einer Webapp zum Testen eines Transfermoduls für medizinische Daten**

Vorgelegt als Bachelorarbeit

Abgabetermin 29.08.2022

Daniel Tura

10. Semester Informatik

Elisabethstr. 34, 44866 Bochum

daniel.tura@live.de

0177 / 7179764

Matrikelnummer: 017200664

**Hochschule Bochum**  
Bochum University  
of Applied Sciences



## Inhalt

Tabellenverzeichnis.....	i
Abbildungsverzeichnis.....	i
Listingverzeichnis.....	<b>Fehler! Textmarke nicht definiert.</b>
Abkürzungsverzeichnis .....	i
1. Einleitung .....	1
2. Technologie Stack.....	2
3. Evaluationsmethode .....	3
3.1. Abbildung des Modelles auf die geforderte Anwendung .....	5
3.2. Nutzungskontext .....	5
3.3. Erstellung der funktionalen Anforderungen .....	5
3.4. Erstellung der nichtfunktionalen Anforderungen .....	6
3.5. Erstellung der User Stories .....	14
4. Erfüllung der funktionalen Anforderungen .....	15
4.1. Allgemeiner Aufbau der TUSCHI .....	15
4.2. Implementierung User Story 1 .....	17
4.3. Implementierung User Story 2.....	20
4.3.1. Backendimplementierung.....	20
4.3.2. Frontendimplementierung .....	23
4.4. Implementierung User Story 3.....	24
4.5. Erfüllung der funktionalen Korrektheit .....	25
5. Erfüllung der nichtfunktionalen Anforderungen.....	25
6. Fazit.....	28
7. Literaturverzeichnis .....	ii
8. Anhang.....	iii
9. Eidesstattliche Erklärung.....	iv
10. Sperrvermerk.....	iv

## Abkürzungsverzeichnis

Die Tuschi – Der Name, der zu entwickelnden Webapp

npm – node package manager

## Abbildungsverzeichnis

Abbildung 1: Produktqualitätsmodell ISO/IEC 25010 (Quelle: Eigene Abbildung)

Abbildung 2: Aufbau der Tuschi (Quelle: Eigene Abbildung)

Abbildung 3: Abbildung der Tuschi Frontend UI (Quelle: Eigene Abbildung)

## Tabellenverzeichnis

Tabelle 1: C-Store status codes (Quelle: <https://pynetdicom2.readthedocs.io/en/latest/statuses.html>)

Tabelle 2: Protokoll der Usability Tests (Quelle: Eigene Tabelle)

## 1. Einleitung

Im Rahmen der Health IT sind die Anforderungen an Anwendungen weitaus höher als in anderen Bereichen der Informatik. Computer werden in immer höherem Maße verwendet, um medizinische Daten zu verarbeiten. Die Entwicklung und Evaluation von Anwendungen, die für diesen Bereich entwickelt werden, ist enorm wichtig, denn Fehler in der Verarbeitung von Daten können von leichten bis hin zu tödlichen Folgen an Patienten führen. Um dies zu vermeiden, müssen medizinische Anwendungen Standards erfüllen. Hierbei ist das Testen dieser Anwendungen einer der Aspekte, der für die Erfüllung jener Standards eine bedeutende Rolle spielt. Ziel dieser Arbeit ist die Entwicklung und Evaluierung einer Webapp zum Testen eines Transfermoduls für medizinische Daten. Grund für die Entwicklung ist der Bedarf an einem Testtool, um die laufende Entwicklung der connect-bridge, des Unternehmens Visus Health IT GmbH, zu begleiten. Dafür soll eine Webbasierte-GUI erstellt werden. Um der Health IT entsprechend eine qualitativ hochwertige Anwendung zu entwickeln, wird ein Evaluationskriterium benötigt, welches den Ansprüchen, für eine Anwendung in diesem Bereich, angemessen ist.

Die Basis der Evaluation bildet die SQuaRE Reihe, welche von der Internationalen Organisation für Standards (ISO) entwickelt wurde. Die Reihe stellt ein Produktqualitätsmodell für Software vor, welches aus 32 Qualitätsmerkmalen besteht, mit denen Anwendungen evaluiert werden können. Außerdem stellt die Reihe eine Anleitung zur Verfügung, wie Anforderungen ermittelt und Evaluationen durchgeführt werden können. Die SQuaRE Reihe soll für diese Arbeit eine Richtlinie bilden, an der sich orientiert werden kann, wenn es um jegliche Art von Entscheidungen geht, die mit der Planung, Entwicklung und Analyse der angeforderten Anwendung zu tun haben.

Die vorliegende Arbeit prüft jedes einzelne Qualitätsmerkmal des Produktqualitätsmodells und erstellt daraus Anforderungen für die zu entwickelnde Webapp. Anhand dieser Anforderungen werden User Stories für das Projekt erstellt. Der praktische Teil dieser Arbeit, ist die Entwicklung der Webapp, welche als mögliche Lösung für die Erfüllung der Anforderungen erforderlich ist. Abschließend werden die ermittelten Ergebnisse zusammen gefasst und betrachtet ob und wie die Anforderungen an das Projekt erfüllt werden. Hierbei sollen auch weiterführende Gedanken und Anregungen für Verbesserungen mit aufgenommen werden.

## 2. Technologie Stack

**React** - Bei React handelt es sich um eine vom Unternehmen Meta entwickelte JavaScript-Bibliothek zur Erstellung von webbasierten Benutzeroberflächen. React kann sowohl in JavaScript als auch TypeScript programmiert werden. Für die vorliegende Arbeit wird die TypeScript Variante verwendet.

**Spring Boot** - Spring Boot ist Teil des Spring Frameworks und bietet Entwicklern die Möglichkeit einen auf Java basierten Webserver aufzusetzen. Das Framework wird als Backend für die Tuschi verwendet und bildet die Schnittstelle bei der Kommunikation zwischen dem Benutzer und der connect-bridge.

**Git** - Git ist eine kostenlose open source Anwendung, welche für die Versionierung von Projekten verwendet wird, was bedeutet, dass mehrere Entwickler gleichzeitig an dem Projekt arbeiten können

**Bitbucket** - Bitbucket ist ein webbasierter Onlinedienst zur Versionsverwaltung für Software-Entwicklungsprojekte.

**DICOM** - Bei DICOM handelt es sich um einen internationalen Standard zum Senden, Speichern, Abrufen, Drucken, Verarbeiten und Anzeigen medizinischer Bildinformationen. DICOM wird von der ISO als eigener Standard anerkannt und kann unter ISO 12052 eingesehen werden.

**DIMSE C-Services** - Ein zum DICOM Standard gehörender Teil sind die DIMSE C-Services, welche es erlauben einer DICOM-Anwendungsentität, explizit eine Operation durch eine andere DICOM-Anwendungsentität auf zusammengesetzten SOP-Instanzen anzufordern. Beim Ausführen der Operationen wird immer ein Status Code erzeugt, der eine Meldung über den Status der Ausführung enthält.

**connect-bridge** - Die connect-bridge ist die von Visus Health IT GmbH entwickelte Anwendung in Form einer Spring Boot Anwendung. Ihr Zweck ist die Prüfung und der Transfer von DICOM Daten. Die connect-bridge wird im Produktionsbetrieb als Windows-Dienst ausgeführt.

Status	Code	Description
Failure	A7xx	Refused: Out of Resources
	A9xx	Error: Data Set does not match SOP Class
	Cxxx	Error: Cannot understand
	0112	Failed: SOP Class Not Supported
	0210	Duplicate Invocation
	0117	Invalid Object Instance
	0212	Mistyped argument
	0211	Unrecognized Operation
	0124	Refused: Not Authorized
Warning	B000	Coercion of Data Elements
	B007	Data Set does not match SOP Class
	B006	Elements Discarded
Success	0000	Success

Tabelle 1

### 3. Evaluationsmethode

Bei Evaluation handelt es sich nach Balzer um eine Bewertung bzw. Begutachtung von Projekten, Prozessen und Funktionseinheiten [1]. Das Projekt ist die Entwicklung der Anwendung zum Testen des Transfermoduls. Als Prozesse sind jene Handlungen abgebildet, die zur Erfüllung der Projektanforderungen dienen. Im Falle der vorliegenden Arbeit sind das die

- Wahl der Tools und Frameworks
- Vorgehen bei der Entwicklung

Die Funktionseinheiten, welche betrachtet werden, sind die Ergebnisse dieser Handlungen. Im vorliegenden Fall ist dies der Programmcode.

Es gibt unterschiedliche Modelle, auf Basis derer Anwendungen bewertet werden können. Lange präsentiert hierfür in einer Ausarbeitung zu Softwarequalitätsmodellen zwei Modelle, die zur Bewertung von Software verwendet werden können [6]. Zum einen das Qualitätsmodell nach McCall. Das Softwarequalitätsmodell nach McCall stammt aus dem Jahre 1977 und ist ein dreistufiges Modell, bestehend aus elf Qualitätshauptzielen, den Qualitätsfaktoren, aus

Qualitätskriterien zu jedem Faktor und aus Kenngrößen und Metriken (Lange 2010). Die Qualitätshauptziele sind folgende:

Korrektheit, Zuverlässigkeit, Effizienz, Integrität, Benutzbarkeit, Wartbarkeit, Testbarkeit, Flexibilität, Portabilität, Wiederverwendbarkeit, Verknüpfbarkeit

Das zweite Modell, welches Lange ausarbeitet ist das Qualitätsmodell nach Norm ISO 9126. Dieses Modell ist allerdings nicht mehr aktuell und wurde in den ISO Standard 25010 überführt und überarbeitet, weshalb sich weitere Erwähnungen auf den aktuelleren Standard beziehen. ISO 25010 definiert ein Produktqualitätsmodell, welches aus 8 Hauptmerkmalen besteht, welche sich wiederum in 32 Untermerkmale aufteilen (Abbildung 1)

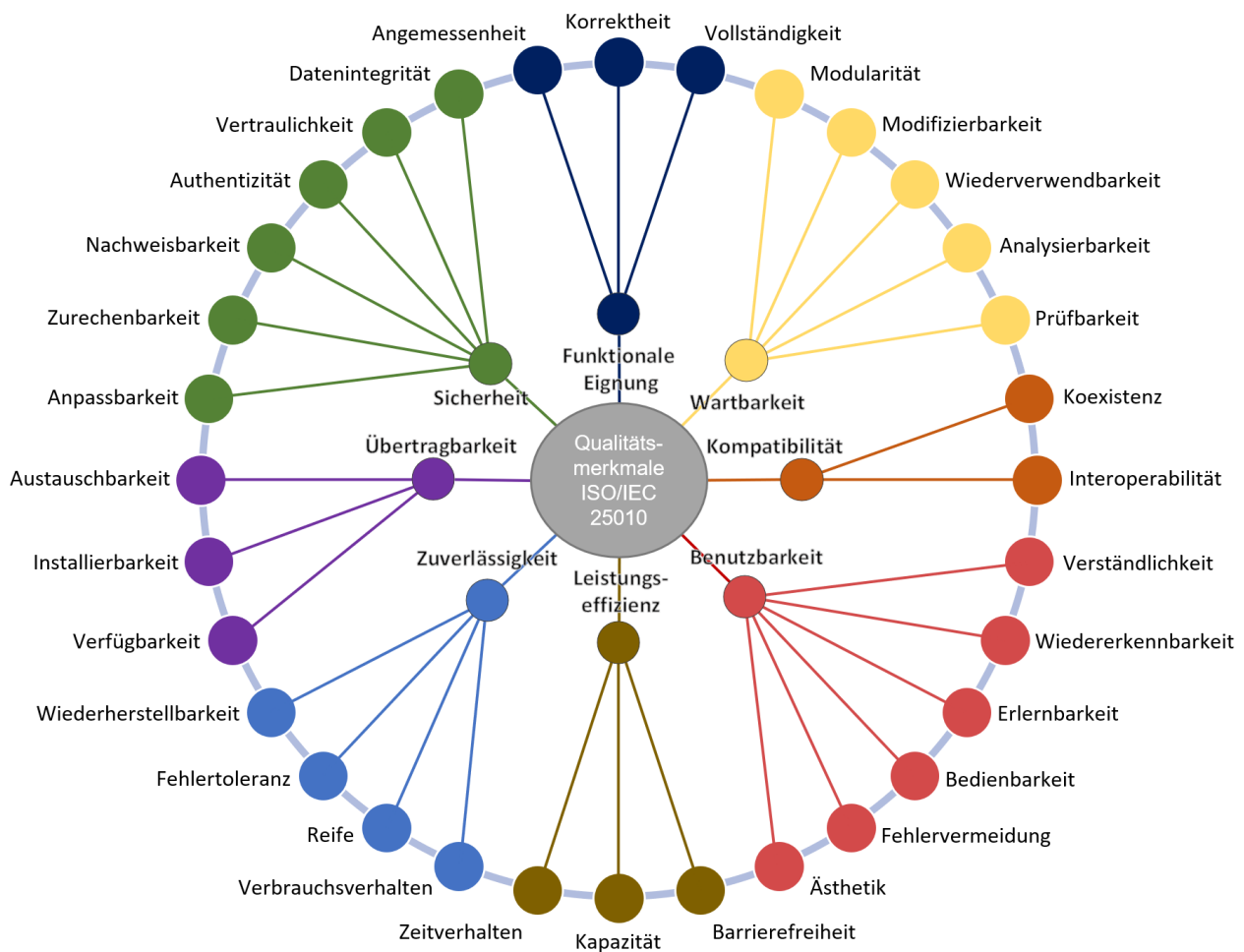


Abbildung 1

Die beiden Modelle sind sich sehr ähnlich. Beide Modelle definieren Kriterien, anhand derer Software bewertet werden kann. Für diese Arbeit wird das Modell nach ISO verwendet, da es alle Kriterien aus dem Modell von McCall aufnimmt und die Aufteilung in die vielen Unterkategorien eine bessere

Evaluation der Anwendung erlauben. Darüber hinaus ist der Standard Teil einer Reihe, die die Anwendung des Modells auf die Realität beschreiben.

### 3.1. Abbildung des Modelles auf die geforderte Anwendung

Nicht alle Qualitätsmerkmale eignen sich für die Anwendung auf Die Tuschi. ISO/IEC 25020 der SQuaRE Reihe beschreibt den Prozess zur Sicherstellung eines Qualitätsmerkmals. Demnach werden für alle benötigten Qualitätsmerkmale eine oder mehrere Qualitätsmessungen durchgeführt. Mögliche Messungen werden hierbei in ISO/IEC 25023 aufgelistet. Aufgrund der Flexibilität der Modelle besteht keine Einschränkung in Hinsicht auf die Erstellung eigener Messfunktionen. Die folgenden Abschnitte legen den Nutzungskontext der Tuschi fest und teilen die Anforderungen an das Projekt und funktionale und nichtfunktionale Anforderungen auf.

### 3.2. Nutzungskontext

- *NK-1*: Die Benutzergruppe, sind Entwickler der Visus Health IT GmbH.
- *NK-2*: Die Entwickler sollen die Tuschi mit ihrem Arbeitsrechner bedienen können.
- *NK-3*: Der Standort des Arbeitsrechners ist variabel. Es ist ein Internetzugang vorhanden.

Benutzersystem: Der durchschnittliche Arbeitsrechner eines Entwicklers hat folgende Parameter:

- Betriebssystem: Windows 10
- Arbeitsspeicher: 32GB
- Prozessor (CPU): Intel Core i7-8750H 2,2 GHz mit 6 Kernen
- Grafikkarte: Intel UHD Graphics 630

### 3.3. Erstellung der funktionalen Anforderungen

Die funktionalen Anforderungen legen fest, was eine Anwendung machen soll [8].

Die funktionalen Anforderungen an die Tuschi, werden im vorliegenden Fall vom Unternehmen gestellt:

- *FA-1*: Die Tuschi soll Dateien an die connect-bridge senden können.
- *FA-2*: Die Tuschi soll die Rückmeldung über den Verlauf der Übertragung dem Benutzer anzeigen.

Die funktionalen Anforderungen lassen sich nach ISO 25010 unter drei Aspekten betrachten:

**Vollständigkeit:** Grad, in dem der Satz von Funktionen alle spezifizierten Aufgaben und Benutzerziele abdeckt.

**Korrektheit:** Grad, in dem ein Produkt oder System die richtigen Ergebnisse mit dem erforderlichen Grad an Präzision liefert.



Funktionale Korrektheit kann erreicht werden, indem Testabdeckung für die Implementierungen der Funktionen geschrieben werden [5]. Dafür kann zum einen ein Prozentsatz festgelegt werden, welcher bei Erreichen eines Schwellwertes die funktionale Korrektheit sicher stellt. Die andere Variante ist die Abdeckung von zuvor definierten Testfällen. Letztere Variante ist aufgrund des geringen Umfangs an Funktionen für die vorliegende Arbeit relevanter. Die zu entwickelnden Testfälle hängen von der Implementation ab und werden in Abschnitt 4.5 erarbeitet.

**Angemessenheit:** Untermerkmal von funktionaler Eignung. Das Ausmaß, in dem die Funktionen das Erreichen Bestimmter Aufgaben und Ziele erleichtern.

### 3.4. Erstellung der nichtfunktionalen Anforderungen

Nichtfunktionale Anforderungen bilden die Randbedingungen für eine Anwendung. Sie beschreiben, wie gut ein System eine Leistung erbringen soll [9]. Im folgenden Teil der Arbeit wird über die Qualitätsmerkmale des Qualitätsmodells, wie es in ISO 25010 beschrieben wird, iteriert und die nichtfunktionalen Anforderungen werden erstellt.

**Leistungseffizienz:** Hauptmerkmal. Leistung im Verhältnis zur Menge der verwendeten Ressourcen unter bestimmten Bedingungen. Ressourcen können andere Softwareprodukte und Hardwarekonfiguration des Systems und Materialien wie Speichermedien umfassen.

**Zeitverhalten:** Untermerkmal von Leistungseffizienz. Grad, in dem die Reaktions- und Verarbeitungszeiten, sowie die Durchsatzraten eines Produkts oder Systems bei der Ausführung seiner Funktionen den Anforderungen entsprechen.

*NFA-1:* Im Produktionsfall kann es vorkommen, dass DICOM Dateien mehrere Gigabytes groß sind. Eine Verarbeitung der Dateien kann nach heutigem Stand der Technik einige Sekunden oder Minuten in Anspruch nehmen. Hierfür kann die Zeit, die der Benutzer auf die Rückmeldung wartet, in drei Klassen aufgeteilt werden [7]:

- 0,1 Sekunden Rückmeldezeit wird vom Benutzer als eine sofortige Ausführung der Aktion empfunden. Der Benutzer benötigt kein spezielles Feedback, dass die Aktion ausgeführt wurde.
- 1,0 Sekunden Rückmeldezeit ist das Limit, um beim Benutzer einen klaren Gedankengang aufrecht zu erhalten. Für Aktionen mit einer Rückmeldezeit von 0,1 bis 1,0 Sekunden wird im Normalfall kein spezielles Feedback benötigt, welches dem Benutzer signalisiert, dass die Aktion ausgeführt wurde. Alles über einer Sekunde hängt vom Anwendungsfall ab.
- 10 Sekunden Rückmeldezeit ist das Limit des Benutzers sich auf eine Aktion zu konzentrieren, ohne eine Rückmeldung zu bekommen. Der Benutzer wird anfangen sich anderen Tätigkeiten zu widmen und braucht deshalb eine Rückmeldung darüber, ob die Aktion ausgeführt wurde.

Da von vornherein weder die Anzahl noch die Größe der Dateien bekannt ist, die der Benutzer mit der Tuschi versenden möchte, kann kein klarer Wert für die Dauer der Übertragung definiert werden. Dem Benutzer soll unabhängig von der Übertragungsdauer eine Rückmeldung gegeben werden, wenn eine Übertragung beendet wird.

**Verbrauchsverhalten:** Untermerkmal von Leistungseffizienz. Grad, in dem die Mengen und Arten von Ressourcen, die von einem Produkt oder System bei der Ausführung seiner Funktionen verwendet werden, den Anforderungen entsprechen.

Aufgrund der Leistung des Benutzersystems in Kombination mit den funktionalen Anforderungen, kann das Verbrauchsverhalten als Qualitätsmerkmal für die Tuschi vernachlässigt werden.

**Kapazität:** Untermerkmal von Leistungseffizienz. Grad, in dem die Höchstwerte eines Produkt- oder Systemparameters den Anforderungen entsprechen.

Aufgrund der Leistung des Benutzersystems in Kombination mit den funktionalen Anforderungen, kann die Kapazität als Qualitätsmerkmal für die Tuschi vernachlässigt werden.

**Kompatibilität:** Hauptmerkmal. Grad, in dem ein Produkt, System oder Bauteil Informationen mit anderen Produkten, Systemen oder Bauteilen austauschen und/oder seine erforderlichen Funktionen ausführen kann, während es dieselbe Hardware- oder Softwareumgebung nutzt.

**Koexistenz:** Untermerkmal von Kompatibilität. Grad, in dem ein Produkt seine erforderlichen Funktionen effizient ausführen kann, während es sich eine gemeinsame Umgebung und Ressourcen mit anderen Produkten teilt, ohne dass dies nachteilige Auswirkungen auf ein anderes Produkt hat.

*NFA-2:* Der Benutzer führt auf seinem System neben der Tuschi auch andere Anwendungen aus. Diese Anwendungen teilen alle die gleichen Systemressourcen wie die Tuschi. Trotz Benutzung der Tuschi sollen alle anderen Anwendungen ihren gewohnten Betrieb fortsetzen können.

Die Tuschi soll nicht den Betrieb anderer Anwendungen des Benutzersystems beeinflussen. Diese Anforderung kann durch die Auswahl der Entwicklungswerkzeuge erfüllt werden.

**Interoperabilität:** Untermerkmal von Kompatibilität. Grad, in dem zwei oder mehr Systeme, Produkte oder Komponenten Informationen austauschen und die ausgetauschten Informationen nutzen können.

*NFA-3:* Die Tuschi steht im engen Zusammenhang mit der connect-bridge, da diese beiden Anwendungen miteinander kommunizieren sollen.

Es soll anhand von Implementierung und Testabdeckung sichergestellt werden, dass die Tuschi und die connect-bridge miteinander kommunizieren können.

**Benutzbarkeit:** Hauptmerkmal. Grad, in dem ein Produkt oder System von bestimmten Nutzern verwendet werden kann, um bestimmte Ziele mit Effektivität, Effizienz und Zufriedenheit in einem bestimmten Nutzungskontext zu erreichen.

**Verständlichkeit:** Untermerkmal von Benutzbarkeit. Grad, in dem die Nutzer erkennen können, ob ein Produkt oder System für ihre Bedürfnisse geeignet ist.

*NFA-4:* Beim Interagieren mit der Tuschi soll der Benutzer selbstständig erkennen können, ob er die Tuschi so benutzen kann, wie es für den Betrieb vorgesehen ist. Ein effektiver Weg das sicherzustellen sind Usability Tests, mit denen der Grad an Verständlich des Nutzers gegenüber der Anwendung gemessen werden kann [2].

Für die Umsetzung der Anforderungen soll ein Usability Test erstellt werden, der alle Anforderungen an die Benutzbarkeit abdeckt. Dies kann mit folgendem selbst erstellten Testablauf abgedeckt werden:

- Um Ressourcen zu sparen, sollen möglichst wenige Teilnehmer für die Tests herangezogen werden. Einer Studie von Nielsen zufolge werden für Usability Tests in den meisten Fällen nur 5 Personen benötigt, um die wichtigsten Probleme in Hinsicht auf Usability aufzudecken [7]. Für die Tests werden entsprechend mindestens 5 Testpersonen ausgewählt, welche der Benutzergruppe entsprechen.
- Testdurchführung:
  - Die Testperson führt den Test unbegleitet durch
  - Die Testperson führt die Schritte 1-4 durch
  - Schritt 1: Die Testperson soll eine beliebige Anzahl an Daten in die Liste laden
  - Schritt 2: Die Testperson soll eine Datei an die connect-bridge senden
  - Schritt 3: Die Testperson soll alle Dateien an die connect-bridge senden
  - Schritt 4: Die Testperson soll die Übertragungsmeldungen vorlesen

**Wiedererkennbarkeit:** Untermerkmal von Benutzbarkeit. Grad, in dem die Nutzer erkennen können, ob ein Produkt oder System für ihre Bedürfnisse geeignet ist.

*NFA-5:* Für die Erstellung der Anforderungen an die Wiedererkennbarkeit der Tuschi können nach folgende Messungen herangezogen werden [5]:

- Ermittlung des Anteils an erklärten oder dokumentieren Benutzerszenarios.
- Ermittlung des Anteils an demonstrierten Funktionselementen zur Erkennung der Funktion.
- Ermittlung des Anteils an selbsterklärenden Funktionen.

Anforderungen:

- NFA-5-1: Die Funktionsweise der Tuschi soll dokumentiert sein.

- Wenn nötig, sollen Elemente eingebracht werden, die die Funktionen der Tuschi erklären.
- Die Funktionen der Tuschi sollen möglichst selbsterklären sein.

**Erlernbarkeit:** Untermerkmal von Benutzbarkeit. Grad, in dem ein Produkt oder System von bestimmten Nutzern verwendet werden kann, um bestimmte Ziele zu erreichen, nämlich zu lernen, das Produkt oder System mit Effektivität, Effizienz, Risikofreiheit und Zufriedenheit in einem bestimmten Nutzungskontext zu verwenden.

*NFA-6:* Für die Erstellung der Anforderungen an die Erkennbarkeit kann die Verständlichkeit der Fehlermeldungen eines Systems zur Messung herangezogen werden [5]. Für die Tuschi ist dies besonders wichtig, da der Benutzer, die Fehlermeldungen eindeutig der Ursache zuordnen können soll. Dies soll bei der Implementierung sichergestellt werden.

**Bedienbarkeit:** Untermerkmal von Benutzbarkeit. Grad, in dem ein Produkt oder System Eigenschaften aufweist, die seine Bedienung und Kontrolle erleichtern.

*NFA-7:* Bedienbarkeit kann in folgende Prinzipien kategorisiert werden:

- NFA-7-1: Eignung der Software für die Aufgabe
- NFA-7-2: Grad an Selbstbeschreibung der Software
- NFA-7-3: Kontrolle über die Software
- NFA-7-4: Konformität der Software mit Benutzererwartungen

Alle Anforderungen an die Bedienbarkeit können mit Usability Tests sichergestellt werden [5].

**Fehlervermeidung:** Untermerkmal von Benutzbarkeit. Grad, in dem ein System die Benutzer vor Fehlern schützt.

*NFA-8:* Teil der funktionalen Anforderungen ist, dass Dateien zwischen zwei Systemen transferiert werden. Dabei können Fehler bei der Übertragung auftreten.

Der Benutzer soll Fehlermeldungen eindeutig der Quelle des Fehlers zuordnen können

Wie bei der funktionalen Korrektheit auch, kann die Anforderung erfüllt werden, durch Erstellung von Unit- oder UI-Tests für die Tuschi [5].

**Ästhetik:** Untermerkmal von Benutzbarkeit. Ausmaß, in dem eine Benutzeroberfläche eine für den Benutzer angenehme und zufriedenstellende Interaktion ermöglicht

*NFA-9:* Auf Basis von *NK-1* gibt es keine speziellen Anforderungen an die Ästhetik. Es ist ausreichend, wenn die UI der Tuschi vom Benutzer als „nicht störend“ empfunden wird.

Zum sicherstellen dieses Qualitätsmerkmals kann eine Umfrage an Benutzern, die die Benutzergruppe der Tuschi widerspiegelt, durchgeführt werden, um festzustellen, wie das UI empfunden wird.

**Barrierefreiheit:** Untermerkmal von Benutzbarkeit. Grad, in dem ein Produkt oder System von Menschen mit den unterschiedlichsten Eigenschaften und Fähigkeiten genutzt werden kann, um ein bestimmtes Ziel in einem bestimmten Nutzungskontext zu erreichen.

Aufgrund des Nutzungskontexts sind keine speziellen Anforderungen an Barrierefreiheit notwendig.

**Zuverlässigkeit:** Hauptmerkmal. Grad, in dem ein System, Produkt oder Bauteil bestimmte Funktionen unter bestimmten Bedingungen über einen bestimmten Zeitraum erfüllt.

**Verfügbarkeit:** Untermerkmal von Zuverlässigkeit. Grad, in dem ein System, ein Produkt oder eine Komponente betriebsbereit und zugänglich ist, wenn es für die Nutzung erforderlich ist.

*NFA-10:* ISO 25023 definiert folgende Maßnahmen zum Erstellen der Anforderungen an Verfügbarkeit:

- Dauer der Verfügbarkeit der Anwendung / Dauer der Verfügbarkeit des Betriebssystems
- Durchschnittliche Zeit, die die Anwendung nicht zur Verfügung steht

Es ist nicht notwendig, dass die Tuschi über die gesamte Lebensdauer des Benutzersystems aktiv ist. Der Benutzer soll selbst entscheiden, wann er die Tuschi benötigt, und kann diese dann selbstständig aktivieren.

**Wiederherstellbarkeit:** Untermerkmal von Zuverlässigkeit. Grad, in dem ein Produkt oder System im Falle einer Unterbrechung oder eines Ausfalls die unmittelbar betroffenen Daten wiederherstellen und den gewünschten Zustand des Systems wiederherstellen kann.

ISO 25023 definiert folgende Maßnahmen zum Erstellen der Anforderungen an Wiederherstellbarkeit:

- Ermittlung der durchschnittlichen Dauer zur Wiederherstellung der Betriebsfähigkeit der Anwendung
- Ermittlung des Anteils an Daten bei denen ein Backup durchgeführt wird

Im Rahmen des Nutzungskontext ist eine schnelle Wiederherstellung der Tuschi nach Absturz nicht notwendig.

**Fehlertoleranz:** Untermerkmal von Zuverlässigkeit. Grad, in dem ein System, ein Produkt oder eine Komponente trotz des Vorhandenseins von Hardware- oder Softwarefehlern wie vorgesehen funktioniert.

*NFA-11:* ISO 25023 definiert folgende Maßnahmen zum Erstellen der Anforderungen an Fehlertoleranz:

- Fehlervermeidung: Ermittlung des Anteils an Fehlermustern, welche unter Kontrolle gebracht werden müssen, um kritische Ausfälle der Anwendung und des ausführenden Systems zu vermeiden.
- Redundanz von Komponenten: Ermittlung des Anteils an Komponenten, welche redundant installiert sind, um einem Anwendungsausfall vorzubeugen.
- Ermittlung der durchschnittlichen Antwortzeit, nachdem ein Fehler aufgetreten ist.

Zur Fehlervermeidung sollen bei der Entwicklung der Tuschi Fehlermuster identifiziert und unter Kontrolle gebracht werden. Es sollen keine redundanten Komponenten vorhanden sein, da ein Ausfall der Anwendung unkritisch ist. Der Benutzer soll sofort darüber informiert werden, wenn ein Fehler in der Anwendung auftritt. Die Tuschi soll einen normalen Betrieb wieder aufnehmen können, nachdem ein Fehler aufgetreten ist.

**Reife:** Untermerkmal von Zuverlässigkeit. Grad, in dem ein System, Produkt oder Bauteil die Anforderungen an die Zuverlässigkeit bei normalem Betrieb erfüllt.

Das Konzept der Reife kann auf alle anderen Qualitätsmerkmale angewendet werden und beschreibt den Grad, in dem diese Qualitätsmerkmale die Bedürfnisse im normalen Betrieb der Tuschi erfüllen [5]. Dafür kann nach ISO 25023 die Testabdeckung zur Erfüllung der Anforderungen an Reife gezogen werden. Die Anforderungen zur Testabdeckung können dem Teil zur funktionalen Korrektheit entnommen werden.

**Sicherheit:** Hauptmerkmal. Grad, in dem ein Produkt oder System Informationen und Daten schützt, so dass Personen oder andere Produkte oder Systeme den ihrer Art und Berechtigung entsprechenden Grad an Datenzugriff haben.

Aufgrund des Nutzungskontext der Tuschi ist Sicherheit kein Teil der Anforderungen.

**Modifizierbarkeit:** Untermerkmal von Wartbarkeit. Grad, in dem ein Produkt oder System effektiv und effizient verändert werden kann, ohne dass es zu Fehlern oder einer Verschlechterung der bestehenden Produktqualität kommt.

*NFA-12:* Modifizierbarkeit ist eines der höher priorisierten Qualitätsmerkmale. Da die Tuschi mit der connect-bridge kommuniziert, soll sichergestellt sein, dass die Kommunikation nach einer Änderung an der connect-bridge auch weiterhin möglich ist. Dafür muss die Tuschi leicht und schnell modifizierbar sein. Sie soll dafür von jedem beliebigen Entwickler und von jedem beliebigen Betriebssystem aus angepasst werden können.

Zur Umsetzung dieser Maßnahmen sollen, wenn nötig, Tools verwendet werden, die zur Erfüllung dieser Anforderung beitragen.

**Wiederverwendbarkeit:** Untermerkmal von Wartbarkeit. Grad, in dem ein Vermögenswert in mehr als einem System oder beim Bau anderer Vermögenswerte verwendet werden kann.

*NFA-13:* Das Umfeld in dem die Tuschi genutzt wird profitiert stark von der Wiederverwendbarkeit von Komponenten, um Ressourcen zu sparen. Nach ISO 25023 zählen hierzu die Wiederverwendung von Code und die Einhaltung von Code Styles.

Bei der Implementierung der Tuschi soll darauf geachtet werden, dass Code möglichst wiederverwertbar erstellt wird und ein einheitlicher Code Style verwendet wird, um ihn für spätere Projekte des Unternehmens wiederverwenden zu können.

**Analysierbarkeit:** Untermerkmal von Wartbarkeit. Grad an Effektivität und Effizienz, mit dem es möglich ist, die Auswirkungen einer beabsichtigten Änderung an einem oder mehreren Teilen eines Produkts oder Systems zu bewerten, ein Produkt auf Mängel oder Fehlerursachen hin zu diagnostizieren oder zu ändernde Teile zu identifizieren.

Um Analysierbarkeit sicherzustellen, können Diagnosefunktionen entwickelt werden, welche Fehler loggen und in einer Diagnosedatei abspeichern [5]. Um Ressourcen zu sparen, soll auf die Implementierung solcher Funktionen verzichtet werden.

**Prüfbarkeit:** Untermerkmal von Wartbarkeit. Grad der Effektivität und Effizienz, mit dem Prüfkriterien für ein System, ein Produkt oder ein Bauteil festgelegt und Prüfungen durchgeführt werden können, um festzustellen, ob diese Kriterien erfüllt wurden.

Anforderungen an Prüfbarkeit werden bereits von den Anforderungen an die funktionale Korrektheit abgedeckt.

**Übertragbarkeit:** Hauptmerkmal. Grad der Effektivität und Effizienz, mit dem ein System, ein Produkt oder eine Komponente von einer Hardware-, Software- oder sonstigen Betriebs- oder Nutzungsumgebung auf eine andere übertragen werden kann.

*NFA-14:* Aufgrund des Nutzungskontext wird ein hoher Grad an Übertragbarkeit an die Tuschi gestellt. Da die Benutzergruppe von verschiedenen Orten aus operieren und gleichzeitiger Zugriff auf die Tuschi sichergestellt sein soll, muss auch sichergestellt sein, dass die Anwendung zu jeder Zeit von überall abrufbar ist.

Zur Umsetzung dieser Anforderungen sollen Entwicklungswerkzeuge verwendet werden, die die Übertragbarkeit der Tuschi sicherstellen.

**Anpassbarkeit:** Untermerkmal von Übertragbarkeit. der Grad, in dem ein Produkt oder System effektiv und effizient an unterschiedliche oder sich entwickelnde Hardware-, Software- oder andere Betriebs- oder Nutzungsumgebungen angepasst werden kann.

*NFA-15:* ISO 25023 definiert hier einen Grad an Anpassbarkeit der Software in Bezug auf:

- Hardwareumgebung
  - Anforderungen an die Hardwareumgebung sind besonders hoch, da die Entwickler nicht alle mit der gleichen Hardware arbeiten. Die Tuschi soll an alle Hardware-Varianten angepasst sein, welche, in Bezug auf die Leistungsparameter des Benutzersystems, mit dem durchschnittlichen Benutzersystem vergleichbar oder besser sind.
- Softwareumgebung
  - Da es sich bei der Tuschi um eine Webapp handelt, wird vorausgesetzt, dass sich diese im Webbrowser öffnen lässt. Die Tuschi soll dementsprechend mit allen gängigen Webbrowsern aufrufbar sein. Da Webbrowser immer wieder Updates unterliegen, soll die Tuschi auch nach diesen Updates noch funktionsfähig sein.
- Betriebssystem
  - Die Tuschi soll an beliebige Betriebssysteme anpassbar sein, da im Entwicklungsumfeld Betriebssystemversionen und ganze Betriebssysteme ausgetauscht werden könnten.

**Austauschbarkeit:** Untermerkmal von Übertragbarkeit. Grad, in dem ein Produkt ein anderes spezifiziertes Softwareprodukt für denselben Zweck in derselben Umgebung ersetzen kann.

Austauschbarkeit ist aufgrund des Nutzungskontext für die Tuschi nicht relevant.

**Installierbarkeit:** Untermerkmal von Übertragbarkeit. Grad der Effektivität und Effizienz, mit der ein Produkt oder System in einer bestimmten Umgebung erfolgreich installiert und/oder deinstalliert werden kann.

*NFA-16:* Da die Tuschi im Entwicklungsprozess verwendet wird, soll der Benutzer eine möglichst einfache Methode haben die Tuschi zu starten. Dafür kann die Dauer der Installation, sowie die Einfachheit der Installation als Messung zur Festlegung der Anforderungen in Betracht gezogen werden [5].

Einer der Vorteile einer Webapp ist, dass die Möglichkeit besteht, diese auf einem Server in einem Netzwerk zu starten, um sie mit dem Webbrowser des Benutzers ansteuern zu können. Dadurch ist keine Installation auf dem Benutzersystem notwendig.



Die Tuschi soll so aufgebaut sein, dass die Möglichkeit besteht, diese zentral auf einem Server zur Verfügung zu stellen.

### 3.5. Erstellung der User Stories

Anhand der funktionalen Anforderungen und in Hinblick auf die nichtfunktionalen Anforderungen können folgende User Stories erstellt werden:

User Story 1:

Als Entwickler möchte ich eine oder mehrere Dateien von meiner Festplatte in eine Liste hinzufügen können, um später eine, mehrere oder alle davon an die connect-bridge senden zu können.

Akzeptanzkriterien:

- AK-1: Es muss keine Prüfung des Dateityps stattfinden, da das Zielsystem falsche Dateien ablehnt.
- AK-2: Doppelt hinzugefügte Dateien sollen nicht erneut zur Liste hinzugefügt werden.
- AK-3: Es soll nur der Dateiname - ohne Pfad - in der Liste angezeigt werden.

User Story 2:

Als Entwickler möchte ich eine Datei auswählen und an die connect-bridge senden, um die laufende Entwicklung der connect-bridge zu testen.

Akzeptanzkriterien:

- AK-4: Falls das empfangende System (connect-bridge) mit einer Fehlermeldung antwortet, möchte ich diese direkt angezeigt bekommen.
- AK-5: Die Fehlermeldung vom empfangenden System kann 1:1 weitergegeben werden und muss nicht übersetzt werden o.ä.
- AK-6: Unabhängig von Erfolg oder Fehler, soll die zuletzt ausgewählte Datei ausgewählt und der Fokus auf dem Absenden-Button bleiben, so dass ich die Datei sofort wieder absenden kann.
- AK-7: Im Erfolgsfall soll eine Meldung "OK" erscheinen

User Story 3:

Als Entwickler möchte ich mehrere Dateien auf einmal auswählen können, um sie mit einem Klick nacheinander an die connect-bridge senden zu können.

Akzeptanzkriterien:

- AK-8: Funktionalität wie beim einzelnen Senden, aber nur eine Datei auf einmal soll gesendet werden.
- AK-9: Während eine Übertragung von mehreren Dateien läuft, soll der Button gesperrt sein, damit keine parallelen Übertragungen gestartet werden können.

## 4. Erfüllung der funktionalen Anforderungen

### 4.1. Allgemeiner Aufbau der Tuschi

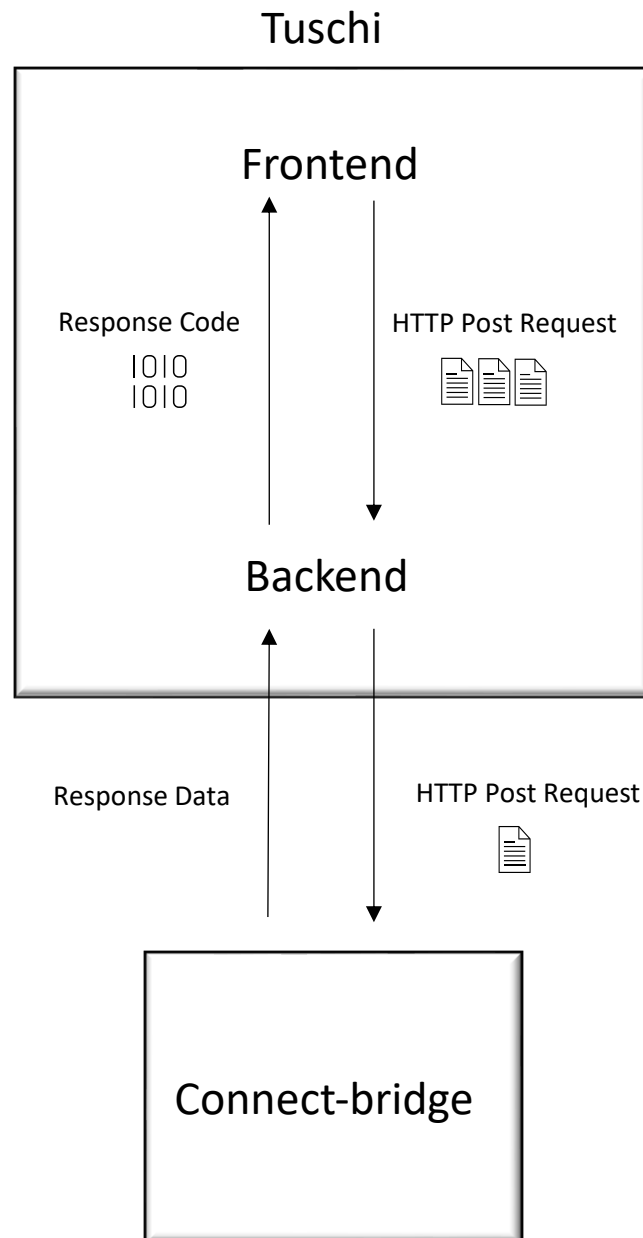


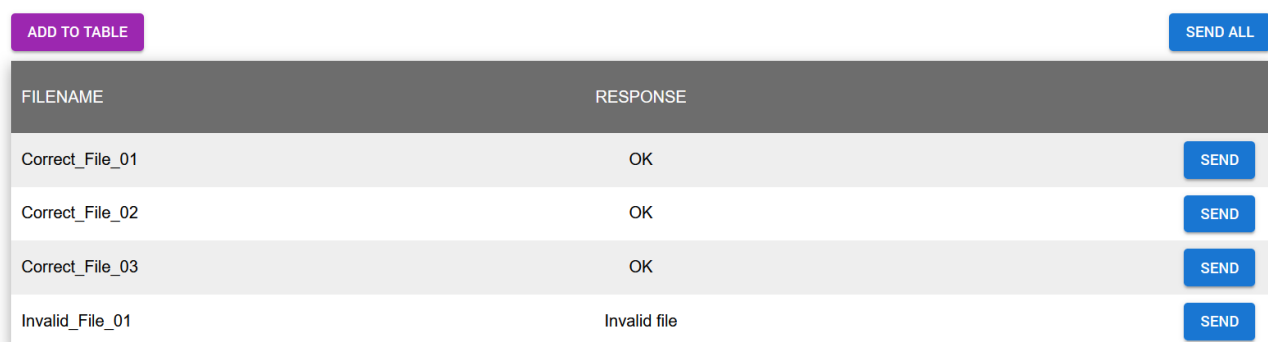
Abbildung 2 Aufbau der Tuschi (Quelle: Eigene Abbildung)

Der folgende Teil der Arbeit vermittelt ein tieferes Verständnis für den Aufbau der Tuschi. Eine Skizzierung des Aufbaus der Tuschi kann Abbildung 2 entnommen werden.

Das Frontend (Abbildung 3) besteht aus einer React Anwendung mit Typescript als Programmiersprache. Der Benutzer interagiert mit dem React Frontend, um die Dateien von seiner Festplatte in die Tuschi zu laden. Diese Dateien können anschließend via HTTP Post Request an das Spring Backend gesendet werden. Das Ansteuern des Backends führt dazu, dass eine weitere HTTP Post Request an die connect-bridge gesendet wird. Die Verarbeitung der Dateien in der connect-bridge findet mit Unternehmensproprietärer Software statt. Die connect-bridge antwortet auf die Request des Backends nach Verarbeitung der Daten. Die dadurch erhaltene Response enthält den Status der Übertragung, welcher zurück an das React Frontend gesendet und dem Benutzer angezeigt wird.

Die Entscheidung dafür die React Bibliothek zur Entwicklung der Tuschi zu verwenden, liegt in der einfachen Erweiterbarkeit der Anwendung über node modules, welche über den mit React bereitgestellten node package manager hinzugefügt werden können. Dadurch wird die Tuschi den Qualitätsanforderungen für Wartbarkeit und Modifizierbarkeit gerecht.

Die Entscheidung dafür Spring Boot für die direkte Interaktion mit der connect-bridge einzusetzen ergibt sich aus der Suche nach einer funktional angemesseneren Lösung, die vom Frontend allein übernommen werden kann. Zum Zeitpunkt der Entwicklung existiert keine den Anforderungen entsprechende Lösung, um das Frontend allein mit der connect-bridge kommunizieren zu lassen. Da es bereits eine vom Unternehmen selbst entwickelte Lösung gibt, welche auf Spring Boot basiert und mit der connect-bridge kommuniziert, ist dies die nächst-angemessene Lösung zur Erfüllung der funktionalen Angemessenheit.



The screenshot shows a web interface with a table. At the top left is a purple button labeled 'ADD TO TABLE'. At the top right is a blue button labeled 'SEND ALL'. The table has two columns: 'FILENAME' and 'RESPONSE'. There are four rows of data. Each row has a blue 'SEND' button on the right side.

FILENAME	RESPONSE
Correct_File_01	OK
Correct_File_02	OK
Correct_File_03	OK
Invalid_File_01	Invalid file

Abbildung 3 Abbildung der Tuschi Frontend UI (Quelle: Eigene Abbildung)

Die folgenden Abschnitte erklären die Implementierung der Tuschi, welche zur Erfüllung der funktionalen Anforderungen FA-1 und FA-2 benötigt wird.

## 4.2. Implementierung User Story 1

Das Frontend besteht aus zwei funktionalen React Komponenten. Zum einen der FileTable Komponente und der FileTableRow Komponente. Die beiden Komponenten haben eine 1 zu n Beziehung. 1 FileTable beinhaltet n FileTableRows.

Es wird ein Element zum Interagieren für den Benutzer benötigt, damit er Dateien von seiner Festplatte in die Anwendung laden kann. Hierfür wird ein Button Element angelegt (Listing 1), welches ein input Element beinhaltet.

```

1. <div className="actions__element">
2.   <Button
3.     variant="contained"
4.     component="label"
5.     color="secondary"
6.   >
7.     Add files
8.     <input
9.       id="file-upload"
10.      type="file"
11.      multiple
12.      onChange={e => {
13.        if (e.target.files) {
14.          const files = Array.from(e.target.files);
15.          handleAddFiles(DicomFile.arrayFrom(files));
16.        }
17.      }}
18.    />
19.   </Button>
20. </div>

```

Listing 1

Dem *type* Attribut des input Elements wird der Wert "file" zugewiesen. Die Zuweisung dieses Schlüsselwortes weist dem input Element die Funktion zu das File Explorer Fenster des Benutzersystems zu öffnen, wenn es angeklickt wird. Das *multiple* Attribut erlaubt es dem Benutzer mehrere Dateien gleichzeitig auszuwählen. Durch das Weglassen des *accept* Attributes werden alle Dateitypen akzeptiert, was für Akzeptanzkriterium AK-1 verlangt wird. Die Funktion in *onChange* wird aufgerufen, sobald der Benutzer Dateien ausgewählt hat. In Listing 1 Zeile 14 wird aus allen Dateien ein Array erzeugt. In Listing 1 Zeile 15 wird anschließend der Handler aufgerufen, welcher die Dateien in einem state der FileTable Komponente speichert. Der Handler in Listing 1 Zeile 15 wandelt das Array vom Typ File in ein Array vom Typ DicomFile um. Dieser selbst erstellte Datentyp dient als Hilfsklasse und beinhaltet zum einen die Datei, die der Benutzer ausgewählt hat und zum anderen den Übertragungsstatus der Datei, nachdem diese gesendet wird. Der Aufbau des Datentyps kann Listing 2 entnommen werden.

```

1. class DicomFile {
2.   _file: File;
3.
4.   _transmissionResponse: string;
5.
6.   constructor(file: File) {
7.     this._file = file;
8.     this._transmissionResponse = '';
9.   }
10.
11.   get name() {
12.     return this.file.name;
13.   }
14.
15.   get file() {
16.     return this._file;
17.   }
18.
19.   get transmissionResponse() {
20.     return this._transmissionResponse;
21.   }
22.
23.   set transmissionResponse(res: string) {
24.     this._transmissionResponse = res;
25.   }
26.
27.   public static arrayFrom(fileList: File[]): DicomFile[] {
28.     return fileList.map(fileElement => new DicomFile(fileElement));
29.   }
30. }
31.
32. export default DicomFile;

```

Listing 2

Um Akzeptanzkriterium AK-2 zu erfüllen, muss vor dem Speichern der Dateien in den state eine Überprüfung stattfinden, um bereits vorhandene Dateien aus der Liste auszusortieren. Mit den Funktionen *containsFile* (Listing 3) und *mergeFiles* (Listing 4) wird sichergestellt, dass nur, dem Namen nach, einzigartige Dateien in den state aufgenommen werden.

```

1. const containsFile = (file: DicomFile, list: DicomFile[]) =>
2.   list.some(elem => elem.name === file.name);

```

Listing 3

```

1. const mergeFiles = (currentFiles: DicomFile[], uploadedFiles: DicomFile[]): DicomFile[]
  => {
2.   const newFiles = uploadedFiles.filter(elem => !containsFile(elem, currentFiles));
3.
4.   return [...currentFiles, ...newFiles];
5. };

```

Listing 4

Listing 5 implementiert die Liste in Form einer Tabelle, in der die Dateien referenziert werden. Listing 5 Zeilen 6 – 16 bilden den Header der Tabelle, welche aus drei Spalten besteht:

- *Filename*: In dieser Spalte wird der Dateiname angezeigt.

- *Response*: Diese Spalte beinhaltet den Übertragungsstatus der Datei, nachdem diese an die connect-bridge gesendet wurde.
- Die dritte Spalte des Headers hat keinen Titel. In dieser Spalte werden die Aktionen aufgeführt, welche mit der Datei durchgeführt werden können.

```

1. <div
2.   className="container"
3.   data-testid="table-container"
4. >
5.   <div className="table">
6.     <div className="table__header">
7.       <div className="table__header-item table__header-item--filename">
8.         <div className="header-text header-text-filename">Filename</div>
9.       </div>
10.      <div className="table__header-item">
11.        <div className="header-text">Response</div>
12.      </div>
13.      <div className="table__header-item">
14.        <div className="header-text" />
15.      </div>
16.    </div>
17.    <div className="table__content">
18.      {fileStore.map(file => (
19.        <FileTableRow
20.          key={file.name}
21.          dicomFile={file}
22.          tableIsTransmitting={tableIsTransmitting}
23.        />
24.      )}}
25.    </div>
26.  </div>
27. </div>

```

#### Listing 5

Listing 5 Zeilen 17 – 25 bilden den Körper der Tabelle. In der Implementation wird über den state mit den Dateien in der FileTable Komponente iteriert und für jedes Element eine FileTableRow angelegt. Jede FileTableRow hat folgende props:

- *key*: Da es sich hier um eine Liste handelt benötigt React ein key prop, um die Komponenten voneinander unterscheiden zu können. Hierfür kann der Name der Datei genommen werden, da dieser innerhalb des state, aufgrund der Implementierung, immer einzigartig ist.
- *dicomFile*: die im fileStore gespeicherte Datei wird an die Komponente übergeben.
- *tableIsTransmitting*: Jeder FileTableRow Komponente wird mitgeteilt, ob die Tabelle bereits einen Transfer aller Dateien an die connect-bridge durchführt.

Eine FileTableRow Komponente besteht dem Header der Tabelle entsprechend aus 3 Elementen (Listing 6).

```

1. <div className="table__row-table-data table-data--filename">{dicomFile.name}</div>
2. <div className="table__row-table-data">{dicomFile.transmissionResponse}</div>
3. <div className="table__row-table-data table-data--actions">
4.   <Button
5.     variant="contained"
6.     disabled={tableIsTransmitting || rowIsTransmitting}
7.     onClick={handleSetResponse}
8.   >
9.     SEND
10.  </Button>
11.</div>

```

Listing 6

Das erste Element beinhaltet den Dateinamen. Dies wird zur Erfüllung von Akzeptanzkriterium AK-3 benötigt. Das zweite Element beinhaltet die Rückmeldung der Übertragung an die connect-bridge, nachdem sie versendet wird. Das dritte Element beinhaltet einen Button, der eine Funktion zum Senden der Datei enthält.

## 4.3. Implementierung User Story 2

### 4.3.1. Backendimplementierung

Mit der zweiten User Story soll eine Möglichkeit bereitgestellt werden, die im React Frontend gesammelten Dateien an die connect-bridge zu senden. An dieser Stelle kommt das Spring Boot Backend zum Einsatz. Listing 7 zeigt den Quellcode des DicomControllers, welcher die Request an die connect-bridge sendet.

```

1. @RestController
2. public class DicomController {
3.
4.     private static final Logger LOG = LogManager.getLogger(DicomController.class);
5.     private final DicomTransferService dicomTransferService;
6.     private final ObjectMapper mapper;
7.
8.     @Autowired
9.     public DicomController(
10.         DicomTransferService dicomTransferService,
11.         ObjectMapper jacksonObjectMapper
12.     ) {
13.         this.dicomTransferService = dicomTransferService;
14.         this.mapper = jacksonObjectMapper;
15.     }
16.
17.     @CrossOrigin(origins = "http://localhost:3000")
18.     @PostMapping("transmit")
19.
20.     public ResponseEntity<Object> forwardDicomData(
21.         @RequestHeader("dicomHost") String host,
22.         @RequestHeader("dicomPort") int port,
23.         @RequestHeader("calledAeTitle") String calledAeTitle,
24.         @RequestHeader("maxPduSize") int maxPduSize,
25.         @RequestParam("files[]") List<MultipartFile> files
26.     ) {
27.         ArrayNode responseNode = mapper.createArrayNode();
28.         for (MultipartFile file : files) {
29.             try {
30.                 Integer responseCode = dicomTransferService.transfer(

```

```

31.                                     host,
32.                                     port,
33.                                     calledAeTitle,
34.                                     maxPduSize,
35.                                     file.getBytes()
36.                                     );
37.         if (responseCode == null) {
38.             addToResponseNode(responseNode, file, "Invalid file");
39.             continue;
40.         }
41.         addToResponseNode(responseNode, file, responseCode.toString());
42.     } catch (IOException e) {
43.         addToResponseNode(responseNode, file, e.getMessage());
44.     } catch (CommonDicomException e) {
45.         addToResponseNode(responseNode, file, e.getMessage());
46.     }
47. }
48. return new ResponseEntity(responseNode, HttpStatus.OK);
49. }
50.
51. private void addToResponseNode(
52.     ArrayNode responseNode,
53.     MultipartFile file,
54.     String transmissionResponse
55. ) {
56.     ObjectNode objectNode = mapper.createObjectNode();
57.     objectNode.put("name", file.getOriginalFilename());
58.     objectNode.put("transmissionResponse", transmissionResponse);
59.     responseNode.add(objectNode);
60. }
61. }

```

Listing 7

Der Controller verfügt über genau einen Endpunkt welcher über localhost:8080/transmit angesprochen werden kann. Er bekommt als Parameter eine Liste vom Typ MultipartFile übertragen (Listing 7 Zeile 25). Des Weiteren müssen im Header der Request die Adressdaten der connect-bridge übergeben werden (Listing 7 Zeilen 21-24). Die Header übergeben folgende Daten an die Request:

- *dicomHost*: Die Host-Adresse der connect-bridge. Läuft die connect-bridge als Windows-Dienst auf dem ausführendem Betriebssystem, hört sie auf ‚localhost‘.
- *dicomPort*: Der Port der connect-bridge. Zum Zeitpunkt der Entwicklung hört die connect-bridge auf Port 104. Dieser Wert kann sich im Laufe der Zeit ändern.
- *calledAeTitle*: Dieser Wert entspricht einer Passphrase und hat zum Zeitpunkt der Entwicklung den Wert ‚connect-box‘. Auch dieser Wert kann sich im Laufe der Zeit ändern.
- *maxPduSize*: Dieser Wert beträgt ‚32768‘.

Der *files* Parameter in Listing 7 Zeile 25 ist das Array an Dateien, welches vom Frontend übertragen wird. In Listing 7 Zeile 38, 41, 43 und 45 wird, abhängig vom verarbeiteten Zustand der Datei, eine *responseNode* erzeugt, die hinterher wieder an das Frontend gesendet werden soll. Damit wird Akzeptanzkriterium AK-5 umgesetzt. Die Response hat folgenden beispielhaften Aufbau:



```
[
    { filename: "Dateiname1", transmissionResponse: "0" },
    { filename: "Dateiname2", transmissionResponse: "Invalid file" }
]
```

Es wird über jede Datei innerhalb der Liste iteriert. Während jeder Iteration versucht der dicomTransferService die Datei an die connect-bridge zu senden. Ist die übermittelte Datei keine DICOM Datei wird auch kein Status Code zurück übermittelt. In dem Fall wird als transmissionResponse der String "Invalid file" eingetragen und zusammen mit dem Dateinamen der responseNode übergeben. Dieser Prozess wiederholt sich, bis alle Dateien der Liste übertragen wurden. Anschließend wird die responseNode an den das Frontend zurückgegeben (Listing 7 Zeile 48).

Der Sinn des dicomTransferService ist, die Dateien an die vorgesehene Stelle zu senden. Dafür wird die C-Store Operation des DIMSE-Service aufgerufen, welche die Datei transferiert und anschließend die Rückmeldung des Transfers erhält. In Listing 8 Zeile 16 wird die Verbindung zur connect-bridge aufgebaut. In Listing 8 Zeilen 22-28 wird anschließend die C-Store Request erstellt, die die Datei an die connect-bridge sendet. Listing 8 Zeilen 30-31 enthalten die Response der C-Store Request. In dieser ist auch der Status Code enthalten, welcher an das Frontend weitergeleitet werden soll.

```
1. public Integer transferToPacs(
2.     String host, int port, String calledAeTitle, int maxPduSize, byte[] data
3. ) throws CommonDicomException {
4.     PacsConnectionFactory connectionFactory =
5.         new PacsConnectionFactory()
6.             .addDestination(host, port)
7.             .addDicomSendMetaData(
8.                 callingAeTitle, calledAeTitle, maxPduSize
9.             );
10.
11.     LOG.debug("Beginning transfer via dicom send process");
12.     DicomObject dicomObject = new DicomObject(data);
13.     Integer dicomResponseCode = null;
14.
15.     try (PacsConnection connection = openConnection(connectionFactory, dicomObject)) {
16.         connection.connect();
17.
18.         PresentationContext selectedPresentationContext = connection
19.             .getUser()
20.             .getSelectedPresentationContext();
21.
22.         CStoreRQ cStoreRQ = new CStoreRQ(
23.             DimseHandler.getMessageID(),
24.             getAffectedSOPClass(dicomObject),
25.             dicomObject.getDicomElement(BaseDicomTagNames.TAG_sopInstanceUID)
26.                 .getValueAsString(0), data,
27.             selectedPresentationContext
28.         );
29.
30.         CStoreRP dicomResponse = connection.sendDicomMessage(cStoreRQ,
```

```

31.         FileMetaHeader.getFileMetaHeaderLenght(data));
32.         dicomResponseCode = dicomResponse.getStatus().getStatusCode();
33.         connection.getUser().release();
34.     } catch (CommonDicomException e) {
35.         return dicomResponseCode;
36.     } catch (NullPointerException e) {
37.         return dicomResponseCode;
38.     }
39.     return dicomResponseCode;
40. }

```

Listing 8

### 4.3.2. Frontendimplementierung

Das React Frontend soll eine Möglichkeit bereit stellen mit dem Spring Boot Backend zu kommunizieren. Die Implementation des Transmitters kann Listing 9 entnommen werden.

```

1. import axios from 'axios';
2. import DicomFile from '../models/DicomFile';
3.
4. const backendConfig = {
5.   dicomHost: 'localhost',
6.   dicomPort: '104',
7.   calledAeTitle: 'connect-box',
8.   maxPduSize: '32768',
9. };
10.
11. export const transmit = async (dicomFiles: DicomFile[]) => {
12.   const url = 'http://localhost:8080/transmit';
13.   const formData = new FormData();
14.   dicomFiles.forEach(dicomFile => {
15.     formData.append('files[]', dicomFile.file);
16.   });
17.
18.   const config = {
19.     headers: {
20.       'content-type': 'multipart/form-data',
21.       dicomHost: backendConfig.dicomHost,
22.       dicomPort: backendConfig.dicomPort,
23.       calledAeTitle: backendConfig.calledAeTitle,
24.       maxPduSize: backendConfig.maxPduSize,
25.     },
26.   };
27.
28.   return axios
29.     .post(url, formData, config)
30.     .then(
31.       response =>
32.         // response.data is of Type Array<Object>
33.         response.data
34.     )
35.     .catch(() => 'No Connection');
36. };

```

Listing 9

Listing 9 Zeilen 4-9 beschreiben die Konfiguration der Header der Anfrage. Listing 9 Zeilen 11-37 beschreiben die Funktion *transmit*, welche eine asynchrone Funktion ist, die dem Aufrufer einen Promise mit den Daten der Übertragung zurück gibt, wie in Abschnitt 4.3.1 dargestellt. Für das

Erstellen der Request wird “axios“ verwendet, welches über den node package manager dem Projekt hinzugefügt wird. Das Modul erlaubt das Erstellen von HTTP Requests.

Die *transmit* Funktion wird anschließend den Komponenten bereitgestellt. In dem Handler des Send Buttons der FileTableRow Komponente wird die *transmit* Funktion aufgerufen und die Datei, die von der Komponente gehalten wird, übergeben (Listing 10).

```

1.  const handleSetResponse = () => {
2.    setRowIsTransmitting(true);
3.    transmit([dicomFile]).then(data => {
4.      setTransmissionResponse(dicomFile, data[0]);
5.      setRowIsTransmitting(false);
6.    });
7.  };

```

Listing 10

Bekommt die Anwendung eine Rückmeldung von “0“ ist die Übertragung erfolgreich verlaufen und dem Benutzer soll im Frontend der Wert “OK“ angezeigt werden. In allen anderen Fällen soll die Rückmeldung 1:1 weitergegeben werden. Damit werden die Akzeptanzkriterien AK-4 und AK-7 erfüllt.

#### 4.4. Implementierung User Story 3

Zum Erfüllen der dritten User Story soll dem Benutzer die Möglichkeit gegeben werden, alle Dateien gleichzeitig an das System zu senden. Der Endpunkt im Spring Boot Backend ist bereits dafür angepasst mehrere Dateien gleichzeitig entgegen zu nehmen. Im Frontend kann wieder die *transmit* Methode aus Listing 9 verwendet werden, da auch diese bereits ein Array an Dateien entgegen nehmen kann. Listing 11 implementiert den Handler, der aufgerufen wird, wenn der „Send All“ Button in der UI geklickt wird. Die Implementierung dieser Funktion erfüllt Akzeptanzkriterium AK-8.

```

1.  const handleSetResponse = () => {
2.    setTableIsTransmitting(true);
3.    transmit(fileStore).then(data => {
4.      data.forEach((response: Response) => {
5.        fileStore.forEach(dicomFile => {
6.          if (dicomFile.name === response.name) {
7.            setTransmissionResponse(dicomFile, response);
8.          }
9.        });
10.     });
11.     setTableIsTransmitting(false);
12.   });
13.   };

```

Listing 11

Zur Erfüllung von Akzeptanzkriterium AK-9 wird zunächst der FileTable Komponente ein neuer state hinzugefügt, der einen boolschen Wert hält, welcher initial auf false gesetzt ist. Dieser state wird dem disabled Attribut des „Send All“ Buttons hinzugefügt. Sobald der Handler in der onClick Funktion

des Buttons aufgerufen wird, wird der Wert des states auf true gesetzt und der Button kann nicht mehr angeklickt werden, bis die transmit Funktion alle Daten verarbeitet hat. Anschließend wird der state wieder auf false gesetzt und der Button kann wieder angeklickt werden.

#### 4.5. Erfüllung der funktionalen Korrektheit

Für die Erfüllung der funktionalen Korrektheit sollen folgende Fälle auf Basis der Implementierung in Betracht gezogen werden:

- Testfall T-1: Wenn ein Benutzer eine Datei von seiner Festplatte in die Tuschi lädt, soll diese in der FileTable Komponente angezeigt werden.
- Testfall T-2: Wenn ein Benutzer mehrere Dateien von seiner Festplatte in die Tuschi lädt, sollen diese in der FileTable Komponente angezeigt werden.
- Testfall T-3: Wenn ein Benutzer eine Datei von seiner Festplatte in die Tuschi lädt, soll diese in der FileTable Komponente angezeigt werden. Anschließend lädt der Benutzer die gleiche Datei erneut in die Tuschi. Diese soll nicht erneut in der FileTable Komponente enthalten sein.
- Testfall T-4: Wenn der „Send“ Button in der FileTableRow Komponente geklickt wird, soll, nach Abschluss des Transfers, die korrekte Rückmeldung dem Benutzer angezeigt werden.
- Testfall T-5: Wenn der „Send“ Button in der FileTableRow Komponente geklickt wird, soll der Button nicht anklickbar sein, bis der Transfer abgeschlossen ist.
- Testfall T-6: Wenn der „Send All“ Button in der FileTable Komponente geklickt wird, soll, nach Abschluss des Transfers, die korrekte Rückmeldung dem Benutzer angezeigt werden.
- Testfall T-7: Wenn der „Send All“ Button in der FileTable Komponente geklickt wird, soll der Button nicht anklickbar sein, bis der Transfer abgeschlossen ist.

Die Abdeckung der Testfälle erfolgt mittels Implementierung und kann den jeweiligen Anwendungen entnommen werden.

### 5. Erfüllung der nichtfunktionalen Anforderungen

#### Erfüllung des Zeitverhaltens (NFA-1)

Die Anforderungen werden durch Implementation von AK-9 erfüllt.

#### Erfüllung der Koexistenz (NFA-2)

Nach momentanem Stand der Entwicklung belegt die Tuschi zwei lokale Ports auf dem Benutzersystem:

- localhost:3000
- localhost:8080

Es ist darauf zu achten, dass keine anderen Anwendungen auf dem Benutzersystem diese Ports nutzen. Die Ports können jederzeit vom Benutzer selbst in der Konfiguration der Anwendung anpassen, sofern das benötigt wird.

### **Erfüllung der Interoperabilität (NFA-3)**

Interoperabilität wird mit Implementation von User Story 2 bereitgestellt.

### **Erfüllung der Verständlichkeit (NFA-4)**

Die Ergebnisse der Usability Tests können Anhang 1 entnommen werden. Zusätzlich zu der Erfüllung des Tests allein haben einige Testpersonen weitere Anmerkungen hinterlassen, welche im Hinblick auf die weitere Entwicklung der Tuschi relevant sein könnten. Diese können ebenfalls Anhang 1 entnommen werden.

### **Erfüllung der Wiedererkennbarkeit (NFA-5)**

NFA-5-1: Erfüllung durch Dokumentation der Ergebnisse in vorliegender Arbeit.

### **Erfüllung der Erlernbarkeit (NFA-6)**

Erfüllung durch Implementierung von AK-4 und AK-5

### **Erfüllung der Bedienbarkeit (NFA-7)**

Erfüllung durch Durchführung von Usability Tests. Die Ergebnisse können dem Anhang entnommen werden.

### **Erfüllung der Fehlervermeidung (NFA-8)**

Erfüllung anhand von Unit Tests. Ergebnisse können Anhang 1 entnommen werden.

### **Erfüllung der Ästhetik (NFA-9)**

Erfüllung anhand von Unit Tests. Ergebnisse können Anhang 1 entnommen werden.

### **Erfüllung der Verfügbarkeit (NFA-10)**

Für die Erfüllung dieses Qualitätsmerkmals hat der Benutzer die Möglichkeit sich sowohl Frontend, als auch Backend aus dem Bitbucket zu holen und auf dem eigenen System auszuführen.

### **Erfüllung der Fehlertoleranz (NFA-11)**

Erfüllung durch Implementierung der User Stories 1 bis 3.

### **Erfüllung der Modifizierbarkeit (NFA-12)**

Zur Erfüllung zählt die Wahl der Frameworks für Frontend und Backend in Kombination mit Git und Bitbucket.

Für das Frontend gilt: TypeScript hat JavaScript gegenüber den Vorteil, dass es beim Programmieren Typsicherheit gibt. So können Fehler in der Entwicklung bereits früh abgefangen werden.

Git und Bitbucket erlauben es mehreren Entwicklern gleichzeitig an verschiedenen Versionen der Software zu arbeiten.

#### **Erfüllung der Wiederverwendbarkeit (NFA-13)**

Die Implementierung weist einen einheitlichen und sauberen Programmierstil auf. Alle Komponenten des React Frontends, sowie die Funktionen des Spring Boot Backends können für gleiche oder andere Zwecke wiederverwendet werden.

#### **Erfüllung der Übertragbarkeit (NFA-14)**

Erfüllung durch Nutzung von Git und Bitbucket. Jeder Entwickler hat Zugriff auf die Tuschi, sofern er sich im Unternehmensinternen Netzwerk befindet

#### **Erfüllung der Anpassbarkeit (NFA-15)**

React und Spring Boot erfüllen die Anforderungen an Anpassbarkeit.

#### **Erfüllung der Installierbarkeit (NFA-16)**

Installierbarkeit ist nicht Teil der Entwicklung und liegt nicht im Rahmen dieser Arbeit.

## 6. Fazit

Das Hauptaugenmerk dieser Arbeit lag vor allem auf der Entwicklung der Tuschi. Die SQuaRE Reihe zur Evaluation von Software konnte als Schablone zum Erstellen der Anforderungen an die Tuschi dienen, und um die Qualität der Anwendung auf Basis dieser Anforderungen zu sichern. Dies war in der Planungsphase des Projektes besonders hilfreich, da so früh erkannt wurde, welche Aspekte in Betracht gezogen werden müssen, um eine qualitativ hochwertige Anwendung zu entwickeln.

Anhand der Abschnitte 4 und 5 lässt sich außerdem gut erkennen, welche Teile der Tuschi die Anforderungen bereits erfüllen und welche Teile noch ausgebaut oder verändert werden müssen, damit die Tuschi im Produktionsbetrieb eingesetzt werden kann. Die vorliegende Arbeit kann in dem Sinne als Checkliste betrachtet werden.

Die Tuschi erfüllt die nichtfunktionalen Anforderungen, welche nicht auf die funktionale Korrektheit zurückzuführen sind, wie sie in Abschnitt 3.4 festgelegt wurden. Die Anforderungen an die funktionale Vollständigkeit sind zwar erfüllt, jedoch fehlt die vorgegebene Testabdeckung, welche durch die funktionale Korrektheit und einige nichtfunktionale Anforderungen gefordert werden. Der momentane Implementierungsstand deckt die Testfälle *T-1* bis *T-3* ab. Somit ist eine funktionale Korrektheit erst gegeben, wenn auch die Testfälle *T-4* bis *T-7* abgedeckt sind.

Eine Kritik an dieser Arbeit kann vor allem dem Evaluationsmodell zugetragen werden. Da es sich um ein 11 Jahre altes Modell handelt, besteht die Möglichkeit, dass es, durch einen schnellen Wandel der technologischen Landschaft, gewisse Qualitätsmerkmale vermisst, welche bei der Evaluation und dem Erstellen von Anforderungen nützlich gewesen wären.

## 7. Literaturverzeichnis

- [1] Balzer, Lars, Andreas Frey, und Peter Nenniger. „Was ist und wie funktioniert Evaluation? Empirische Pädagogik, Zeitschrift zu Theorie und Praxis erziehungswissenschaftlicher Forschung.“ *Zeitschrift zu Theorie und Praxis*, 1999: 21.
- [2] Bastien, J.M. Christian. „Usability testing: a review of some methodological and technical aspects of the method.“ *International Journal of Medical Informatics*, 2008: e18-e23.
- [3] ISO/IEC. *ISO 25010, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. 2011.
- [4] ISO/IEC. *ISO 25020, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Quality measurement framework*. 2019.
- [5] ISO/IEC. *ISO 25023, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Measurement of system and software product quality*. 2016.
- [6] Lange, Clara. „matthes.in.tum.de.“ 2010. <https://www.matthes.in.tum.de/file/6ikz2i550193/sebis-Public-Website/-/Proseminar/Lange-Qualitaetsmodelle-Ausarbeitung.pdf>.
- [7] Nielsen, Jakob. *Usability Engineering*. London: Academic Press, 1993.
- [8] Suzanne Robertson, James Robertson. *Mastering the Requirement Process*. Harlow: Addison-Wesley, 2006.
- [9] vdf-Wirtschaftsinformatik. *System-Entwicklung in der Wirtschaftsinformatik*. Zürich: vdf Hochschulverlag AG, 2022.



## 8. Anhang

### Anhang 1: Protokoll der Usability Tests

Testperson	Konnte alle Schritte ohne Hilfe befolgt werden?	Wird die Oberfläche der Tuschi als „Störend“ empfunden?	Sonstige Anmerkungen
1	Ja	Nein	
2	Ja	Nein	„Es könnte angezeigt werden, ob eine Datei bereits erfolgreich versendet wurde.“
3	Ja	Nein	„Ein Button zum Löschen der Tabelle oder einzelner Dateien wäre hilfreich“
4	Ja	Nein	
5	Ja	Nein	

Tabelle 2

## 9. Eidesstattliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt und mich anderer als der in den beigefügten Verzeichnissen angegebenen Hilfsmittel nicht bedient habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit entspricht der eingereichten schriftlichen Fassung exakt. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen hat.

Ort, den

---

Vorname, Name

## 10. Sperrvermerk

Diese Arbeit enthält vertrauliche Daten der Firma Visus Health IT GmbH und ist daher einschließlich aller ihrer Teile urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verfassers/ bzw. der Visus Health IT GmbH unzulässig. Bei der elektronischen Prüfung der wissenschaftlichen Arbeit durch den Fachbereich Elektrotechnik und Informatik wird die Geheimhaltung gewahrt.