technische universität dortmund

IT in Produktion und Logistik

# Masterarbeit

# Backend zu einer datenbankbasierten Webapplikation für den Einsatz in der digitalen Lehre

B.Sc. Džemal Alić
Matrikelnummer: 167000
Studiengang: Maschinenbau

ausgegeben am:
09.09.2020
eingereicht am:
24.01.2021

Betreuer:
Dr.-Ing. Dipl.-Inf. Anne Antonia Scheidler
M. Sc. Joachim Hunker

Technische Universität Dortmund
Fakultät Maschinenbau
IT in Produktion und Logistik
http://www.itpl.mb.tu-dortmund.de

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| ACID | Atomicity, Consistency, Isolation, Durability |
| API | Application Programming Interface |
| DB | Datenbank |
| DoD | Department of Defence |
| HTTP | Hypertext Transfer Protocol |
| IP | Internet Protocol |
| ISO | International Standard Organisation |
| JSON | JavaScript Object Notation |
| Ms | Millisekunden |
| OSI | Open System Interconnection |
| REST | Representational State Transfer |
| SOAP | Simple Object Acess Protocol |
| SQL | Structured Query Language |
| TCP/IP | Transmission Control Protocol / Internet Protocol |
| UDP | User Datagram Protocol |
| UML | Unified Modeling Language |
| URI | Unified Resource Identifier |
| WSDL | Web Services Description Language |

# Tabellenverzeichnis

# Symbolverzeichnis

| | |
|---|---|
| Ø | Durchschnitt |

# 1 Einleitung

Durch die zunehmende Digitalisierung finden immer mehr Veränderungsprozesse in sämtlichen Bereichen der Gesellschaft statt (Ladel et al., 2018). Zu diesen Bereichen zählt die Bildung (Schimank, 2013). Computergestützte Lernwerkzeuge finden dadurch eine wachsende Anwendung (Ladel et al., 2018). Diese Lernwerkzeuge ermöglichen neue und individuelle Lernformen. Eine dieser Lernformen ist das asynchrone Lernen. Das asynchrone Lernen beschreibt unter anderem die zeitversetzte Kommunikation zwischen den Studierenden und der lehrenden Person. Dadurch können die Studierenden die Zeiteinteilung für das Lernen unabhängig von Lehrveranstaltungen gestalten und im Selbststudium unterstützt werden (Niegemann et al., 2004). Des Weiteren können mit Hilfe von computergestützten Lernwerkzeugen große Datenmengen über das Lernverhalten der Studierenden erfasst werden. Die Lehrenden können diese Daten auswerten, um neue Erkenntnisse über das Lernverhalten der Studierenden und über die Qualität der eingesetzten Lernmethoden zu gewinnen. So können beispielsweise Aufgabenstellungen lokalisiert werden, welche viele Studierende vor eine Herausforderung stellen. Diese Erkenntnisse können daraufhin genutzt werden, um die Lehre zu optimieren (Ladel et al., 2018).
Die aktuelle COVID-19 Pandemie verdeutlicht, dass die Notwendigkeit von digitalen Lernwerkzeugen gegeben ist, um den Lehrbetrieb orts- und zeitunabhängig zu gestalten (Claudia Ricci, 2020). Die dafür erforderlichen Technologien sind bereits vorhanden. Es ist lediglich erforderlich, diese Technologien für den Bereich der Bildung zu nutzen (Ladel et al., 2018). Ein Bereich der Gesellschaft, in welchem die Anpassung an die aktuellen Gegebenheiten deutlich schneller durchgeführt werden konnte als im Bereich der Bildung, ist die Wirtschaft. Dem liegt zu Grunde, dass in dem Bereich der Wirtschaft neue Technologien schneller adaptiert werden als in dem Bereich der Bildung (Tata Consultancy Services Deutschland GmbH, 2020). Aus diesem Grund ist es vor allem jetzt wichtig, die Digitalisierung im Bereich der Bildung zu fördern, sodass es nicht zur Benachteiligung von Generationen kommt, welche zu Zeiten einer Pandemie leben. Dabei bietet es sich besonders im Rahmen der Informationstechnik an, den Lehrbetrieb durch computergestützte Lernwerkzeuge zu unterstützen.

Das Ziel dieser Arbeit ist die Konzeption und Entwicklung des Backends eines computergestützten Lernwerkzeugs. Dabei handelt es sich um eine datenbankbasierte Webapplikation zur Übung der Structured Query Language (SQL), welche eine Sprache zur Abfrage und Verwaltung von relationalen Datenbanken ist (Fuchs, 2018). Diese soll sich aus einem Webserver, einer Datenbank und einem Frontend zusammensetzen. Der Webserver bildet in Kombination mit der Datenbank das Backend der Webapplikation. Das erste Unterziel, welches zur Erfüllung des Hauptziels erreicht werden soll, ist die Auswahl eines geeigneten Architekturansatzes für das Zusammenspiel der drei

Komponenten. Nach der Auswahl des Architekturansatzes soll die Architektur der Webapplikation unter Berücksichtigung der geforderten Funktionalitäten konzipiert werden. So soll die Webapplikation im Übungsbetrieb des Masterstudiums an der Technischen Universität Dortmund eingesetzt werden. Die in der Webapplikation enthaltenen Übungsaufgaben sollen zu dem Lerninhalt, der von dem Lehrstuhl IT in Produktion und Logistik eingesetzt wird, korrespondieren. Dabei soll die Gestaltung der Übungsaufgaben unter der Berücksichtigung didaktischer Grundlagen erfolgen. Den Studierenden soll die Eingabe von SQL Statements ermöglicht werden, wobei der Lernprozess durch eine integrierte Benutzerunterstützung gefördert werden soll. Die Webapplikation soll mehrbenutzerfähig sein, sodass jeder Studierende die Möglichkeit erhält, Datenbanktabellen unabhängig von anderen Studierenden zu manipulieren und abzufragen. So kann sich jeder Studierende in der Webapplikation mit seiner UniMail registrieren und anschließend anmelden, um eigene Tabellen für die jeweiligen Übungsaufgaben zu erhalten. Nach der Eingabe einer SQL Abfrage, wird diese geprüft. Der Studierende erhält nach der Prüfung der SQL Abfrage die Rückmeldung, ob die Eingabe die richtige Lösung zu der entsprechenden Aufgabe ist oder ob die Eingabe semantische oder syntaktische Fehler enthält. Gelingt es den Studierenden nicht die korrekte Eingabe eigenständig zu erarbeiten, bietet die Webapplikation die Möglichkeit, die Lösung zu der jeweiligen Aufgabe anzuzeigen. Des Weiteren soll in der Webapplikation eine Hilfefunktion integriert werden, welche als Anleitung zur Benutzung der Webapplikation fungiert. Nachdem die Architektur der Webapplikation konzipiert ist, soll diese implementiert werden. Das zweite Unterziel, welches zur Erfüllung des Hauptziels erreicht werden soll, ist die Validierung der Implementierung durch geeignete Testfälle. Dazu werden Komponententests, Integrationstests und Systemtests durchgeführt. Bei dem Komponententest werden die einzelnen Softwaremodule isoliert geprüft. Die Integrationstests werden dazu genutzt die Zusammenspiele zwischen den Softwaremodulen zu prüfen. Für die Systemtests wird eine Testumgebung erzeugt, welche dazu genutzt wird die Belastungsgrenzen der Webapplikation zu ermitteln.

Das Forschungsthema wird in einer Arbeitsgruppe bearbeitet. Die Arbeitsgruppe setzt sich aus Dzemal Alić und Osman Alić zusammen. Die Konzeption und Entwicklung des Backends und die Aufbereitung des eingesetzten Datenbanksystems werden von Dzemal Alić durchgeführt, wobei die Konzeption der Übungsaufgaben und die Entwicklung des Frontends von Osman Alić durchführt werden.
Somit wird in dieser Arbeit die Konzeption und Entwicklung des Backends thematisiert. Das Backend setzt sich dabei aus einem Webserver und einer Datenbank zusammen. Zunächst werden mögliche Architekturansätze für die Webapplikation diskutiert. Im Anschluss wird eine geeignete Architektur für die Webapplikation ausgewählt. Nachdem die Architektur für die Webapplikation festgelegt wurde, werden die Programm-schnittstellen definiert, welche für die Kommunikation zwischen dem Webserver und der Datenbank und zwischen dem Backend und dem Frontend eingesetzt werden. Für die

Kommunikation zwischen dem Frontend und dem Backend werden Anfragen über das Hypertext Transfer Protocol (HTTP), welche sich nach den Leitsätzen des Representational State Transfer (REST) richten und das WebSocket-Protokoll verwendet. Für das Datenbankmanagementsystem besteht die Vorgabe, dass PostgreSQL werden soll. In PostgreSQL werden die erforderlichen Datenbanktabellen für das Benutzer- und Aufgabenmanagement angelegt. Der Aufbau und der Inhalt der Datenbanktabellen erfolgt nach der Ausarbeitung der Aufgabenstellungen von Osman Alić. Nachdem die Datenbank mit sämtlichen Tabellen befüllt ist, wird der Webserver entwickelt. Die Kommunikation zwischen dem Webserver und der Datenbank erfolgt mithilfe der von PostgreSQL zur Verfügung gestellten Socket-Schnittstelle. Die Entwicklung des Webservers wird mithilfe der plattformübergreifenden Open-Source-JavaScript-Laufzeitumgebung Node.js realisiert. Nach der Implementierung sämtlicher Komponenten der Webapplikation, wird die Implementierung durch geeignete Testfälle validiert. Dieser Schritt erfolgt in Zusammenarbeit innerhalb der Arbeitsgruppe. Die Softwaretests teilen sich dabei in den Komponententest, den Integrationstest und den Systemtest auf. Bei dem Komponententest werden die einzelnen Softwaremodule der Webapplikation zunächst isoliert getestet. In Abhängigkeit des Testergebnisses einzelner Testfälle, werden entdeckte Fehler korrigiert. Anschließend werden die Tests wiederholt. Dieses Vorgehen wird solange durchgeführt, bis sämtliche Tests erfolgreich verlaufen. Nach den erfolgreichen Komponententests werden die Zusammenspiele zwischen den einzelnen Softwaremodulen in den Integrationstests untersucht. Aufgetretene Fehler werden lokalisiert und korrigiert. Anschließend wird eine Testumgebung erzeugt mit der die Systemtests durchgeführt werden. Dabei werden die Belastungsgrenzen der Webapplikation ermittelt. Darüber hinaus werden von O. Alić Usability-Tests durchgeführt, bei denen mehrere Probanden die Webapplikation nutzen. Die Usability-Tests sollen zeigen, ob die Nutzung der Webapplikation für die Benutzer intuitiv und verständlich ist. Abschließend wird die Webapplikation hinsichtlich der Benutzerfreundlichkeit optimiert.

# 2 Informationstechnische Grundlagen zur Entwicklung des Backends einer Webapplikation

In diesem Kapitel werden die informationstechnischen Grundlagen vorgestellt, welche zur Entwicklung des Backends einer Webapplikation genutzt werden können. Zunächst wird das Grob- und Feindesign von Softwarearchitekturen erarbeitet. Mit diesen Architekturen wird das Grundgerüst einer zu entwickelnden Webapplikation festgelegt. In den einzelnen Abschnitten dieses Kapitels werden anschließend mögliche Konzepte und Produkte aufgeführt, welche die Bestandteile der Softwarearchitektur bilden. So wird das Konzept der relationalen Datenbanken beschrieben, wobei PostgreSQL beispielhaft als Datenbankmanagementsystem vorgestellt wird. Dabei werden das Relationsmodell und die Structured Query Language erörtert. Daraufhin wird auf das Produkt JavaScript als mögliche Skriptsprache zur Entwicklung serverseitiger Anwendungen eingegangen. Abschließend werden in diesem Kapitel grundlegende Datenaustauschkonzepte zwischen den einzelnen Komponenten der Softwarearchitektur aufgezeigt, wobei der Fokus auf das TCP/IP-Netzwerkmodell gesetzt ist.

## 2.1 Grob- und Feindesign der Architektur einer Webapplikation

Die Architektur eines Softwareprodukts definiert die Strukturen eines Softwaresystems und bildet das Fundament für die gesamte Entwicklung des Softwareprodukts. Die Strukturen beinhalten Informationen zu den Komponenten, welche in dem Softwareprodukt enthalten sind oder zu externen Komponenten. Außerdem stellen die Strukturen in Softwarearchitekturen die Zusammenhänge und Beziehungen der aufgeführten Komponenten dar (Bass et al., 2013). Bei den Komponenten kann es sich beispielsweise um eine Datenbank, einen Server, einen Client oder um ein Objekt handeln. Um ein komplexes Softwareprodukt zu entwickeln, sind die Betrachtung des Gesamtsystems und die Spezifikation seiner Strukturen erforderlich. So kann die Softwarearchitektur als Bauplan für die Softwareentwicklung angesehen werden (Dunkel & Holitschke, 2003). Für die Realisierung der Strukturierung von Softwareprodukten ist die Aufteilung der Software in logische Schichten ein etablierter Ansatz (Buschmann et al., 2013). In der Regel setzen sich Softwareprodukte aus den drei Standard-Softwareschichten zusammen. Die Standard-Softwareschichten bestehen aus der Präsentations-, der Anwendungs- und der Persistenzschicht. Zwischen den drei Schichten liegt eine lose Kopplung vor. Das bedeutet, dass die jeweiligen Schichten kaum Informationen zu den übrigen Schichten erhalten. Wie die Schichten zusammenhängen, ist in der folgenden Abbildung ersichtlich.

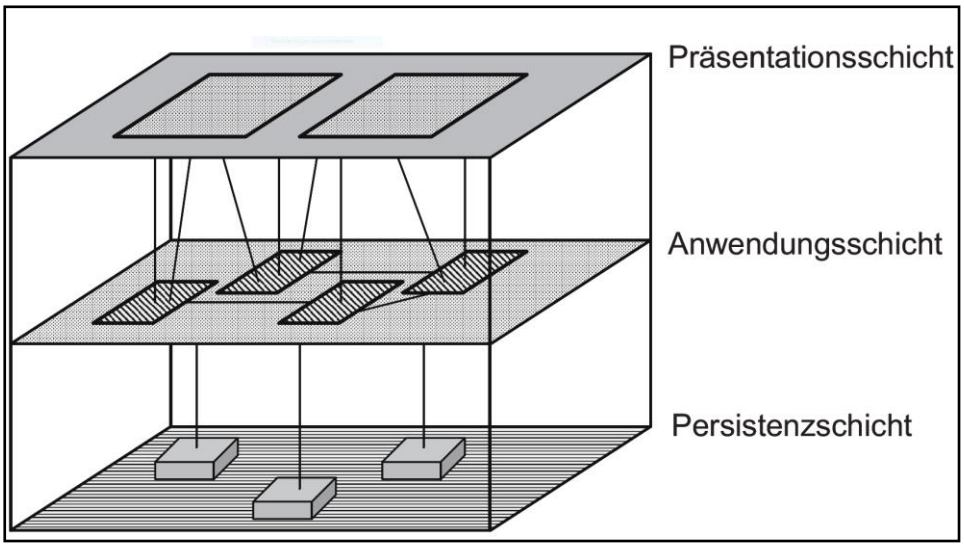


**Abbildung 2.1.1** – Standard-Softwareschichten (Dunkel & Holitschke, 2003)

Die Präsentationsschicht stellt die Schnittstelle zum Benutzer der Software dar. Dazu werden die Daten der Software auf einer Benutzeroberfläche visualisiert. Neben der Visualisierung der Daten, dient die Präsentationsschicht zur Interaktion mit dem Benutzer. So kann der Benutzer beispielsweise auf Buttons klicken oder Texteingaben tätigen. Die Anwendungsschicht wird im Rahmen der Softwareentwicklung auch als Applikations- oder Geschäftslogikschicht bezeichnet. Sie umfasst die fachlichen Funktionalitäten des Softwareprodukts. Zu diesen Funktionalitäten zählen die Geschäftsprozesse, welche entsprechende Geschäftsobjekte beinhalten (Jakob, 2011). Die Persistenzschicht besitzt die Aufgabe Daten dauerhaft zu speichern und diese der Anwendungsschicht zur Verfügung zu stellen. In der Regel wird für die Persistenzschicht ein Datenbanksystem benutzt. Mit Hilfe der Drei-Schichten-Architektur ist eine klare Trennung der Aufgaben und Verantwortlichkeiten bei der Softwareentwicklung gegeben (Dunkel & Holitschke, 2003). Dadurch, dass die einzelnen Schichten eigene Aufgaben besitzen, welche von den übrigen Schichten losgelöst sind, wird die Verständlichkeit des Gesamtsystems gefördert. Außerdem wird die Projektorganisation verbessert, da die Aufgaben innerhalb eines Projektteams klar getrennt sind. Dabei ist das zur Entwicklung benötigte Wissen schichtenspezifisch. Des Weiteren bietet die Drei-Schichten-Architektur eine Grundlage für Redundanzfreiheit, da jede Funktion des Softwareprodukts eindeutig einer Schicht zugewiesen ist. Der Datenaustausch zwischen den Schichten erfolgt durch den Einsatz von Programmschnittstellen (Dunkel & Holitschke, 2003). Die Technologien zum Datenaustausch zwischen den einzelnen Schichten werden in dem Abschnitt 2.4 näher thematisiert.

Je nach Softwareprodukt, sind unterschiedliche Adaptionen der Drei-Schicht-Architektur möglich. So kann es sich bei dem Softwareprodukt unter anderem um eine Desktopanwendung oder um eine Webapplikation handeln. Bei Desktopanwendungen handelt es sich um Anwendungen, welche auf einem Computer installiert werden. Webapplikationen sind hingegen Anwendungen, welche in einem Webbrowser wie

beispielsweise Chrome, Firefox oder Edge ausgeführt werden. So ist keine weitere Softwareinstallation neben dem Webbrowser erforderlich. Webapplikationen werden häufig unabhängig von der Erreichbarkeit über das World Wide Web innerhalb von Unternehmen genutzt. Um eine Anwendung als Webapplikationen zu kategorisieren, ist lediglich der Einsatz von Webtechnologien notwendig. Dabei handelt es sich um Client-Server Anwendungen. Der Webbrowser bildet den Webclient welcher mit dem Webserver kommuniziert (Rohr, 2018). Die für Webapplikationen grundlegende 3-Tier-Architektur ist in der folgenden Abbildung 2.1.2 dargestellt.

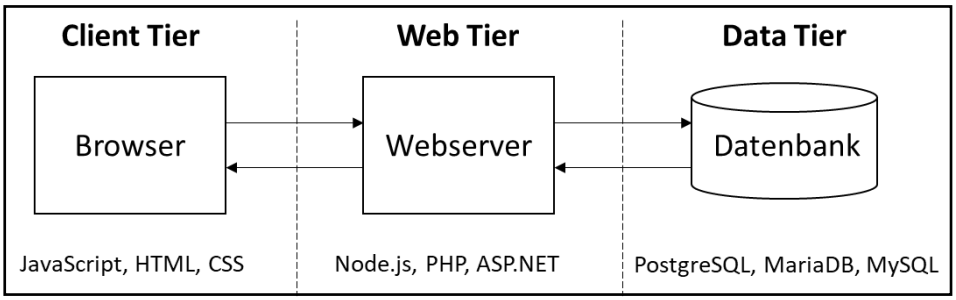


**Abbildung 2.1.2** – 3-Tier-Architektur einer Webapplikation (Rohr, 2018)

Die 3-Tier-Architektur besteht aus der Client-Schicht, der Web-Schicht und der Datenschicht. In Bezugnahme auf die drei Standard-Softwareschichten aus der Abbildung 2.1.1, bildet die Client-Schicht die Präsentationsschicht, die Web-Schicht die Anwendungsschicht und die Datenschicht die Persistenzschicht einer Webapplikation. Im Zusammenhang mit Webapplikationen wird die Client-Schicht auch als Frontend bezeichnet. Die Web-Schicht und die Datenschicht bilden das Backend von Webapplikationen. Diese beiden Schichten können auf unterschiedliche physikalische Server aufgeteilt oder als Programme auf einem physikalischen Server installiert werden. Des Weiteres sind für die einzelnen Schichten in der Abbildung 2.1.2 beispielhaft Technologien aufgeführt, welche für die Umsetzung der jeweiligen Schicht genutzt werden können. Bei komplexen Webapplikationen können die einzelnen Schichten noch weiter unterteilt werden. So können moderne Webapplikationen in der Client-Schicht beispielsweise auch Applikationen von Mobilgeräten vorsehen. Außerdem kann der Webserver bei komplexen Anwendungen um einen Applikations-server erweitert werden. Die Nutzung von mehreren, unterschiedlichen Datenbanken ist ebenfalls möglich. Im Allgemeinen lassen sich jedoch auch diese komplexen Webapplikationen in die 3-Tier-Architektur aufteilen. Die gerichteten Kanten in der Abbildung 2.1.2 repräsentieren die möglichen Datenflüsse zwischen den einzelnen Schichten.

Während die Softwarearchitektur die Struktur des Gesamtsystems beschreibt, befasst sich die Unified Modeling Language (UML) mit der Prozessmodellierung innerhalb der Programme, welche in den jeweiligen Schichten enthalten sind. Die Softwarearchitektur skizziert somit das Grobdesign, wobei die UML das Feindesign eines Softwareprodukts beschreibt. Es existiert eine Vielzahl an UML-Diagrammen, mit welchen die

Komponenten eines Programms visualisiert werden können. Um das dynamische Verhalten eines Systems zu visualisieren, eignet sich das Zustandsdiagramm. Das Zustandsdiagramm beschreibt die einzelnen Objekte in einem Programm oder Programmteil. Des Weiteren werden die Zustände der Objekte zu gegebenen Zeitpunkten dargestellt (Kleuker, 2013). Die Grundstruktur eines Zustandsdiagramms ist in der folgenden Abbildung 2.1.3 dargestellt.

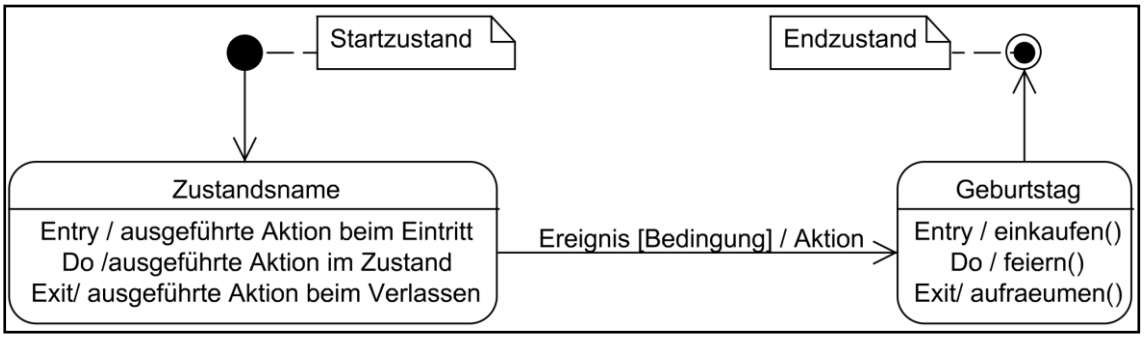


**Abbildung 2.1.3** – Grundstruktur eines Zustandsdiagramms (Kleuker, 2013)

Zustandsdiagramme besitzen grundsätzlich einen Startzustand und mindestens einen Endzustand. Der Startzustand wird als aufgefüllter Kreis dargestellt. Der Endzustand wird als ausgefüllter Kreis mit zusätzlicher Umrandung dargestellt. Ein Zustandsdiagramm wird zur deterministischen Verhaltensbeschreibung genutzt. Daraus folgt, dass ein Objekt zu jedem Zeitpunkt genau einen Zustand annehmen kann. Der darauf folgende Zustand ist dabei in Anhängigkeit der möglichen Folgeereignissen definiert. Zustände sind als Rechtecke mit abgerundeten Ecken dargestellt. Zustände enthalten den Namen des jeweiligen Zustands. Darüber sind auch die Aktionen, die bei Zustandseintritt, Zustandsaustritt und die während des Zustands ausgeführt werden definiert. Aktionen können dabei sowohl einfache Zuweisungen als auch das Aufrufen komplexer Programmabschnitte sein. Es ist nicht zwingend erforderlich, dass ein Zustand sämtliche Aktionen aufweist. Die gerichteten Pfeile beschreiben Transitionen, wobei es sich dabei um Zustandsübergänge handelt. Transitionen können drei Attribute annehmen. Diese Attribute können ein Ereignis, ein Erfüllen einer Bedingung oder die Durchführung einer Aktion sein. Eine Transaktion, welches ein Ereignis als Attribut besitzt, beschreibt eine Zustandsänderung, die außerhalb des Programms eintritt. Diese Zustandsänderung versetzt einen Zustand in einen Folgezustand. Wird eine Transition mit einer Bedingung aufgeführt, so wird eine Zustandsänderung ausgelöst sobald diese Bedingung erfüllt ist. Die Bedingung wird in eckigen Klammern definiert. Ist das Attribut der Transaktion eine Aktion, so wird eine Zustandsänderung vom Programm initiiert. Die Aktion ist für die Transaktion definiert, da solche Aktionen vor dem Zustandseintritt ausgelöst werden. Nicht beschriftete Transaktionen werden sofort nach der Aktion des Zustandsaustritts ausgeführt. Ein in der Abbildung 2.1.3 nicht dargestelltes Element von Zustandsdiagrammen ist die Raute. Dieses Element ist in der Abbildung 2.1.4 dargestellt.

**Abbildung 2.1.4** – Raute-Element in Zustandsdiagrammen (Müller, 2005)

Das Raute-Element wird zur Vereinigung von Ablaufalternativen genutzt. Die Ablaufalternativen sind dabei lediglich von Bedingungen abhängig (Müller, 2005; Unhelkar, 2018).

## 2.2 Relationale Datenbanken

Datenbanken dienen im Allgemeinen zur elektronischen und persistenten Speicherung und Verwaltung von zusammengehörigen Daten (Mertens et al., 2012). In diesem Kapitel wird das relationale Datenbankmodell thematisiert. Dazu wird die Grundidee des Relationsmodells anhand der Mengenlehre verdeutlicht. Im Anschluss werden die Bestandteile einer relationalen Datenbank aufgeführt. Des Weiteren werden in diesem Abschnitt das Datenbankmanagementsystem PostgreSQL und die Datenbanksprache SQL vorgestellt, wobei grundlegende SQL Statements zur Auswertung und Manipulation von Daten betrachtet werden.

### *2.2.1 Das Relationsmodell*

Das Relationsmodell basiert auf dem mathematischen Konzept der Relation (Elmasri & Navathe, 2004). So kann die Struktur relationaler Datenbanken formal beschrieben werden. Eine Relation beschreibt die Beziehung zwischen Objekten, welche in gegebenen Mengen vorhanden sind. Bei Objekten und Mengen handelt es sich dabei um unterschiedliche Entitäten. Objekte teilen sich in atomare Objekte und in Objekte der n-Tupel Form. Für das Relationsmodell sind die Objekte der n-Tupel Form relevant. Ein Tupel beschreibt dabei eine Kollektion von Informationen eines Objektes. Informationen, welche ein Objekt beschreiben, sind als Attribute des Objektes definiert. Eine Menge repräsentiert hingegen eine ungeordnete Kollektion von Objekten. So sind Objekte Elemente einer Menge. Eine Menge kann dabei kein Element einer anderen Menge sein. Es ist jedoch möglich, dass eine Menge eine Teilmenge einer anderen Menge ist. Damit eine Menge A als Teilmenge einer anderen Menge B gilt, ist es erforderlich, dass sämtliche Objekte der Menge A auch in der Menge B enthalten sind. Darüber hinaus können Operationen auf Mengen ausgeführt werden. Die wichtigsten Operationen sind die Vereinigung, die Differenz und der Schnitt von Mengen. Das Ergebnis einer Vereinigung von zwei Mengen sind sämtliche Objekte, welche in beiden Mengen enthalten sind. Das Ergebnis einer Differenz von zwei Mengen enthält die Objekte aus einer Menge, wobei die gleichen Objekte aus den Mengen nicht in dem Ergebnis

enthalten sind. Das Ergebnis des Schnitts von zwei Mengen enthält die Objekte, welche in beiden Mengen vorhanden sind (Studer, 2019).
Nach der Aufführung der erforderlichen Grundlagen der Mengenlehre für das Relationsmodell, wird nun die Anwendung des Relationsmodells im Bereich relationaler Datenbanken betrachtet. Die Daten in einer relationalen Datenbank werden in Form von Relationen gespeichert. Die Relationen können dabei der Ordnung n sein. Eine Relation der Ordnung n beschreibt eine Tabelle mit n Spalten, wobei eine Tabelle einem Objekt entspricht. Eine Datenbank beschreibt eine Menge, welche beliebig viele Objekte enthalten kann (Studer, 2019). So können in einer Datenbank beliebig viele Tabellen vorhanden sein. Die Um die aufgeführten Zusammenhänge anhand eines Beispiels zu verdeutlichen, wird im Folgenden die Menge *Studierende* betrachtet.

*Studierende:= { (167000, Mike, Büker), (269199, Tim, Karge) }*

In dem Beispiel ist die Menge ein Objekt, welches durch die Relation zweiter Ordnung *Studierende* beschrieben wird. Diese Relation kann als Tabelle mit zwei Zeilen und drei Spalten dargestellt werden:

**Tabelle 2.2.1** – Darstellung einer Relation zweiter Ordnung als Tabelle

| | | |
|---|---|---|
| 167000 | Mike | Büker |
| 269199 | Tim | Karge |

Die dargestellte Tabelle 2.2.1 bildet eine Relationsinstanz. Die Zeilen werden als Tupel oder als Datensätze bezeichnet. Damit diese Tabelle zur Speicherung von Daten in einer Datenbank nach dem Relationsmodell genutzt werden kann, ist es notwendig weitere Informationen über die Objekte der Menge miteinzubeziehen. Dabei handelt es sich um die Attribute und den Objektnamen. Aus diesen beiden Komponenten setzt sich das Relationsschema zusammen. Attribute beschreiben Eigenschaften von Objekten. Für die tabellarische Darstellung bedeutet dies, dass jede Spalte mit einem Namen versehen wird. Dieser Name ist das Attribut des Objektes. Innerhalb der Spalte sind die Daten zu dem jeweiligen Attribut enthalten (Elmasri & Navathe, 2004). In dem aufgeführten Beispiel handelt es sich bei den Attributen um die Matrikelnummer, den Vornamen und den Nachnamen von Studierenden. Die Studierenden sind in diesem Fall ein Objekt. Mit diesen Informationen wird die Tabelle 2.2.1 in der Tabelle 2.2.2 erweitert.

**Tabelle 2.2.2** – Darstellung einer Relation zweiter Ordnung als Tabelle mit Attributen

| **Studierende** | | |
|---|---|---|
| **Matrikelnummer** | **Vorname** | **Nachname** |
| 167000 | Mike | Büker |
| 269199 | Tim | Karge |

In der dargestellten Tabelle 2.2.2 ist die Relationsinstanz aus der Tabelle 2.2.1 enthalten. Die Tabelle 2.2.1 ist also eine Relationsinstanz des Relationsschemas

Studierende(Matrikelnummer, Vorname, Nachname). Für die Informationen zu den Attributen sind bestimmte Wertebereiche definiert. So besteht eine Matrikelnummer in jedem Fall aus einer Ganzzahl und kann keine beliebige Zeichenkette sein. Diese Wertebereiche sind als Domänen definiert. Domänen werden meistens durch einen Datentypen beschrieben, wobei es möglich ist weitere Eingrenzungen des Wertebereichs durchzuführen (Steiner, 2017). In der folgenden Tabelle 2.2.3 sind die häufigsten Datentypen, welche als Domänen verwendet werden, aufgelistet.

**Tabelle 2.2.3** – Häufige Datentypen zur Festlegung der Domäne

| Datentyp (Domäne) | Beschreibung |
|---|---|
| CARDINAL | Natürliche Zahlen |
| INTEGER | Ganze Zahlen |
| NUMERIC | Numerische Werte in Dezimaldarstellung |
| CHAR | Einzelne Zeichen |
| STRING | Zeichenketten |
| BOOLEAN | Logische Werte |

Neben den in der Tabelle 2.2.3 aufgeführten Datentypen zur Festlegung der Domäne, kann auch der Fall eintreten, dass der Wertebereich nicht bekannt ist. In diesem Fall wird das entsprechende Feld mit *NULL* gefüllt.
Bei der Konzeption einer Datenbank ist es wichtig Datensätze innerhalb von Tabellen eindeutig identifizieren zu können. Mit einer eindeutigen Kennung ist es beispielsweise möglich, zusammenhängende Datensätze tabellenübergreifend auszuwerten. Dazu werden Attribute zur Identifikation von Datensätzen genutzt. Ein Attribut, welches zur Identifikation von Datensätzen genutzt wird, wird Primärschlüssel-Attribut genannt (Steiner, 2017). In der Tabelle 2.2.2 ist die Matrikelnummer das Primärschlüssel-Attribut, da eine Matrikelnummer nur einmal für jeden Studierenden vergeben werden kann.

### *2.2.2 PostgreSQL als Datenbankmanagementsystem*

Bisher sind relationale Datenbanken im Allgemeinen betrachtet worden. Ein Datenbanksystem setzt sich jedoch aus Datenbanken und einem Datenbank-managementsystem zusammen. Das Datenbankmanagementsystem dient zur Verwaltung und Bearbeitung von Datenbanken. Darüber hinaus bietet das Datenbank-managementsystem die Möglichkeit, mehrere Benutzer für die Verwaltung und Bearbeitung der Datenbanken einzurichten, welche angepasste Zugriffsrechte besitzen (Mertens et al., 2012). Es existieren zahlreiche Produkte, welche als Datenbankmanagementsystem eingesetzt werden können. Gängige Beispiele für Datenbankmanagementsysteme für relationale Datenbanken sind unter anderem MySQL, MariaDB und PostgreSQL. Sämtliche Datenbanksysteme können in der Persistenzschicht einer Webapplikation eingesetzt werden. Das in dieser Arbeit vorgegebene Datenbankmanagementsystem ist PostgreSQL.

PostgreSQL ist ein frei zugängliches Datenbankmanagementsystem, das in den 1980er Jahren erstmals entwickelt worden ist (Riggs & Ciolli, 2018). Seitdem wird dieses Produkt stetig weiterentwickelt und der Funktionsumfang erweitert. So ist PostgreSQL seit dem Jahr 2001 ACID konform (Studer, 2019).

ACID beschreibt dabei erwünschte Charakteristika von Transaktionen in Datenbankmanagementsystemen. Transaktionen sind dabei Operationen, welche die Datenbank von einem konsistenten Zustand in einen konsistenten, eventuell veränderten Zustand überführen (Elmasri & Navathe, 2004). Die Abkürzung ACID steht für atomicity, consistency, isolation und durability. Atomicity (Deutsch: Atomarität) stellt sicher, dass Transaktionen entweder vollständig oder gar nicht ausgeführt werden. Consistency (Deutsch: Konsistenzerhaltung) gewährleistet, dass die Datenbank vor dem Beginn und nach dem Abschließen einer Transaktion in einem konsistenten Zustand ist. Isolation gibt vor, dass eine Transaktion ohne den Einfluss anderer Transaktionen durchgeführt wird. So wird der Datenzugriff auf die betroffenen Daten zu Beginn einer Transaktion für andere Transaktionen gesperrt. Durabilty (Deutsch: Dauerhaftigkeit) beschreibt die persistente Speicherung der Ergebnisse der Transaktionen (Elmasri & Navathe, 2004).

Im Jahr 1994 ist PostgreSQL um einen SQL-Interpreter erweitert worden. Dadurch ist es möglich SQL Statements zur Auswertung und Manipulation von Datenbanken einzusetzen. PostgreSQL ist größtenteils SQL-Standard konform. Mit jedem Update von PostgreSQL wird die Einhaltung des SQL Standards ausgeweitet, wobei das Ziel verfolgt wird, den SQL-Standard in Zukunft vollständig zu gewährleisten (Riggs & Ciolli, 2018).

Die Datenbanken können mithilfe einer grafischen Oberfläche verwaltet und bearbeitet werden. Dazu bieten die Entwickler von PostgreSQL ein weiteres kostenloses Werkzeug namens pgAdmin an. Bei pgAdmin handelt es sich um eine Browseranwendung, welche über die IP Adresse und einen definierten Port des Datenbankservers auf die Datenbanken zugreifen. In pgAdmin können unterschiedliche Datenbankbenutzer angelegt werden. Diese können unterschiedlich berechtig werden, um zu gewährleisten, dass nur befugte Personen gewisse Änderungen an den Datenbanken vornehmen (The PostgreSQL Global Development Group, 2020). Darüber hinaus stellt PostgreSQL eine integrierte Socket-Schnittstelle zur Verfügung, welche externen Anwendungen die Kommunikation mit PostgreSQL vereinfacht ermöglicht. Dabei handelt es sich um eine in C programmierte Anwendung namens libpq. Mithilfe von libpq können externe Anwendungen über einen Socket-Kanal Daten von der Datenbank anfordern oder Daten durch den Einsatz von SQL Statements abfragen oder manipulieren. Um eine Verbindung aufzubauen ist es lediglich erforderlich den in PostgreSQL konfigurierten Port und die IP-Adresse des Datenbankservers mit einem gültigen Login in der externen Anwendung zu hinterlegen. Im Folgenden wird näher auf die Abfrage- und Manipulationsmöglichkeiten von Daten mithilfe von SQL Statements eingegangen.

### *2.2.3 Structured Query Language*

SQL hat sich im Bereich relationaler Datenbanksysteme als wichtigste Sprache etabliert (Fuchs, 2018). Mithilfe dieser Sprache können Datenstrukturen definiert, Daten abgefragt oder geändert und die Sicherheitskonzepte in Bezug auf die Benutzerberechtigungen gesteuert werden. In dieser Arbeit wird der Fokus auf die Abfrage und Manipulation von Daten durch die Nutzung von SQL gesetzt. SQL Statements sind Befehle, mit denen Abfragen oder Manipulationen von Daten ausgelöst werden (Fuchs, 2018). Es werden die grundlegenden SQL Statements vorgestellt, wobei die Ausarbeitung dieser SQL Statements nach dem Buch „Relationale Datenbanken: Von den theoretischen Grundlagen zu Anwendungen mit PostgreSQL“ von Thomas Studer erfolgen (Studer, 2019).
Wie in dem Abschnitt 2.1 beschrieben, werden die Daten innerhalb einer relationalen Datenbank in Tabellen gespeichert. Die Erzeugung einer leeren Tabelle wird mithilfe des SQL Statements CREATE TABLE umgesetzt. Dieses SQL Statement ist im Allgemeinen wie folgt aufgebaut:

```
CREATE TABLE Tabellenname (Attribut1 Domäne1, Attribut2 Domäne2)
```

Mit diesem SQL Statement wird eine leere Tabelle, also eine Relation ohne Tupel erzeugt. Es sind lediglich die Attribute der Tabelle mit der jeweiligen Domäne definiert. Um diese Tabelle im nächsten Schritt um Datensätze zu erweitern, wird das SQL Statement INSERT INTO genutzt. Das INSERT INTO Statement hat die folgende Form:

```
INSERT INTO Tabellenname VALUES (Wert1, Wert2)
```

In diesem verallgemeinerten Beispiel wird der zuvor erstellten Tabelle mit dem Namen „Tabellenname“ der Datensatz (Wert1, Wert2) hinzugefügt. Wert1 gehört dabei zum Attribut1 und Wert2 zum Attribut2. Es ist wichtig, dass Wert1 und Wert2 den jeweils definierten Wertebereich einhalten. Mit einem SQL Statement können mehrere Datensätze angelegt werden. Dazu werden die einzelnen Datensätze mit Kommata voneinander getrennt. Im Folgenden werden die beiden kennengelernten SQL Statements an einem Beispiel verdeutlicht. Dazu werden die beiden aufgeführten SQL Statements nacheinander ausgeführt.

```
1. CREATE TABLE Studierende (Matrikelnummer integer, Nachname String)
2. INSERT INTO Studierende VALUES (167860, Schneider),(216000,
   Peters), (167860, Schneider)
```

Die mit diesen SQL Statements erzeugte Tabelle kann mit dem SELECT Statement ausgegeben werden. Ein SELECT Statement ist im wie folgt aufgebaut:

```
SELECT Attribut1, Attribut2 FROM Tabellenname
```

Nach dem Schlüsselwort SELECT sind die Attribute anzugeben, welche bei der Ausgabe berücksichtigt werden sollen. Die Attribute werden durch Kommata voneinander getrennt. Sollen sämtliche Attribute einer Tabelle ausgegeben werden, so kann dies verkürzt durch die Verwendung von „*“ umgesetzt werden.
Der Inhalt der Beispieltabelle kann somit durch die Ausführung des folgenden Statements ausgegeben werden:

```
SELECT * FROM Studierende
```

Aus diesem Statement resultiert die in der Tabelle 2.2.3.1 dargestellte Ausgabe.

**Tabelle 2.2.3.1** – Beispiel: CREATE TABLE und INSERT INTO Statements

| Matrikelnummer | Nachname |
|---|---|
| 167860 | Schneider |
| 216000 | Peters |
| 167860 | Schneider |

In dem Beispiel ist der Datensatz (167860, Schneider) doppelt vorhanden. Um bei der Ausgabe lediglich unterschiedliche Datensätze zu berücksichtigen, wird das Schlüsselwort DISTINCT nach dem Schlüsselwort SELECT angefügt. Somit resultiert das Statement

```
SELECT DISTINCT * FROM Studierende
```

in der folgenden Ausgabe, die in der Tabelle 2.2.3.2 abgebildet ist.

**Tabelle 2.2.3.2** – Beispiel: Einsatz von DISTINCT

| Matrikelnummer | Nachname |
|---|---|
| 167860 | Schneider |
| 216000 | Peters |

In dem Beispiel ist es sinnvoll die Studierenden eindeutig voneinander zu unterscheiden. Dazu ist es erforderlich ein Attribut als Primärschlüssel-Attribut zu definieren. In dem Beispiel ist die Matrikelnummer ein sinnvolles Primärschlüssel-Attribut. Nach der Festlegung des Primärschlüssel-Attributs ist es nicht mehr möglich mehrere Datensätze mit identischer Matrikelnummer der Tabelle hinzuzufügen.
Die Ausgabe über das SELECT Statement kann auf Datensätze konkretisiert werden, welche eine oder mehrere definierte Bedingungen erfüllen. Dazu wird das SELECT Statement um eine WHERE-Klausel erweitert. Im Statement wird nach der WHERE-Klausel die Bedingung formuliert. Bedingungen können dabei mithilfe der booleschen Operatoren AND, OR, NOT und XOR miteinander verknüpft werden. Welche Bedingung zulässig ist, ist von der Domäne des Attributs abhängig, welches in die Bedingung miteinbezogen wird. Handelt es sich bei der Domäne um eine Zahl, so können die Vergleichsoperatoren <, <=, >, >=, = und <> verwendet werden. Handelt es sich bei der

Domäne um ein Zeichen oder um Zeichenketten, so können die Operatoren =, LIKE oder NOT LIKE verwendet werden. Der Operator LIKE beschreibt dabei keine absolute Übereinstimmung, sondern ermöglicht die Benutzung von Platzhaltern. Darüber hinaus muss die, in der Bedingung gewählte Zeichenkette, zwischen zwei Apostrophen angegeben werden.
Für die folgenden Beispiele wird angenommen, dass lediglich eindeutige Datensätze in der Tabelle 2.2.3.1 existieren. So kann mit einem Statement ausgewertet werden, welche Studierenden eine Matrikelnummer besitzen, die größer ist als 200000. Dazu wird das folgende Statement formuliert:

```
SELECT * FROM Studierende WHERE Matrikelnummer > 200000
```

Die aus dem Statement resultierende Ausgabe ist in der folgenden Tabelle 2.2.3.3 abgebildet.

**Tabelle 2.2.3.3** – Beispiel: Einsatz der WHERE-Klausel für Zahlen

| Matrikelnummer | Nachname |
|---|---|
| 216000 | Peters |

Alternativ kann auch eine Bedingung nach dem Attribut „Nachname“ gebildet werden. Für dieses Beispiel wird ausgewertet, welche Studierenden einen Nachnamen besitzen, welcher mit dem Buchstaben „S“ beginnt. Das Statement folgende Statement generiert die gesuchte Ausgabe:

```
SELECT * FROM Studierende WHERE Nachname LIKE 'S%'
```

Die Ausgabe dieses SQL Statements ist in der Tabelle 2.2.3.4 dargestellt.

**Tabelle 2.2.3.4** – Beispiel: Einsatz der WHERE-Klausel für Zeichenketten

| Matrikelnummer | Nachname |
|---|---|
| 167860 | Schneider |

Die Sortierung von Ausgaben mit mehreren Datensätzen kann durch die ORDER BY Klausel festgelegt werden. Für eine absteigende Sortierung wird das Schlüsselwort DESC verwendet. Für eine aufsteigende Sortierung wird hingegen das Schlüsselwort ASC genutzt. Der allgemeine Aufbau eines SQL Statements, bei dem die Ausgabe sortiert wird, ist in dem folgenden SQL Statement beispielhaft dargestellt:

```
SELECT Attribut FROM Tabellenname ORDER BY Attribut DESC
```

Weitere Auswertungsmöglichkeiten mithilfe von SQL werden durch die GROUP BY Klausel geboten. Diese Klausel ermöglicht es Datensätze zu Gruppieren. Anschließend können die in der Tabelle 2.2.3.5 abgebildeten Aggregatsfunktionen auf die Gruppen angewendet werden.

**Tabelle 2.2.3.5** – Aggregatsfunktionen

| Aggregatsfunktion | Aufgabe |
|---|---|
| COUNT | Gibt die Anzahl nichtleerer Datensätze der übergebenen Spalten an |
| SUM | Addiert alle Werte einer numerischen Spalte |
| MIN | Gibt den kleinsten Wert einer numerischen Spalte aus |
| MAX | Gibt den größten Wert einer numerischen Spalte aus |
| AVG | Berechnet das arithmetische Mittel einer numerischen Spalte |

Die Gruppierung und Anwendung einer Aggregatsfunktion wird im Allgemeinen wie folgt in SQL Statements benutzt:

```
SELECT Attribut, COUNT(*) FROM Tabellenname GROUP BY Attribut
```

Darüber hinaus ist es möglich tabellenübergreifende Auswertungen mit SQL Statements durchzuführen. Dazu wird das Schlüsselwort JOIN verwendet. Bei einem JOIN Befehl handelt es sich dabei um einen Verbund von Datensätzen aus unterschiedlichen Tabellen. JOINS werden in drei Haupttypen unterteilt. Darunter zählen der INNER JOIN, der LEFT JOIN und der RIGHT JOIN. Auf den INNER JOIN wird im Folgenden näher eingegangen. Durch den INNER JOIN wird die Schnittmenge der unterschiedlichen Tabellen ausgegeben. Wichtig dabei ist, dass in den Tabellen Datensätze enthalten sind, welche mindestens ein gemeinsames Attribut nutzen. SQL Statements, welche das JOIN Schlüsselwort beinhalten, haben die folgende Syntax:

```
SELECT Attribut FROM Tabellenname1
INNER JOIN Tabellenname2
ON Gleiches_Attribut
```

Zur Verdeutlichung des INNER JOINs werden die Tabellen 2.2.3.6 und 2.2.3.7 betrachtet.

**Tabelle 2.2.3.6** – Beispiel INNER JOIN | Tabelle Studierende

| Studierende | |
|---|---|
| **Matrikelnummer** | **Nachname** |
| 167860 | Schneider |
| 216000 | Peters |

**Tabelle 2.2.3.7** – Beispiel INNER JOIN | Tabelle Noten

| Noten | |
|---|---|
| **Matrikelnummer** | **Note** |
| 167860 | 1.3 |
| 216000 | 3.0 |

Um sowohl den Nachnamen, als auch die Note der Studierenden in einer Ausgabe zu erhalten, wird der INNER JOIN wie folgt verwendet:

```
SELECT Studierende.Nachname, Noten.Note
FROM Studierende
INNER JOIN Noten
ON Studierende.Matrikelnummer = Noten.Matrikelnummer
```

Die Attribute der einzelnen Tabellen sind über den Tabellennamen wie dargestellt zugänglich. Das gemeinsame Attribut ist in diesem Beispiel die Matrikelnummer. Dadurch ist es möglich die Datensätze der unterschiedlichen Tabellen miteinander zu verknüpfen. Aus dem aufgeführten SQL Statement resultiert die Ausgabe aus der Tabelle 2.2.3.8.

**Tabelle 2.2.3.8** – Beispiel INNER JOIN | Ausgabe

| Nachname | Note |
|---|---|
| Schneider | 1.3 |
| Peters | 3.0 |

Abschließend werden neben dem CREATE TABLE und dem INSERT INTO Statement drei weitere Schlüsselwörter vorgestellt, welche zur Manipulation der Daten genutzt werden können. Dabei handelt es sich um die Schlüsselwörter UPDATE, DELETE und DROP TABLE. Durch die Nutzung des UPDATE Statements können bestehende Datensätze gezielt geändert werden. Dieses Statement hat die folgende Form:

```
UPDATE Tabellenname SET Attribut = Wert WHERE [Bedingung]
```

Nach dem Schlüsselwort UPDATE wird der Tabellenname aufgeführt, in welchem sich der zu ändernde Datensatz befindet. Mit dem Schlüsselwort SET wird festgelegt, welche Änderung an dem genannten Attribut durchgeführt werden soll. Um die durchzuführende Änderung zu konkretisieren, wird das Statement um eine WHERE Klausel erweitert.
Das Löschen von Datensätzen kann durch die Verwendung eines DELETE Statements veranlasst werden. DELETE Statements haben die folgende Syntax:

```
DELETE FROM Tabellenname WHERE [Bedingung]
```

Nach den Schlüsselwörtern DELETE FROM, wird der Tabellenname der Tabelle aufgeführt, aus welcher Datensätze gelöscht werden sollen. Anschließend werden die zu löschenden Datensätze mithilfe einer WHERE Klausel eingegrenzt. Um nicht nur den Inhalt einer Tabelle, sondern die Tabelle als Relation der Datenbank zu löschen, wird der folgende Befehl verwendet: DROP TABLE Tabellenname.

## 2.3 JavaScript zur Entwicklung serverseitiger Anwendungen

JavaScript ist eine Skriptsprache, welche im Jahr 1995 von dem Unternehmen Netscape unter dem Namen LiveScript entwickelt und veröffentlicht worden ist. Um die Popularität der Programmiersprache Java zu nutzen, ist der Produktname von LifeScript zu JavaScript geändert worden (Avci, 2003). Die Skriptsprache JavaScript unterscheidet sich dabei stark von der Programmiersprache Java. Bei Skriptsprachen handelt es sich um Programmiersprachen, welche mithilfe eines Interpreters ausgeführt werden. Programmiersprachen werden hingegen mithilfe eines Compilers ausgeführt. Interpreter und Compiler sind Programme, welche einen Quellcode in Maschinencode übersetzten, um diesen anschließend auszuführen. Der Unterschied zwischen einem Interpreter und einem Compiler ist, dass ein Interpreter den Quellcode zeilenweise in Maschinencode übersetzt und ausführt, wogegen ein Compiler zunächst den gesamten Quellcode vor der Ausführung in Maschinencode übersetzt. Interpreter haben den Vorteil, dass die Fehlerlokalisierung durch die zeilenweise Vorgehensweise besser als bei Compilern ist. Der Vorteil von Compilern ist hingegen, dass das ausgeführte Programm effizienter läuft, da der gesamte Quellcode bereits in Maschinencode umgewandelt worden ist (Bewersdorff, 2018). Heutzutage sind sämtliche Webbrowser in der Lage JavaScript zu interpretieren. Die Motivation für die Entwicklung von JavaScript ist die Ermöglichung von interaktiven Funktionalitäten auf Webseiten. So ist es mit JavaScript möglich HTML-Elemente von Webseiten zu manipulieren, da es sich im Gegensatz zu HTML um keine Auszeichnungssprache handelt. Durch HTML-Elemente wird die statische Erscheinung einer Webseite definiert, wogegen der Einsatz von JavaScript ein dynamisches Verhalten einer Webseite ermöglicht. Um ein Skript in einem HTML-Dokument zu nutzen, wird der Programmabschnitt zwischen zwei <script>-Tags aufgeführt. Die Nutzung von JavaScript ist ursprünglich für die clientseitige Ausführung im Webbrowser vorgesehen worden, wobei der serverseitige Einsatz von JavaScript zur Entwicklung des Webservers und somit des Backends im letzten Jahrzehnt immer mehr an Bedeutung gewonnen hat (Steyer, 2020). Dem liegt zu Grunde, dass es durch den Einsatz von JavaScript im Backend und Frontend möglich ist, eine komplexe Webseite mit einer Client-Server-Architektur mit nur einer Skriptsprache zu entwickeln. Ein Produkt, welches dazu verwendet werden kann, ist Node.js.

Node.js ist eine plattformübergreifende Open-Source-JavaScript-Laufzeitumgebung und ist von der Node.js Foundation im Jahr 2010 veröffentlicht worden. Dabei ist Node.js als asynchrone und ereignisbasierte Laufzeitumgebung insbesondere für die Entwicklung von skalierbaren Netzwerkanwendungen entworfen worden. Node.js basiert auf der V8 JavaScript-Engine. Die V8-JavaScript-Engine ist die Kernkomponente des aktuell meistverwendeten Webbrowsers Google Chrome. Node.js nutzt die V8-JavaScript-Engine außerhalb von Webbrowsern und ist dadurch ein Leistungsstarkes Werkzeug zur Entwicklung serverseitiger Anwendungen. Die Nutzung von JavaScript für serverseitige Anwendungen bietet den Vorteil, dass durch die ereignisorientierte Architektur von

JavaScript weniger Arbeitsspeicher pro Verbindungsaufbau benötigt wird als es bei ähnlichen Anwendungen der Fall ist. Eine weitere Besonderheit ist dass sämtliche Verbindungseingänge und Verbindungsausgänge über nur einen Prozess verarbeitet werden. Im Standard nutzen ähnliche Anwendungen einen Prozess pro Verbindungsaufbau. Durch diese Handhabung von Verbindungen erweist sich Node.js als besonders perfomantes Produkt. Durch die einfache Struktur von Node.js hat es sich zu einem beliebten Produkt der Community im JavaScript-Umfeld entwickelt. So stellen Entwickler aus der ganzen Welt selbstprogrammierte Bibliotheken kostenfrei zur Verfügung. Diese Bibliotheken werden als Module bezeichnet. Aktuell existieren mehr als eine Millionen frei zugängliche Module, wodurch die Anwendungsentwicklung immens vereinfacht wird, da zu gängigen Aufgabenstellungen bereits Module bestehen, welche den Arbeitsaufwand der Entwicklung reduzieren. Die Installation der Module erfolgt mithilfe des Modulmanagementsystems namens „Node Packaged Modules“. Dieses ist in der Installation von Node.js integriert. Um ein Modul zu installieren, ist es lediglich erforderlich einen Befehl der folgenden Struktur auszuführen:

```
npm install <modulname>
```

Anschließend werden sämtliche Softwarepakete, die für die Verwendung des Moduls erforderlich sind, automatisch installiert. Für die Entwicklung von Webapplikationen sind die Module Express und Socket.io häufig Bestandteile eines Backends. Mithilfe des Moduls Express, ist die Erstellung eines Webservers mit wenigen Zeilen Code möglich:

```
1  var app = require('express')();
2  var http = require('http').createServer(app);
3
4  app.get('/', (req, res) => {
5  res.sendFile(__dirname + '/index.html');
6  });
7
8  http.listen(3000, () => {
9   console.log('listening on Port:3000');
10 });
```

(Socket.IO, 2020)

Wie in dem Codeabschnitt gezeigt wird, werden mit dem Express-Modul die Variablen app und http initialisiert. Die Variable app ist dabei eine Instanz des Express-Moduls. Die Variable http ist eine Instanz eines Servers, welches app als Argument besitzt. In den Zeilen vier bis sechs wird angegeben, dass bei einer GET-Anfrage an die IP-Adresse des Servers unter einem Port das Dokument index.html aufgerufen werden soll. Die GET-Anfrage ist eine HTTP-Standardmethode, welche in dem Abschnitt 2.4.2 thematisiert wird. In der achten Zeile wird festgelegt, dass der Webserver auf Verbindungseingänge auf dem Port 3000 hört. In den Zeilen neun und zehn wird bei der

Ausführung des Codes die Meldung protokolliert, dass der Webserver aktiv ist und auf den Port 3000 hört. Um den Webserver für eine Echtzeitkommunikation zum Aufbau von Netzwerkanwendungen zu erweitert, wird das Modul Socket.io benutzt. Mithilfe des Moduls Socket.io wird die Nutzung von WebSockets ermöglicht. Auf die Thematik der WebSockets wird in dem Abschnitt 2.4.3 näher eingegangen. Ein weiteres Node.js-Modul, welches für die Anbindung des Webservers an PostgreSQL entwickelt worden ist, ist node-postgres. Dieses Modul ermöglicht die Kommunikation einer Node.js-Anwendung zu PostgreSQL mithilfe der in PostgreSQL integrierten Socket-Schnittstelle libpq. Sämtliche Informationen einer Anwendung, welche mit Node.js entwickelt wird, werden in der Datei package.json gespeichert. Diese Datei beisitzt das Format JSON. Die Abkürzung JSON steht für JavaScript Object Notation. Das JSON Format findet häufige Anwendung beim Informationsaustausch zwischen Webclients und Webservern. JSON ist ein textbasiertes und für Menschen leserliches Format, welches einen geringen Speicherbedarf benötigt. Insbesondere wird dieses Dateiformat für Webapplikationen oder für mobile Anwendungen in Verbindung mit JavaScript benutzt. Dabei ist es unabhängig davon, welche Programmier- oder Skriptsprache verwendet wird, da es sich bei JSON um ein Dateiformat und nicht um eine Programmier- oder Skriptsprache handelt (Sriparasa, 2013). JSON-Dateien sind im Allgemeinen wie in dem folgenden Beispiel dargestellt aufgebaut:

```
1  [
2     {
3      matrikelnummer: 123456,
4      nachname: "Klaus",
5      vorname: "Felix",
6      studiengang: "Maschinenbau",
7      studienbeginn: 2015,
8     universität: "TU München"
9     },
10    {
11     matrikelnummer: 145794,
12     nachname: "Said",
13     vorname: "Abdel",
14     studiengang: "Logistik",
15     studienbeginn: 2017,
16     universität: "TU Dortmund"
17    },
18  ]
```

In dem aufgeführten Beispiel sind Informationen zu zwei Studenten in dem JSON-Format als Array mit zwei Elementen dargestellt. Dabei sind die Studenten Elemente, dessen Informationen in geschwungenen Klammern enthalten sind. Die Informationen zu einem Element werden durch Schlüsselfelder definiert, wobei den Schlüsselfeldern Werte zugewiesen werden. So ist beispielsweise in der dritten Zeile der Matrikelnummer des ersten Studenten der Wert 123456 zugewiesen. Im Fall, dass ein Element mehrere Schlüsselfelder besitzt, werden diese durch Kommata voneinander getrennt.

## 2.4 Programmschnittstellen

Um die Kommunikation zwischen den Schichten einer Softwarearchitektur oder anderen Softwareprodukten zu ermöglichen ist die Definition von Programmschnittstellen erforderlich. Diese Schnittstellen werden als Application Programming Interface (API) bezeichnet. Eine API ist dabei ein Programmteil, welcher anderen Programmen zur Anbindung zur Verfügung gestellt wird. Um eine API umzusetzen ist grundlegendes Wissen über die Vernetzung von Computersystemen erforderlich. So wird im Folgenden das TCP/IP-Netzwerkmodell vorgestellt. Des Weiteren wird insbesondere auf die Netzwerkprotokolle HTTP und WebSocket eingegangen.

### *2.4.1 Das TCP/IP –Netzwerkmodell*

Bis zu den 1970er Jahren ist keine herstellerübergreifende Vereinbarung zur Vernetzung von Computern vorhanden gewesen. Jeder Hersteller hat eigene Lösungen konzipiert, um die eigenen Computer miteinander zu vernetzen. Dies ist zunächst als Marketingstrategie genutzt worden, um Kunden dazu zu animieren sämtliche Computer von einem Hersteller zu beziehen. Zum Ende der 1960er Jahre haben die Hersteller jedoch erkannt, dass die Notwendigkeit besteht, die Vernetzung von Computern unabhängig vom Hersteller des Computers zu gestalten. Im Laufe der 1970er Jahre hat die International Organization for Standardization (ISO) die Aufgabe übernommen die Open-System-Interconnection zu entwickeln. Im deutschen Sprachraum ist diese als das OSI-Netzwerkmodell bekannt. Zur selben Zeit hat das US-Verteidigungsministerium die Entwicklung eines zweiten, standardisierten Netzwerkmodells veranlasst. Dieses Netzwerkmodell trägt den Namen TCP/IP. Dieses Netzwerkmodell hat sich neben dem OSI-Netzwerkmodell und weiteren Modellen durchgesetzt und gilt auch in der heutigen Zeit als das Standard-Netzwerkmodell (Jarzyna, 2013).
Durch TCP/IP ist eine Vielzahl von Protokollen definiert, welche die Kommunikation zwischen Computern ermöglichen. Der Name des Netzwerkmodells basiert dabei auf den zwei wichtigsten Protokollen. Dabei handelt es sich zum einen um das Transmission Control Protokol (TCP) und um das Internet Protokol (IP). Die durch TCP/IP definierten Protokolle werden in unterschiedliche Schichten aufgeteilt. Aus dieser Aufteilung resultiert das TCP/IP-Schichtenmodell. Da die Entwicklung von TCP/IP vom US-Verteidigungsministerium veranlasst worden ist, ist das TCP/IP-Schichtenmodell auch unter dem Namen Department of Defence-Model (DoD-Modell) bekannt (Lienemann & Larisch, 2014). In der folgenden Tabelle 2.4.1 sind die einzelnen Schichten des TCP/IP-Schichtenmodells aufgeführt.

**Tabelle 2.4.1** – TCP/IP-Schichtenmodell

| TCP/IP-Schicht | Beispielprotokolle |
|---|---|
| Anwendungsschicht | HTTP, SOCKS, SMTP, FTP, WebSockets |
| Transportschicht | TCP, UDP |
| Internetschicht | IPv4, IPv6 |
| Netzzugangsschicht | Ethernet, Token-Ring, FDDI |

Die Anwendungsschicht umfasst die Dienste, welche von Anwendungen verwendet werden. So fungiert die Anwendungsschicht als Schnittstelle zwischen einer Anwendung auf einem Computer und dem Netzwerk. Bei den meistgenutzten TCP/IP-Anwendungen handelt es sich um Webbrowser (Loshin, 2003). Um die Funktion der Anwendungsschicht zu verdeutlichen, wird der Prozess des Aufrufens einer Webseite im Folgenden als Beispiel verwendet. Webbrowser nutzen in der Regel das Hypertext Transfer Protocol (HTTP) in der Anwendungsschicht. Damit eine Webseite über einen Webbrowser aufgerufen wird, ist es notwendig einen HTTP-Header an den Webserver zu senden. Der HTTP-Header enthält die Information, was vom Webserver angefragt wird. Bei dem Besuch einer Webseite wird eine HTML-Datei angefragt, welche die Webseite darstellt. Erhält der Webserver die Anfrage, so sendet er dem Webbrowser als Antwort ebenfalls einen HTTP-Header. In diesem HTTP-Header, sendet der Webserver die angefragte HTML-Datei an den Webbrowser und die Webseite wird im Webbrowser dargestellt. Kann der Webserver die angefragte HTML-Datei nicht zur Verfügung stellen, so wird ein Fehlercode an den Webbrowser gesendet (Jarzyna, 2013). Dieses Beispiel zeigt eine der bedeutendsten Grundlagen von Netzwerkmodellen. Kommunizieren zwei Computer auf derselben TCP/IP-Schicht miteinander, so werden Header von beiden Computern verwendet, um Informationen an den jeweils anderen Computer zu übertragen. Dieser Prozess trägt den Namen same layer interaction. Dabei ist dieses Konzept der Kommunikation unabhängig von dem Anwendungsschichtprotokoll und trifft auf jeder Schicht des TCP/IP-Schichtenmodell zu. Zu der Anwendungsschicht zählt eine Vielzahl von Protokollen (Trick & Weber, 2009). In den Abschnitten 2.4.1 und 2.4.2 wird näher auf die Anwendungsschicht-protokolle HTTP und WebSocket eingegangen.

Die Transportschicht umfasst hingegen lediglich die zwei Protokolle TCP und User Datagram Protocol (UDP) (Jarzyna, 2013). In dem TCP/IP-Schichtenmodell befindet sich die Transportschicht unter der Anwendungsschicht. Die Anordnung der Schichten im TCP/IP-Schichtenmodell definiert, welche Schicht einer anderen Schicht einen Dienst zur Verfügung stellt. So dient jede Schicht der darüber aufgeführten Schicht. Über der Transportschicht befindet sich die Anwendungsschicht, was bedeutet, dass die Transportschicht der Anwendungsschicht einen Dienst erweist. Die Anwendungsschicht beauftragt die Transportschicht damit, die Anfrage vom Webbrowser zum Webserver zu transportieren. Dabei ist es in den meisten Fällen wichtig sicherzustellen, dass kein Informationsverlust während der Übertragung auftritt. Aus diesem Grund besitzt das Transportprotokoll TCP eine Fehlerbehebungsfunktion, welche Acknowledgments

(deutsch: Bestätigungsnummern) nutzt. In der folgenden Abbildung 2.4.1 ist ein Datenaustauschprozess für den Aufruf einer Webseite schematisch dargestellt, wobei lediglich die Anwendungs- und Transportschicht betrachtet werden.

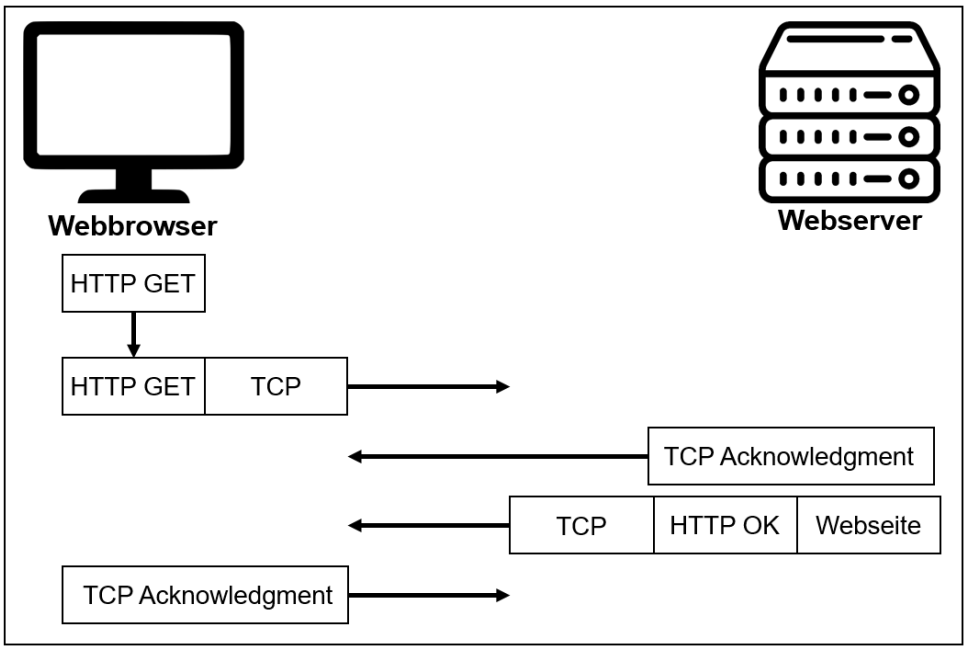


**Abbildung 2.4.1** – Aufruf einer Webseite (Anwendungs- und Transportschicht) (Jarzyna, 2013)

Die linke Seite der Abbildung repräsentiert die Prozesse, die auf dem Webbrowser stattfinden und die rechte Seite repräsentiert die Prozesse, die auf dem Webserver stattfinden. Der Webbrowser möchte eine HTTP-Anfrage zum Aufrufen einer Webseite an den Webserver senden. Dazu beauftragt die Anwendungsschicht die Transportschicht dazu diese Anfrage zu versenden. Um den Empfang der der HTTP-Anfrage zu bestätigen, sendet der Webserver eine Empfangsbestätigung, also ein TCP Acknowledgment an den Webbrowser. Anschließend sendet der Webserver dem Webbrowser einen HTTP-Header, welcher alle notwendigen Informationen zur Darstellung der Webseite beinhaltet. Nach erfolgreichem Empfangen der Informationen, sendet der Webbrowser eine Empfangsbestätigung an den Webserver (Jarzyna, 2013). Weltweit existieren aktuell mehr als 1,7 Milliarden Webseiten (Seeboth, 2020). Damit eine Datenübertragung vom Webbrowser zum Webserver stattfinden kann, muss sichergestellt werden, dass die Daten an den richtigen Webserver gesendet werden. Diese Aufgabe übernimmt die Internetschicht. Diese ist für die logische Adressierung der physischen Netzwerkschnittstelle verantwortlich. Dazu wird das zu sendende Datenpaket um einen IP-Header erweitert. Der IP-Header enthält die IP-Adresse des Webservers, an den die Anfrage übertragen werden soll, wobei die Anfrage als Datenpaket übertragen wird. Eine IP-Adresse ist eine eindeutige Kennung einer Netzwerkkomponente innerhalb eines Netzwerks. Zwischen den einzelnen Netzwerkkomponenten befinden sich Netzwerkrouter, welche Datenpakete zwischen mehreren Netzwerken weiterleiten können. Der Prozess der Weiterleitung von

Datenpaketen wird als IP-Routing bezeichnet (Jarzyna, 2013). Um das IP-Routing zu verdeutlichen, wird die folgende Abbildung 2.4.2 betrachtet.

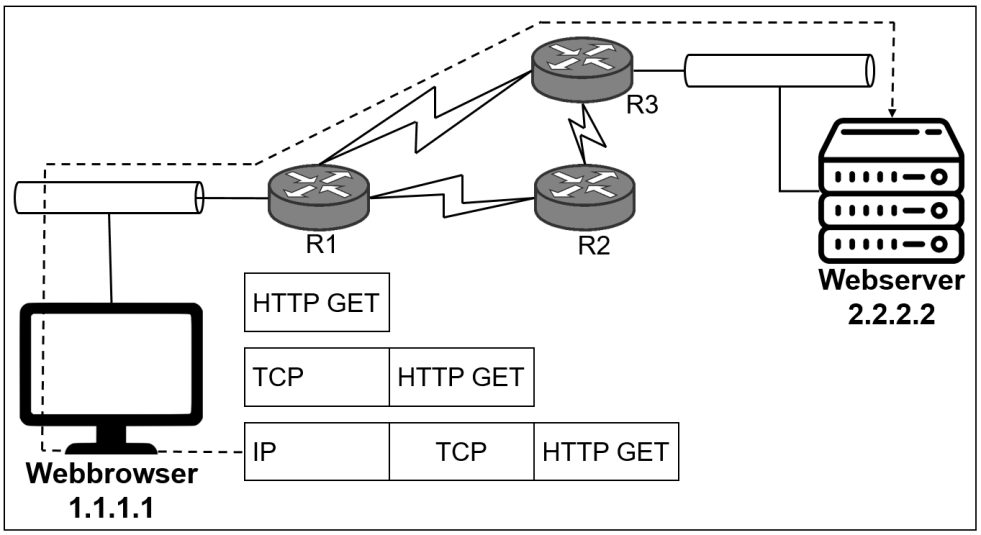


**Abbildung 2.4.2** – IP-Routing (Jarzyna, 2013)

Die Abbildung 2.4.2 bezieht sich erneut auf das Aufrufen einer Webseite in einem Webbrowser. Erneut beauftragt die Anwendungsschicht die Transportschicht mit der Versendung einer HTTP-Anfrage. Daraufhin beauftragt die Transportschicht die Internetschicht dazu die Anfrage um einen IP-Header zu erweitern, um die Kommunikation mit dem Webserver aufzunehmen. In diesem Beispiel besitzt der Webserver die IP-Adresse 2.2.2.2. Damit die HTTP-Anfrage zum Webserver gesendet wird, sendet der Computer auf dem der Webbrowser ausgeführt wird, die Anfrage an den eigenen Netzwerkrouter R1. Der Netzwerkrouter R1 trifft anschließend die Entscheidung das Datenpaket an den Netzwerkrouter R3 weiterzuleiten. Dieser leitet das Datenpaket anschließend an den Webserver weiter. Im Fall eines Verbindungsausfalls zwischen den Netzwerkroutern R1 und R3, kann das Datenpaket alternativ über den Netzwerkrouter R3 an R2 weitergeleitet werden (Jarzyna, 2013).
Um die physische Übertragung des Datenpakets durchzuführen, wird die Netzzugangsschicht von der Internetschicht beauftragt. Die Internetschicht besitzt Informationen zu der Netzwerktopologie und kennt sämtliche Netzwerkrouter und Computer, die physisch miteinander verbunden sind. Die Einzelheiten des physischen Netzwerks sind der Internetschicht jedoch unbekannt. Die Netzzugangsschicht legt die erforderliche Hardware und Protokolle für die Datenübertragung über ein physisches Netzwerk fest. Das in der Netzzugangsschicht am häufigsten verwendete Protokoll ist das Ethernet. Das Ethernet umfasst die Informationen zu der Verkabelung, der Adressierung und Protokolle für ein Ethernet-Local-Area-Network, welche zur physischen Übertragung von Datenpaketen erforderlich sind. Des Weiteren werden weitere Protokolle auf der Netzzugangsschicht verwendet, welche Stecker, Kabel,

Stromstärken und sonstige physische Übertragungsmedien definieren (Lienemann & Larisch, 2014).

### *2.4.2 Hypertext Transfer Protocol & WebSocket-Protocol*

Wie bereits diskutiert handelt es sich bei dem HTTP um ein Protokoll der Anwendungsschicht des TCP/IP-Schichtenmodells (Jarzyna, 2013). Es ist im Jahr 1996 unter der Version 1.0 veröffentlicht worden. So ist die Kommunikation zwischen Webclients und Webservern ermöglicht worden. Wie in dem im vorherigen Abschnitt aufgeführten Beispiel verdeutlicht worden ist, basiert diese Kommunikation auf einem Anfrage-Antwort-Konzept. In der Version 1.0 ist es erforderlich gewesen für jede Anfrage einen gesonderten Verbindungsaufbau vom Webclient zum Webserver durchzuführen. So ist es zum Beispiel für das Laden einer Webseite mit fünf Elementen notwendig gewesen sechs Verbindungen aufzubauen. Es ist schnell erkannt worden, dass diese Vorgehensweise ineffizient ist. Aus diesem Grund ist im Jahr 1999 die HTTP-Version 1.1 veröffentlicht worden. In dieser Version ist es dem Webclient ermöglicht worden einen zusätzlichen Header an den Webserver zu senden, welcher den Befehl gibt, die aufgebaute Verbindung aufrecht zu erhalten und nicht zu schließen. Dieser Befehl wird als Keepalive-Befehl bezeichnet. So ist für das genannte Beispiel der Webseite mit fünf Elementen nur noch ein Verbindungsaufbau erforderlich. Dabei ist das Anfrage-Antwort-Konzept auch in der Version 1.1 aufrechterhalten worden. Für das Aufrufen einer Webseite ist es erforderlich gewesen eine Verbindung vom Webclient zum Webserver aufzubauen und für jedes Element eine Anfrage zu senden. Dabei ist es stets die Aufgabe des Webclients gewesen Anfragen an den Webserver zu senden. Im Jahr 2015 ist die heute aktuelle HTTP-Version 2 veröffentlich worden. In dieser Version ist es möglich mehrere Anfragen zu einer Anfrage zusammenzufassen. Darüber hinaus ist nun auch eine Datenübertragung möglich, welche vom Webserver initiiert wird (Pollard, 2019). HTTP nutzt weiterhin das TCP als Transportprotokoll für die Übertragung der Anfragen. Die gängigsten HTTP-Anfragen sind GET, POST, PUT, PATCH und DELETE. Mit einer GET-Anfrage fordert der Webclient Daten vom Webserver an ohne eine Manipulation von Daten auszulösen. Die POST-Anfrage bietet die Möglichkeit Daten vom Webclient an den Webserver zu übertragen. Hierdurch ist es möglich, dass neue Ressourcen auf dem Webserver angelegt oder bestehende Ressourcen auf dem Webserver verändert werden. Um eine Datei vom Webclient an den Webserver zu übertragen, kann eine PUT-Anfrage gesendet werden. Falls eine Datei aktualisiert werden soll ohne diese vollständig zu ersetzen, kann dies mithilfe einer PATCH-Anfrage realisiert werden. Mit einer DELETE-Anfrage kann der Webclient das Löschen von Daten am Webserver veranlassen (Shiflett, 2003). Bei dem HTTP handelt es sich um ein zustandsloses Protokoll. Das bedeutet, dass eine HTTP-Anfrage keine Informationen über vorherige HTTP-Anfragen besitzt. Somit sind die einzelnen Anfragen unabhängig voneinander. Die Zustandslosigkeit des HTTP ermöglicht eine uneingeschränkte

Skalierbarkeit, da die HTTP-Anfragen nicht miteinander verknüpft sind. Außerdem ist es dadurch möglich eine Lastenverteilung umzusetzen, da die unterschiedlichen Anfragen von unterschiedlichen, zusammenhängenden Servern beantwortet werden können. Der Nachteil der Zustandslosigkeit des HTTP ist, dass es nicht möglich ist Sitzungen aufrechtzuhalten, da die einzelnen Anfragen keine Informationen über die übrigen Anfragen verfügen. Das Protokoll HTTP verfügt über einen Kompressionsmechanismus, welcher es dem Webserver ermöglicht die Datenmenge in den jeweiligen Antworten an den Webclient zu reduzieren (Wong, 2000). Dabei wird dieser Kompressions-mechanismus hauptsächlich für die Komprimierung textbasierter Daten genutzt. Ein Beispiel dafür ist das in dem Abschnitt 2.3 kennengelernte Dateiformat JSON.
Um die Kommunikation zwischen Webclients und Webservern zu vereinheitlichen, ist es erforderlich Leitsätze für die Kommunikation auf der Anwendungsschicht festzulegen. Architekturstile, welche solche Leitsätze definieren sind unter anderem REST, das Simple Object Acess Protocol (SOAP) und die Web Services Description Language (WSDL) (Jin et al., 2018). Im Folgenden wird näher auf REST eingegangen. REST wird häufig in Verbindung mit dem Protokoll HTTP genannt. Dies liegt daran, dass die Umsetzung von REST in den meisten Fällen mit dem HTTP erfolgt. In der Kombination von REST und HTTP ist von dem RESTful HTTP die Rede. Die in REST definierten Leitsätze können jedoch auch mit einer Vielzahl an weiteren Protokollen verwendet werden. Die RESTful-Grundprinzipien umfassen die folgenden fünf Aspekte (Spichale, 2019):

1) Eindeutige Identifikation von Ressourcen
2) Nutzung von Hypermedia
3) Nutzung von HTTP-Standardmethoden
4) Unterschiedliche Repräsentationen von Ressourcen
5) Zustandslose Kommunikation

Das erste RESTful-Grundprinzip gibt vor, dass sämtliche Ressourcen eindeutig identifiziert werden können müssen. Die eindeutige Identifikation von Ressourcen wird dabei mithilfe von eindeutigen Schlüsseln realisiert. Für Webapplikationen werden dafür die Uniform Resource Identifiers (URIs) verwendet. Diese besitzen die folgende Form:

http://beispiel.de/antworten/1

Mithilfe dieser URI ist es möglich eine Ressource eindeutig zu identifizieren. So wird mit der beispielhaften URI eine Antwort mit der ID 1 identifiziert. Der Inhalt der Antwort ist somit eine Ressource, auf die zugegriffen werden kann. Das zweite RESTful-Grundprinzip gibt die Nutzung von Hypermedia vor. Bei Hypermedia handelt es sich um eine Kombination von Hypertext und Multimedia. Der Präfix „Hyper“ impliziert die Verknüpfung mit weiteren Objekten. Bei den Objekten kann es sich dabei beispielsweise um Texte, Dokumente, Bilder und weitere multimediale Inhalte handeln. HTML ist im

Bereich von Webapplikationen ein typisches Hypermediaformat. So stellt der Webserver die möglichen Aktionen und Navigationspfade über den Webclient in Form von Hypermedia zur Verfügung. Das vierte RESTful-Grundprinzip impliziert die Nutzung von HTTP-Standardmethoden. Darunter zählen die bereits beschriebenen Methoden GET, POST, PUT, PATCH und DELETE. So werden clientseitig Funktionen programmiert, welche die HTTP-Methoden benutzen. In der folgenden Tabelle 2.4.1.1 sind die Funktionen und die dazugehörigen RESTful HTTP-Anfragen abgebildet, die beispielsweise für eine Benutzerverwaltung implementiert werden können. Die Ressourcen sind dabei unterschiedliche Benutzer.

**Tabelle 2.4.1.1** – Beispiel zur Verwendung der RESTful-Schnittstelle (Spichale, 2019)

| **Clientseitige Funktion** | **RESTful HTTP** |
|---|---|
| BenutzerAufrufen() | GET /benutzer |
| BenutzerAktualisieren() | PUT /benutzer{id} |
| BenutzerHinzufügen() | POST /benutzer |
| BenutzerLöschen() | DELETE /benutzer/{id} |

In der Tabelle 2.4.1.1 sind beispielhaft die Funktionen zum Aufrufen, Aktualisieren, Hinzufügen und Löschen von Benutzern aufgeführt. Diese Funktionen senden die jeweiligen RESTful HTTP Anfragen vom Webclient aus zum Webserver. Auf dem Webserver sind die dazugehörigen Aktionen zu den einzelnen Anfragen implementiert. Dieses Konstrukt bildet somit die Schnittstelle zwischen dem Webclient und dem Webserver. Das vierte RESTful-Grundprinzip umfasst die unterschiedlichen Repräsentationen der Ressourcen. Wie bereits erarbeitet, sendet der Webserver dem Webclient Antworten zu den jeweiligen Anfragen, wobei der Webclient vorgibt in welchem Format die Daten in den Antworten bereitgestellt werden sollen. Dabei kann es sich beispielsweise um Daten im Format JSON handeln. Die Form, in welcher die Daten aufbereitet werden, wird als Repräsentation bezeichnet. So werden die Daten, welche vom Webserver beispielsweise aus einer Datenbank extrahiert werden in einem vorgegebenen Datenformat zur Verfügung gestellt. In der folgenden Abbildung 2.4.1.1 ist das Zusammenspiel von den ersten vier RESTful-Grundprinzipien abgebildet.

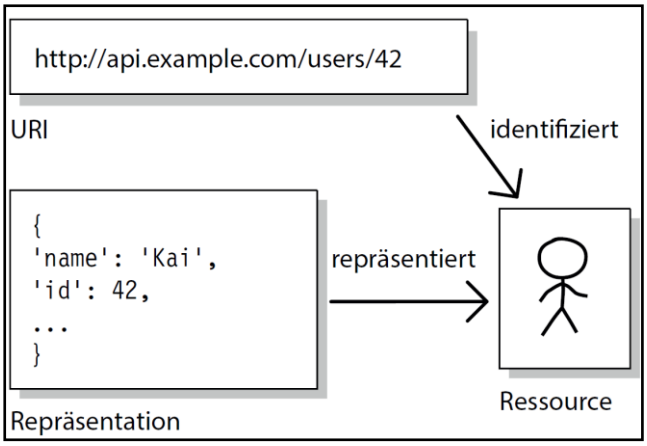


**Abbildung 2.4.1.1** – Beispiel zur Verwendung der RESTful-Schnittstelle (Spichale, 2019)

In der Abbildung 2.4.1.1 wird die Ressource „Benutzer“ über die URI http://api.example.com/users/42 identifiziert. Mit Hilfe einer RESTful HTTP GET-Anfrage werden die Daten von dem Webserver angefragt. Auf diese Anfrage ruft der Webserver die vorhandenen Daten zu diesem Benutzer aus einer Datenbank ab und stellt diese Daten als Repräsentation im JSON-Format zur Verfügung. Diese Daten repräsentieren die Ressource „Benutzer“. Dabei ist es möglich, dass eine Repräsentation die Daten zu mehreren Ressourcen liefert. Somit stehen die Daten dem Webclient zur Verwendung zur Verfügung. Das fünfte und letzte RESTful-Grundprinzip gibt die zustandslose Kommunikation vor, die bereits bei der Beschreibung des HTTP erläutert worden ist.
Für das Aufrufen von Webseiten und für die Nutzung von einfachen Webapplikationen mit geringem Datenaustausch eignet sich das HTTP sehr gut. Das Anfrage-Antwort-Konzept bildet in Kombination mit der Zustandslosigkeit des HTTP den Grundstein für eine klare und skalierbare Kommunikation zwischen einem Webclient und einem Webserver. Für den Einsatz im Bereich von Echtzeitapplikationen wie beispielsweise Chatanwendungen oder Online-Spielen, ist das HTTP nicht gut geeignet. Zum einen enthalten HTTP-Anfragen aufgrund der Zustandslosigkeit, immer alle für die Anfrage erforderlichen Informationen, wodurch die Anfragen ein großes Datenvolumen besitzen. Das verwendete Datenvolumen ist zu groß, als dass eine Echtzeitkommunikation mit HTTP umgesetzt werden könnte. Zum anderen erfolgt die Kommunikation zwischen dem Webclient und dem Webserver im halbduplex Betrieb. So können zwar Daten sowohl vom Webclient zum Webserver als auch vom Webserver zum Webclient gesendet werden, jedoch können Daten nicht zum selben Zeitpunkt in beide Richtungen ausgetauscht werden. Diese Übertragungsart von Daten verringert die Kommunikationsgeschwindigkeit stark. Um Webseiten und Webapplikationen, welche die Echtzeitkommunikation zwischen Webclients und Webservern nutzen sollen, zu realisieren, ist es erforderlich auf ein alternatives Anwendungsschichtprotokoll zurückzugreifen (Spichale, 2019). Eine Möglichkeit ist dabei die Verwendung des WebSocket-Protokolls (Lombardi, 2015).

Bei dem WebSocket-Protokoll handelt es um ein weiteres Anwendungsschichtprotokoll, welches das TCP in der Transportschicht nutzt. Dieses Protokoll unterstützt wie das HTTP die bidirektionale Kommunikation zwischen Webclients und Webservern. Im Gegensatz zum HTTP erfolgt die Kommunikation jedoch im Duplexbetrieb, welcher auch als Gegenbetrieb bezeichnet wird. Dadurch ist der gleichzeitige Datenaustausch vom Webclient zum Webserver und umgekehrt möglich. Um dies zu realisieren wird eine Verbindung vom Webclient zum Webserver aufgebaut und aufrechterhalten. Das WebSocket-Protokoll ist somit kein zustandsloses Protokoll. Um die Kommunikation zwischen einem Webclient und einem Webserver über das WebSocket-Protokoll umzusetzen, ist es zunächst erforderlich, dass eine HTTP-Anfrage vom Webclient an den Webserver gesendet wird. In dieser Anfrage teilt der Webclient dem Webserver mit, dass das Protokoll von HTTP auf das WebSocket-Protokoll umgestellt werden soll. Diese Anfrage wird als Opening-Handshake oder auch als WebSocket-Handshake bezeichnet. Unterstützt der Webserver das WebSocket-Protokoll, so sendet er dem Webclient die Bestätigung, dass die Kommunikation auf der Anwendungsschicht auf das WebSocket-Protokoll umgestellt worden ist. Somit wird eine Verbindung durch einen WebSocket-Kanal zwischen dem Webclient und dem Webserver aufgebaut. Nun können beide Seiten Daten senden, ohne dass diese von der Gegenseite angefragt werden müssen. Die WebSocket-Verbindung bleibt solange erhalten, bis diese von dem Webclient oder von dem Webserver abgebrochen wird. Die Daten, welche über einen WebSocket-Kanal ausgetauscht werden, enthalten aufgrund der Zustandslosigkeit keine Header. Dies hat zur Folge, dass das Datenvolumen im Vergleich zur Kommunikation mit dem HTTP um ein Vielfaches reduziert wird. Während ein HTTP-Header etwa 700 Byte groß ist, umfassen Nachrichten, welche über das WebSocket-Protokoll versendet werden lediglich 2 bis 6 Bytes. Dadurch wird die Kommunikationsgeschwindigkeit zwischen dem Webclient und Webserver deutlich erhöht. Durch die erhöhte Kommunikationsgeschwindigkeit in Kombination mit dem Duplexbetrieb eignet sich das WebSocket-Protokoll sehr gut für den Einsatz im Bereich von Echtzeitapplikationen (Gorski et al., 2015; Lombardi, 2015).

# 3 Konzeption der Architektur der Webapplikation und Einrichtung und Entwicklung des Backends

In diesem Kapitel wird die Architektur der Webapplikation konzipiert. Es werden sowohl das Grobdesign als auch das Feindesign der Webapplikation erarbeitet. In der Ausarbeitung des Grobdesigns werden die eingesetzten Technologien aufgeführt, die zur Entwicklung der Webapplikation verwendet werden. Außerdem wird die Einrichtung der einzelnen Komponenten der Softwarearchitektur beschrieben, wobei die Programmschnittstellen zwischen den Komponenten ebenfalls berücksichtigt werden. In dieser Arbeit liegt dabei der Fokus auf das Backend der Webapplikation. Das Feindesign der Architektur der Webapplikation setzt sich aus Zustandsdiagrammen zu den einzelnen Softwaremodulen zusammen. So werden die möglichen Zustände, Ereignisse und Aktionen der Webapplikation vorgestellt. Des Weiteren verdeutlichen die Zustandsdiagramme der einzelnen Softwaremodule das Zusammenspiel des Backends mit dem Frontend.

## 3.1 Grobdesign der Architektur der Webapplikation und Einrichtung und Entwicklung des Backends

Die in dem Abschnitt 2.1 vorgestellte 3-Tier-Architekur lässt sich auch für die Webapplikation dieser Arbeit anwenden. In der folgenden Abbildung 3.1.1 wird diese Architektur dargestellt.

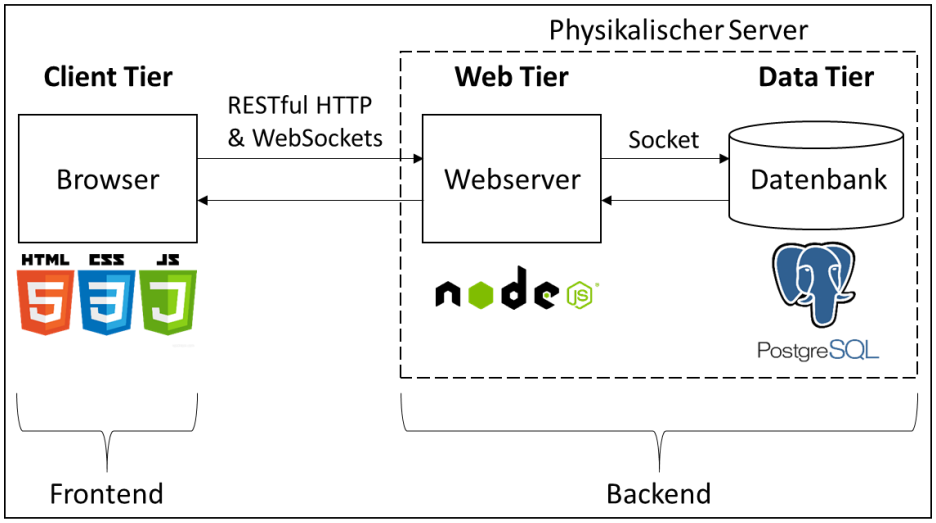


**Abbildung 3.1.1** – 3-Tier-Architektur der Webapplikation

Das Grobdesign der Architektur der Webapplikation ist in drei Schichten aufgeteilt. Dabei handelt es sich um die Client-Schicht, die Web-Schicht und die Datenschicht. Die Client-Schicht bildet die Präsentationsschicht und somit das Frontend der Webapplikation. Die

zur Client-Schicht gehörige Komponente ist der Webbrowser. Über diesen wird auf die Webapplikation zugegriffen. Der Webbrowser bildet die Benutzerschnittstelle der Webapplikation. In dem Webbrowser werden HTML- und CSS-Dateien ausgelesen, um die Webapplikation darzustellen. Außerdem führt der Browser JavaScript-Code aus, um die Interaktion des Anwenders mit der Webapplikation zu ermöglichen. In der Arbeit von O. Alić wird näher auf die Konzeption und Entwicklung des Frontends der Webapplikation eingegangen (Alić, 2021). Das Backend setzt sich aus der Web-Schicht und aus der Datenschicht zusammen. Die Web-Schicht bildet dabei die Anwendungsschicht und die Datenschicht die Persistenzschicht der Webapplikation. Die Komponente der Web-Schicht ist ein Webserver. Dieser dient dazu die Anfragen des Frontends zu bearbeiten und zu beantworten. Aufgrund der relativ geringen Komplexität der Webapplikation, kann auf einen separaten Applikationsserver verzichtet werden. So genügt es für die Webapplikation dieser Arbeit die Geschäftslogik als Programmabschnitt innerhalb der Laufzeitumgebung des Webservers abzubilden. Für die Entwicklung des Webservers wird dabei die Open-Source-JavaScript-Laufzeitumgebung Node.js verwendet. Die Kommunikation vom Webbrowser mit dem Webserver und umgekehrt erfolgt mithilfe von HTTP-Anfragen und WebSockets. Diese beiden TCP/IP-Anwendungsschichtprotokolle nutzen jeweils das TCP in der Transportschicht und IPv4 in der Internetschicht. Für die HTTP-Anfragen werden die standardisierten REST-Leitsätze befolgt. Somit ist eine der zwei Schnittstellen zwischen dem Webbrowser und dem Webserver die RESTful HTTP-Schnittstelle. Um HTTP-Anfragen mithilfe von Node.js bearbeiten zu können, wird das Node.js-Modul Express verwendet. Die HTTP GET- und HTTP POST-Anfragen, die der Webserver bearbeitet, sind in den folgenden Tabellen 3.1.1 und 3.1.2 aufgeführt.

**Tabelle 3.1.1** – Umgesetzte HTTP GET-Anfragen

| **RESTful HTTP** | **Antwort** |
|---|---|
| GET / | index.html |
| GET /aufgaben | JSON: Aufgabenstellungen |
| GET /[AufgabentabellenName] | JSON: Aufgabentabellen |
| GET /UniMailRegistriert | JSON: UniMail registriert oder nicht |
| GET /PasswortZuUniMail | JSON: Korrektes Passwort zur UniMail oder nicht |
| GET /UserID | JSON: UserID des angemeldeten Benutzers |
| GET /MaxUserID | JSON: Wert der höchsten UserID |

**Tabelle 3.1.2** – Umgesetzte HTTP POST-Anfragen

| **RESTful HTTP** | **Inhalt** | **Antwort** |
|---|---|---|
| POST /user | UserID, Vorname, Nachname, UniMail und Passwort des Benutzers | JSON: Registrierung erfolgreich oder nicht |
| POST /ergebnis | SQL Statement | JSON: Ausgabe des SQL Statements oder Fehlermeldung |

Die Tabelle 3.1.1 enthält die am Webserver umgesetzten HTTP GET-Anfragen und die Tabelle 3.1.2 die HTTP POST-Anfragen. Die URIs, mit denen die Anfragen ausgelöst werden können, sind in den Tabellen aufgeführt. Des Weiteren ist zu jeder Anfrage-URI die vom Server bereitgestellte Ressource in der Spalte „Antwort“ dargestellt. Da HTTP POST-Anfragen mit einem Inhalt an den Webserver gesendet werden, ist in der Tabelle 3.1.2 eine Spalte enthalten, welche den Inhalt der jeweiligen Anfrage wiedergibt. Die Webapplikation ist so aufgebaut, dass sie ein mehrbenutzerfähiges System ist. Die Benutzer können sich mit der eigenen UniMail der Technischen Universität Dortmund an der Webapplikation registrieren und anmelden. Nach erfolgreicher Registrierung und Anmeldung können die Benutzer die Übungsaufgaben zum Thema SQL bearbeiten. Die Übungsaufgaben setzen sich aus Aufgabenstellungen zusammen, zu welchen jeweils eine oder zwei Aufgabentabellen zugeordnet sind. Ein Benutzer hat die Möglichkeit sein SQL Statement in ein Eingabefeld einzutragen und das SQL Statement auszuführen. Mithilfe der Anfrage-URI „/“ fragt der Webbrowser die Datei index.html an. Diese Datei enthält sämtliche Informationen und Verknüpfungen zu weiteren Dateien, die für die Darstellung der Webapplikation erforderlich sind. Bei den übrigen serverseitigen Antworten handelt es sich um JSON-Dateien, welche dem Webbrowser unter anderem die Informationen zu den Aufgabenstellungen und den Benutzerinformationen bereitstellen. Neben der RESTful HTTP-Schnittstelle werden auch WebSockets zum Datenaustausch zwischen dem Webbrowser und Webserver verwendet. Um die Verbindung von dem Webbrowser zum Webserver über WebSockets zu realisieren, wird das Node.js-Modul Socket.io eingesetzt. Mit Hilfe dieses Node.js-Moduls sendet der Webbrowser beim Aufrufen der Webapplikation eine HTTP-Anfrage den Befehl das Protokoll von HTTP auf WebSockets zu ändern. Der Webserver ist so programmiert, dass er diese Anfrage versteht und das Protokoll umstellt. Bei einer erfolgreichen Anmeldung eines Benutzers wird eine Verbindung zu einem WebSocket-Kanal geöffnet. Dieser wird dazu genutzt die Echtzeitinformation, welche Benutzer die Webapplikation aktuell nutzen, an den Server zu übertragen. Weshalb diese Information von hoher Bedeutung für die Effizienz der Webapplikation ist, wird in dem Abschnitt 3.2 verdeutlicht. Als Komponente für die Datenschicht wird das Datenbankmanagementsystem PostgreSQL benutzt. Sowohl der Webserver als auch das Datenbankmanagement-system sind auf einem physikalischen Server als Programme installiert. So werden die Latenzen bei der Kommunikation zwischen den beiden Komponenten minimiert. In PostgreSQL wird ein Datenbankserver eingerichtet. Dabei wird die Socket-Schnittstelle libpq konfiguriert. Durch diese Schnittstellenkonfiguration wird die Anbindung von PostgreSQL an den Webserver ermöglicht. Um die Anbindung vom Webserver aus durchzuführen wird das Node.js-Modul node-postgres eingesetzt. In diesem Node.js-Modul werden die IP-Adresse und der Port für die Socket-Schnittstelle hinterlegt. Im nächsten Schritt ist es erforderlich eine Datenbank und einen Benutzer für diese Datenbank auf dem eingerichteten Datenbankserver zu erstellen. Es wird die Datenbank „itpl_sql“ erstellt. Daraufhin wird ein Datenbank-Benutzer erstellt, welcher die vollen

Berechtigungen zur Abfrage und Manipulation der Datenbank besitzt. Dieser Benutzer erhält den Anmeldenamen „postgres“. Der Datenbankname und die Anmelde-informationen des Datenbankbenutzers werden anschließend am Webserver hinterlegt. Somit ist die Anbindung von dem Webserver an die Datenbank erfolgreich durchgeführt. Über diese Anbindung kann der Webserver nun die Daten der Datenbank abfragen und manipulieren. Die Antwort der Datenbank wird ebenfalls über die Socket-Schnittstelle an den Webserver gesendet. Dabei wird die Datenbank so konfiguriert, dass die Antworten im JSON-Format an den Webserver zur weiteren Verarbeitung an den Webserver gesendet werden. Die Datenbank enthält noch keine Daten. Um Daten in die Datenbank einpflegen zu können, wird ein Datenbankschema angelegt. In diesem Datenbankschema können nun Datenbanktabellen angelegt werden. Es werden die folgenden 14 Datenbanktabellen angelegt:

1) Benutzer
2) Aufgaben
3) Ergebnisse
4) Klausurergebnisse
5) Mechanikergebnisse
6) Studierende
7) Studierendeninformation
8) Studierendeninformationen
9) Studierendenliste
10) Studierende2a
11) Studierende2b
12) Studierende2c
13) Studierende3b
14) Studierendenliste4c

Die Datenbanktabelle „Benutzer“ enthält die Informationen, die für das Benutzermanagement der Webapplikation erforderlich sind. Die Attribute der Datenbanktabelle sind in der folgenden Tabelle 3.1.3 abgebildet.

**Tabelle 3.1.3** – Attribute der Tabelle „Benutzer“

| **Attribut** | **Domäne** |
|---|---|
| UserID (Primärschlüssel) | Integer |
| Vorname | Character Varying |
| Nachname | Character Varying |
| Mail | Character Varying |
| Passwort | Character Varying |
| Erreichtes_Level | Integer |
| Letzte_Anmeldung | Character Varying |
| Registriert_am | Character Varying |

Registriert sich ein Benutzer mit seiner UniMail an der Webapplikation, wird ein Datensatz für den Benutzer erzeugt und in der Datenbank gespeichert. Jedem Benutzer ist eine UserID zugewiesen, über die eine eindeutige Identifikation erfolgen kann. Aus dem Format der UniMail der Technischen Universität Dortmund, kann auf den Vor- und Nachnamen des Benutzers geschlossen werden. Bei erfolgreicher Registrierung, wird auch das vom Benutzer vergebene Passwort in der Datenbank gespeichert. Auf dieses Passwort wird bei der Anmeldung des Benutzers zurückgegriffen, um festzustellen, ob sich der Benutzer mit dem korrekten Passwort anmeldet. Das Attribut „Erreichtes_Level“ enthält die Information zu den Aufgaben, die der jeweilige Benutzer bereits gelöst hat. Nach einer erfolgreichen Registrierung wird außerdem der Zeitpunkt der Registrierung protokolliert. Bei jeder Anmeldung wird der Zeitpunkt der Anmeldung ebenfalls protokolliert. Diese Zeitstempel können bei der Verwaltung der Webapplikation für Analysen verwendet werden. So kann beispielsweise untersucht werden wie viele Benutzer die Webapplikation in einem gewissen Zeitraum genutzt haben.
Die Datenbanktabelle „Aufgaben“ enthält die Informationen zu den Aufgabenstellungen, welche in der Webapplikation bearbeitet werden können. Der Aufbau dieser Datenbanktabelle ist in der Tabelle 3.1.4 dargestellt.

**Tabelle 3.1.4** – Attribute der Tabelle „Aufgaben“

| **Attribut** | **Domäne** |
|---|---|
| AufgabenID (Primärschlüssel) | Integer |
| Aufgabenstellung | Character Varying |
| Lösung | Character Varying |

Die Datenbanktabelle „Aufgaben“ setzt sich aus drei Attributen zusammen. Dabei entspricht jeder Datensatz einer Aufgabe in der Webapplikation. Das erste Attribut ist der Primärschlüssel der Datenbanktabelle und enthält die eindeutige ID der Aufgabenstellung. Der Datentyp dieser Spalte ist Integer, wodurch Ganzzahlen als ID definiert werden können. In der zweiten Spalte ist der Text der Aufgabenstellung enthalten, welche dem Benutzer der Webapplikation angezeigt wird. In der letzten Spalte ist das SQL Statement enthalten, welches zur Lösung der Aufgabe führt. Die Datentypen der Spalten „Aufgabenstellung“ und „Lösung“ ist Character Variying, wodurch Zeichenketten von unterschiedlichen Längen definiert werden können.
Die aufgeführten Datenbanktabellen vom dritten bis zum neunten Aufzählungspunkt sind die Aufgabentabellen. Diese Tabellen sind neben dem Text der Aufgabenstellung ein weiterer Bestandteil der Aufgaben. So werden sie in Abhängigkeit der, von dem Benutzer ausgewählten Aufgabe, angezeigt. Die Konzeption und der Aufbau dieser Datenbanktabellen sind in der Arbeit von O. Alić ausgearbeitet worden (Alić, 2021). Die übrigen Datenbanktabellen aus der Aufzählung enthalten Lösungen zu Aufgaben, bei denen die Datenbanktabellen manipuliert werden sollen. So nutzt die Webapplikation diese Tabellen als Vergleichsreferenz, um das Ergebnis des SQL Statements des Benutzers

zu prüfen. Die letzten zwei Zeichen beschreiben dabei für welche Aufgabe die Tabelle jeweils genutzt wird. Bei den übrigen Aufgabenstellungen sind keine separaten Datenbanktabellen erforderlich, da der Webserver SQL Statements, welche lediglich Daten abfragen, ausführen kann, ohne dabei die Datenbank zu manipulieren.

## 3.2 Feindesign der Architektur der Webapplikation

Nachdem das Grobdesign mit den einzelnen Schichten, Komponenten und Schnittstellen erarbeitet worden ist, wird nun auf das Feindesign der Architektur der Webapplikation eingegangen. Das Feindesign bezieht sich dabei auf die implementierten Softwaremodule der Webapplikation. Die Webapplikation umfasst insgesamt sieben Softwaremodule, welcher in der Abbildung 3.2.1 dargestellt sind.

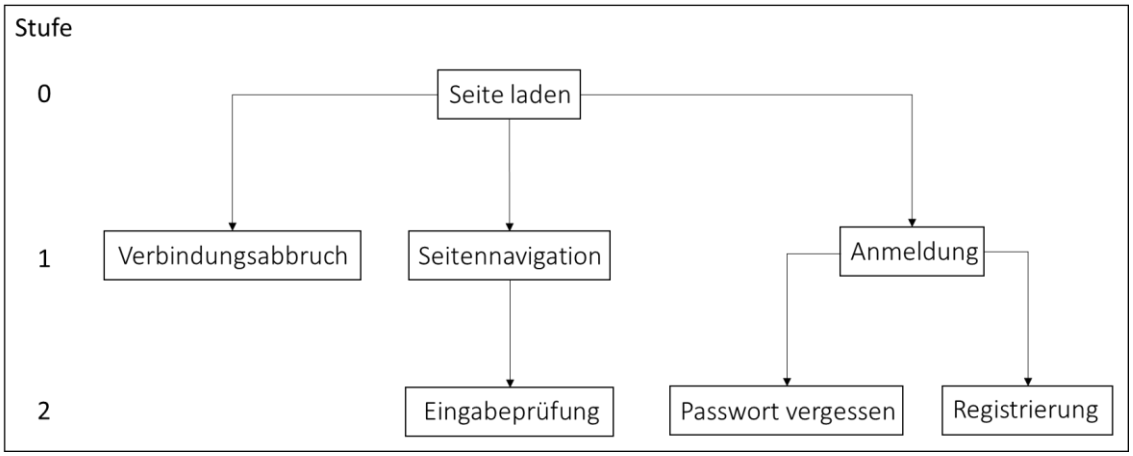


**Abbildung 3.2.1** – 2-stufige hierarchische Gliederung der Softwaremodule der Webapplikation

Bei der Darstellung handelt es sich um eine 2-stufige hierarchische Gliederung der Softwaremodule der Webapplikation. Die Softwaremodule einer Stufe rufen dabei die Softwaremodule der nächst höheren Stufe auf. So ist das Wurzelmodul „Seite laden“. Dieses Softwaremodul bildet die Stufe 0 in der Darstellung. Dieses Softwaremodul ruft die Softwaremodule der Stufe 1 auf. Dabei handelt es sich um die Softwaremodule „Verbindungsabbruch“, „Seitennavigation“ und „Anmeldung“. Die Softwaremodule „Seitennavigation“ und „Anmeldung“ rufen jeweils weitere Softwaremodule der Stufe 2 auf. So wird das Softwaremodul „Eingabeprüfung“ von der Seitennavigation aufgerufen. Die Anmeldung ruft hingegen die Softwaremodule „Passwort vergessen“ und „Registrierung“ auf. Die Abbildung 3.2.1 bildet eine Übersicht über die einzelnen Softwaremodule und deren hierarchischen Zusammenhänge, wobei nicht auf den Inhalt der Softwaremodule eingegangen wird. Der Inhalt und die Funktionen der einzelnen Softwaremodule werden im Folgenden mithilfe von Zustandsdiagrammen, welche im Abschnitt 2.1 eingeführt worden sind, veranschaulicht. Durch die Zustandsdiagramme werden die Zustände, Ereignisse und Aktionen der einzelnen Softwaremodule verdeutlicht. Dabei sind die Zustandsdiagramme durch eine horizontale, gestrichelte

Linie geteilt, welche das Frontend und Backend voneinander abgrenzt. Die Zustandsdiagramme geben lediglich Auskunft über die technischen Abläufe der Webapplikation und nicht über das Design der Benutzeroberfläche der Webapplikation. Das Design und das Verhalten der Benutzeroberfläche wird in der Arbeit von O. Alić behandelt (Alić, 2021).

Seite laden

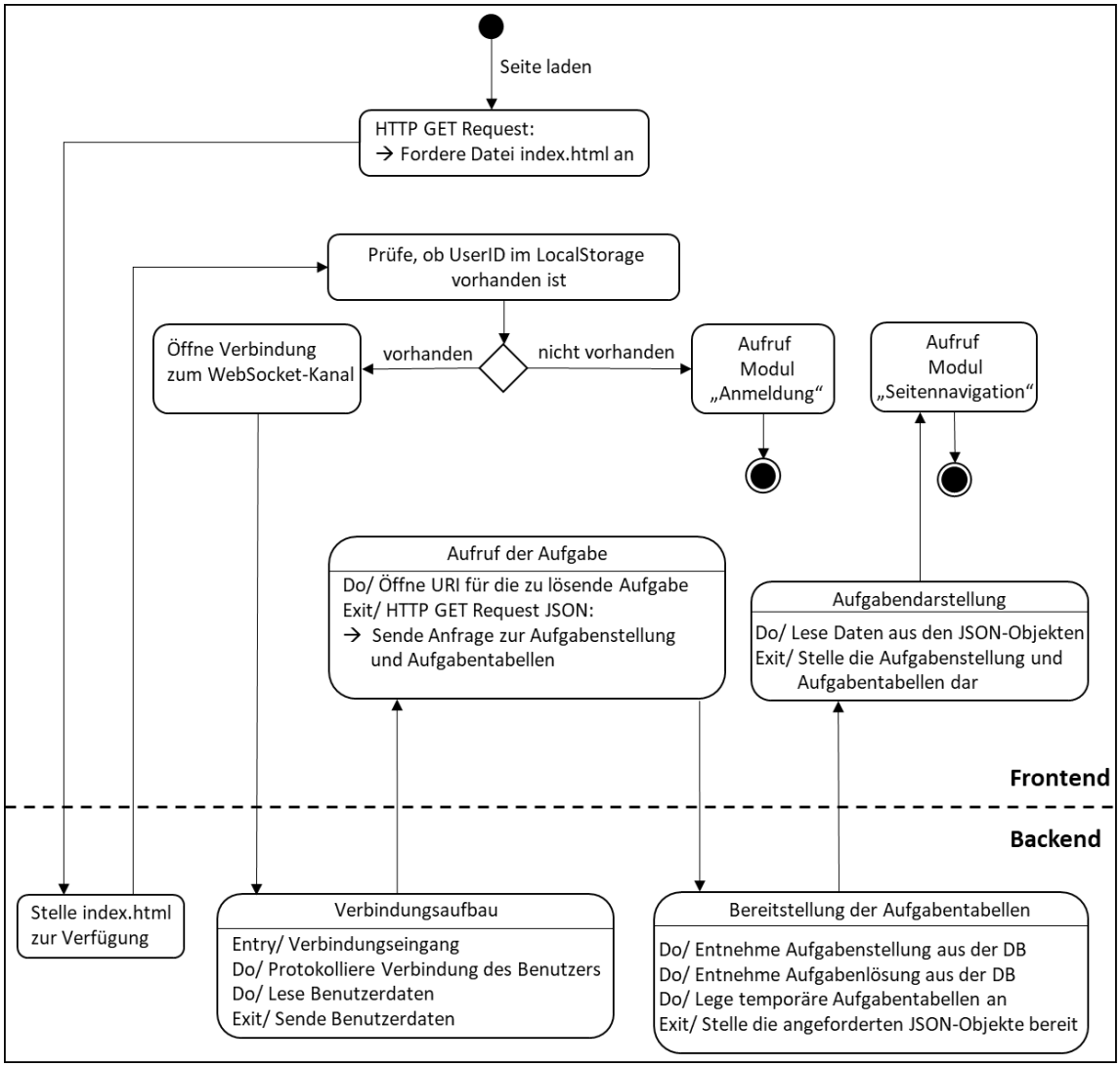


**Abbildung 3.2.2** – Zustandsdiagramm zum Softwaremodul „Seite laden“

Wie bereits erörtert, handelt es sich bei dem Softwaremodul „Seite Laden“ um das Wurzelmodul der Webapplikation. Der Startzustand trifft ein, sobald ein Benutzer die URI der Webapplikation in einem Webbrowser ausführt. Dadurch wird eine HTTP GET-Anfrage vom Webbrowser an den Webserver gesendet. In dieser Anfrage wird die Datei index.html angefordert, welche Daten zur Visualisierung der Webapplikation und Verknüpfungen zu weiteren Dateien enthält. Nachdem die Anfrage des Webbrowsers von dem Webserver beantwortet und die angeforderte Datei index.html zur Verfügung

gestellt wurde, wird die Webapplikation in dem Webbrowser geladen. Während des Ladevorgangs wird geprüft, ob eine UserID im LocalStorage des Webservers vorhanden ist. Falls keine UserID im LocalStorage des Webbrowsers lokalisiert werden kann, wird das Softwaremodul „Anmeldung“ angerufen. Falls eine UserID im LocalStorage des Webbrowsers vorliegt, bedeutet dies, dass sich ein registrierter Benutzer bereits von diesem Webserver aus angemeldet hat. Infolgedessen sendet der Webbrowser eine HTTP-Anfrage zum umschalten des TCP/IP-Anwendungsschichtprotokolls von HTTP zu WebSocket. Diese Anfrage wird vom Webserver akzeptiert. Im Anschluss wird das TCP/IP-Anwendungsschichtprotokoll auf WebSocket umgestellt. Der Webserver ermittelt anhand der UserID, welcher Benutzer die Webapplikation aufruft und protokolliert dessen Verbindung in der Datenbank. Daraufhin liest der Webserver die Benutzerdaten aus der Datenbank aus und sendet diese an den Webbrowser. Zu diesen Daten zählen der Vor- und Nachname des Benutzers, die UniMail des Benutzers und die Anzahl an bereits gelösten Aufgaben. Anhand der Information zu den bereits vom Benutzer gelösten Aufgaben, wird im Webbrowser die URI zu der nächsten, noch nicht gelösten Aufgabe aufgerufen. Dadurch wird eine HTTP GET-Anfrage an den Webserver gesendet. In dieser Abfrage fordert der Webbrowser den Text der Aufgabenstellung und die dazugehörigen Aufgabentabellen im JSON-Format an. Nach Eingang der Anfrage am Webserver, entnimmt dieser die angeforderten Informationen aus der Datenbank. Um auszuschließen, dass die Benutzer der Webapplikation die Aufgabentabellen von anderen Benutzern manipulieren, ist es erforderlich, dass jeder Benutzer eigene Aufgabentabellen erhält, auf die die eingegebenen SQL Statements ausgeführt werden können. Es wäre jedoch nicht effizient für jeden Benutzer permanent gespeicherte Datenbanktabellen anzulegen, da dies zu einem hohen Speicherbedarf in der Datenbank führen würde. Es wäre erforderlich pro Benutzer sieben permanent gespeicherte Datenbanktabellen anzulegen. Um den Benutzern eigene Datenbanktabellen zur Verfügung zu stellen und gleichzeitig die Effizienz der Webapplikation nicht zu mindern, erstellt der Webserver temporäre Datenbanktabellen. Dabei werden nur die Aufgabentabellen als temporäre Datenbanktabelle zur Verfügung gestellt, welche für die Bearbeitung der ausgewählten Aufgabe notwendig sind. Die temporären Datenbanktabellen werden in der folgenden Form in der Datenbank angelegt:

```
temp_[UserID]_[Aufgabentabellenname]
```

Durch die Einbindung der UserID im Namen der Datenbanktabelle wird sichergestellt, dass eine eindeutige Zuordnung von Benutzer zu Datenbanktabelle gewährleistet ist. Die temporären Datenbanktabellen werden so lange in der Datenbank gespeichert,

solange eine WebSocket-Verbindung vom Webbrowser zum Webserver besteht. Die temporären Datenbanktabellen werden mit dem Aufruf des Softwaremoduls „Verbindungsabbruch“ gelöscht. Nachdem die temporären Datenbanktabellen für den Benutzer der Webapplikation erstellt worden sind, werden diese mit der Aufgabenstellung zu der entsprechenden Aufgabe im JSON-Format dem Webbrowser zu Verfügung gestellt. Der Webserver kann mit diesen Informationen die Aufgabenstellung darstellen und anschließend das Softwaremodul „Seitennavigation“ aufrufen.

Verbindungsabbruch

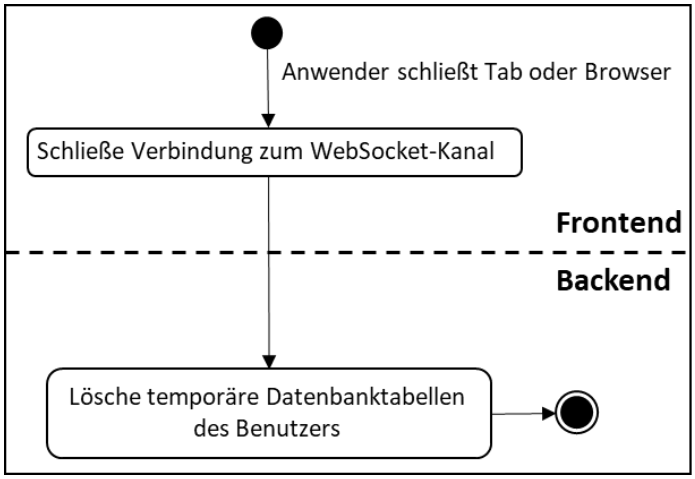


**Abbildung 3.2.3** – Zustandsdiagramm zum Softwaremodul „Verbindungsabbruch“

Der Benutzer der Webapplikation kann die Webapplikation zu jedem Zeitpunkt verlassen. Ist während der Benutzung der Webapplikation ein WebSocket-Kanal geöffnet worden, so erhält der Webserver die Information wenn dieser WebSocket-Kanal vom Webbrowser geschlossen wird. Das Schließen der Verbindung zum WebSocket-Kanal erfolgt durch das Verlassen der Webapplikation. Dieses Ereignis löst das Softwaremodul „Verbindungsabbruch“ aus. Nachdem die WebSocket-Verbindung getrennt worden ist, löscht der Webserver die temporären Datenbanktabellen des Benutzers. Somit ist gewährleistet, dass lediglich die Datenbanktabellen in der Datenbank gespeichert sind, welche zu einem Zeitpunkt benötigt werden.

Anmeldung

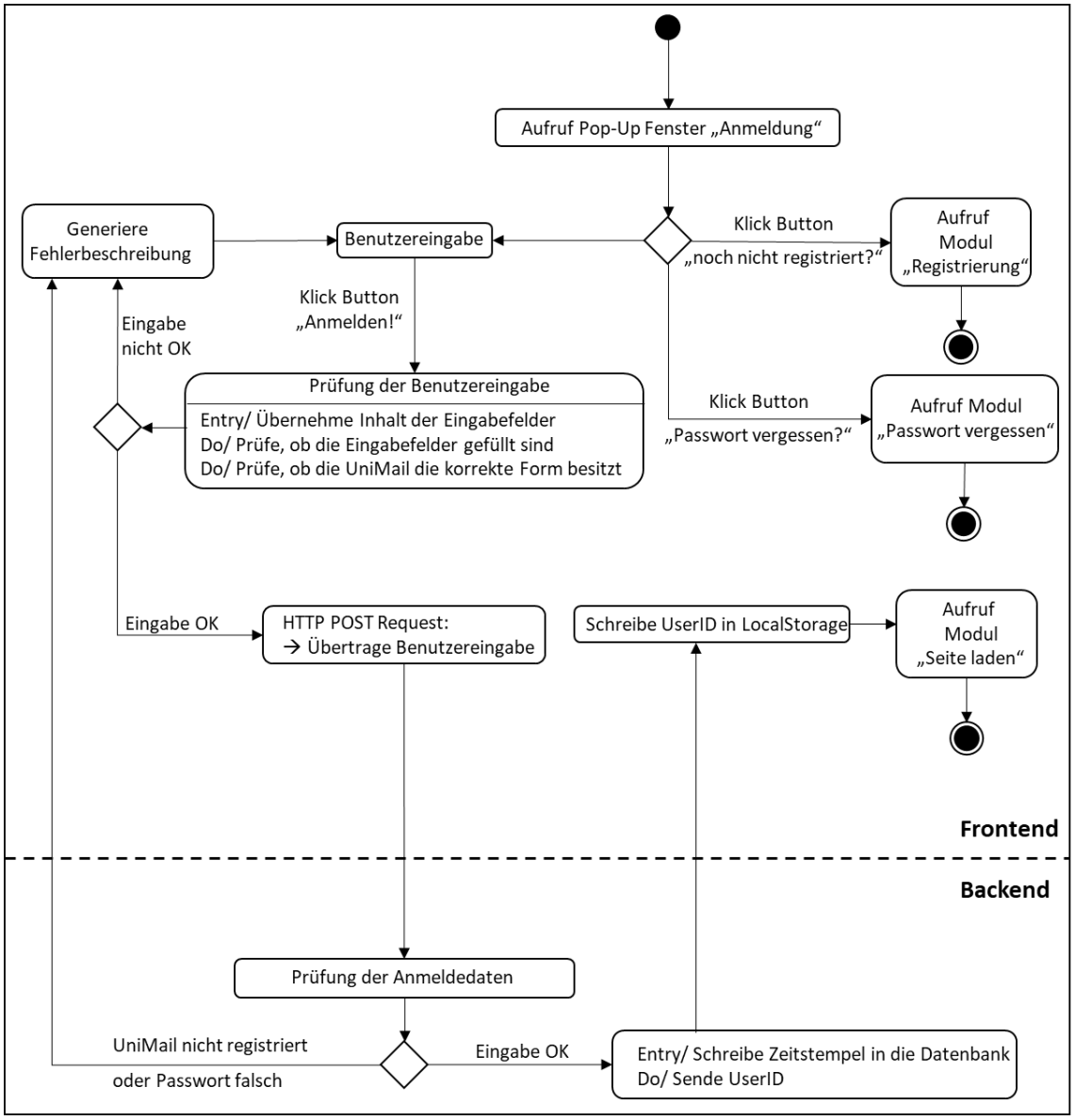


**Abbildung 3.2.4** – Zustandsdiagramm zum Softwaremodul „Anmeldung“

Wie beschrieben tritt der Startzustand des Softwaremoduls „Anmeldung“ ein, wenn keine UserID im LocalStorage des Webbrowsers hinterlegt ist. Durch das Aufrufen dieses Softwaremoduls wird ein Pop-Up Fenster aufgerufen. In diesem Pop-Up Fenster hat der Benutzer der Webapplikation drei Handlungsmöglichkeiten. Nutzt der Benutzer die Webapplikation zum ersten Mal, so kann er sich über den Button „noch nicht registriert“ an der Webapplikation registrieren. Durch einen Klick auf den Button wird das Softwaremodul „Registrierung“ aufgerufen. Ist der Benutzer bereits registriert kann er im Fall, dass er sein Passwort vergessen hat, das Passwort durch das anklicken des Buttons „Passwort vergessen?“ anfordern. Durch den Klick auf diesen Button wird das Softwaremodul „Passwort vergessen“ aufgerufen. Ist der Benutzer registriert und kennt

er seine Anmeldedaten, so kann er diese in die dafür vorgesehenen Felder des Pop-Up Fensters eintragen. Nach der Eingabe der Anmeldeinformationen kann er die Anmeldung per Klick auf den Button „Ausführen“ veranlassen. Durch den Klick auf diesen Button, werden die Inhalte aus den Eingabefeldern übernommen. Zunächst wird geprüft, ob die Eingabefelder gefüllt sind. Anschließend wird die Form der eingetragenen UniMail auf Korrektheit geprüft. Sollte die UniMail nicht in der korrekten Form eingetragen oder die Eingabefelder nicht gefüllt sein, wird eine Fehlermeldung generiert, welche bei der Benutzereingabe ausgegeben wird. Der Benutzer kann nun seine Eingabe korrigieren. Sind die Eingabefelder gefüllt und besitzt die UniMail die korrekte Form, so wird eine HTTP POST-Anfrage an den Webserver gesendet. Der Inhalt dieser Anfrage ist die vom Benutzer getätigte Eingabe. Der Webserver prüft, ob eine UserID zu der UniMail in der Datenbank vorhanden ist. Falls keine UserID zu der UniMail in der Datenbank hinterlegt ist, bedeutet dies, dass die UniMail noch nicht registriert ist. Ist eine UserID zu der UniMail vorhanden, wird anschließend geprüft, ob das eingegebene Passwort mit dem in der Datenbank hinterlegten Passwort übereinstimmt. Sollte der UniMail keine UserID zugewiesen werden können oder das eingegebene Passwort falsch sein, wird diese Information an den Webbrowser gesendet. Dort wird eine Fehlermeldung generiert und zu Ausgabe an den Webbrowser gesendet. Ist die UniMail registriert und ist das eingegebene Passwort korrekt, so wird der Anmeldezeitpunkt in der Datenbank protokolliert. Anschließend sendet der Webserver die entsprechende UserID des Benutzers an den Webbrowser. Diese UserID wird vom Webbrowser in dem LocalStorage hinterlegt. Daraufhin wird das Softwaremodul „Seite laden“ aufgerufen. Dadurch, dass die UserID nun im LocalStorage hinterlegt ist, wird der User an der Webapplikation angemeldet und die Seite wird im Webbrowser dargestellt.

Registrierung

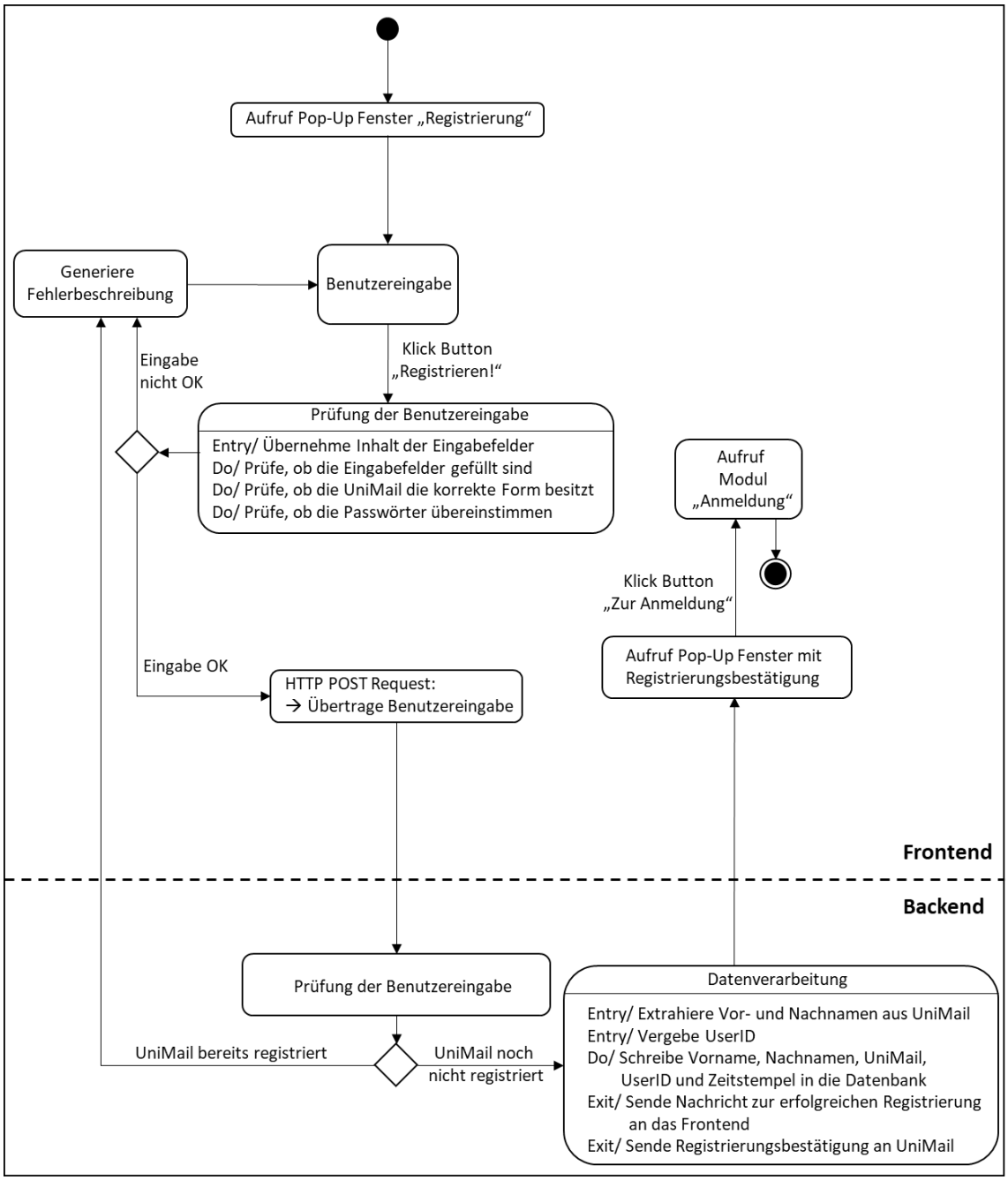


**Abbildung 3.2.5** – Zustandsdiagramm zum Softwaremodul „Registrierung“

Der Startzustand des Softwaremoduls „Registrierung“ löst den Aufruf des Pop-Up Fensters für die Registrierung des Benutzers aus. In diesem Pop-Up Fenster kann der Benutzer seine UniMail eintragen und ein Passwort hinterlegen. Um sicherzustellen, dass der Benutzer das Passwort korrekt vergibt, wird eine wiederholte Eingabe des Passworts verlangt. Der Registrierungsvorgang wird durch das anklicken des Buttons „Registrieren!“ ausgelöst. Die Inhalte der Eingabefelder werden übernommen und es wird geprüft, ob sämtliche Eingabefelder gefüllt sind. Anschließend wird geprüft, ob die

eingetragene UniMail die korrekte Form besitzt. Des Weiteren wird geprüft, ob die beiden Passworteingaben miteinander übereinstimmen. Sollte mindestens eine der drei Prüfungen negativ sein, so wird eine Fehlermeldung generiert und ausgegeben. Der Benutzer hat nun die Möglichkeit seine Eingabe zu korrigieren. Sind die drei Prüfungen positiv, so ist die Form der Eingabe korrekt. Diese wird mit einer HTTP POST-Anfrage an den Webserver übertragen. Der Webserver prüft, ob die eingetragene UniMail bereits in der Datenbank hinterlegt ist. Falls die UniMail in der Datenbank hinterlegt ist, bedeutet dies, dass der Benutzer bereits registriert ist. Diese Information wird zum Webbrowser gesendet, welcher die entsprechende Fehlermeldung generiert und ausgibt. Falls die UniMail nicht in der Datenbank hinterlegt ist so wird die Registrierung des Benutzers am Webserver fortgeführt. Aus der UniMail entnimmt der Webserver den Vor- und Nachnamen des Benutzers. Anschließend wird dem Benutzer eine noch nicht verwendete UserID zugewiesen. Daraufhin werden sämtliche Informationen zu dem Benutzer in die Datenbank geschrieben. Die Registrierung des Benutzers ist somit erfolgreich abgeschlossen. Der Webserver sendet eine E-Mail zur Registrierungsbestätigung an die UniMail des Benutzers. Außerdem sendet der Webserver die Bestätigung zu der erfolgreichen Registrierung an den Webbrowser. Im Webbrowser wird ein Pop-Up Fenster mit der Registrierungsbestätigung aufgerufen. Mit einem Klick auf den Button „Zur Anmeldung“ wird das Softwaremodul „Anmeldung“ aufgerufen. Dort kann sich der Benutzer mit seinen Anmeldeinformationen an der Webapplikation anmelden.

Passwort vergessen

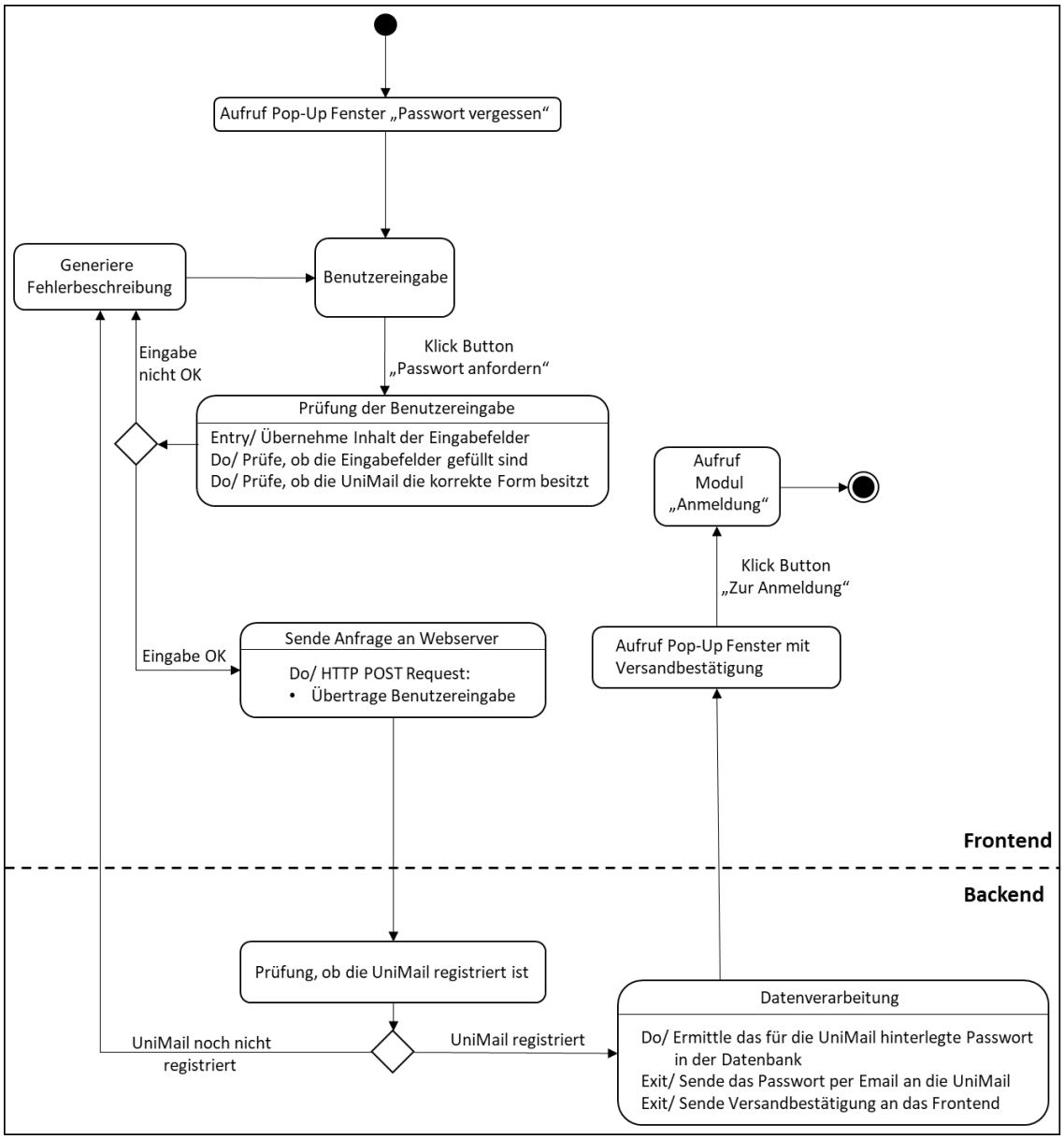


**Abbildung 3.2.6** – Zustandsdiagramm zum Softwaremodul „Passwort vergessen“

Hat sich ein Benutzer bereits an der Webapplikation registriert und sein Passwort vergessen, kann er sich sein hinterlegtes Passwort per E-Mail zukommen lassen. Dazu kann er im Softwaremodul „Anmeldung“ auf den Button „Passwort vergessen?“ klicken. Dadurch wird das Softwaremodul „Passwort vergessen“ aufgerufen und es erscheint ein Pop-Up Fenster. In diesem Pop-Up Fenster kann der Benutzer seine registrierte UniMail eingeben. Nach einem Klick auf den Button „Passwort anfordern“, wird der Inhalt des Eingabefelds übernommen. Anschließend wird geprüft, ob das Eingabefeld gefüllt ist und ob die Form der eingegebenen UniMail korrekt ist. Sollte mindestens eine der beiden Prüfungen negativ sein, so wird eine Fehlermeldung generiert und ausgegeben. Der Benutzer kann anschließend seine Eingabe korrigieren. Sind beide Prüfungen positiv,

so wird die Eingabe mit einer HTTP POST-Anfrage an den Webserver übertragen. Der Webserver prüft, ob die UniMail in der Datenbank hinterlegt ist. Falls dies nicht der Fall ist, bedeutet dies, dass die UniMail noch nicht registriert ist. Der Webserver sendet anschließend diese Nachricht an den Webbrowser, welcher eine Fehlermeldung generiert und ausgibt. Ist die UniMail in der Datenbank hinterlegt, so ist der Benutzer registriert. Daraufhin ermittelt der Webserver das Passwort, welches für die UniMail hinterlegt ist und sendet das Passwort per E-Mail an den Benutzer. Des Weiteren sendet der Webserver eine Versandbestätigung an den Webbrowser. Der Webbrowser ruft anschließend ein Pop-Up Fenster mit der Versandbestätigung auf. Mit einem Klick auf den Button „Zur Anmeldung“, wird das Softwaremodul „Anmeldung“ aufgerufen und der Benutzer kann sich an der Webapplikation anmelden.

Seitennavigation

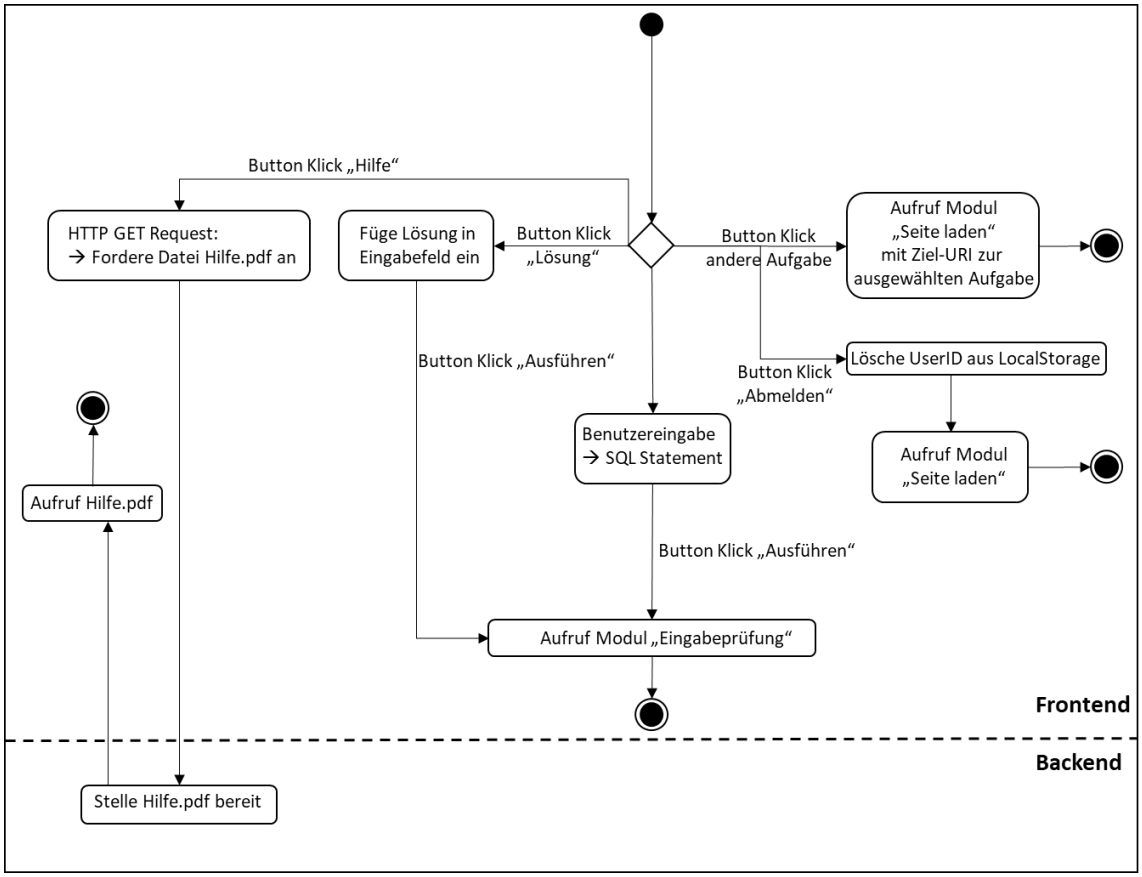


**Abbildung 3.2.7** – Zustandsdiagramm zum Softwaremodul „Seitennavigation“

Hat sich ein Benutzer erfolgreich an der Webapplikation registriert und angemeldet, wird das Softwaremodul „Seitennavigation“ aufgerufen. Dieses Softwaremodul bietet dem Benutzer vier Handlungsmöglichkeiten. Der Benutzer kann auf den Button „Abmelden“ klicken, wodurch die UserID aus dem LocalStorage des Webbrowsers gelöscht wird. Anschließend wird das Softwaremodul „Seite laden“ aufgerufen. Da nun keine UserID

mehr im LocalStorage des Webbrowsers hinterlegt ist, wird das Softwaremodul „Anmeldung“ aufgerufen. Des Weiteren kann der Benutzer per Klick die Aufgabe auswählen, die er bearbeiten möchte. Dabei werden die entsprechende URI für diese Aufgabe und das Softwaremodul „Seite laden“ aufgerufen. Falls der Benutzer die Anleitung zu der Webapplikation lesen möchte, kann er diese mit einem Klick auf den Button „Hilfe“ aufrufen. Dadurch wird eine HTTP GET-Anfrage an den Webserver gesendet. In dieser Anfrage wird die Datei Hilfe.pdf angefordert. Der Webserver bearbeitet diese Anfrage und stellt die angeforderte Datei zu Verfügung. Diese wird anschließend im Webbrowser dargestellt. Falls der Benutzer die Aufgabe nicht eigenständig lösen kann, kann er auf den Button „Lösung“ klicken und sich das SQL Statement, welches zur Lösung der Aufgabe führt, ausgeben lassen. Andernfalls kann der Benutzer das SQL Statement in ein Eingabefeld eintragen. Durch Anklicken des Buttons „Ausführen“, kann er das SQL Statement ausführen lassen. Dadurch wird das Softwaremodul „Eingabeprüfung“ ausgeführt.

Eingabeprüfung

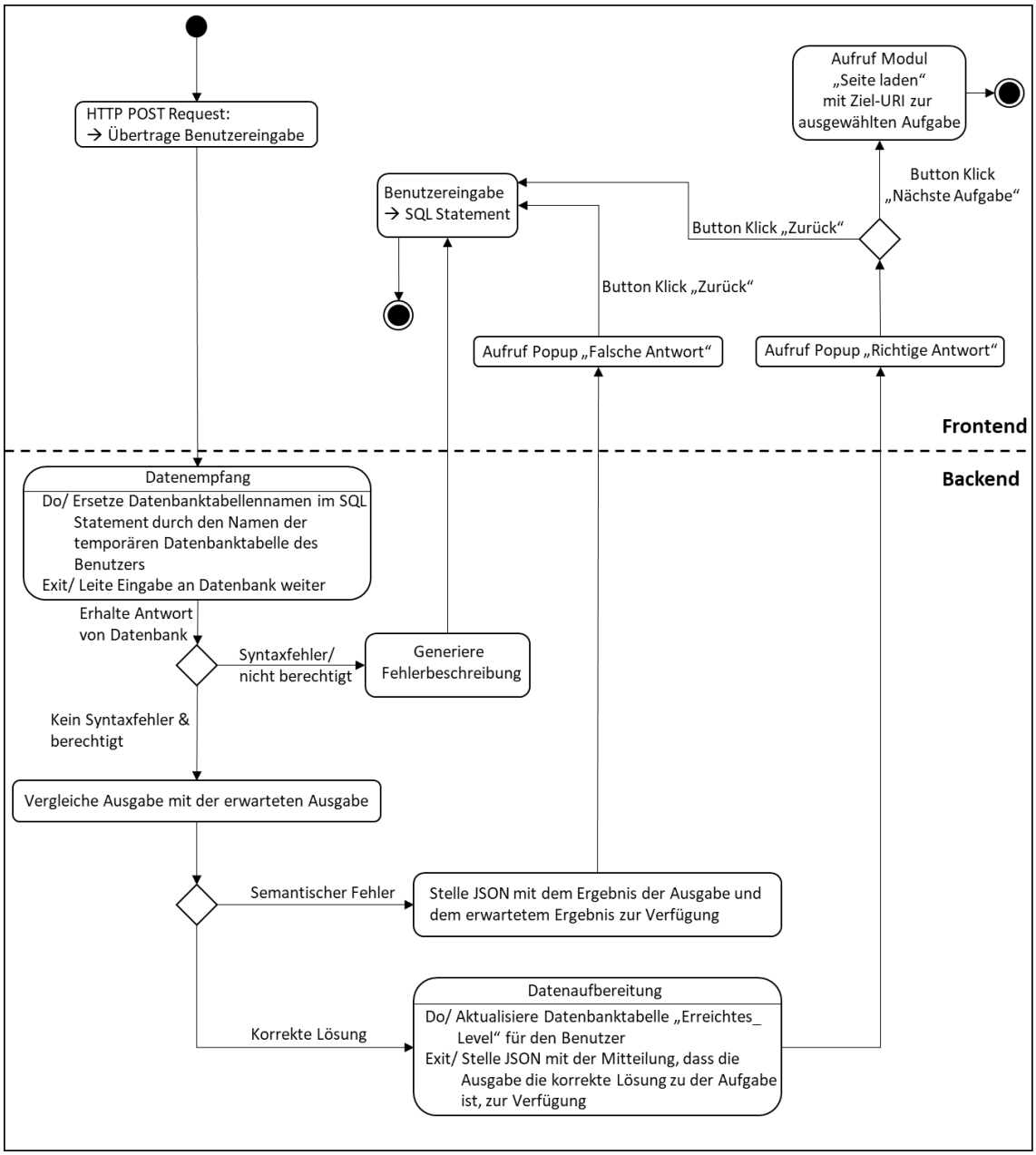


**Abbildung 3.2.8** – Zustandsdiagramm zum Softwaremodul „Eingabeprüfung“

Das Softwaremodul „Eingabeprüfung“ wird aufgerufen, wenn der Benutzer der Webapplikation sein eingegebenes SQL Statement ausführt. Das eingegebene Statement wird mithilfe einer HTTP POST-Anfrage an den Webserver gesendet. Der Webserver passt das SQL Statements des Benutzers an, indem er die Datenbanktabellennamen durch die Namen der temporären Datenbanktabellen des Benutzers ersetzt. Somit hat das SQL Statement lediglich Auswirkung auf die Datenbanktabellen des Benutzers, ohne dass er diese im SQL Statement konkret angeben muss. Anschließend wird die SQL Anfrage an die Datenbank weitergeleitet, in welcher diese ausgeführt wird. Die Datenbank verarbeitet das SQL Statement und

übergibt dem Webserver die Fehler des SQL Statements oder die Ausgabe des SQL Statements. Dabei wird zunächst geprüft, ob der Benutzer für die Ausführung der im SQL Statement enthaltenen Schlüsselwörter berechtigt ist. In der Webapplikation sind Berechtigungseinschränkungen für die Benutzer enthalten, um unter anderem zu verhindern, dass Datenbanktabellen gelöscht werden. Falls ein Syntaxfehler in dem SQL Statement enthalten ist, generiert der Webserver die entsprechende Fehlerbeschreibung und sendet diese an den Webbrowser. Der Webbrowser gibt die Fehlerbeschreibung aus. Der Benutzer kann sein SQL Statement nun korrigieren. Enthält das eingegebene SQL Statement keine Syntaxfehler, so erhält der Webserver die Ausgabe des SQL Statements von der Datenbank im JSON-Format. Daraufhin vergleicht der Webserver diese Ausgabe mit der erwarteten Ausgabe zur Lösung der Aufgabe. Falls diese beiden Ausgaben nicht übereinstimmen, liegt ein fachlicher Fehler vor. Anschließend stellt der Webserver die Ausgabe des eingegebenen SQL Statements und die erwartete Ausgabe im JSON-Format zur Verfügung. Der Webbrowser ruft das Pop-Up Fenster „Falsche Antwort“ auf und gibt dem Benutzer der Webapplikation die falsche und die erwartete Ausgabe aus. Durch einen Klick auf den Button „Zurück“ gelangt der Benutzer zum Eingabefeld und kann seine Eingabe korrigieren. Ergibt der Vergleich der beiden Ausgaben, dass die Ausgabe des eingegebenen SQL Statements mit der erwarteten Ausgabe übereinstimmt, so ist die Aufgabe gelöst worden. Der Webserver stellt dem Webbrowser die Ausgabe im JSON-Format zur Verfügung. Daraufhin wird die Datenbanktabelle „Erreichtes_Level“ aktualisiert, sodass der Benutzer die nächste Aufgabe bearbeiten kann. Anschließend wird im Webbrowser das Pop-Up Fenster „Richtige Antwort“ aufgerufen. In diesem Pop-Up Fenster hat der Benutzer zwei Handlungsmöglichkeiten. Er kann zum einen auf den Button „Zurück“ klicken und die Aufgabe erneut bearbeiten. Zum anderen kann er auf den Button „Nächste Aufgabe“ klicken. Dadurch wird das Softwaremodul „Seite laden“ mit der URI der nächsten Aufgabe aufgerufen.

# 4 Softwaretests der Webapplikation

Die Implementierung der Webapplikation erfolgt in Kooperation mit O. Alić. Nach der Implementierung der Softwaremodule aus der Abbildung 3.2.1, erfolgen in diesem Kapitel die Softwaretests der Webapplikation. Mithilfe der Softwaretests soll die Implementierung durch geeignete Testfälle validiert werden. Dabei werden die sieben Grundsätze des Softwaretestens des International Software-Testing Qualification Board berücksichtigt. So ist unter anderem bekannt, dass ein vollständiges Testen der Webapplikation nicht möglich ist. Durch die Softwaretests kann die Existenz von Fehlern nachgewiesen, wobei nicht nachgewiesen werden kann, dass die Webapplikation fehlerfrei ist. In diesem Kapitel wird der Komponententest, der Integrationstest und der Systemtests der Webapplikation durchgeführt. Die Grundlagen zu diesen Softwaretests sind in dem Abschnitt 2.3 der Arbeit von O. Alić ausgearbeitet worden. In dem Komponententest wird die Implementierung der einzelnen Softwaremodule auf Abweichungen zu dem Sollverhalten geprüft. Das Sollverhalten der einzelnen Softwaremodule ist dabei in der Feinarchitektur der Webapplikation in Kapitel 3 vorgestellt worden. Im Integrationstest wird geprüft, ob das Zusammenspiel der einzelnen Softwaremodule die Anforderungen aus der Feinarchitektur der Webapplikation erfüllen. Abschließend wird das System als Ganzes getestet. In der Arbeit von O. Alić wird über die aufgeführten Testverfahren hinaus, ein Usability-Test der Webapplikation durchgeführt. Nach der Lokalisierung der genannten Abweichungen, wird die Implementierung der Softwaremodule angepasst, woraufhin erneut Softwaretests durchgeführt werden. Dieses Vorgehen wird so oft durchgeführt, bis sämtliche behandelte Testfälle positiv sind.

## 4.1 Komponententest und Integrationstest

Wie bereits eingeführt, ist das erste anzuwendende Testverfahren der Komponententest. Bei den Komponenten handelt es sich dabei um die Softwaremodule aus der Abbildung 3.2.1. Die Komponenten werden voneinander isoliert betrachtet, um komponentenexterne Einflüsse auszuschließen. Für die einzelnen Komponenten werden Testfälle untersucht. Je nach Testfall erfolgen die Tests programmcodenah oder durch die manuelle Nutzung der Benutzeroberfläche. Es werden Testfunktionen als Codeabschnitt hinzugefügt, welche die einzelnen Komponenten aufrufen. Dabei übergeben die Testfunktionen Eingabewerte an die einzelnen Komponenten und prüfen, ob die von den Komponenten gelieferten Ausgaben mit der erwarteten Ausgabe übereinstimmen. Vor der Durchführung der Tests wird der Programmcode um Druckanweisungen erweitert. Es werden nach fast jeder Zeile im Code Druckanweisungen hinzugefügt, um das Verhalten der Software Schritt für Schritt nachvollziehen zu können. Darüber hinaus erleichtern die Druckanweisungen die

Lokalisierung von auftretenden Fehlern in den einzelnen Softwaremodulen. Wird ein Fehler im Komponententest entdeckt, so kann dieser dem getesteten Softwaremodul eindeutig zugeordnet werden. Für die Tests jeder Komponente werden separate Tabellen erstellt, in denen die Testfälle aufgeführt sind. Im der folgenden Tabelle 4.1.1 sind die Testfälle des Softwaremoduls „Passwort vergessen“ dargestellt.

**Tabelle 4.1.1** – Testfälle zum Softwaremodul „Passwort vergessen“

| **Testkomponente: Passwort vergessen** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Befüllen des Eingabefelds | Beispiel Eingabe | Eingabe möglich | Eingabe möglich | OK |
| 2 | Button Klick "Passwort anfordern" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 3 | Hover über Button „Passwort anfordern“ | Hover | Button-Farbe ändert sich | Button-Farbe ändert sich | OK |
| 4 | Übernahme der Inhalte der Eingabefelder | Beispiel Inhalte | Inhalte der Eingabefelder | Inhalte der Eingabefelder | OK |
| 5 | Prüfung, ob Eingabefeld "UniMail" gefüllt ist | 1) UniMail | OK | OK | OK |
| 6 | Prüfung, ob Eingabefeld "UniMail" gefüllt ist | 2) keine Eingabe | Bitte UniMail eingeben | UniMail nicht im korrekten Format | NOK |
| 7 | Prüfung, ob die Form der UniMail korrekt ist | 1) korrekte UniMail | OK | OK | OK |
| 8 | Prüfung, ob die Form der UniMail korrekt ist | 2) Nicht korrekte UniMail | UniMail nicht im korrekten Format | UniMail nicht im korrekten Format | OK |
| 9 | Inhalt an den Webserver übertragen | Beispielinhalt | Ausgabe des Beispielinhalts am Webserver | Ausgabe des Beispielinhalts am Webserver | OK |
| 10 | Prüfung, ob UniMail registriert ist | 1) Nicht registrierte UniMail | UniMail ist nicht registriert | UniMail ist nicht registriert | OK |
| 11 | Prüfung, ob UniMail registriert ist | 2) registrierte UniMail | OK | OK | OK |
| 12 | UserID in der DB ermitteln | UniMail | UserID zur UniMail | UserID zur UniMail | OK |
| 13 | Passwort in der DB ermitteln | UniMail | Passwort zur UniMail | Passwort zur UniMail | OK |
| 14 | Sende Passwort per E-Mail | Registrierte UniMail | E-Mail mit dem Passwort im Postfach der UniMail | Keine E-Mail erhalten | NOK |
| 15 | Aufruf des Pop-Up Fensters "Versandbestätigung" | Erfolgreiche Anmeldung | UserID im LocalStorage | UserID im SessionStorage | NOK |

Jede Zeile der Tabelle 4.1.1 enthält die Informationen zu einem Testfall. Jedem Testfall ist eine eindeutige ID zugewiesen. Des Weiteren wird der Inhalt des Testfalls in jeder

Zeile beschrieben. Zu jedem Testfall ist die getätigte Eingabe aufgeführt, welche dem Softwaremodul übergeben wird. Das Softwaremodul verarbeitet diese Eingabe und generiert eine Ausgabe. Diese Ausgabe wird mit der erwarteten Ausgabe abgeglichen. Entspricht die tatsächliche Ausgabe des Softwaremoduls der erwarteten Ausgabe, so wird das Testergebnis in der Spalte „E“ zu dem Testfall als „OK“ gekennzeichnet und das Istverhalten stimmt mit dem Sollverhalten überein. Unterscheidet sich die tatsächliche Ausgabe des Softwaremoduls von der erwarteten Ausgabe, so wird das Testergebnis zu dem Testfall als „NOK“ (kurz für „Nicht OK“) gekennzeichnet. Darüber hinaus wird die Zeile des Testfalls rot hervorgehoben. In der Tabelle 4.1.1 ist somit ersichtlich, dass drei der 15 Testfälle die im Feindesign der Architektur der Webapplikation definierten Anforderungen nicht erfüllen. Es wird keine korrekte Fehlermeldung von der Webapplikation ausgegeben, wenn das Eingabefeld für die UniMail nicht gefüllt wird. Außerdem funktioniert das Versenden der E-Mail mit dem Passwort des Benutzers nicht. Daraus resultiert, dass das Pop-Up Fenster „Versandbestätigung“ nicht aufgerufen wird. Nachdem die Testfälle, die Fehler aufweisen, identifiziert worden sind, werden die Fehler mithilfe der Druckanweisungen im Programmcode lokalisiert. Anschließend erfolgt das Debugging der aufgetretenen Fehler. Dazu wird der Programmcode angepasst und getestet, bis die aufgetretenen Fehler behoben sind. Abschließend werden erneut sämtliche Testfälle zu dem Softwaremodul geprüft, um auszuschließen, dass die Anpassung des Programmcodes neue Fehler erzeugt hat. In dem Fall des Softwaremoduls „Passwort vergessen“, sind die Fehler bereits nach dem ersten Durchlauf der Programmcodeanpassung behoben worden. Es kann jedoch sein, dass mehrere Durchläufe erforderlich sind, um sämtliche Fehler zu beheben. Dieses Vorgehen wird für die sieben Softwaremodule durchgeführt. Die Testfälle der einzelnen Softwaremodule sind in den Tabellen A.1 bis A.6 aufgeführt. In den Tabellen sind die die Testergebnisse des ersten Testdurchlaufs dargestellt. Die dabei entdeckten Fehler sind nach der Lokalisierung behoben worden. Da es bei den Softwaretests lediglich möglich ist, Stichprobenmengen zu untersuchen, ist es erforderlich den Testaufwand gegenüber den Prioritäten und Risiken abzuwägen (Spillner & Linz, 2012). Für den Komponententest werden insgesamt 212 Testfälle untersucht. Die hohe Anzahl der Testfälle ergibt sich aus der Granularität der Testfälle. So sind die Prüfungen der Ausführbarkeit jedes Buttons in einzelne Testfälle aufgeteilt. Es wird geprüft, ob die Softwaremodule die korrekten Fehlermeldungen generieren, wenn beispielsweise Eingabefehler vorliegen. Weitere Testfälle beziehen sich nicht auf die Funktionalität der Komponenten, sondern auf die Nutzung der Webapplikation als Anwender. So wird getestet, ob sich die Farben der Buttons ändern, wenn der Benutzer den Mauszeiger über diesen Button führt. Für die Funktion des Softwaremoduls spielt die Farbe des Buttons keine Rolle. Dennoch ist es wichtig, dass der Anwender der Webapplikation die gewohnten Rückmeldemechanismen von Webseiten vorfindet, damit er die Webapplikation als intuitiv und verständlich empfindet (Chlebek, 2006). Mit der hohen Anzahl an Testfällen kann zwar nicht ausgeschlossen werden, dass keine

weiteren unentdeckten Fehler existieren, jedoch wird die Wahrscheinlichkeit für die Existenz solcher Fehler minimiert (Witte, 2016).
Nachdem der Komponententest abgeschlossen ist, wird im nächsten Schritt der Integrationstest durchgeführt. Der Integrationstest dient dazu das Zusammenspiel der voneinander abhängigen Softwaremodule zu untersuchen. Um den Integrationstest durchzuführen ist das Verständnis über die Zusammenhänge der einzelnen Softwaremodule erforderlich (Witte, 2016). Die Zusammenhänge der einzelnen Softwaremodule sind in der Abbildung 3.2.1 ersichtlich. Die Integration der Softwaremodule erfolgt nach dem Bottom-Up-Prinzip. So beginnt die Integration in der Stufe 2 der hierarchischen Gliederung der Softwaremodule. Zunächst werden also die Zusammenspiele zwischen den Softwaremodulen „Eingabeprüfung“ mit dem Softwaremodul „Seitennavigation“ und den Softwaremodulen „Passwort vergessen“ und „Registrierung“ mit dem Softwaremodul „Anmeldung“ getestet. Für die Tests werden Tabellen zu den einzelnen Testfällen analog zu dem Komponententest erstellt. In der folgenden Tabelle 4.1.2 sind die Testfälle für das Zusammenspiel der Softwaremodule „Eingabeprüfung“ und „Seitennavigation“ dargestellt.

**Tabelle 4.1.2** – Testfälle „Seitennavigation“ → „Eingabeprüfung“

| **Seitennavigation → Eingabeprüfung** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Button Klick "Ausführen" | Klick | Aufruf Softwaremodul "Eingabeprüfung" | Aufruf Softwaremodul "Eingabeprüfung" | OK |
| 2 | Datenübertragung von dem Softwaremodul "Seitennavigation" zu "Eingabeprüfung" | Beispiel-daten | Ausgabe der Daten im Softwaremodul "Eingabeprüfung" | Daten werden nicht im Softwaremodul "Eingabeprüfung" ausgegeben | NOK |

Für den Integrationstest der Webapplikation existieren deutlich weniger Testfälle, die geprüft werden müssen als beim Komponententest. Dies liegt daran, dass die übergeordneten Softwaremodule die untergeordneten Softwaremodule in den meisten Fällen lediglich aufrufen. Finden Datenübertragungen statt, so erfolgen diese ohne weitere Datentransformationen, was die Komplexität des Zusammenspiels zwischen den Softwaremodulen reduziert. Die Softwaremodule werden sukzessive integriert. Tritt ein Fehler auf so muss dieser lokalisiert werden. Da die Tests im Integrationstest komponentenübergreifend stattfinden, ist der Aufwand zur Lokalisierung der Fehlerquelle größer als beim Komponententest. Es ist möglich, dass die Fehler, die im Integrationstest entdeckt werden, auf nicht abgedeckte Testfälle im Komponententest zurückzuführen sind. In den Tabellen A.7 bis A.10 sind die Testfälle zu sämtlichen Zusammenspielen der einzelnen Softwaremodule enthalten. Die Durchführung der Tests und die Anpassung des Programmcodes erfolgt iterativ. So wird der Programmcode nach einem Testdurchlauf angepasst, woraufhin erneut getestet wird. Dieses Vorgehen wird so oft durchgeführt, bis sämtliche Testfälle in einem positiven Ergebnis resultieren. Mit der Durchführung des Komponententests und des Integrationstests sind die

programmcodenahen Tests abgeschlossen. Die Komponenten erfüllen für sich und im Zusammenspiel mit den übrigen Komponenten die Anforderungen der Fein-Architektur der Webapplikation.

## 4.2 Systemtests

Nach der Durchführung des Komponententests und des Integrationstests, ist die nächste Teststufe der Systemtest. Bei diesem wird das System als Ganzes getestet. Dazu wird eine Testumgebung erzeugt. Für die Testumgebung wird ein virtueller Server bei der Strato AG gemietet. Die Systemeigenschaften des Testservers sind in der Tabelle 4.2.1 dargestellt.

**Tabelle 4.2.1** – Systemeigenschaften des Testservers

| Systemkomponente | Eingesetzte Ressource |
|---|---|
| Betriebssystem | Windows Server 2019 Standard |
| CPU | Intel® Xeon® CPU E5-2680 v3 @ 2.50GHz 2.50GHz (2 Kerne) |
| Arbeitsspeicher | 4GB DDR4 RAM |
| Festplattenspeicher | 200 GB HDD |
| Netzwerkanbindung | Bis zu 1.000 Mbit/s |
| Virtualisierer | Microsoft Hyper-V |

Bei der Produktivumgebung wird es sich um einen Server der Technischen Universität Dortmund handeln. Die Systemeigenschaften des Testservers können jedoch aufschlussreich für die erforderliche Skalierung des Produktivservers sein. Auf dem Testserver wird das Backend der Webapplikation installiert und eingerichtet. Dazu zählen der Webserver und das Datenbankmanagementsystem. Die Webapplikation wird auf der Testumgebung auf Interoperabilität, Stabilität und Robustheit geprüft. Die Interoperabilität beschreibt das fehlerfreie Arbeiten der Webapplikation auf der Testumgebung (Witte, 2016). Um diesen Test durchzuführen werden sämtliche Funktionalitäten der Webapplikation über einen Webbrowser getestet. Weitere funktionale Tests werden im Rahmen der Usability-Tests von O. Alić durchgeführt (Alić, 2021). Nachdem der erste manuelle Test erfolgreich abgeschlossen ist, werden die nichtfunktionalen Tests durchgeführt. Somit werden Lasttests durchgeführt, um die Stabilität und Robustheit der Webapplikation zu prüfen. Um die Lasttests durchzuführen wird ein Simulationsprogramm entwickelt. Dabei handelt es sich um ein Python-Skript, welches das Python-Modul Locust verwendet. Mithilfe dieses Skripts kann eine Vielzahl an Benutzern simuliert werden, welche die Webapplikation gleichzeitig verwenden. Diese Benutzer werden im weiteren Verlauf der Arbeit als virtuelle Benutzer bezeichnet. Um die Nutzung der Webapplikation realitätsnah zu simulieren, werden in dem Simulationsprogramm sämtliche URIs zu den 19 Aufgaben der Webapplikation hinterlegt. Diese URIs werden von den virtuellen Benutzern in zufälliger Reihenfolge

aufgerufen. Dabei verweilt ein virtueller Benutzer zwischen einer Sekunde und zwei Minuten auf der aufgerufenen Seite der Webapplikation. Das erste Simulationsexperiment untersucht das Verhalten der Webapplikation, wenn diese von zehn virtuellen Benutzern über einen Zeitraum von 24 Stunden parallel verwendet wird. Dabei wird die Anzahl der Anfragen auf jede URI, die Anzahl der aufgetretenen Fehler und die durchschnittliche Antwortzeit des Webservers protokolliert. In der Tabelle 4.2.2 sind die Ergebnisse des aufgeführten Simulationsexperiments dargestellt.

**Tabelle 4.2.2** – Testfall: 10 virtuelle Benutzer; Testzeitraum: 24 Stunden

| URI | Anzahl Anfragen | Anzahl Fehler | Ø Antwortzeit [Ms] |
|---|---|---|---|
| /aufgabe=1 | 758 | 0 | 343 |
| /aufgabe=2 | 734 | 0 | 312 |
| /aufgabe=3 | 727 | 0 | 312 |
| /aufgabe=4 | 804 | 0 | 312 |
| /aufgabe=5 | 768 | 0 | 343 |
| /aufgabe=6 | 813 | 0 | 343 |
| /aufgabe=7 | 707 | 0 | 342 |
| /aufgabe=8 | 768 | 0 | 312 |
| /aufgabe=9 | 721 | 0 | 296 |
| /aufgabe=10 | 830 | 0 | 344 |
| /aufgabe=11 | 781 | 0 | 327 |
| /aufgabe=12 | 750 | 0 | 358 |
| /aufgabe=13 | 755 | 0 | 312 |
| /aufgabe=14 | 792 | 0 | 328 |
| /aufgabe=15 | 776 | 0 | 327 |
| /aufgabe=16 | 764 | 0 | 312 |
| /aufgabe=17 | 718 | 0 | 312 |
| /aufgabe=18 | 773 | 0 | 297 |
| /aufgabe=19 | 755 | 0 | 328 |
| Aggregiert: | 14.494 | 0 | 324 |

In der Tabelle 4.2.2 sind die URIs, die von den zehn virtuellen Benutzern angefragt worden sind aufgeführt. Zu jeder URI ist die getätigte Anzahl an Anfragen, die Anzahl der aufgetretenen Fehler und die durchschnittliche Antwortzeit des Servers in Millisekunden (Ms) eingetragen. In diesem Simulationsexperiment haben die zehn virtuellen Benutzer insgesamt 14.494 Anfragen an den Webserver gesendet. Sämtliche Anfragen sind dabei fehlerfrei beantwortet worden. Die durchschnittliche Antwortzeit des Servers beträgt 324 Ms. Die Antwortzeit beschreibt die Zeit vom Versenden einer Anfrage des Webbrowsers bis zum Eingang der Antwort des Webservers am Webbrowser. Dabei liefert der Webserver sämtliche Informationen, die zur Darstellung der Webseite erforderlich ist. Die Zeit, die für die Darstellung der Webapplikation in dem

Webbrowser benötigt wird, ist kein Bestandteil der Antwortzeit (Rohr, 2018). Antwortzeiten, die im Bereich zwischen 200 Ms und 350 Ms liegen, werden als schnell angesehen. Antwortzeiten zwischen 400 Ms und 700 Ms werden als durchschnittliche Antwortzeiten von Webseiten eingestuft. Antwortzeiten, die größer als 700 Ms sind, werden hingegen als langsam eingestuft (*Why Server Response Time Is Important*, 2020). Somit ist die Antwortzeit des Testservers mit 324 Ms eine schnelle Antwortzeit. Die Webapplikation ist für den Einsatz am Lehrstuhl für IT in Produktion und Logistik der Technischen Universität Dortmund vorgesehen. An dem Lehrstuhl wird das Thema SQL unter anderem in dem Fach Informationsaustausch produzierender Unternehmen gelehrt. Zum Zeitpunkt der Erstellung dieser Arbeit sind für dieses Fach 171 Teilnehmer eingeschrieben. Um einen aussagekräftigen Lasttest durchzuführen wird von der maximalen Systemlast ausgegangen. In dem Fall der Webapplikation dieser Arbeit bedeutet dies, dass davon ausgegangen wird, dass 171 virtuelle Benutzer die Webapplikation gleichzeitig nutzen. Um das Verhalten der Webapplikation in Abhängigkeit der Anzahl der Benutzer, die die Webapplikation parallel verwenden zu untersuchen, werden mehrere Lasttests mit variierenden Benutzerzahlen durchgeführt. So werden jeweils Tests über 24 Stunden für zehn, 50, 100, 200 und 300 Benutzer durchgeführt. Die Ergebnisse der Tests sind in der folgenden Tabelle 4.2.3 zusammengeführt.

**Tabelle 4.2.3** – Verhalten der Webapplikation in Abhängigkeit der Benutzeranzahl

| Anzahl Benutzer | Anzahl Anfragen | Anzahl Fehler | Ø Antwortzeit [Ms] |
|---|---|---|---|
| 10 | 14.494 | 0 | 324 |
| 50 | 72.336 | 0 | 349 |
| 100 | 144.750 | 0 | 346 |
| 200 | 285.468 | 0 | 670 |
| 300 | 427.857 | 16 | 1.024 |

In der Tabelle 4.2.3 ist ersichtlich, dass bei zehn, 50, 100 und 200 virtuellen Benutzern keine Fehler auftreten. Bei dem Test mit 300 virtuellen sind bei 427.857 Anfragen an den Server 16 Fehler aufgetreten. Die ausgegebene Fehlermeldung ist in sämtlichen Fällen „ERR_CONNECTION_TIMED_OUT“. Dieser Fehler tritt auf, wenn die Antwortzeit des Webservers eine im Webbrowser definierte Zeit überschreitet. Dieser Fehler ist jedoch unkritisch, da der Webbrowser die Antwort vom Webserver, wenn auch verspätet, erhält. Anschließend wird die Webapplikation im Webbrowser dargestellt. Der Anwender der Webapplikation muss dazu keine Aktion durchführen, sondern lediglich einige Sekunden länger warten. Da der Fehler nur sehr selten auftritt und sich selbst behebt, wird er als unkritisch eingestuft. Durch dieses Verhalten der Webapplikation bei dem Auftreten eines Fehlers, ist die Robustheit der Webapplikation gewährleistet. Des Weiteren ist aus der Tabelle 4.2.3 die Abhängigkeit der Antwortzeit des Servers von der Anzahl der Benutzer ersichtlich. Bei bis zu 100 parallelen Benutzer ist die Antwortzeit annähernd konstant und befindet sich im Bereich von schnellen Antwortzeiten. Bei dem Test mit

200 parallelen Benutzern verdoppelt sich die Antwortzeit in etwa. Diese liegt bei 670 Ms und befindet sich im Bereich von durchschnittlichen Antwortzeiten von Webseiten. Somit ist die Webapplikation für den Einsatz am Lehrstuhr IT in Produktion und Logistik der Technischen Universität Dortmund gut geeignet, da sie einwandfrei funktionieren würde selbst wenn sie sämtliche 171 Teilnehmer des Fachs Informationsaustausch produzierender Unternehmen gleichzeitig nutzen würden. Somit ist die Stabilität der Webapplikation gewährleistet. Erst ab einer Benutzeranzahl von 300 überschreitet die Antwortzeit eine Sekunde und wird somit als langsam eingestuft. Um die Belastungsgrenzen der Webapplikation zu ermitteln, werden weitere Lasttests durchgeführt. Dazu werden jeweils 1000 und 2000 Benutzer simuliert, die die Webapplikation eine Stunde lang parallel nutzen. Die Ergebnisse dieser Tests sind in der Tabelle 4.2.4 dargestellt.

**Tabelle 4.3.4** – Belastungsgrenzen der Webapplikation in Abhängigkeit der Benutzeranzahl

| **Anzahl Benutzer** | **Anzahl Anfragen** | **Anzahl Fehler** | **∅ Antwortzeit [Ms]** |
|---|---|---|---|
| 1000 | 28.110 | 740 | 3.476 |
| 2000 | 57.596 | 1.508 | 11.503 |

Bei diesen hohen Benutzeranzahlen steigt die Anzahl der auftretenden Fehler. Jedoch treten selbst bei 2000 parallelen Benutzern lediglich bei 2,5% der Anfragen Fehler auf. Die durchschnittliche Antwortzeit vervielfacht sich bei den hohen Benutzerzahlen. So beträgt sie bei 1000 parallelen Benutzern mehr als drei Sekunden. Diese langsame Antwortzeit wird von dem Benutzer der Webapplikation zwar als langsam, jedoch nicht als unbedienbar empfunden. Bei 2000 parallelen Benutzern ist die Belastungsgrenze der Webapplikation erreicht, da die Antwortzeit 10 Sekunden überschreitet und somit als unbedienbar eingestuft wird. Um eine derart hohe Anzahl von parallelen Benutzern zu handhaben, ohne dass die Antwortzeit zu langsam wird, wäre es erforderlich die Architektur der Webapplikation anzupassen. So wäre beispielsweise der Einsatz mehrerer Webserver sinnvoll, welche die Anfragen von einem Lastenverteiler zugeordnet bekommen (Rohr, 2018). Da Benutzerzahlen von 1000 oder 2000 für diese Arbeit nicht realistisch sind, ist eine Änderung der Architektur der Webapplikation nicht erforderlich.

Wie bereits erläutert bezieht sich die Antwortzeit lediglich auf die Zeit, die der Webserver zur Bereitstellung der vom Webbrowser angefragten Informationen und Ressourcen benötigt. Um das Verhalten der Webapplikation im Webbrowser zu untersuchen, wir der Dienst PageSpeed Insights der Google LLC verwendet (*PageSpeed Insights*, 2020). Das Testergebnis ist in der folgenden Abbildung 4.2.1 dargestellt.

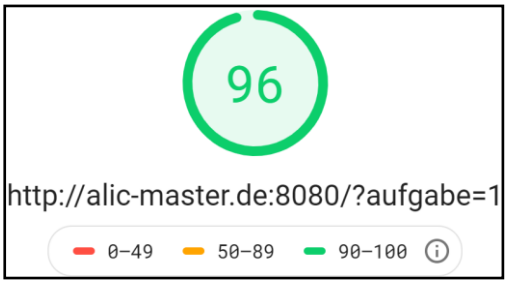


**Abbildung 4.3.4** – Testergebnis PageSpeed Insights der Google LLC

Die Webapplikation erreicht 96 von 100 möglichen Punkten und befindet sich somit im sehr guten Bereich. Der Test prüft unter anderem die Zeit, die ein Webbrowser zum Rendern der Webapplikation benötigt. Außerdem wird auch die Geschwindigkeit zur Ausführung der Skripte der Webapplikation geprüft. Insgesamt ergibt der Systemtest der Webapplikation, dass die Webapplikation eine sehr gute Leistung aufweist.

## 4.3 Fazit

In diesem Abschnitt werden die Ergebnisse dieser Arbeit zusammengeführt. Dabei wird geprüft, ob die Ziele dieser Arbeit erfüllt worden sind. Das Hauptziel dieser Arbeit ist die Konzeption und Entwicklung einer datenbankbasierten Webapplikation gewesen. Diese soll als digitales Lernwerkzeug zum Thema SQL dienen und im Übungsbetrieb des Masterstudiums am Lehrstuhl IT in Produktion und Logistik der Technischen Universität Dortmund eingesetzt werden. Die Webapplikation sollte dabei mehrbenutzerfähig sein und den Studierenden die Möglichkeit bieten eigene SQL Statements einzugeben und auszuführen. Das Forschungsthema ist in einer Arbeitsgruppe behandelt worden. Dabei ist die Konzeption und Entwicklung des Backends der Webapplikation Gegenstand dieser Arbeit. Die Konzeption und Entwicklung des Frontends, ist in der Arbeit von O. Alić thematisiert worden (Alić, 2021). Um das Hauptziel zu erfüllen, ist es notwendig gewesen zwei Unterziele inklusive derer Anforderungen zu erfüllen.
Das erste Unterziel ist dabei die Ausarbeitung einer geeigneten Architektur der Webapplikation gewesen. Um dieses Ziel zu erfüllen, ist das Grob- und Feindesign der Webapplikation konzipiert worden. Das Grobdesign der Webapplikation enthält Informationen zu den einzelnen Komponenten der Webapplikation und zu den eingesetzten Technologien, die für die Umsetzung der Komponenten verwendet worden sind. So basiert die Architektur auf der 3-Tier-Architektur und setzt sich aus einem Webbrowser, einem Webserver und einer Datenbank zusammen. Der Webserver bildet mit der Datenbank, welche ein Bestandteil eines Datenbankmanagementsystems ist, das Backend der Webapplikation, wogegen der Webbrowser das Frontend der Webapplikation bildet. Das Frontend kommuniziert mit dem Backend mithilfe von HTTP-Anfragen, welche sich nach den standardisierten REST-Leitsätzen richten. Darüber hinaus kommuniziert das Frontend zusätzlich über WebSockets mit dem Backend, um Informationen zu angemeldeten Benutzern der Webapplikation in Echtzeit zu übertragen. Für die Entwicklung des Webservers ist die Open-Source-JavaScript-

Laufzeitumgebung Node.js eingesetzt worden. In Node.js sind die möglichen HTTP-Anfragen, die der Webserver bearbeiten können muss, programmiert worden. Außerdem ist die Anbindung an die Datenbank im Webserver implementiert worden. Um die Latenz zwischen dem Webserver und der Datenbank zu minimieren, befinden sich beide Komponenten auf demselben physikalischen Server. Das eingesetzte Datenbankmanagementsystem ist PostgreSQL. PostgreSQL bietet die einfache Anbindung über eine Socket-Schnittstelle an, über welche der Webserver SQL Statements zum Abfragen und Manipulieren der Datenbank übertragen kann. Die erforderlichen Datenbanktabellen für das Aufgaben- und Benutzermanagement sind in dieser Arbeit konzipiert und in PostgreSQL angelegt worden. Nachdem das Grobdesign der Webapplikation ausgearbeitet worden ist, wurde das Feindesign konzipiert. Das Feindesign umfasst dabei die Softwaremodule der Webapplikation. Die Softwaremodule sind anhand von Zustandsdiagrammen veranschaulicht worden. Insgesamt sind sieben Softwaremodule konzipiert worden. Drei Softwaremodule sind dabei dafür konzipiert worden die Mehrbenutzerfähigkeit der Webapplikation zu ermöglichen. Hat sich ein registrierter Benutzer an der Webapplikation angemeldet, so legt die Webapplikation temporäre Datenbanktabellen für den Benutzer an. Um die Effizienz der Webapplikation zu erhöhen, werden diese temporären Datenbanktabellen gelöscht, sobald der Benutzer die Webapplikation verlässt. Der Benutzer kann nach der erfolgreichen Anmeldung insgesamt 19 Aufgaben zum Thema SQL bearbeiten. Das Softwaremodul „Eingabeprüfung“ bildet den Kern der Webapplikation. Mithilfe dieses Softwaremoduls ist dem Benutzer der Webapplikation die Möglichkeit gegeben worden, eigene SQL Statements auszuführen. Diese SQL Statements werden an die Datenbank geleitet und auf Fehler überprüft. Führt der Benutzer ein fehlerhaftes SQL Statement aus, so erhält er eine Fehlermeldung mit einem Hinweis zu der Fehlerursache. Die Fehlerursachen werden dabei in semantische und syntaktische Fehler unterschieden. Gelingt es einem Benutzer nicht eigenständig die Lösung zu einer Aufgabe zu erarbeiten, kann er sich die Lösung zu jeder Aufgabenstellung ausgeben lassen. Hat der Benutzer Schwierigkeiten bei der Bedienung der Webapplikation, so kann er sich über die Hilfe-Funktion eine Anleitung zu der Webapplikation anzeigen lassen. In weiteren Softwaremodulen sind die Vorgänge, die zur Darstellung der Webapplikation im Webbrowser erforderlich sind, beschrieben worden. Das Unterziel der Ausarbeitung einer geeigneten Architektur der Webapplikation ist somit erfüllt worden.

Nachdem die Architektur der Webapplikation definiert worden ist, wurde die Webapplikation in Kooperation mit O. Alić implementiert. Daraufhin ist das zweite Unterziel behandelt worden. Dabei handelt es sich um die Validierung der Implementierung durch geeignete Testfälle. So sind in dieser Arbeit Komponententests, Integrationstests und Systemtests durchgeführt worden. Die Komponententests haben dazu gedient die einzelnen Softwaremodule isoliert auf ihre Funktionalität zu prüfen. Dazu sind Testfälle definiert worden, welche anschließend getestet wurden. Insgesamt sind 212 Testfälle untersucht worden. Nach einem Testdurchlauf sind Anpassungen an

dem Programmcode durchgeführt worden, um entdeckte Fehler zu beseitigen. Anschließend sind weitere Testdurchläufe und Programmcodeanpassungen durchgeführt worden, bis sämtliche Testfälle ein positives Ergebnis aufgewiesen haben. Daraufhin sind Integrationstests durchgeführt worden, um das Zusammenspiel zwischen den Komponenten auf Korrektheit zu prüfen. Dazu sind ebenfalls Testfälle formuliert worden. Diese Testfälle sind daraufhin geprüft worden. Die aufgetretenen Fehler sind anschließend behoben worden. Nachdem die Komponententests und Integrationstests abgeschlossen worden sind, sind die Systemtests durchgeführt worden. Für die Systemtests ist die Webapplikation auf einem Testserver installiert worden. Die Webapplikation ist anschließend auf Interoperabilität, Stabilität und Robustheit geprüft worden. Durch Lasttests sind die aufgeführten Testpunkte validiert worden. Für die Lasttests ist ein Simulationsprogramm unter Verwendung der Skriptsprache Python entwickelt worden. Mithilfe des Simulationsprogramms ist es möglich gewesen eine Vielzahl an virtuellen Benutzern zu simulieren, welche die Webapplikation parallel benutzen. Dabei sind mehrere Testdurchläufe getätigt worden, bei denen die Anzahl der virtuellen Benutzer variiert worden ist. So konnte das Verhalten der Webapplikation in Abhängigkeit der Anzahl an parallelen Benutzern untersucht werden. Bis zu einer Anzahl von 200 parallelen Benutzern, sind sämtliche Anfragen der Benutzer fehlerfrei verarbeitet und beantwortet worden. Bei einer Anzahl von 300 Benutzern, die die Webapplikation parallel benutzt haben sind 16 von 427.857 Anfragen mit einem Fehler beantwortet worden. Dabei hat die Antwortzeit des Webservers eine für ein Timeout definierte Zeit überschritten. Selbst diese wenigen Fehler haben sich durch das Warten des Benutzers selbst behoben, da die Webapplikation die Anfragen lediglich verspätet beantwortet hat. Da im Produktivbetrieb voraussichtlich maximal 171 Benutzer Zugriff zu der Webapplikation besitzen werden, die die Webapplikation mit geringer Wahrscheinlichkeit gleichzeitig nutzen werden, ist die Stabilität und Robustheit der Webapplikation gewährleistet. Um die Belastungsgrenzen der Webapplikation zu ermitteln sind weitere Lasttests mit unrealistischen Benutzerzahlen durchgeführt worden. Erst ab einer Anzahl von 2000 Benutzern, die die Webapplikation parallel benutzen, ist die Webapplikation als unbedienbar eingestuft worden. Somit ist auch das zweite Unterziel dieser Arbeit erfüllt worden. Mit der Erfüllung der beiden Unterziele unter Berücksichtigung der aufgeführten Aspekte ist auch das Hauptziel erreicht worden.

# 5 Zusammenfassung und Ausblick

In dieser Arbeit ist das Backend zu einer datenbankbasierten Webapplikation konzipiert und entwickelt worden. Diese soll als digitales Lernwerkzeug zum Thema SQL dienen und im Übungsbetrieb des Masterstudiums am Lehrstuhl IT in Produktion und Logistik der Technischen Universität Dortmund eingesetzt werden. Das Frontend der Webapplikation ist dabei von O. Alić konzipiert und Entwickelt worden (Alić, 2021).
In dem Kapitel 2 sind die informationstechnischen Grundlagen, die für die Entwicklung des Backends einer Webapplikation erforderlich sind, vorgestellt worden. So sind allgemeine Architekturansätze behandelt worden, welche für moderne Webapplikationen eingesetzt werden. Darüber hinaus ist die Thematik der relationalen Datenbanken behandelt worden, wobei auf das Datenbankmanagementsystem PostgreSQL näher eingegangen worden ist. Darüber hinaus sind die Grundlagen der Datenbanksprache SQL aufbereitet worden. Es sind Technologien vorgestellt worden, die für die Umsetzung der Architektur der Webapplikation verwendet werden können. Außerdem sind die möglichen Programmschnittstellen zusammengestellt worden. Dazu ist grundlegendes Wissen zu der Kommunikation zwischen Computersystem über ein Netzwerk anhand des TCP/IP-Netzwerkmodells vermittelt worden.
In dem Kapitel 3 ist die Konzeption der Architektur der Webapplikation und die Einrichtung und Entwicklung des Backends thematisiert worden. So ist das Grob- und Feindesign der Webapplikation ausgearbeitet worden. Das Grobdesign basiert auf der 3-Tier-Architektur, welche sich aus einem Webbrowser, Webserver und einer Datenbank zusammensetzt. Für die Entwicklung des Backends ist die Open-Source-JavaScript-Laufzeitumgebung eingesetzt worden. Als Datenbankmanagementsystem ist PostgreSQL verwendet worden. Die Programmschnittstellen zwischen den drei aufgeführten Komponenten sind ausgearbeitet worden. So ist die Kommunikation von dem Webbrowser zum Webserver über zwei unterschiedliche Anwendungsschicht-protokolle des TCP/IP-Netzwerkmodells realisiert worden. Zum einen sendet der Webbrowser HTTP-Anfragen an den Webbrowser, wobei die REST-Leitsätze befolgt werden. Zum anderen kommunizieren der Webbrowser und der Webserver über WebSockets um Echtzeitinformationen auszutauschen. Der Webserver bildet gemeinsam mit der Datenbank das Backend der Webapplikation, wogegen der Webbrowser das Frontend der Webapplikation repräsentiert. Die Kommunikation zwischen dem Webserver und der Datenbank ist über eine Socket-Schnittstelle umgesetzt worden. Nachdem das Grobdesign der definiert worden ist, wurde das Feindesign der Webapplikation ausgearbeitet. Dazu sind insgesamt sieben Softwaremodule konzipiert worden, welche anhand von Zustandsdiagrammen veranschaulicht worden sind. Die Softwaremodule enthalten die Informationen über die Zustände, Ereignisse und Aktionen, welche in den einzelnen Softwaremodulen auftreten können. So wird die Mehrbenutzerfähigkeit mithilfe der Softwaremodule „Registrierung“,

„Anmeldung“ und „Passwort vergessen“ gewährleistet. Die Softwaremodule „Seite laden“ und Seitennavigation“ haben die Aufgabe die Webapplikation im Webbrowser darzustellen. So kann der Benutzer insgesamt 19 Übungsaufgaben zum Thema SQL bearbeiten. Sollte die Webapplikation für einen Benutzer nicht intuitiv verständlich sein, so kann er sich mit der Hilfe-Funktion eine Anleitung anzeigen lassen. Falls der Benutzer die Lösung zu einer Aufgabe nicht eigenständig erarbeiten kann, kann er sich diese ausgeben lassen. Das Modul „Eingabeprüfung“ bildet die Kernfunktion der Webapplikation ab. So kann der Benutzer eigene SQL Statements ausführen lassen. Diese Benutzereingabe wird an die Datenbank weitergeleitet und auf Fehler geprüft. Ist die Benutzereingabe fehlerbehaftet, so erhält der Benutzer eine Fehlermeldung mit einem Hinweis auf die Fehlerursache. Nachdem die Architektur der Webapplikation ausgearbeitet worden ist, wurde die Webapplikation in Kooperation mit O. Alić implementiert.

In dem Kapitel 4 sind die Softwaretests der Webapplikation durchgeführt worden, um die Implementierung der Webapplikation zu validieren. Dazu sind Komponententests, Integrationstests und Systemtests durchgeführt worden. Bei den Komponententests sind die Softwaremodule isoliert getestet worden. Dabei sind insgesamt 212 Testfälle untersucht worden. Nach der Entdeckung von Fehlern, sind Anpassungen des Programmcodes erfolgt, woraufhin erneut getestet worden ist. Dieses Vorgehen ist so oft wiederholt worden, bis sämtliche Testfälle in einem positiven Ergebnis resultiert sind. Nach der Durchführung der Komponententest, sind Integrationstests durchgeführt worden. Dabei ist geprüft worden, ob das Zusammenspiel der einzelnen Komponenten so funktioniert, wie es im Feindesign der Architektur der Webapplikation definiert worden ist. Entdeckte Fehler sind durch Anpassungen des Programmcodes korrigiert worden. Abschließend sind Systemtests durchgeführt worden. Dazu ist die Webapplikation auf einem Testserver installiert worden. Um die Webapplikation auf Interoperabilität, Stabilität und Robustheit zu testen, ist ein Simulationsprogramm unter Verwendung der Skriptsprache Python entwickelt worden. Das Simulationsprogramm lässt virtuelle Benutzer parallel auf die Webapplikation zugreifen. Die Durchführung von Lasttests hat ergeben, dass die Webapplikation bei bis zu 200 gleichzeitig agierenden Benutzern sehr gute Ergebnisse erzielt. Aus diesem Grund ist die Webapplikation für die Produktivnutzung am Lehrstuhl IT in Produktion und Logistik an der Technischen Universität Dortmund als geeignet eingestuft worden. Erst bei einer Benutzerzahl von 2000 Benutzern ist die Belastungsgrenze der Webapplikation erreicht worden. Somit hat die Webapplikation sämtliche Anforderungen erfüllt.

Computergestützte Lernwerkzeuge finden im Zuge der zunehmenden Digitalisierung eine wachsende Anwendung in sämtlichen Bereichen der Gesellschaft. Darunter zählt auch der Bereich der Bildung (Schimank, 2013). Durch die Anwendung computergestützter Lernwerkzeuge kann die Lehre an Schulen und Universitäten orts- und zeitunabhängig gestaltet werden. Insbesondere zu der Zeit der aktuellen COVID-19 Pandemie haben sich Bildungseinrichtungen als Flexibel erwiesen, welche eine

Infrastruktur für das E-Learning bereits vor der Pandemie ausgebaut haben. Bildungseinrichtung ohne solch eine Infrastruktur haben dabei weniger Möglichkeiten den Lehrbetrieb ohne Nachteile fortzuführen (Claudia Ricci, 2020). Aus diesem Grund ist es vor allem jetzt wichtig, die Digitalisierung im Bereich der Bildung zu fördern. Die in dieser Arbeit konzipierte und entwickelte Webapplikation ist ein Beitrag zur Förderung der Digitalisierung im Bereich der Bildung. Die Webapplikation deckt jedoch lediglich ein Themengebiet ab. Dennoch können große Teile des Programmcodes für die Entwicklung weiterer Webapplikation für weitere Themengebiete genutzt werden. So können die Darstellung, das Benutzermanagement und das Aufgabenmanagement der Webapplikation übernommen werden. Um die Webapplikation für andere Themengebiete zu nutzen, können die Aufgabenstellungen und die Prüfmechanismen dank des modularen Aufbaus der Webapplikation gezielt angepasst werden. Außerdem kann die entwickelte Webapplikation erweitert werden. Eine sinnvolle Erweiterung wäre zum Beispiel eine dynamische Aufgabenverwaltung, welche es Dozenten ermöglicht, neue Aufgabenstellungen zu definieren ohne den Programmcode anzupassen. Eine weitere denkbare Erweiterung der Webapplikation wäre ein Prüfungsmodus. Solch ein Prüfungsmodus könnte die Prüfungsbewertung in kurzer Zeit automatisiert durchführen. Neben den aufgeführten Erweiterungsmöglichkeiten existiert noch eine Vielzahl an Möglichkeiten für den Einsatz von computergestützten Lernwerkzeugen.

# Literaturverzeichnis

Alić, O. (2021). *Frontend zu einer datenbankbasierten Webapplikation für den Einsatz in der digitalen Lehre*. Dortmund.

Avci, O. (Hg.). (2003). *Web-Programmierung: Softwareentwicklung mit Internet-Technologien - Grundlagen, Auswahl, Einsatz - XHTML & HTML, CSS, XML, JavaScript, VBScript, PHP, ASP, Java* (1. Aufl.). Vieweg. Wiesbaden.

Bass, L., Clements, P. & Kazman, R. (2013). *Software architecture in practice* (3. Aufl.). *Always learning*. Addison-Wesley/Pearson. Upper Saddle River.

Bewersdorff, J. (2018). *Objektorientierte Programmierung mit JavaScript: Direktstart für Einsteiger* (2. Auflage). Springer Vieweg. Wiesbaden.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. & Stal, M. (2013). *Pattern-Oriented Software Architecture, A System of Patterns* (1. Aufl.). *Wiley Software Patterns Series*. Wiley. New York.

Chlebek, P. (2006). *User-Interface-orientierte Softwarearchitektur: Bauentwurfslehre für interaktive Softwareoberflächen ; Kompass für die Entwicklung dialogintensiver Anwendungen ; Leitfaden für erlebbare User-Interfaces* (1. Aufl.). Vieweg. Wiesbaden.

Claudia Ricci. (2020). *Die Corona-Pandemie als Digitalisierungsbooster*. https://blog.iao.fraunhofer.de/die-corona-pandemie-als-digitalisierungsbooster/*2020, Okt 25.

Dunkel, J. & Holitschke, A. (2003). *Softwarearchitektur für die Praxis*. *Xpert.press*. Springer. Berlin.

Elmasri, R. & Navathe, S. (2004). *Fundamentals of database systems* (Internat. ed., 4. ed.). Pearson Addison-Wesley. Boston.

Fuchs, E. (2018). *SQL - Grundlagen und Datenbankdesign* (5. Ausgabe). Bodenheim.

Gorski, P. L., Lo Iacono, L. & Nguyen, H. V. (2015). *WebSockets: Moderne HTML5-Echtzeitanwendungen entwickeln* (1. Aufl.). Hanser. München.

Jakob, M. (2011). *Model-based engineering of web applications : the flashWeb method*. Stuttgart.

Jarzyna, D. (2013). *TCP/IP: Grundlagen, Adressierung, Subnetting* (1. Aufl.). mitp Verl.-Gruppe Hüthig Jehle Rehm. Hamburg.

Jin, B., Shevat, A. & Sahni, S. (2018). *Designing Web APIs: Building APIs that developers love* (First edition). O'Reilly Media. Sebastopol.

Kleuker, S. (2013). *Grundkurs Software-Engineering mit UML: Der pragmatische Weg zu erfolgreichen Softwareprojekten* (3. Aufl.). *Lehrbuch*. Springer Vieweg. Wiesbaden.

Ladel, S., Knopf, J. & Weinberger, A. (Hg.). (2018). *Digitalisierung und Bildung*. Springer VS. Wiesbaden.

Lienemann, G. & Larisch, D. (2014). *TCP/IP - Grundlagen und Praxis: Protokolle, Routing, Dienste, Sicherheit* (2., aktualisierte Aufl.). Heise. Hamburg.

Lombardi, A. (2015). *WebSocket: Lightweight client-server communications* (First edition). O'Reilly Media. Sebastopol.

Loshin, P. (2003). *TCP/IP clearly explained* (4th ed.). Morgan Kaufmann Publishers. Amsterdam.

Mertens, P., Bodendorf, F., König, W., Picot, A., Schumann, M. & Hess, T. (2012). *Grundzüge der Wirtschaftsinformatik* (11. Aufl.). *Springer-Lehrbuch*. Springer Gabler. Berlin.

Müller, J. (2005). *Workflow-based Integration: Grundlagen, Technologien, Management ; mit 21 Tabellen*. *Xpert.press*. Springer. Berlin.

Niegemann, H. M., Hessel, S., Hochscheid-Mauel, D., Aslanski, K., Deimann, M. & Kreuzberger, G. (2004). *Kompendium E-Learning*. *X.media.press*. Springer. Berlin.

*PageSpeed Insights*. https://developers.google.com/speed/pagespeed/insights/?hl=de*2020, Dez 30.

Pollard, B. (2019). *HTTP/2 in action*. New York.

The PostgreSQL Global Development Group. (2020). *PostgreSQL 13.1 Documentation*. https://www.postgresql.org/files/documentation/pdf/13/postgresql-13-A4.pdf*2020, Nov 05.

Riggs, S. & Ciolli, G. (2018). *PostgreSQL 10 Administration Cookbook* (1st edition). Packt Publishing. Birmingham.

Rohr, M. (2018). *Sicherheit von Webanwendungen in der Praxis: Wie sich Unternehmen schützen können - Hintergründe, Maßnahmen, Prüfverfahren und Prozesse* (2., vollständig überarbeitete und aktualisierte Auflage). Springer Vieweg. Wiesbaden.

Schimank, U. (2013). *Gesellschaft*. *Einsichten Soziologische Themen - Themen der Soziologie*. Transcript-Verl. Bielefeld.

Seeboth, A. (22. November 2020). *Wie viele Webseiten gibt es?* https://optimondo.de/news/wie-viele-webseiten-gibt-es/*2020, Nov 22.

Shiflett, C. (2003). *HTTP developer's handbook*. *Developer's library*. Sams. Indianapolis.

Socket.IO. (2020). *Get started*. https://socket.io/get-started/chat/*2020, Sep 24.

Spichale, K. (2019). *API-Design: Praxishandbuch für Java- und Webservice-Entwickler* (2. Auflage). dpunkt.verlag. Heidelberg.

Spillner, A. & Linz, T. (2012). *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester ; Foundation Level nach ISTQB-Standard* (5., überarbeitete und aktualisierte Auflage). dpunkt.verlag GmbH. Heidelberg.

Sriparasa, S. S. (2013). *JavaScript and JSON essentials: Successfully build advanced JSON-fueled web applications with this practical, hands-on guide*. *Community experience distilled*. Packt Pub. Birmingham.

Steiner, R. (2017). *Grundkurs relationale Datenbanken: Einführung in die Praxis der Datenbankentwicklung für Ausbildung, Studium und IT-Beruf* (9. Aufl.). *Lehrbuch*. Springer Vieweg. Wiesbaden.

Steyer, R. (2020). *JavaScript Grundlagen: Interaktive Webseiten mit JavaScript erstellen* (4. Ausgabe, März 2020). HERDT. Bodenheim.

Studer, T. (2019). *Relationale Datenbanken: Von den theoretischen Grundlagen zu Anwendungen mit PostgreSQL*. Springer Verlag Berlin. Berlin.

Tata Consultancy Services Deutschland GmbH. (2020). *Gelassen zur Digitalisierung: Wie sich deutsche Unternehmen in der neuen Zeit orientieren: Die Trendstudie von Tata Consultancy Services (TCS) und Bitkom Research*. https://downloads.studie-digitalisierung.de/2019/de/Trendstudie_TCS_2019_Bericht_DE.pdf*2020, Okt 25.

Trick, U. & Weber, F. (2009). *SIP, TCP/IP und Telekommunikationsnetze: Next generation networks und VoIP konkret* (4., überarb. und erw. Aufl.). Oldenbourg. München.

Unhelkar, B. (2018). *Software Engineering with UML* (First edition). CRC Press. Boca Raton.

*Why Server Response Time Is Important*. https://www.hrank.com/why-server-response-time-is-important*2020, Dez 30.

Witte, F. (2016). *Testmanagement und Softwaretest: Theoretische Grundlagen und praktische Umsetzung*. Springer Vieweg. Wiesbaden.

Wong, C. (2000). *HTTP Pocket Reference: Hypertext Transfer Protocol*. *Pocket Reference (O'Reilly)*. O'Reilly Media. Sebastopol.

# Anhang

**Tabelle A.1** – Testfälle zum Softwaremodul „Registrierung“

| **Testkomponente: Registrierung** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Befüllen der Eingabefelder | Beispiel Eingabe | Eingabe möglich | Eingabe möglich | OK |
| 2 | Button Klick "Registrieren" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 3 | Übernahme der Inhalte der Eingabefelder | Beispiel Inhalte | Inhalte der Eingabefelder | Inhalte der Eingabefelder | OK |
| 4 | Hover über Button "Registrieren" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 5 | Prüfung, ob Eingabefeld "UniMail" gefüllt ist | 1) UniMail | OK | OK | OK |
| 6 | Prüfung, ob Eingabefeld "UniMail" gefüllt ist | 2) keine Eingabe | Bitte UniMail eingeben | UniMail nicht im korrekten Format | NOK |
| 7 | Prüfung, ob Eingabefeld "Passwort" gefüllt ist | 1) Passwort | OK | OK | OK |
| 8 | Prüfung, ob Eingabefeld "Passwort" gefüllt ist | 2) keine Eingabe | Bitte Passwort eingeben | Bitte Passwort eingeben | OK |
| 9 | Prüfung, ob die Form der UniMail korrekt ist | 1) korrekte UniMail | OK | OK | OK |
| 10 | Prüfung, ob die Form der UniMail korrekt ist | 2) Nicht korrekte UniMail | UniMail nicht im korrekten Format | UniMail nicht im korrekten Format | OK |
| 11 | Prüfung, ob Passwörter übereinstimmen | 1) Gleiche Passwörter | OK | OK | OK |
| 12 | Prüfung, ob Passwörter übereinstimmen | 2) Ungleiche Passwörter | Passwörter stimmen nicht überein | Passwörter stimmen nicht überein | OK |
| 13 | Inhalte der Eingabefelder werden zum Webserver übertragen | Beispielinhalt | Ausgabe des Beispielinhalts am Webserver | Ausgabe des Beispielinhalts am Webserver | OK |
| 14 | Prüfung, ob UniMail registriert | 1) Nicht registrierte UniMail | OK | OK | OK |
| 15 | Prüfung, ob UniMail registriert | 2) registrierte UniMail | UniMail bereits registriert | UniMail bereits registriert | OK |
| 16 | Entnahme des Vor- und Nachnamen aus der UniMail | UniMail | Vor- und Nachname | Vor- und Nachname | OK |
| 17 | Benutzerdaten in die Datenbank schreiben | Benutzerdaten | Benutzerdaten in Datenbank geschrieben | Benutzerdaten in Datenbank geschrieben | OK |
| 18 | Sende Registrierungs- | Registrierungsbestätigung | Registrierungs-bestätigung im | keine Email erhalten | NOK |

| | | | | | |
|---|---|---|---|---|---|
| | bestätigung an UniMail | | Postfach der UniMail | | |
| 19 | Aufruf des Pop-Up Fensters mit der Registrierungsbestätigung | Registrierungsbestätigung | Aufruf des Pop-Up Fensters | Aufruf des Pop-Up Fensters | OK |

**Tabelle A.2** – Testfälle zum Softwaremodul „Anmeldung“

| **Testkomponente: Anmeldung** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Befüllen der Eingabefelder | Beispiel Eingabe | Eingabe möglich | Eingabe möglich | OK |
| 2 | Button Klick "Anmelden!" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 3 | Hover über Button "Anmelden!" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 4 | Übernahme der Inhalte der Eingabefelder | Beispiel Inhalte | Inhalte der Eingabefelder | Inhalte der Eingabefelder | OK |
| 5 | Prüfung, ob Eingabefeld "UniMail" gefüllt ist | 1) UniMail | OK | OK | OK |
| 6 | Prüfung, ob Eingabefeld "UniMail" gefüllt ist | 2) keine Eingabe | Bitte UniMail eingeben | UniMail nicht im korrekten Format | NOK |
| 7 | Prüfung, ob Eingabefeld "Passwort" gefüllt ist | 1) Passwort | OK | OK | OK |
| 8 | Prüfung, ob Eingabefeld "Passwort" gefüllt ist | 2) keine Eingabe | Bitte Passwort eingeben | Bitte Passwort eingeben | OK |
| 9 | Prüfung, ob die Form der UniMail korrekt ist | 1) korrekte UniMail | OK | OK | OK |
| 10 | Prüfung, ob die Form der UniMail korrekt ist | 2) Nicht korrekte UniMail | UniMail nicht im korrekten Format | UniMail nicht im korrekten Format | OK |
| 11 | Inhalte an den Webserver übertragen | Beispielinhalt | Ausgabe des Beispielinhalts am Webserver | Ausgabe des Beispielinhalts am Webserver | OK |
| 12 | UserID in der DB ermitteln | UniMail | UserID zur UniMail | UserID zur UniMail | OK |
| 13 | Passwort in der DB ermitteln | UniMail | Passwort zur UniMail | Passwort zur UniMail | OK |
| 14 | Prüfung, ob Passwörter übereinstimmen | 1) Gleiche Passwörter | OK | OK | OK |
| 15 | Prüfung, ob Passwörter übereinstimmen | 2) Ungleiche Passwörter | Falsches Passwort | Falsches Passwort | OK |
| 16 | Prüfung, ob UniMail registriert ist | 1) Nicht registrierte UniMail | UniMail ist nicht registriert | UniMail ist nicht registriert | OK |
| 17 | Prüfung, ob UniMail registriert ist | 2) registrierte UniMail | OK | OK | OK |
| 18 | Zeitstempel der Anmeldung in DB schreiben | Erfolgreiche Anmeldung | Zeitstempel in der DB | Zeitstempel in der DB | OK |
| 19 | Prüfung, ob die UserID in den LocalStorage geschrieben wird | Erfolgreiche Anmeldung | UserID im LocalStorage | UserID im SessionStorage | NOK |

**Tabelle A.3** – Testfälle zum Softwaremodul „Eingabeprüfung“

| **Testkomponente: Eingabeprüfung** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Übertragung Benutzereingabe an Webserver | Benutzereingabe | Ausgabe am Webserver | Ausgabe am Webserver | OK |
| 2 | DB-Tabellennamen durch Namen der temporären Tabelle ersetzen | DB-Tabellenname | Name der temporären DB-Tabelle | Name der temporären DB-Tabelle | OK |
| 3 | Eingabe an DB leiten | Beispieleingabe | DB führt Beispieleingabe aus | DB führt Beispieleingabe aus | OK |
| 4 | Semantischer Fehler: leere Ausgabe erkennen | SQL Statement, dass in leerer Ausgabe resultiert | Semantischer Fehler: leere Ausgabe | Semantischer Fehler: leere Ausgabe | OK |
| 5 | Semantischer Fehler: falsche Ausgabe erkennen | SQL Statement, dass in falscher Ausgabe resultiert | Semantischer Fehler: falsche Ausgabe | Semantischer Fehler: falsche Ausgabe | NOK |
| 6 | Syntaxfehler erkennen | SQL Statement mit Syntaxfehler | Syntaxfehler | Syntaxfehler | OK |
| 7 | Lexikalischen Fehler erkennen | SQL Statement mit lexikalischem Fehler | Lexikalischer Fehler | Lexikalischer Fehler | OK |
| 8 | Korrektes SQL Statement erkennen | Korrektes SQL Statement | OK | OK | OK |
| 9 | Ergebnis in JSON-Format zur Verfügung stellen | HTTP-GET Anfrage | Ergebnis in JSON-Format | Ergebnis in JSON-Format | OK |
| 10 | Erwartetets Ergebnis in JSON-Format zur Verfügung stellen | HTTP-GET Anfrage | Erwartetes Ergebnis in JSON-Format | Erwartetes Ergebnis in JSON-Format | OK |
| 11 | Aufruf Pop-Up "Falsche Antwort" | Falsche Antwort | Aufruf des Pop-Up Fensters | Aufruf des Pop-Up Fensters | OK |
| 12 | Aufruf Pop-Up "Richtige Antwort" | Richtige Antwort | Aufruf des Pop-Up Fensters | Aufruf des Pop-Up Fensters | OK |
| 13 | Hover über Button "Zurück" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 14 | Hover über Button "Nächste Aufgabe" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 15 | Button Klick "Zurück" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 16 | Button Klick "Nächste Aufgabe" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |

**Tabelle A.4** – Testfälle zum Softwaremodul „Seite laden“

| **Testkomponente: Seite laden** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Index.html wird bereitgestellt | Aufruf der Webapplikation im Browser | Darstellung der index.html | Darstellung der index.html | OK |
| 2 | Verbindung zum WebSocket-Kanal wird geöffnet | Webapplikation wird im Browser geladen | Verbunden | Verbunden | OK |
| 3 | Prüfung, ob UserID im LocalStorage ist | UserID im LocalStorage | OK | keine UserID hinterlegt | NOK |
| 4 | Prüfung, ob UserID im LocalStorage ist | leeres LocalStorage | keine UserID hinterlegt | keine UserID hinterlegt | OK |
| 5 | Benutzerdaten aus DB auslesen | UserID | Benutzerdaten | Benutzerdaten | OK |
| 6 | Aufgabenstllungen im JSON-Format zur Verfügung stellen | HTTP GET Request | Daten im Browser einsehbar | Daten im Browser einsehbar | |
| 7 | Aufabenstellung und Aufgabentabelle zu Aufgabe 1a) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 8 | Aufabenstellung und Aufgabentabelle zu Aufgabe 1b) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 9 | Aufabenstellung und Aufgabentabelle zu Aufgabe 1c) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 10 | Aufabenstellung und Aufgabentabelle zu Aufgabe 2a) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 11 | Aufabenstellung und Aufgabentabelle zu Aufgabe 2b) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 12 | Aufabenstellung und Aufgabentabelle zu Aufgabe 2c) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 13 | Aufabenstellung und Aufgabentabelle zu Aufgabe 3a) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Flasche Aufgabenstellung | NOK |
| 14 | Aufabenstellung und Aufgabentabelle zu Aufgabe 3b) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 15 | Aufabenstellung und Aufgabentabelle zu Aufgabe 4a) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 16 | Aufabenstellung und Aufgabentabelle zu Aufgabe 4b) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |

| | | | | | |
|---|---|---|---|---|---|
| 17 | Aufabenstellung und Aufgabentabelle zu Aufgabe 4c) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 18 | Aufabenstellung und Aufgabentabelle zu Aufgabe 5a) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 19 | Aufabenstellung und Aufgabentabelle zu Aufgabe 5b) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 20 | Aufabenstellung und Aufgabentabelle zu Aufgabe 5c) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 21 | Aufabenstellung und Aufgabentabelle zu Aufgabe 6a) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Flasche Aufgabenstellung | NOK |
| 22 | Aufabenstellung und Aufgabentabelle zu Aufgabe 6b) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 23 | Aufabenstellung und Aufgabentabelle zu Aufgabe 6c) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 24 | Aufabenstellung und Aufgabentabelle zu Aufgabe 6d) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |
| 25 | Aufabenstellung und Aufgabentabelle zu Aufgabe 6e) | URI der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | Aufgabenstellung und Aufgabentabelle der Aufgabe | OK |

**Tabelle A.5** – Testfälle zum Softwaremodul „Seitennavigation“

| **Testkomponente: Seitennavigation** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Button Klick "Hilfe" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 2 | Bereitstellung der Hilfe.pdf | HTTP GET Anfrage | Darstellung der Hilfe.pdf im Browser | Darstellung der Hilfe.pdf im Browser | OK |
| 3 | Button Klick "Abmelden" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 4 | UserID aus LocalStorage löschen | Button Klick "Abmelden" | UserID aus LocalStorage gelöscht | UserID aus LocalStorage gelöscht | OK |
| 5 | Button Klick "Ausführen" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 6 | Mehrzeilige Eingabe | Mehrzeilige Eingabe | OK | OK | OK |
| 7 | Groß- und Kleinschreibung möglich | Eingabe in Großbuchstaben | OK | OK | OK |
| 8 | Groß- und Kleinschreibung möglich | Eingabe in Kleinbuchstaben | OK | OK | OK |
| 9 | Groß- und Kleinschreibung möglich | Eingabe in Groß- und Kleinbuchstaben | OK | OK | OK |
| 10 | Button Klick "Lösung" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 11 | Lösung zu Aufgabe 1a) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 12 | Lösung zu Aufgabe 1b) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 13 | Lösung zu Aufgabe 1c) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 14 | Lösung zu Aufgabe 2a) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 15 | Lösung zu Aufgabe 2b) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 16 | Lösung zu Aufgabe 2c) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 17 | Lösung zu Aufgabe 3a) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Falsche Lösung im Eingabefeld | NOK |
| 18 | Lösung zu Aufgabe 3b) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 19 | Lösung zu Aufgabe 4a) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 20 | Lösung zu Aufgabe 4b) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 21 | Lösung zu Aufgabe 4c) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 22 | Lösung zu Aufgabe 5a) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 23 | Lösung zu Aufgabe 5b) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 24 | Lösung zu Aufgabe 5c) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |

| | | | | | |
|---|---|---|---|---|---|
| 25 | Lösung zu Aufgabe 6a) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Falsche Lösung im Eingabefeld | NOK |
| 26 | Lösung zu Aufgabe 6b) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 27 | Lösung zu Aufgabe 6c) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 28 | Lösung zu Aufgabe 6d) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 29 | Lösung zu Aufgabe 6e) im Eingabefeld eintragen | Button Klick "Lösung" | Lösung im Eingabefeld | Lösung im Eingabefeld | OK |
| 30 | Hover über Button "Aufgabe 1a)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 31 | Hover über Button "Aufgabe 1b)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 32 | Hover über Button "Aufgabe 1c)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 33 | Hover über Button "Aufgabe 2a)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 34 | Hover über Button "Aufgabe 2b)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 35 | Hover über Button "Aufgabe 2c)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 36 | Hover über Button "Aufgabe 3a)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 37 | Hover über Button "Aufgabe 3b)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 38 | Hover über Button "Aufgabe 4a)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 39 | Hover über Button "Aufgabe 4b)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 40 | Hover über Button "Aufgabe 4c)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 41 | Hover über Button "Aufgabe 5a)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 42 | Hover über Button "Aufgabe 5b)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 43 | Hover über Button "Aufgabe 5c)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 44 | Hover über Button "Aufgabe 6a)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 45 | Hover über Button "Aufgabe 6b)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 46 | Hover über Button "Aufgabe 6c)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 47 | Hover über Button "Aufgabe 6d)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 48 | Hover über Button "Aufgabe 6e)" | Hover | Button-Farbe ändert sind | Button-Farbe ändert sind | OK |
| 49 | Button Klick "Aufgabe 1a)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 50 | Button Klick "Aufgabe 1b)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 51 | Button Klick "Aufgabe 1c)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 52 | Button Klick "Aufgabe 2a)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |

| | | | | | |
|---|---|---|---|---|---|
| 53 | Button Klick "Aufgabe 2b)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 54 | Button Klick "Aufgabe 2c)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 55 | Button Klick "Aufgabe 3a)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 56 | Button Klick "Aufgabe 3b)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 57 | Button Klick "Aufgabe 4a)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 58 | Button Klick "Aufgabe 4b)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 59 | Button Klick "Aufgabe 4c)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 60 | Button Klick "Aufgabe 5a)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 61 | Button Klick "Aufgabe 5b)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 62 | Button Klick "Aufgabe 5c)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 63 | Button Klick "Aufgabe 6a)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 64 | Button Klick "Aufgabe 6b)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 65 | Button Klick "Aufgabe 6c)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 66 | Button Klick "Aufgabe 6d)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 67 | Button Klick "Aufgabe 6e)" | Klick | Klick-Ereignis ausgelöst | Klick-Ereignis ausgelöst | OK |
| 68 | SQL Schlüsselwort SELECT hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 69 | SQL Schlüsselwort FROM hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 70 | SQL Schlüsselwort WHERE hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 71 | SQL Schlüsselwort DELETE hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 72 | SQL Schlüsselwort INSERT hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 73 | SQL Schlüsselwort INTO hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 74 | SQL Schlüsselwort VALUES hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 75 | SQL Schlüsselwort GROUP hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 76 | SQL Schlüsselwort BY hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 77 | SQL Schlüsselwort JOIN hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 78 | SQL Schlüsselwort INNER JOIN hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 79 | SQL Schlüsselwort OUTER JOIN hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |

| | | | | | |
|---|---|---|---|---|---|
| 80 | SQL Schlüsselwort ON hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 81 | SQL Schlüsselwort ASC hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 82 | SQL Schlüsselwort DESC hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 83 | SQL Schlüsselwort DISTINCT hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | NOK |
| 84 | SQL Schlüsselwort UPDATE hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 85 | SQL Schlüsselwort SET hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 86 | SQL Schlüsselwort OR hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 87 | SQL Schlüsselwort AND hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 88 | SQL Schlüsselwort LIKE hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | NOK |
| 89 | SQL Schlüsselwort COUNT hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 90 | SQL Schlüsselwort SUM hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 91 | SQL Schlüsselwort ORDER hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 92 | SQL Schlüsselwort DROP hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 93 | SQL Schlüsselwort ROUND hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 94 | SQL Schlüsselwort MIN hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 95 | SQL Schlüsselwort MAX hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 96 | SQL Schlüsselwort NOT hervorgehoben | Schlüsselwort | Schlüsselwort hervorgehoben | Schlüsselwort hervorgehoben | OK |
| 97 | Berechtigungseinschränkung für Schlüsselwort DROP | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | NOK |
| 98 | Berechtigungseinschränkung für Schlüsselwort TRUNCATE | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 99 | Berechtigungseinschränkung für Schlüsselwort CREATE | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 100 | Berechtigungseinschränkung für Schlüsselwort REPLACE | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 101 | Berechtigungseinschränkung für Schlüsselwort PROCEDURE | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 102 | Berechtigungseinschränkung für Schlüsselwort EXEC | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 103 | Berechtigungseinschränkung für Schlüsselwort INDEX | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |

| | | | | | |
|---|---|---|---|---|---|
| 104 | Berechtigungseinschränkung für Schlüsselwort CONSTRAINT | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 105 | Berechtigungseinschränkung für Schlüsselwort ADD | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 106 | Berechtigungseinschränkung für Schlüsselwort ALTER | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 107 | Berechtigungseinschränkung für Schlüsselwort BACKUP | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 108 | Berechtigungseinschränkung für Schlüsselwort REVOKE | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 109 | Berechtigungseinschränkung für Schlüsselwort PASSWORD | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 110 | Berechtigungseinschränkung für Schlüsselwort BENUTZER | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 111 | Berechtigungseinschränkung für Schlüsselwort AUFGABEN | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 112 | Berechtigungseinschränkung für Schlüsselwort STUDIERENDE2a | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 113 | Berechtigungseinschränkung für Schlüsselwort STUDIERENDE2b | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 114 | Berechtigungseinschränkung für Schlüsselwort STUDIERENDE2c | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 115 | Berechtigungseinschränkung für Schlüsselwort STUDIERENDE3b | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |
| 116 | Berechtigungseinschränkung für Schlüsselwort STUDIERENDENLISTE4c | Schlüsselwort | Sie sind für diesen Befehl nicht berechtigt | Sie sind für diesen Befehl nicht berechtigt | OK |

**Tabelle A.6** – Testfälle zum Softwaremodul „Verbindungsabbruch“

| **Testkomponente: Verbindungsabbruch** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Schließen der Verbindung zum WebSocket-Kanals wird erkannt | Browser schließen | Rückmeldung, dass die Verbindung zum WebSocket-Kanal geschlossen wurde | Rückmeldung, dass die Verbindung zum WebSocket-Kanal geschlossen wurde | OK |
| 2 | Temporäre Datenbanktabellen löschen | Browser schließen | Temporäre DB-Tabellen werden gelöscht | Temporäre DB-Tabellen werden gelöscht | OK |

**Tabelle A.7** – Testfälle „Anmeldung“ ←→ „Registrierung“

| **Anmeldung ←→ Registrierung** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Button Klick "Noch nicht registriert?" | Klick | Aufruf Modul "Registrierung" | Aufruf Modul "Registrierung" | OK |
| 2 | Button Klick „Zur Anmeldung“ | Klick | Aufruf Modul „Anmeldung“ | Aufruf Modul „Anmeldung“ | OK |

**Tabelle A.8** – Testfälle „Anmeldung“ ←→ „Passwort vergessen“

| **Anmeldung ←→ Passwort vergessen** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | Button Klick "Passwort vergessen" | Klick | Aufruf Modul "Passwort vergessen" | Aufruf Modul "Passwort vergessen" | OK |
| 2 | Button Klick „Zur Anmeldung“ | Klick | Aufruf Modul „Anmeldung“ | Aufruf Modul „Anmeldung“ | OK |

**Tabelle A.9** – Testfälle „Seite laden“ → „Seitennavigation“

| **Seite laden → Seitennavigation** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | UserID im LocalStorage | Aufruf des Moduls "Seite laden", mit einer UserID im LocalStorage | Aufruf Modul "Seitennavigation" | Aufruf Modul "Seitennavigation" | OK |

**Tabelle A.10** – Testfälle „Seite laden“ → „Anmeldung“

| **Seite laden → Anmeldung** | | | | | |
|---|---|---|---|---|---|
| **ID** | **Testfall** | **Eingabe** | **Erwartete Ausgabe** | **Tatsächliche Ausgabe** | **E** |
| 1 | UserID nicht im LocalStorage | Aufruf des Moduls "Seite laden", ohne dass eine UserID im LocalStorage gespeichert ist | Aufruf Modul "Anmeeldung" | Aufruf Modul "Anmeldung" | OK |