Entwicklung einer webbasierten Client-Server Anwendung zur Unterstützung von interaktiven Unterrichtsmethoden

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor of Science

an der HTW Berlin

Hochschule für Technik und Wirtschaft Berlin

Fachbereich Informatik, Kommunikation und Wirtschaft

internationaler Studiengang Medieninformatik

Eingereicht von

Jannes Julian Brunner

geb. 21.06.1991

Eingereicht am:	05.08.2019
Betreuender Hochschuldozent:	Prof. Dr. Gefei Zhang
Zweitgutachter:	Prof. Dr.-Ing. Kai Uwe Barthel

Abstract

Im Zuge der Digitalisierung sind Bildungseinrichtungen mit neuartigen, nie dagewesen Problemen konfrontiert. Wie können sowohl bereits bewährte, als auch neue Unterrichtsmethoden sinnvoll durch digitale Technik unterstützt werden? Wie kann der finanzielle Rahmen von zur Verfügung stehenden Mittel optimal genutzt werden? Ist Datenschutz gewährleistet?

Viele Anbieter setzen auf Cloud-Lösungen, welche eine Internetanbindung zur Nutzung obligatorisch macht. Schulen mit noch ausbaufähiger digitaler Infrastruktur benötigen jedoch skalierbaren Hard- und Softwarelösungen, die zur Not auch offline einsetzbar sind, um auch zeitnah kosteneffizient digitale Technik im Unterrichtsalltag zu integrieren.

Im Rahmen dieser Arbeit wurde die gegenwärtige Situation in Sachen Digitalisierung hinsichtlich Soft- und Hardware sowie dem damit verbundenen Einsatz von Softwareanwendungen zu Unterstützung von interaktiven Unterrichtsmethoden analysiert. Anschließend darauf aufbauend ist eine Client-Server Web-Anwendung zur Lösung implementiert worden, welche in erster Linie offline funktioniert, wenig Anforderungen an die technische Infrastruktur stellt, aber sich auch in ausgebauter Umgebung sinnvoll nutzen lässt.

Inhaltsverzeichnis

Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ARPANET	Advanced Research Projects
CA	Certificate Authority
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICT	Interactive course teaching
IoC	Inversion of Control
IoT	Internet of things
ISO	International Standardization Organisation
JS	JavaScript
JSON	JavaScript Object Notation
LAN	Local Area Network
NPM	Node Packet Manager
ORM	Object Relationship Mapper
OSI	Open System Interconnection
REST	Representational State Tranfer
RIA	Rich Internet Application
SaaS	Software-as-a-Service
SchulG	Schulgesetz
SDN	Software-defined Networking
SML	Standard Generalized Markup Language
SOAP	Simple Object Access Protocoll
SQL	Structured Query Language
SSH	Secure Shell
URL	Uniform Resource Locator
WAF	Web-Application-Framework
WASM	WebAssembly
WDSL	Web Services Description Language
WLAN	Wireless Local Area Network
WS	WebSocket
WWW	World Wide Web
XML	Extensible Markup Language

1 Einführung

1.1 Motivation

Bildung ist ein wichtiges Element der Persönlichkeitsentwicklung und unter Artikel 26 der allgemeinen Erklärung der Menschenrechte als solches definiert. Ohne Bildung ist das Ausüben eines gewählten Berufes und das Entwickeln einer Meinung zu komplexen Sachverhalten unmöglich [1]. Heute sieht sich Bildung durch den digitalen Wandel der letzten Jahre sich noch nie vorher dagewesenen Problemen gegenübergestellt. Wie können Lehrende an Schulen digitale Technik effizient und preiswert im Unterricht einsetzen und so neue Bildungskonzepte erfolgreich in den Lehrplan integrieren?

Ursprünglich bezeichnet der Begriff Digitalisierung das Umwandeln von Analog nach Digital. Wurde früher Musik auf Schallplatten vertrieben, so wurde diese von der Compact Disc vom Markt verdrängt, welche die Musik auf kleinerem Raum digital abspeichert. Auch wenn der Begriff im Zusammenhang mit Schule längst nicht mehr das Ursprüngliche meint, halte Ich es für sehr wichtig, früher dagewesene Unterrichtskonzepte nicht einfach zu digitalisieren, sondern es erfordert ein Neudenken. Bewährte pädagogische Methoden sollten durch Digitalisierung profitieren sowie neue Konzepte müssen erforscht und entwickelt werden.

1.2 Ist-Zustand

Am 04.04.2019 trat die Änderung des Artikel 104c des Grundgesetz für die Bundesrepublik Deutschland in Kraft und ebnete so den Weg für den von Bund und Ländern beschlossenen Digitalpakt Schule [2]. Dieser Beschluss macht deutlich, dass digitale Kompetenz im Bildungssektor von hoher Bedeutung ist, was von einer Förderungssumme von mindestens 5,5 Milliarden Euro unterstrichen wird. Legt man diese Summe auf die ca 40.000 Schulen um, erhält jede Schule einen Durchschnittsbeitrag von 137.000 Euro. Bei ca. 11 Millionen Schülerinnen und Schülern ergibt dies eine Förderungssumme von ca. 500 Euro pro Schülerin bzw. Schüler. Einer der Hauptförderungspunkte des Digitalpakt Schule sieht den Ausbau der technischen Infrastruktur an deutschen Schulen vor, z.B. Bereitstellung von drahtlosen Netzwerken, schnellen Internetzugangspunkten und digitalen Unterrichtsmedien wie interaktiven Whiteboards.

Nach dem Bundesministerium für Bildung und Forschung (BMBF) fördert kein digitales Medium alleine gute Bildung, sondern immer dahinterstehende pädagogische Konzepte aus einer Vielfalt von Angeboten sind entscheiden [3]. Ergänzend dazu kritisiert Dennis Horn (Experte für digitale Themen der ARD) den zu starken Fokus auf Hardware und mahnt an, dass zu wenig darüber gesprochen wurde, wie diese denn auch sinnvoll genutzt werden kann [4].

Diese Kritikpunkte wurden auch auf der Podiumsdiskussion der *re:publica 2018* - „Was kommt in den digitalen Schulranzen?“ angeschnitten. Tobias Hübner, Lehrer und Autor im Bereich Medienistik, zeigt dort ebenfalls auf, dass der Wille Geld auszugeben zu begrüßen sei, es aber an Konzepten und Materialien mangele. Als Lehrer würde er den Investitionsfokus auf Lehrerfortbildung setzen.

Besonders beliebte Hardware an Schulen ist der populäre Tablet Computer *iPad* der Firma Apple inc. Er wird u.A. an der Oberschule Gehrden bei Hannover von Schülerinnen und Schülern genutzt, wird hier aber von Eltern finanziert [5]. In der günstigsten Variante liegen die Kosten bereits bei mindestens 449€ [6] (Stand April 2019), was schon knapp 90% des Förderungsvolumens pro Schülerin und Schüler ausmachen würde. Als günstigere Alternative ist der Einplatinencomputer *Raspberry Pi* zu nennen (siehe Abbildung 1.1), welcher bereits für 33 Euro erwerblich ist (Stand April 2019) und genug Rechenkapazitäten bereitstellt um zahlreiche Projekte im Bildungsbereich durchzuführen.

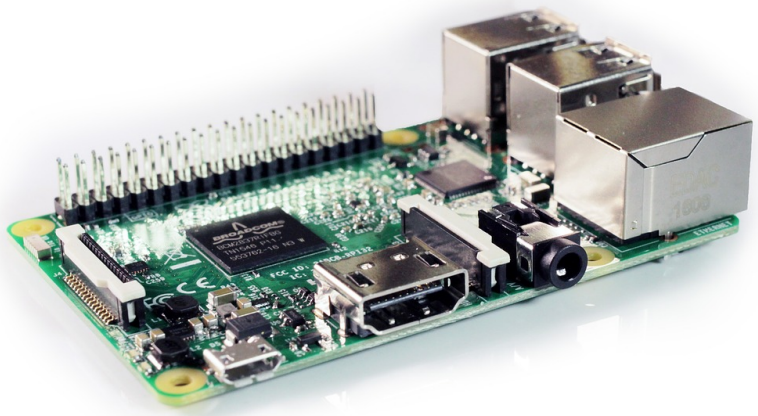

Abb. 1.1: Der Raspberry Pi 3 - Einplantinencomputer [7]

Mit Touchscreenmodul und Schutzhülle liegt der Preis insgesamt bei ca. 150 Euro, was immer noch weniger als die Hälfte des Fördervolumens beträgt.

Besuch der Grundschule am Rüdesheimer Platz Berlin Im Rahmen der Vorrecherche zu dieser Arbeit wurde einem Unterrichtstag in einer Jahrgangsübergreifenden (JüL) Klasse 1 bis 3 an der Grundschule am Rüdesheimer Platz beigewohnt um ein differenzierteres Bild der gegenwärtigen Lern- und Digitalisierungssituation an einer Berliner Schule zu bekommen. An dieser Stelle eine große Dankaussagung an Frau Wewer, Grundschullehrerin, welche diese Erfahrung möglich gemacht hat und in einem anschließenden Gespräch das Interesse an einer kostengünstigen und einfach nutzbaren Lösung zur Unterstützung von interaktiven Unterrichtsmethoden unterstrichen hat.

1.3 Aufbau der Arbeit

Im Anschluss an dieses Kapitel folgt die Formulierung einer **Zielsetzung**. Darauf folgend werden **Grundlagen** erörtert. Dies umfasst die Themengebiete Digitalisierung an Schulen und einen Überblick über Webtechnologie. Ersteres ist für den späteren potentiellen Einsatz der Software maßgebend, letzteres bildet das technologische Fundament, welches die Implementierung erst möglich macht. Anschließend wird in Kapitel **Analyse** ein Vergleich zwischen existierenden kommerziellen und nichtkommerziellen Plattformen gezogen. Darauf aufbauend folgt eine Anforderungs- und Systembeschreibung. Im darauffolgenden Kapitel **Konzept** wird ebendieses erörtert und darauffolgend der Prozess der **Implementierung** beschrieben. In einer folgenden **Auswertung** werden die Ergebnisse mit den geplanten Zielen verglichen und ein Fazit gezogen. Schlussendlich wird im letzten Abschnitt ein **Ausblick** formuliert, welcher die Zukunft des Projekts betrifft.

Genderhinweis: Aus Gründen der Lesbarkeit wurde in dieser Arbeit die männliche Form gewählt, nichtsdestoweniger beziehen sich die Angaben auf Angehörige beider Geschlechter.

2 Zielsetzung

Das Ziel der Arbeit ist es, eine auf Webtechnologien basierende Softwarelösung zur Unterstützung von interaktiven Unterrichtsmethoden zu entwickeln.
Dabei soll auch untersucht werden, ob eine gänzlich vom Internet unabhängige Lösung umsetzbar ist. Sollte keine ausreichende Netzwerkinfrastruktur vorhanden sein, soll diese von der Software selbst in Form eines drahtlosen Netzwerks eingerichtet und gestellt werden. Dies könnte Schulen mit langsamer oder gar nicht vorhandene Internetanbindung und Netzwerkinfrastruktur den Einsatz von digital gestützten Unterrichtsmethoden erleichtern.

Darauf aufbauend sollen die Mindesthardwareanforderungen gering gehalten werden. Ein niedriger Einrichtungspreis könnte den Rahmen der im *Digitalpakt Schule* fließenden Gelder effizienter ausschöpfen und Schulen mehr finanzielle Flexibilität einräumen. Ebenso soll den Nutzenden der Software die Handhabung einfach gemacht werden, um auch technisch weniger Versierten die Nutzung zu ermöglichen. Die zu entwickelnde Software soll Benutzerkomponenten für Lehrende sowie Schülerinnen und Schüler bieten. Darüber hinaus soll eine optional nutzbare angepasste Darstellungsoption der Software für Klassenräumen implementiert werden, wie z.B. Projektoren, Fernseher u.Ä. Lehrenden soll damit die Möglichkeit gegeben werden, vorhandene Hardware im Klassenraum flexibel einsetzen zu können. Die soll Schülerinnen und Schüler eine visuell ergonomische Nutzung ermöglichen. Im Rahmen der Entwicklung soll generell auf gute Gebrauchstauglichkeit (*Usability*) und ein gutes Nutzungserlebnis (*User Experience*) geachtet werden.

Diese Arbeit widmet sich in diesem Zusammenhang der Umsetzung von Software, welche diese Unterrichtsmethoden abbildet. Damit verbundene pädagogische Aspekte werden nur im Rahmen der Softwareentwicklung beleuchtet.
Der Begriff Unterrichtsmethoden meint in dieser Arbeit Lerninhalte, welche sich digital gestützt im Rahmen einer Unterrichtssituation von Dozierenden mit Teilnehmenden interaktiv durchführen lassen, wie z.B. Quiz-Anwendungen, Brainstorming, Memory-Spiele etc. Weiterführend hierzu sei an dieser Stelle der Methodenpool von *Netzwerk Digitale Bildung* [8] genannt, welcher unter folgender Webadresse erreichbar ist:
`www.netzwerk-digitale-bildung.de/information/methoden/methodenpool/`

3 Grundlagen

Dieses Kapitel soll einen grundlegenden Überblick über die zwei wichtigsten Themengebiet dieser Arbeit bieten: Digitalisierung an Schulen und Webtechnologie.

3.1 Digitalisierung an Schulen

Die folgenden drei Abschnitte sollen einen Einblick und eine Momentaufnahme über den Stand der Digitalisierung an deutschen Schulen (Stand 2018/2019) und den möglichen Potential von digital gestützten interaktiven Unterrichtsmethoden bieten. Ebenso wird das Thema Datenschutz an Schulen näher betrachtet.

3.1.1 Momentaufnahme

Laut einer aktuellen Studie von *Citrix*, sind deutsche Schüler mit Abstand am schlechtesten ausgestattet, was Technologie im Unterricht angeht [9]. Die Studie hat dabei einen direkten Vergleich zwischen den vier europäischen Ländern Frankreich, Großbritannien, Niederlande und Deutschland gezogen und pro Land mehr als 1000 Schülerinnen und Schüler befragt (Niederlande 500). 22% gaben an, gar keine Technologie im Unterricht einzusetzen, die über das Anzeigemedium Projektor hinausgeht. Innovative Technologien wie der im Abschnitt 1.2 erwähnte Einplantinencomputer *Raspberry Pi*, mit denen u.A. IoT-Projekte umgesetzt werden können, stehe nur 13% der Schülerinnen und Schülern an deutschen Schulen zur Verfügung.

Oftmals ist der normale Zustand an einer Schule jener, dass ein IT-interessierter Lehrer oder sogar der Hausmeister selbst, administrative Aufgaben die Schul-IT betreffend übernimmt, was an einem Fachpersonalmangel festzumachen sei, so Ralf Koenzen, Gründer und Geschäftsführer der *LANCOM Systems GmbH* im Fachartartikel IT-Infrastrukturen an Schulen: „Von der Kreidezeit ins digitale Zeitalter“ [10]. Darüber hinaus seien funktionstüchtige Projektoren und Computer vielerorts Mangelware ebenso das Funknetzwerk (WLAN) nur begrenzt, wenn überhaupt, verfügbar. Dies wird auch durch die Erhebung in Abbildung 3.1 über Internet- und WLAN Verfügbarkeit in Klassenräumen gestützt.

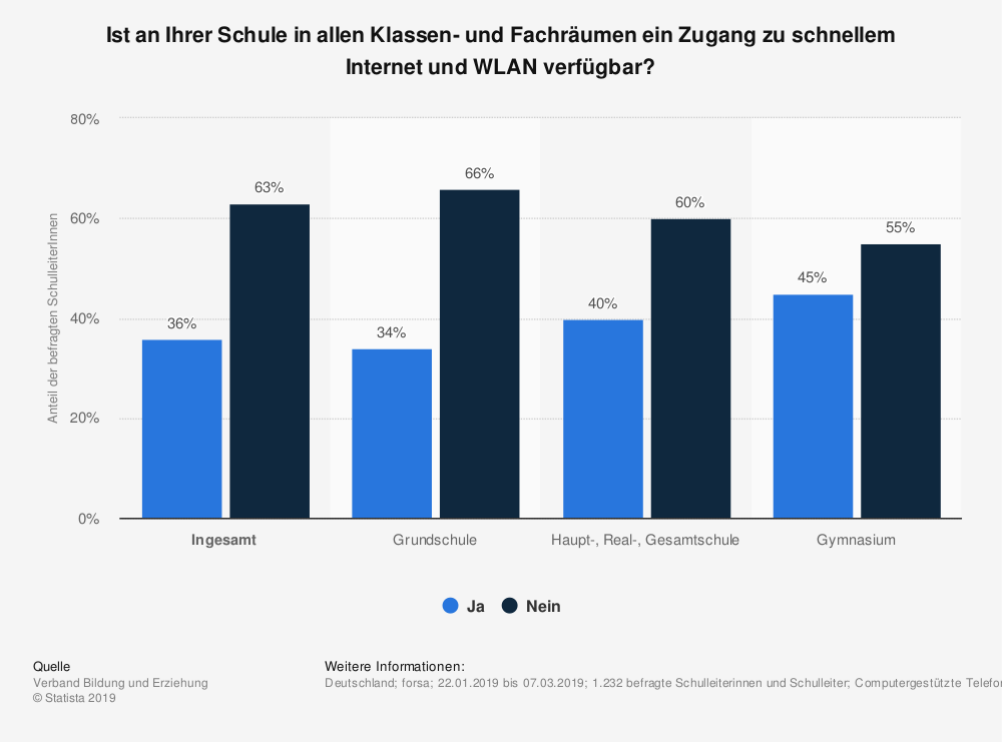

Abb. 3.1: Verfügbarkeit von schnellem Internet und WLAN in Klassenräumen: Bundesweiten Erhebung zum Thema Digitalisierung an allgemeinbildenden Schulen in Deutschland. [11]

Allerdings scheinen die ersten Barrieren durch den, in der Einleitung dieser Arbeit bereits erwähnten, Digitalpakt Schule zu fallen. Dieser sieht vor fünf Millarden Euro in die IT-Ausstattung der deutschen Schulen fliesen zu lassen. Eine Schule mit mehr als tausend Schülern und entsprechendem Lehrkörper steht der Anforderungskomplexität an IT-Systeme eines größeren Wirtschaftsunternehmen kaum nach. Eine Ausnahme bilden hier Schulen, die auf professionelle Betreuung durch ein Systemhaus oder eigene Netzwerktechniker setzen [10].

Eine interessante Alternative könnte hier das Nutzen von Cloud-Technologie sein. Cloud-basierte Netzwerkmanagementlösungen und Software-defined Networking (SDN) ist im Wirtschaftssektor bereits auf dem Vormarsch zu sein. Diese Technologien könnte Schulen dabei unterstützen den Digitalisierungsfortschritt voranzutreiben. Hierbei werden notwendige infrastrukturelle Geräte wie Access Points, Router, Switches und die nötige Verkabelung direkt vor Ort in der Schule installiert. Die Betreuung und Wartung erfolgt jedoch höchstmöglich automatisiert mit geringerem Aufwand aus der Ferne.

3.1.2 Ausblick digital gestützte interaktive Unterrichtsmethoden

Die erwähnte Technik im Abschnitt 3.1 macht den Einsatz von digital gestützten Interaktiven Unterrichtsmethoden möglich. Der online Lernvideo Anbieter Sofatutor hat im Jahr 2016 auf dem Educamp Leipzig Lehrerinnen und Lehrer über Software befragt, welche diese erfolgreich in ihren Unterricht integriert haben [12]. Neben zahlreicher Software, welche der Unterrichtsvorbereitung dient, lässt sich eine umfangreiche Liste in der Sektion „Interaktion“ finden. Es lassen sich verschiedene Aufgabenformen ausmachen, die an einer digitalen Tafel von den Schülern gelöst werden sollen. Dies umfasst z.B. das Markieren, Sortieren, Zuordnen (Paare finden) von Bildern, Multiple-Choice Aufgaben, Brainstorming, Quiz-Anwendungen, u.v.m. Generell lässt sich feststellen, dass die Palette von Anwendungsmöglichkeiten enorm ist. Viele klassische Konzepte, die sich bereits analog interaktiv durchführen lassen konnten, stehen auch in einem digitalem Pendant bereit. Beispielsweise lassen sich Unterrichtsmethoden wie ein Lern-Quiz auch mit Papier und Stift durchführen, digitale Technik kann hier jedoch viel Arbeit abnehmen und lässt die Ausführung der Unterrichtsmethode deutlich immersiver und medial interaktiver zu. Abseits spielerischer Szenarien, kann z.B. die Mathematik Software *GeoGebra* das Visualisieren von mathematischen Zusammenhängen. So Auswirkungen von anderen Werten gezeigt werden. Das Konzept „bring-your-own-device“, welches vorsieht, dass Teilnehmende eigene Geräte mitbringen, wie z.B. ein Smartphone, steigert das Maß von Interaktivität zusätzlich. So ist es möglich, dass eine ganze Lerngruppe oder Schulklasse an einer interaktiven Unterrichtsmethode simultan teilnimmt. Ebenso können Dozierende in einer Prüfungssituation auf Software zurückgreifen, welche die Prüfung der Teilnehmenden unterstützt und eine anschließende Auswertung der Antworten wesentlich automatisiert. An vielen Universitäten wird die Software *Moodle* für diesen Zweck eingesetzt. Zusammenfassend lässt sich feststellen, dass das Potential von digital gestützten Unterrichtsmethoden deutlich gegeben und das Angebot sowie das Potenzial der Anwendung vielfältig ist. Eine bewusste Einbettung in den Unterricht kann Dozierende unterstützen und entlasten. Als positiven Nebeneffekt lässt sich das Steigern von digitalen Kompetenzen seitens der Schülern vermerken, welche im Zuge der immer fortschreitenden weltweiten Digitalisierung eine nicht zu unterschätzende Fähigkeit ist.

3.1.3 Datenschutz an Schulen

Seit dem 25. Mai 2018 gilt die neue EU-Datenschutzverordnung, welche für alle Personen, Behörden oder sonstigen Stellen anzuwenden ist, wenn personenbezogene Daten verarbeitet werden. Dies betrifft also auch Schulen und ist sofern nichts neues, da die Datenverarbeitung und Auskunftsrechte in §64 des SchulG (Schulgesetz) im Falle des Bundeslands Berlin geregelt ist und dieser weiterhin anzuwenden ist. Neu ist allerdings, dass die Verantwortlichen für die Verarbeitung von personenbezogenen Daten weitere Aspekte berücksichtigen müssen, welche die neue Datenschutzverordnung mit sich bringt. Weiterführend sei hierzu die Quelle [13, Datenschutz in der Schule] zu nennen. In diesem Zusammenhang ist es wichtig, dass auch eingesetzte Software an Schulen zu diesen Bestimmungen kompatibel sein muss, wenn diese personenbezogene Daten verarbeitet. Im Jahr 2018 an der Düsseldorfer Gemeinschaftsschule haben Unklarheiten um den Schutz von Schülerdaten dafür gesorgt, dass die Zeugnisse der rund 300 Schülerinnen und Schüler wieder per Hand geschrieben wurden. Auch die im vorherigen Abschnitt genannten Bring-your-own-device Praxis befindet sich Stand 2018 noch in einer rechtlichen Grauzone, sollten personenbezogene Daten verarbeitet werden[14]. Die im Abschnitt 3.1.1 genannte Auslagerung in eine Cloud könnte hier ebenfalls helfen, wenn der Cloud-Anbieter EU-Datenschutzverordnung konform arbeitet. Zur Einhaltung der gesamten Datensicherheit, ist es auf jeden Fall ratsam, wenn Cloud-Anbieter und Server ihren Standort in Deutschland haben. Verlassen Daten alternativ gar nicht erst das Schulgelände, trägt dies ebenso positiv zum Erhalt des Datenschutzes bei.

3.2 Überblick Webtechnologie

In den folgenden Untersektionen 3.2.1 ff. wird ein Überblick über Webtechnologie und Web-Softwareentwicklung gegeben.

3.2.1 Popularität

Seit dem Erfolgskurs des Web 2.0[1] in den frühen 2000er Jahren, zeichnet sich zunehmend der Trend des Software-as-a-Service Geschäftsmodells ab. Dies beschreibt die Bereitstellung von Software im Internet oder durch ein lokal laufenden Servers, ohne dass Benutzende die Software selbst noch lokal installiert haben müssen. Im Jahr 2015 setzten bereits über drei Viertel von 102 befragten Unternehmen Software dieser Form aktiv im Geschäft ein [15]. Viele Arten von Software können mittlerweile in einer im Webbrowser lauffähigen Alternative substituiert werden. Ein populäres Beispiel ist die Office-Suite *Google Docs* der Firma *Google inc.* Hier lassen sich Textverarbeitung, Tabellenkalkulation und das erstellen von Präsentationen ohne Installation und direkt im Webbrowser des Benutzenden ausführen. Ein anderes Beispiel ist die Web-Software *Photopea* welche ebenfalls komplett im Web-Browser ausgeführt wird und dem nur lokal installiert ausführbaren Bildbearbeitungsprogramm *Photoshop* der Firma *Adobe inc.* sehr nahe kommt. Im Vergleich zu lokal installierter Software ist die Bereitstellung von Web-Software einfacher, da solange ein moderner Webbrowser lauffähig ist, das Betriebssystem des Client-Computers zu vernachlässigen ist. Ebenso stellt potente Hardware keine zwingende Voraussetzungen, da etwaige rechenintensive Aufgaben auf der Serverseite getätigt werden können oder hier eine Balance zwischen Client und Server angestrebt werden kann.

3.2.2 Intranet und Internet

Einfach ausgedrückt, ist das Internet ein Netzwerk von Computern, welche weltweit miteinander vernetzt sind. Seine Anfänge lassen sich auf das Ende der 1960er in den USA datieren, als die DARPA (Defense Advanced Research Projects Agency) eine weltweite Verknüpfung von Datennetzen anstrebte. Das hier draus resultierende ARPANET (Advanced Research Projects) kann als Ursprung angesehen werden. Dabei beschreibt der Begriff Internet streng genommen ein 'interconnected network', also ein international vernetztes Netzwerk, ohne dabei die Hardware- und Netzwerktechnologie genauer zu beschreiben [16].

[1] Web 2.0 ist ein Schlagwort, das für eine Reihe interaktiver und kollaborativer Elemente des Internets, speziell des World Wide Webs, verwendet wird. Dabei konsumiert der Nutzer nicht nur den Inhalt, er stellt als Prosument selbst Inhalte zur Verfügung. - Wikipedia.org

Die wohl populärste Anwendung des Internets ist das World Wide Web, welche gegen das Jahr 1989 von einer Forschungsgruppe rund um Sir Tim Berners-Lee ins Leben gerufen wurde und heute oftmals als Synonym für das gesamte Internet sprachlich genutzt wird.

In unser heutigen globalisierten Welt lässt sich das Internet mitsamt World Wide Web nicht mehr wegdenken und ist ein integraler Bestandteil der Kultur und Struktur des weltweiten Informationsaustausches.

Das Intranet beschreibt analog dazu ein lokal abgeschlossenes Netzwerk von Computern, bspw. innerhalb eines Unternehmens. Dabei endet ein Intranet an seinen Grenzen und ein Gateway fungiert als Übergabepunkt ins Internet (siehe Abbildung 3.2). Die Vernetzung der Endgeräte erfolgt kabelgebunden (LAN) oder kabellos (WLAN). Die Kommunikationsgeschwindigkeit innerhalb eines Intranets sind i.d.R. deutlich höher als im Internet, da Daten nicht erst nach außen an einen Internet Service Provider übermittelt werden müssen. Ein Intranet funktioniert unabhängig vom öffentlichem Internet (erhöhte Ausfallsicherheit), ist nicht öffentlich zugänglich und bietet oft andere oder zusätzliche Funktionen [17].

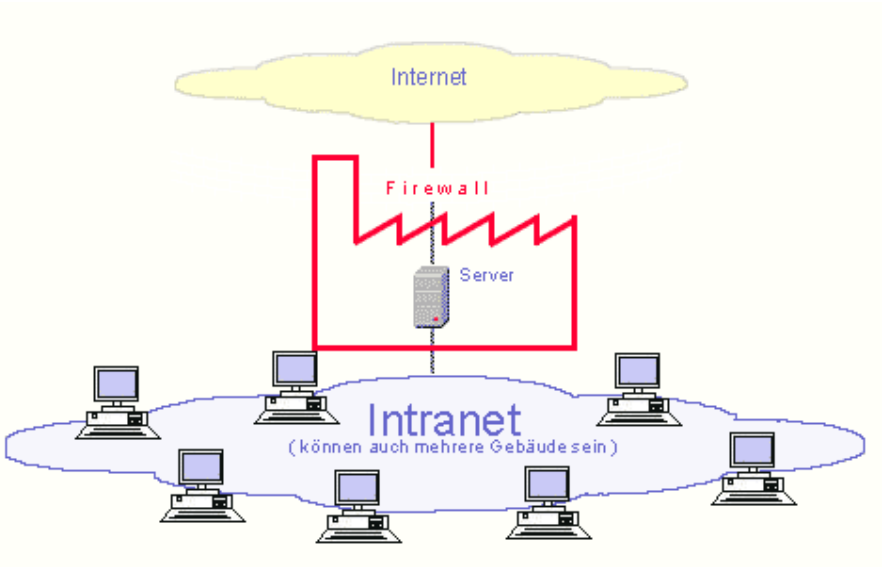

Abb. 3.2: Schematische Darstellung Intranet und Internet [**intranet2019**]

3.2.3 Client-Server Modell

Das Client-Server Modell beschreibt das Prinzip der Kommunikation zwischen zwei Teilnehmern innerhalb eines Netzwerks. Grundlegend unterscheidet das Modell hierbei zwischen einer Anbieterseite (Server) und einer Benutzerseite (Client). Der

Client betreibt auf seinem Endgerät (Computer, Smartphone, etc.) eine Client-Software mit der die Verbindung zum Server aufgebaut wird. Im Fall des WWW (siehe 3.2.5) ist dies in den meisten Anwendungsszenarien ein Webbrowser. Der Client fordert dabei eine Ressource an, welche auf dem Server vorliegt oder dort speziell für die Anfrage des Clients generiert wird (siehe auch Sektion 3.2.6). Das Client-Server Modell sieht vor, dass immer der Client die Verbindung aufbaut, nie andersherum [18]. Die Anfrage des Clients wird *Request* genannt, die Antwort des Servers *Response* oder *Reply*, welche bei ausreichender Berechtigung des Clients auch Daten enthält. Server-Computer sollen rund um die Uhr erreichbar sein, während Client-Endgeräte auch abgeschaltet werden können, ohne die Integrität des Netzwerks zu beeinflussen.

3.2.4 Kommunikation

Die Kommunikation im Internet und Intranet erfolgt über Protokolle. Ein Protokoll kann als ein Satz von Kommunikationsregelvorschriften [16] verstanden werden, welche den Netzwerkverkehr auf unterschiedlichen Schichten reglementieren. Diese Schichten werden im *OSI-Modell* (Open System Interconnection) der *ISO* (International Standardization Organisation), der internationalen Standardisierungsorganisation beschrieben.
Das *ISO-OSI-Modell* ist dabei in sieben Schichten eingeteilt, während die Erste als physikalische Schicht definiert ist und die Siebte als Anwendungsschicht. Protokolle sind dabei jeweils nur über Protokolle benachbarter Schichten in Kenntnis gesetzt. Es lässt sich grob in anwendungsorientierte Schichten (1 bis 4) und transportorientierte Schichten (5 bis 7) unterteilen. Die im Rahmen dieser Arbeit genutzten Webtechnologien nutzen kommunikativ nur anwendungsorientierte Schichten des *ISO-OSI Modells*.

3.2.5 World Wide Web

Das *World Wide Web* (WWW) ist die wohl populärste Anwendung des Internets [16] und wird oftmals fälschlicherweise als Synonym für das gesamte Internet genannt. Das WWW ist eine Sammlung von verteilten Dokumenten, welche sich gegenseitig über sog. Hyperlinks referenzieren und von Web-Servern zur Verfügung gestellt werden. Auf der Client Seite (siehe 3.2.3) stellt der Web-Browser die wichtigste Software da. Mit ihr werden Web Server angesprochen (Request) und Antworten (Response) für den Nutzenden dargestellt. Die wichtigsten sprachlichen Komponenten des WWW sind:

- **HTML:** Hypertext Markup Language - eine reine Beschreibungssprache, welche Hypertext Dokumente durch Tags codiert.
- **CSS:** Cascading Style Sheet - Eine Stylesheet-Sprache, welche das äußere Erscheinungsbild von Hypertext Dokumenten beschreibt
- **JS:** JavaScript: Eine Skriptsprache, welche u.A. Interaktion sowie Dynamik hinzufügt und clientseitig interpretiert wird.

Die Techniken des WWW können auch lokal im Intranet genutzt werden. Das zur Verständigung zwischen Client und Server genutzte Protokoll (siehe 3.2.4) ist das *Hypertext Transfer Protocol* (HTTP) bzw. in verschlüsselter Form *Hypertext Transfer Protocol Secure* (HTTPS), da eine Übermittlung im Klartext nicht immer wünschenswert ist. HTTP/HTTPS ist ein Zustandsloses Protokoll, das bedeutet dass jede Anfrage unabhängig voneinander geschieht und betrachtet wird. Dies und die Tatsache, dass jede Anfrage von der Client-Seite aus gestartet werden muss (siehe 3.2.3), stellen oftmals Hürden für die Entwicklung von Webanwendungen und Webservices da. Techniken wie *Cookies* und *Sessions*, sowie das wiederholte Abfragen von aktualisierten Daten seitens des Clients wirken hier entgegen. *Cookies* stellen persistent gespeicherte Daten auf der Client-Seite da, mit deren Hilfe der Webserver einen Client eindeutig zuordnen kann. Bei einer *Session* sendet der Client bei jeder Anfrage eine eindeutige ID an den Server. Im Normalfall endet eine Session beim Beenden des Webbrowser, während Cookie-Dateien eine längere Lebensdauer besitzen.

3.2.6 Webanwendungen und Webservices

Im Laufe der Entwicklung des WWW (siehe 3.2.5) stieg der Anspruch vom reinen Anbieten statischer Dokumenten in Richtung dynamischer Inhalte, welche einer Programmlogik folgend von einem Webserver für jede Anfrage generiert werden. Webanwendungen sind Computerprogramme, welche auf einem Webserver ausgeführt werden und den Webbrowser des Clients als Schnittstelle nutzen [16]. Dies bietet den großen Vorteil, dass etwaige Anpassungen von Programmlogik nur serverseitig erfolgen müssen und jeder Client mit Webbrowser als Benutzerschnittstelle ausreicht.

Webservices sind eine spezialisierte Art von Webanwendung. Der Fokus hier liegt auf dem Bereitstellen von Daten für andere Applikationen, welche die gewonnen Daten selbst auswerten und dem Nutzenden bereitstellen. Dies geschieht i.d.R.

über eine einheitlich beschriebene Schnittstelle (API - *Application Programming Interface*), über welche fremde Applikationen angefragte Daten abrufen können. Der Austausch der Daten erfolgt hier meist über Formate wie JSON (*JavaScript Object Notation*) oder XML (*Extensible Markup Language*), da Aussehen und Lesbarkeit der Daten irrelevant sind und somit eine Ausgabe in HTML nicht von Nöten ist.
Bei der Implementierung eines Webservices bieten sich folgende zwei technologische Arten der Umsetzung an:

SOAP/WSDL: Hier werden Nachrichten über das *Simple Object Access Protocol* (SOAP) ausgetauscht und deren Beschreibung über die *Web Services Description Language* (WSDL) definiert. Anfrage- und Antwortformat der Daten ist XML, eine Auszeichnungssprache, welche HTML sehr ähnelt aber deutlich allgemeiner ist. XML kann also mehr als Regelwerk verstanden werden, mit dessen Hilfe Entwickelnde ihre eigene hierarchische Beschreibung einer Datenstruktur vornehmen können. XML und HTML leiten sich bei der von der SGML (*Standard Generalized Markup Language*) ab, welches ihre Ähnlichkeit zusätzlich begründet [19].

REST: Representational State Tranfer - Hier kann jede einzelne Funktion des Webservices über eine jeweils zugeordnete URL (*Uniform Resource Locator*) abgerufen werden, umgangssprachlich als Webadresse bekannt. Das WWW selbst kann als REST-Webservice verstanden werden [20].

3.3 Websockets

Bezugnehmend auf die Problematik, welche durch die Kommunikationsstrategie über das http-Protokoll entsteht (siehe Sektion 3.2.5), wirken *Websockets* dieser entgegen. Als Kommunikationskanal verknüpft ein *Websocket* Server und Client. Zwar muss die Kommunikation zunächst über den Client initiiert werden, bleibt dann jedoch bestehen und der Server kann diese nutzen um aktiv neue Daten zu emittieren. Ein Nachteil ist jedoch, dass im Gegensatz zum http-Protokoll hier auch Daten hin- und hergeschickt werden, wenn dies eventuell nicht gewünscht ist [21], was insbesondere bei mobilen Applikationen kritisch sein kann. Ein generellen Vorteil bieten *Websockets* auch in Sachen Performanz, da das Protokoll, ist erst einmal eine Verbindung zustande gekommen, deutlich schlanker ist.
Die folgende Abbildung 3.3 zeigt einen Performanz Vergleich in Anbetracht des zusätzlichen *Payloads* von REST- und *WebSocket* Nachrichten.

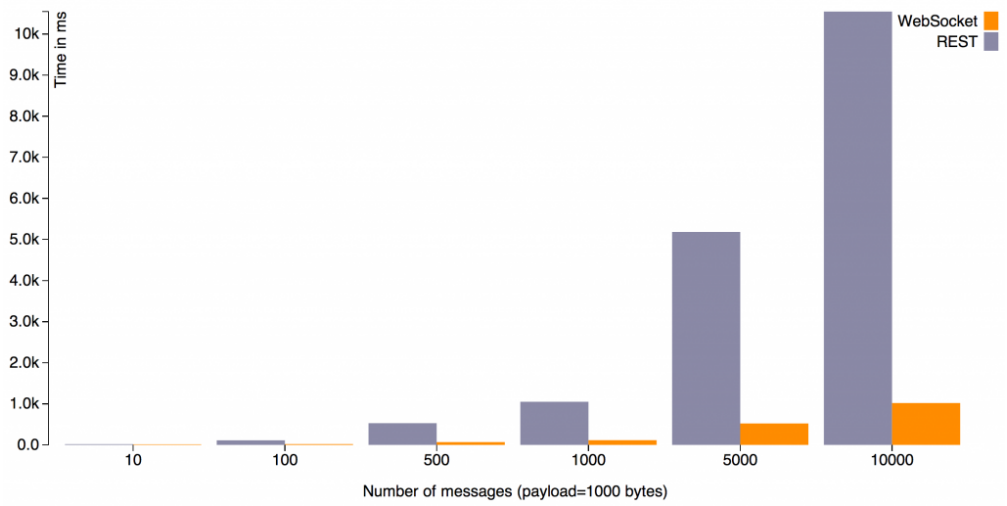

Abb. 3.3: Performanzvergleich REST versus WebSockets [22]:
Benötigte Zeit um N Nachrichten einer konstanten Nachrichtengröße zu verarbeiten.

3.4 Webapplikationsentwicklung

Dieses Kapitel beleuchtet wesentliche Begriffe hinsichtlich der Entwicklung von Webapplikationen. Webanwendungen und Webservices können unter diesem Begriff zusammengefasst werden.

3.4.1 Web-Application-Frameworks

Bei der Entwicklung von Webapplikationen wird oftmals auf Frameworks, spezifischer Web-Application-Frameworks (WAF) zurückgegriffen. Ein WAF bezeichnet ein Programmgrundgerüst, welches als Grundlage im Implementierungsprozess zum Einsatz kommt [23]. Dies erleichtert die Entwicklung ungemein, da auf bereits vorgefertigte Ansätze und Programmbausteine zurückgegriffen werden kann. Diese WAFs reflektieren zumeist auch Modelle und Prinzipien, was einen gewissen Grad an Konformität gewährleistet und das Verständnis für den Quellcode erhöht. Ein für Frameworks bekanntes Paradigma stellt das Umsetzungsparadigma
Inversion of Control (IoC), z. Dt. Umkehrung der Steuerung dar, welches u.a. auch in der objektorientierten Programmierung Anwendung findet. Hierbei wird eine Funktion/Unterprogramm bei der Hauptprogrammbibliothek registriert und von dieser zu einem späteren Zeitpunkt aufgerufen. Dies ist umgangssprachlich auch als 'Hollywood'-Prinzip bekannt („Don't call us! We call you“ z. Dt. „Ruf nicht uns an! Wir rufen dich an!“). Das Framework behält also die Programmflusssteuerung bei. Ein Nachteil, der durch den Einsatz von eines WAFs bedingt ist,

stellt die Einschränkung der Freiheit während des Implementierungsprozesses dar. Dieser wird jedoch billigend in Kauf genommen, da sich ein Reduktion des Zeit- und Kostenaufwands erhofft wird. Die Wahl des richtigen WAFs ist ein wichtiger Entschcheidungsprozess, bei dem mehrere Faktoren beachtet werden müssen, wie z.B. benötigte Einarbeitungszeit und Lizenzen.

3.4.2 Serverseitiger Ansatz

Anknüpfend an Sektion 3.2.6 sind Webapplikationen Software, welche serverseitig ausgeführt werden, wobei der Webbrowser eines Nutzenden als Benutzerschnittstelle dient. Eine Webapplikation kann jedoch auch clientseitig implementiert werden, wie in Sektion 3.4.3 beschrieben.
Serverseitige Webapplikationen verfolgen oftmals den *Multi-Page* Ansatz, das heißt pro Anfrage (Request) wird dem Client (Webbrowser) ein anderes Dokument übergeben. Wichtige Programmiersprachen für den Ansatz sind *php*, *Ruby*, *Python*, *Java* und auch *JavaScript*. Letzteres kam zuvor nur auf der Clientseite zur Anwendung.
Die **Model - View - Controller** Architektur (MVC) ist die vorherrschende Architektur, auf welche sich der Großteil der serverseitigen WAFs stützen. Hierbei wird die Programmlogik klar in drei große Bestandteile unterteilt:

Model: Das Model oder z. Dt. Modell (im restlichen Teil der Ausarbeitung *Model* geschrieben) beschreibt eine Datenstruktur an sich. In einem Webshop wären dies z.B. die Produkte und deren Eigenschaften wie ID, Name, Preis usw.
View: Sie beschreibt die reine Ansicht eines *Models*. In einem Webshop wäre dies z.B. die Detailseite eines Produkts.
Controller: Der Controller dient als Bindeglied zwischen *Model* und *View*. Er handelt ankommende *Requests* (Anfragen) ab und übergibt der *View* aus dem Modell die notwendigen Daten.
Neben der MVC Architektur existieren weitere, andere Architekturen und Ableitungen der MVC Architektur, wie z.B. der im *Django* WAF genutzten *Model - View - Presenter* Architektur.
Die folgende Tabelle 3.1 zeigt einen groben Überblick über bekannte WAFs, welche den serverseitigen Multi-Page Ansatz verfolgen [24].

Tab. 3.1: Überblick einiger serverseitiger Web-Application-Frameworks

Name	Sprache	Architektur
Symfony	php	Model - View - Controller
Laravel	php	Model - View - Controller
Phalcon	php	Model - View - Controller
Codeigniter	php	Model - View - Controller
Django	Python	Model - View - Presenter
Ruby on Rails	Ruby	Model - View - Controller

Aus der Tabelle lässt sich eine starke Popularität der Programmiersprache *php* ableiten und deren auf dieser Sprache basierenden WAFs. Die Tabelle stellt keinen Anspruch auf Vollständigkeit, da noch unzählig viele andere serverseitige WAFs existieren. Ebenso wurde das WAF *ExpressJS*, welches auf der serverseitigen Plattform *NodeJS* basiert, bewusst nicht mit in die Tabelle aufgenommen, da dies ein Sonderfall darstellt. Diese Thematik wird in Kapitel 5 ausführlich behandelt.

3.4.3 Clientseitiger Ansatz

Das Programmiermodell des WWW, welches durch die Architektur des HTTP geprägt ist, wird bei der Entwicklung von Webapplikationen übernommen. Dies sieht eine Anfrage immer seitens des Clients vor (siehe auch Sektion 3.2.5). Dies schränkt das Ausmaß von Interaktion und generieren von dynamisch ladenden Webseiten ein. Der clientseitige Ansatz der Webapplikationsentwicklung kommt meistens bei sog. Single-Page-Applikationen zur Anwendung. Hierbei wird o.g. Problem damit umgangen, indem bei Aufruf einer Internetseite die gesamte HTML Benutzeroberfläche inklusive Programmlogik in Form von JS Code als Ganzes an die Client übergeben wird. Dies bietet den großen Vorteil, dass die Logik nun auf dem Client ausgeführt wird und dieser dynamisch Daten nachladen bzw. Anfragen kann. Oftmals ändert sich auf einer Single-Page Applikation die Webadresse in der Adresszeile des Browsers nicht. Die ganze Applikation läuft also auf einer einzelnen Website ab, die sich dynamisch ändert. Dieses dynamische Nachladen von Inhalten wird **AJAX** - *Asynchronous JavaScript and XML* genannt. Die einzig nativ unterstützte Programmiersprache seitens der Webbrowser ist *JavaScript* und daher vorherrschend [16]. Jeder moderne Webbrowser hat einen JS Interpreter integriert. Über Plugins können zwar auch andere Sprachen genutzt werden, in Form von *Java-Applets* (Programmiersprache dort *Java*) oder das früher sehr populäre *Flash* des Unternehmens *Adobe*, welches *ActionScript* als Programmiersprache nutzt.

Beides gilt aber Stand 2019 als veraltet und der Einsatz derartigen Technologien wird nicht empfohlen. Es gibt sehr viele JavaScript WAFs und Bibliotheken, zu den bekanntesten zählen:

Angular ist ein clientseitiges *JavaScript* WAF, entwickelt und bereitgestellt von dem Unternehmen *Google inc.* Es hat vergleichsweise eine steile Lernkurve und setzt etwas Einarbeitungszeit voraus.

React ist streng genommen kein WAF, sondern lediglich eine JS Bibliothek. Es bietet aber über Erweiterungen die Möglichkeit, wie ein WAF genutzt werden zu können, was seine Flexibilität noch erhöht. Entwickelt und Betrieben wird *React* von der Firma *Facebook inc.*

Vue ist ein clientseitiges *JavaScript* WAF, ursprünglich entwickelt von dem Entwickler Evan You. Es gilt als einfacher zu erlernen als *Angular* und ist sehr flexibel.

Das Entwickeln von clientzentrischen JS Anwendungen ist mittlerweile so fortgeschritten, dass oftmals beim Nutzenden ein Gefühl entsteht, es würde ein lokal installiertes Programm ausgeführt werden. Populäre Beispiele wäre das in Kapitel 2 erwähnte *Google Docs*, welches eine voll umfassende Textverarbeitungslösung im Browser bietet. Derartige Applikationen werden *Rich Internet Application* (RIA) genannt.

3.4.4 Hardware Anforderungen

Auf der **Serverseite** ist der Anspruch an die Hardware sehr abhängig vom gewünschten Anwendungsfall und benötigter Skalierbarkeit. Das beliebte Server *Linux* Derivat *Debian* benötigt bspw. mindestens 128 Megabyte Ram-Speicher und 2 Gigabyte Festplattenspeicher. Es ist aber durchaus möglich mit noch sehr viel weniger potenter Hardware ein Server zu betreiben [3]. Für den Einsatz in einzelnen Unterrichtsklassen an Schulen würde ein *Raspberry Pi 3* Einplantinencomputer bereits genügend Leistung für Webanwendungen und einen günstigen Anschaffungspreis bieten. Auch besitzen bereits 67% der 10 bis 11 jährigen Jugendlichen Smartphones [25], welche ebenfalls genug Leistung für Webanwendungen aufweisen und als Clients genutzt werden können.

3.4.5 Vergleich zu anderen Entwicklungsansätzen

Der klassische Ansatz der Software Entwicklung wäre das Implementieren eine Desktop-Anwendung, welche lokal auf dem Computer des Anwendenden installiert wird. Typischerweise wird die Software programmiert und anschließend von einem Compiler in Maschinencode übersetzt bzw. von einer Laufzeitumgebung zur Ausführung interpretiert. Die Software wird also normalerweise auf dem Computer installiert und an die Gegebenheiten des Betriebssystems angepasst. Dies hat den Vorteil bei Bedarf sehr hardwarenah und performant entwickeln zu können, was durch das vorherige kompilieren des Codes in Maschinencode begünstigt wird. Nachteilig ist es jedoch, dass die Software zunächst überhaupt installiert werden muss. Eine noch vergleichsweise neue aber vielversprechende Webtechnologie ist in dieser Hinsicht *Webassembly* (WASM). Dies ist ein Bytecode, welcher neben der virtuellen JS-Maschine im Webbrowser ausgeführt wird und aus Programmiersprachen wie *C*, *C++* oder *Rust* kompiliert wird. Dies soll die Monopolstellung von JS als einzig ausführbare Sprache im Webbrowser adressieren und vor allem auf rechenintensiven Anwendungsgebieten einen Vorteil erbringen [26].
Die folgende Tabelle 3.2 vergleicht Web- und Desktopapplikationen hinsichtlich gängiger Kriterien.

Tab. 3.2: Vergleich von Web- und Desktopapplikationen [15]

Kriterium	**Webapplikation**	**Desktopapplikation**
Struktur	Modularer Aufbau	Meist als Gesamtpaket vertrieben
Verfügbarkeit	Weltweit dank Internet, lokal eingeschränkt möglich	Nur bei lokaler Installation verfügbar
Installation	Nicht erforderlich	Erforderlich
Speicher	Kein Zusätzlicher Speicher benötigt	Installation benötigt Speicherplatz
Updates	Live-Aktualisierung möglich	Teil- oder Neuinstallation notwendig
Teamarbeit	Zeitgleiches und schnelleres Arbeiten leicht möglich	Teamarbeit nur über Synchronisation möglich

4 Analyse

In diesem Kapitel werden zunächst existierende Plattformen am Markt verglichen und darauf aufbauend Anforderungen an das Projekt formuliert.

4.1 Vergleich mit existierenden Plattformen

Im folgenden Abschnitt werden ausgewählte existierende Plattformen, die im Bereich digital gestützte Unterrichtsmethoden angesiedelt sind, beleuchtet und anschließend gegenübergestellt. Eine klare Trennung zwischen kommerziell und nicht-kommerziell ist schwierig bis unmöglich, da viele Plattformen im Bereich des Freemium[2] Geschäftsmodells vermarktet werden. Generell gibt es sehr viele Anbieter und Plattformen und somit ist eine Beschränkung der Auswahl unabdingbar.

SMART Learning Suite Online Der Anbieter *SMART (Smart Technologies Corporation)* ist in Deutschland vor allem für sein Angebot von interaktiven Whiteboards, welche unter dem Namen *SMART Board* vermarktet werden, bekannt. Ergänzend bietet *SMART* auch die *SMART Learning Suite* an, welche sowohl online als auch lokal installiert genutzt werden kann [27]. Positiv hervorzuheben ist, dass bei der Cloud Variante Nutzende vorab ihre Server Region vorab festlegen. Wird hier Europa gewählt, ist anzunehmen dass europäische Richtlinien im Bezug auf Datenschutz und Speicheranforderungen berücksichtigt werden. Gespeichert werden die Daten generell auf *Amazon*- oder *Google*-Servern, wobei *SMART* angibt, dass in der europäischen Service Region hierbei *Amazon*- oder *Google*-Server mit Standort Deutschland genutzt werden [28]. Die *SMART Learning Suite* kann sowohl online als auch offline installiert werden und kostenlos getestet werden. Getestet wurde jedoch nur die online Version, da nur diese im Webbrowser läuft und sich in einigen Punkten von dem offline Pendant auch unterscheidet, was in Hinsicht auf dieses Projekt relevant ist.
Das Angebot umfasst viele Funktionalitäten und unterschiedlichste Implementierungen von interaktiven Unterrichtsmethoden, wie Quiz/Befragungen, Brainstorming, Memory, Karteikarten u.v.m. Viele Anwendungen funktionieren im Einzelanwender-Betrieb; Lehrende und Schüler nutzen das gleiche Gerät. Andere Anwendungen erfordern zusätzliche Clients, sprich Geräte wie Smartphones oder Computer. Sie laufen im Mehrbenutzerbetrieb. Ebenso können Lehrende Prüfungsaufgaben erstellen und diese dann Abfragen und Auswerten. Eine strikte Unterscheidung zwischen

[2] Freemium ist ein Geschäftsmodell, bei dem das Basisprodukt gratis angeboten wird, während das Vollprodukt und Erweiterungen kostenpflichtig sind.

Lehrer-, Klassen-, und Studierendenansicht findet nicht statt. Ein besonderes Feature der Software ist, dass Lehrende ganze Sets an Aktivitäten (siehe Abbildung 4.1) für den Unterricht erstellen können und diese anschließend schrittweise durchlaufen werden. Es kann bspw. mit einem Test begonnen werden, anschließend erfolgt eine Folie mit einem Begriff und darauffolgend wird eine interaktive Unterrichtsmethode ausgeführt usw.

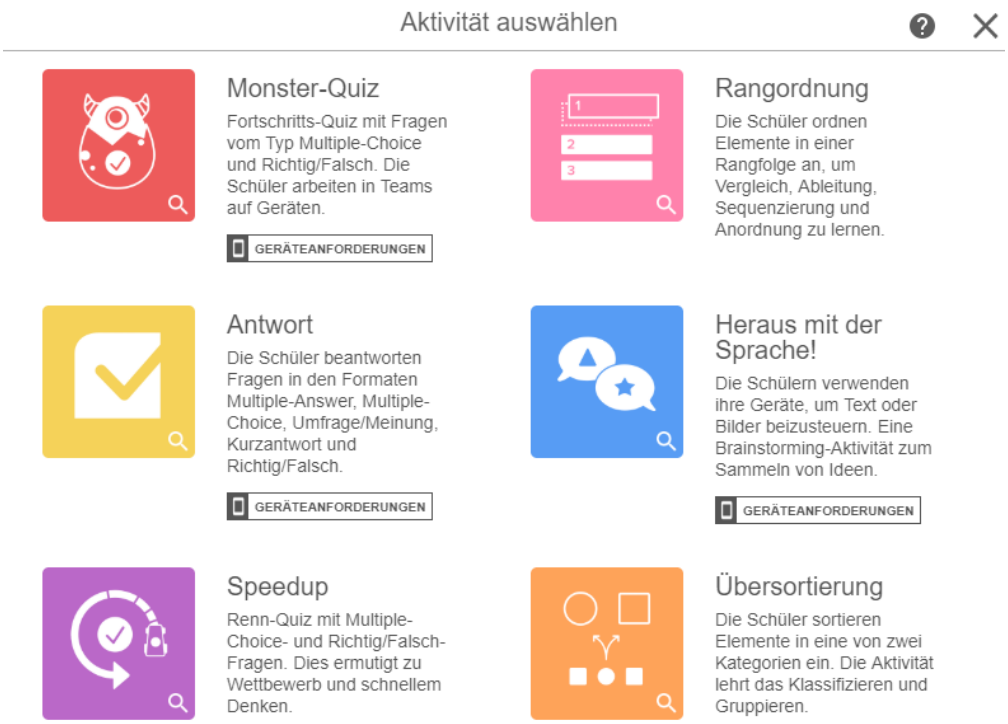

Abb. 4.1: Screenshot SMART Learning Suite Online [27]:
Im Bild ist die Maske zur Aktivitätserstellung zu sehen.
Benötigen die Schüler ein Endgerät, so wird dies gekennzeichnet.

ClassFlow Ähnlich zu *SMART Learning Suite Online* ist *ClassFlow* eine Software, welche die Durchführung von interaktivem Unterricht ermöglicht. Lehrende erstellt hierzu Sitzungen, welche ähnlich einer Präsentation durchlaufen werden. An jeder Stelle kann der Lehrende den Bildschirminhalt an die Schülerinnen und Schüler Endgeräte schicken und interaktive Unterrichtsmethoden starten, welche z.B. Umfragen, Brainstormings, Kreuzworträtsel u.v.m. sein können. Die Software läuft in der Cloud, es existiert keine Offline Variante. Für eine reine Datenspeicherung innerhalb der EU garantiert Anbieter *Promethean Limited* nicht [29]. Die meisten interaktiven Unterrichtsmethoden, in *ClassFlow* Aktivitäten genannt (siehe Abbildung 4.2), können auch im Einbenutzerbetrieb genutzt werden, d.h. die Einheit wird auf einem Computer gestartet und dort auch ausgeführt. Weitere Geräte

seitens der Schülerinnen und Schüler sind dann nicht notwendig. Eine interaktive digitale Tafel ist in diesem Modus jedoch empfehlenswert. Lehrende können auch schon vorgefertigte Unterrichtseinheiten aus dem sog. Marktplatz erwerben. Es gibt kostenlose wie auch kostenpflichtige Einheiten.

Abb. 4.2: Screenshot ClassFlow [30]:
Aktivitäten können i.d.R. am gleichen Gerät ausgeführt sowie an Endgeräte der Schüler geschickt werden.

Google Classroom Die online Software *Google Classroom* ist eng in die Produktpalette der Firma *Google inc.* eingebettet. Technisch betrachtet lässt sich *Google Classroom* eher mit der Software *Moodle* vergleichen, da eher das Ziel der Organisation einer Bildungsreinrichtung bzw. derer Kurse und Klassen verfolgt wird, obgleich das Erstellen von Fragen an alle Kursteilnehmer sowie von Quiz Aufgaben möglich ist. Beim Quiz wird die hauseigene online Software *Google Formulare* verwendet. Bildungseinrichtungen müssen sich zunächst als solche registrieren, bevor eine Nutzung erlaubt wird. Es ist möglich *Google Classroom* allerdings auch privat mit einem *Google* Konto nutzen, wenn explizit angegeben wird, dass die Software nicht in einer Bildungseinrichtung genutzt wird. Schulen und Universitäten müssen ein *G Suite for Education* Konto eröffnen, welches die Verwendung und Verwaltung weiterer Google Software mit sich bringt, so z.B. *Google Kalender*, *G-Mail* und weitere. Der *Google* eigene Cloudspeicherdienst *Google Drive* ist angebunden und somit werden z.B. erstellte Quizze dort abgespeichert. Beim Speichern der Daten kann von nicht EU-zentralen *Google* Cloud-Servern ausgegangen werden.

Sonstige Neben den o.g. größeren Anbietern existieren viele kleinere Online Angebote, welche sich meist auf die Bereitstellung einer Dienstleistung bzw. Ausführung einer interaktiven Unterrichtsmethode beschränken, hierbei jedoch oftmals auch interessante Ansätze bieten. Gerade wenn Dozierende unkompliziert eine bestimmte

interaktive Unterrichtsmethode im Unterricht einsetzen möchten, bietet es sich an auf einen kleineren Anbieter zurückzugreifen. Zu nennen wäre hier z.B. die Software **Plickers** [31], welche das Prinzip „bring-your-own-device“ etwas abändert. Schüler benötigen hier lediglich Papier auf dem spezielle QR-Codes abgebildet sind. Bei Fragestellungen wird das Papier nach oben gehalten und je nachdem welche Seite des Papiers (und somit auch des QR-Codes) nach oben zeigt, wird entschieden ob für Antwort A, B, C oder D plädiert wird. Eine Kamera vom Smartphone oder Tablet des Dozierenden erkennt dies und kann somit die Daten auswerten. Ähnlich verfährt die Anwendung **Poll Everywhere** [32], hier ist allerdings ein Endgerät pro Schüler notwendig. Wer nur teilnimmt muss jedoch keine Applikation installieren, hier reicht ein spezieller Link der in einem Webbrowser aufgerufen wird.

4.2 Gegenüberstellung

Nach der Präsentation der ausgewählten Angebote in Abschnitt 4.1 werden diese in folgenden Hauptkriterien miteinander verglichen:

1. **Serverstandort**: Befinden sich die Server des Anbieters innerhalb der Europäischen Union, so dass die dort geltenden datenschutzrechtliche Regelungen Anwendungen finden?

2. **Online Nutzung**: Ist die Nutzung über das Internet möglich?

3. **Offline Nutzung**: Ist die Nutzung offline über das Intranet möglich?

4. **Betriebsart E/M**: Ist die Nutzung im Einzelbenutzerbetrieb und/oder im Mehrbenutzerbetrieb möglich? Ersteres bedeute, dass eventuell mehrere Nutzer die Software ggf. an einem Computer verwenden können, eine Interaktion über mehrere angebundene Clients ist nicht möglich. Im Mehrbenutzerbetrieb können sich Nutzende über Clients mit der Software verbinden und gemeinsam interaktiv werden.

5. **Betriebsmodus Single/Set**: Können mehrere Unterrichtsmethoden als Set angelegt werden, welches später sukzessiv durchlaufen wird oder können nur einzeln angelegte Unterrichtsmethoden nach und nach manuell gestartet werden?

6. **Clients**: Gibt es unterschiedliche Arten von Clients für Lehrende, Teilnehmende und spezielle, die nur zur Anzeige gedacht sind?

7. **Registrierung**: Ist für die Nutzung eine Registrierung für Lehrende notwendig? Ebenso für Schülerinnen und Schüler?

8. **Unterstützte Unterrichtsmethoden**: Welche Arten/Typen lassen sich nutzen? Bei mehr als zwei wird hier die reine Zahl genannt.

Tab. 4.1: Tabellarischer Vergleich existierender Plattformen

Produkt	**SMART LSO**	**ClassFlow**	**Google Classroom**	**Plickers**	**Poll Everywhere**
Serverstandort EU	Ja*	Nein	Nein	Nein	Unbekannt
Online Nutzung	Ja	Ja	Ja	Ja	Ja
Offline Nutzung	Ja	Nein	Nein	Nein	Nein
Betriebsmodus	Single/Set	Single/Set	Single	Single	Single
Clients	2	2	2	1**	2
Client-Typen	Lehrer / Schüler	Lehrer / Schüler	Lehrer / Schüler	Lehrer	Lehrer / Schüler
Registrierung	Ja/Nein	Ja/Ja	Ja/Ja	Ja/Nein	Ja/Nein
Unterrichtsmethoden	12	10	Frage/Quiz	Quiz	Umfrage

* Option ist innerhalb der Software wählbar.

** Der „Client“ für die Schülerinnen und Schüler stellt in diesem Fall Papier mit aufgedruckten QR Codes da.

Aus Tabelle 4.1 lässt sich schließen, dass *SMART Learning Suite Online* und *ClassFlow* einen ähnlichen Ansatz nutzen und in unmittelbarer Konkurrenz zueinander stehen.
Beide bieten viele Arten von interaktiven Unterrichtsmethoden an und sehen einen Betrieb vor, einen Unterrichtsblock zu begleiten und unterstützen. Es können einzelne Unterrichtsmethoden gestartet werden, doch die hauptsächliche Stärke findet sich in dem Erstellen von Sets, welche dass iterative Durchgehen von Folien und interaktiven Unterrichtsmethoden vorsieht. Die zwei größten Vorteil des Angebots von *SMART* sind einerseits die verfügbare Offline-Applikation und der optionale Serverstandort innerhalb der EU. Eine Installation der Desktop-Applikation von *SMART* ist allerdings mit den genannten Vor- und Nachteilen aus Sektion 3.4.5 verbunden.
ClassFlow bietet im Vergleich noch mehr Interaktion auch außerhalb der Ausführung von interaktiven Unterrichtsmethoden an, da der Lehrende z.B. jederzeit das aktuelle Tafelbild an die angeschlossenen Clients der Schüler schicken kann. Dabei kann auch zwischen Gruppen unterschieden werden. Wird das Tafelbild von einem Client verändert, kann der Lehrende dies akzeptieren und als neue Grundlage für den weiteren Unterricht verwenden. Schüler können die *SMART*

Learning Suite Online auch anonym nutzen, während dies bei *ClassFlow* nicht möglich ist. Die Anbieter *Plickers* und *Pull Everywhere* sind fokussiert auf das Anbieten einer interaktiven Unterrichtsmethode. Beide Angebote können auch gut im Rahmen einer Vortragssituation abseits von Schulen genutzt werden. Da nur eine Unterrichtsmethode existiert, ist kein Set-Betrieb möglich. Alle Anbieter haben gemein, dass sie keinen speziellen Client oder sonstige Lösung für reine Präsentationsgeräte wie Projektoren oder andere Großbild-Anzeigen anbieten. Dies würde jedoch mehr Flexibilität beim Einsatz bedeuten, da vielen Vortragsräumen zwei Anzeigegeräte vorhanden sind. Der Lehrer könnte so eine Kontrollansicht auf einem nur ihm ersichtlichem Bildschirm einsehen, während das Publikum eine angepasste Großbildansicht zu sehen bekommt. Dieser Punkt soll in der zu entwickelnden Software gelöst werden. Keiner der genannten Anbieter lässt den Betrieb als im Intranet laufende Webanwendung zu, dies erschwert den Zugang für minder technisch ausgestattet Bildungseinrichtungen. Gehobener Datenschutz, bei dem die Daten gar nicht erst das Intranet verlassen, ist ebenfalls nicht gegeben.

Aus dem Vergleich lassen sich folgende wünschenswerte Eigenschaften für die im Rahmen dieses Projektes zu implementierende Software aufstellen:

- Ein Betrieb unabhängig vom Internet ist wünschenswert um den Einsatz in Einrichtungen ohne gut ausgebaute oder gar nicht vorhandene Internetanbindung zu ermöglichen. Hierzu sollte, wenn möglich, autark ein Intranet durch einen WLAN-Zugangspunkt generiert werden, welcher selbstständig ein Netzwerk aufbaut und Clients die Verbindung ermöglicht. Ein Betrieb im Intranet/WLAN würde ggf. das Aufkommen datenschutzrechtlicher Fragen deutlich eindämmen und wäre unabhängig von online Ressourcen.

- Ein zusätzlicher Präsenter Client, welcher bei Bedarf auf einer anderen Maschine ausgeführt wird, bringt mehr Flexibilität für den Dozierenden. So kann der Lehrende bspw. sein Smartphone als Kontrollclient nutzen und den im Klassenraum installierten Rechner, welcher an ein Projektor oder eine interaktive Tafel angeschlossenen ist zur reinen Anzeige der relevanten Inhalte nutzen. Diese Anzeige kann zusätzlich auf die visuellen Ansprüche einer Großbildanzeige optimiert sein, sodass ein Betrachten auch aus weiterer Entfernung problemlos möglich ist. Je nach Situation können Server und Clients aber auch einfach auf demselben Computer ausgeführt werden, z.B. in verschiedenen Browsertabs.

- Die Installation und Inbetriebnahme der Serversoftware sollte so einfach wie

möglich gestaltet werden, um auch weniger technisch versierten Dozierenden die Nutzung zu vereinfachen. Deshalb sollte eine Installation, welche tiefer ins System eingreift, möglichst vermieden werden. Ein rein im Internet angebotener Dienst muss diesen Teil natürlich nicht berücksichtigen und kann die technische Wartung selbst übernehmen, da das Angebot auf einem hauseigenen Server läuft.

- Das Anbieten mehrere interaktiver Unterrichtsmethoden innerhalb der Software ist wünschenswert, um die Nutzung und Inbetriebnahme attraktiver zu machen. Lehrende bekommen mehr Abwechslung geboten und sind weniger auf andere Softwareangebote angewiesen.

4.3 Systembeschreibung

Die Software soll als Web Server-Applikation implementiert sein. Diese soll skalierbar sein, d.h. ein Betrieb ausschließlich im Intranet soll ohne großen Installationsaufwand möglich sein wie auch der Betrieb auf einem entfernten, via Intranet oder Internet angebundenen Server. Die Software soll auch ohne aktive Internetanbindung im Intranet (resp. LAN/WLAN) nutzbar sein. Über zwei Web-Client Lösungen kann mit dem Server interagiert werden, eine für Lehrende, eine für Schüler. Ein dritter Client ist für den Einsatz auf Anzeigegeräten wie Projektoren und sonstigen Großbild-Anzeigen gedacht.
Über ein Backend Zugang können Administratoren und Dozierende den Server verwalten sowie erstmalig initialisieren. Neue Dozierende können einen Benutzeraccount anlegen, welcher von Administratoren freigeschaltet werden muss. Alternativ können Administratoren neue Benutzerkonten anlegen. Dozierende ist es möglich Lehreinheiten zu erstellen, diese zu starten sowie zu beenden. Eine Lehreinheit enthält zunächst eine interaktive Unterrichtsmethode. Während einer aktiven Lehreinheit, ist es Dozierenden möglich, diese zu leiten. Dies umfasst z.B. den Fortschritt, Verwalten von verbundenen Schülern, Speichern, etc. Als erste Umsetzung von interaktiven Unterrichtsmethoden erfolgt in diesem Projekt die Implementierung eines Brainstormings und Quiz'. Der parallele Betrieb von mehreren, unabhängig am System authentifizierten Lehrenden, welche Lehreinheiten starten und zu denen sich Schülerinnen und Schülern einschreiben, soll möglich sein.
Ein Präsenter-Client zeigt relevanten Informationen während der Durchführung einer Lehreinheit und ihrer interaktiven Unterrichtsmethode an. Dieser ist zur Anzeige auf einem Großflächenanzeigegerät ausgelegt (Projektor, Fernseher, Smart Board). Es soll problemlos möglich sein, mehrere Präsenter Clients anzukoppeln,

falls z.B. mehr als ein Projektor im Raum installiert ist.

Schüler/Studenten geben einen frei wählbaren Namen an. Ein Benutzerregistrierung ist nicht notwendig. Sie können anschließend aktiven Lehreinheiten beitreten und nach dem Start an deren interaktiven Unterrichtsmethoden partizipieren.

Eine detaillierte Aufführung der Anforderungen und Eigenschaften dieses Projekts erfolgt in den nachfolgenden Abschnitten.

4.4 Zielgruppe

Das Software-Projekt richtet sich an Bildungseinrichtungen jeglicher Art, welche eine lokal ausgeführte Softwarelösung für das Durchführen von interaktiven Unterrichtsmethoden bevorzugen. Darüber hinaus auch an jene, die digital gestützte interaktive Unterrichtsmethoden nutzen möchten. Dabei ist eine flexible Skalierbarkeit des Servers gegeben (siehe Abschnitt 4.6). Des Weiteren ist das Software-Projekt attraktiv für die Open-Source Community, welche das Projekt weiter ausbauen kann sowie neue Typen von interaktiven Unterrichtsmethoden hinzufügen kann.

4.5 Abgrenzung

Der Prototyp des Softwareprojekts (interne Bezeichnung Node ICT[3]) soll das Anlegen und Ausführen von Lehreinheiten mit zunächst einer interaktiven Unterrichtsmethode ermöglichen. Der Prototyp wird auf dem lokalen Host getestet (Server und Client auf demselben Computer ausgeführt) sowie im Intranet (LAN, Server und Clients auf unterschiedlichen Computern ausgeführt). Eine verschlüsselte Kommunikation zwischen Server und Client ist erwünscht, wird jedoch nicht im Prototyp implementiert. Eine Nutzung über öffentlicher IP-Adresse oder Domain im Internet ist prinzipiell möglich, wird jedoch nicht getestet. Das UI-Design wird leicht anpassbar sein. Wie in Abschnitt 4.3 erwähnt, wird sich auf das Implementieren von zwei interaktiven Unterrichtsmethoden beschränkt, der Prototyp wird aber das Umsetzen und Hinzufügen weitere Unterrichtsmethoden ermöglichen, da prinzipielle nur die Logik der neuen Unterrichtsmethode geschrieben werden muss. Eine Wiederverwendbarkeit von grundlegender Funktionalität (z.B. Anbindung an die Datenbank, Management verbundener Clients etc.) wird bereitgestellt. Zunächst

[3]Node ICT steht für *Node.js - interactive course teaching*. Das *Node.js* Server-Framework bildet den Grundstein des Softwareprojekts

wird pro Lehreinheit (Set) nur eine interaktive Unterrichtsmethode zugewiesen werden können.

4.6 Systemanforderungen

Aufbauend auf das Analysekapitel (Kapitel 4) sowie dem Abschnitten 3.1 und 3.1.3 lassen sich funktionale und nicht-funktionale Anforderungen an das System formulieren.

4.6.1 Nicht Funktional

Die Erstellung der nicht funktionalen Anforderungen an das System wurde aufbauend auf dem Vergleich existierender Plattformen aus Abschnitt 3.4.5 angefertigt, wobei noch weitere für das Projekt als wichtig erachtete Aspekte mit eingeflossen sind.

Tab. 4.2: Nicht Funktionale Anforderungen an die Projektsoftware

ID	Name	Beschreibung
NFA01	Erreichbarkeit	Das System soll auch ohne jeglichen Internetzugang im Intranet betrieben werden können
NFA02	Wartbarkeit	Der Code der Software soll verständlich und einfach zu warten und erweitern sein
NFA03	Nutzung von Open-Source Software	Die Nutzung von Open-Source Modulen und Frameworks ist zu bevorzugen
NFA04	Performanz	Das System soll ohne nennenswerte Verzögerung auf Eingaben reagieren
NFA05	Portierbarkeit	Der Betrieb unter unterschiedlichen Betriebssystemen soll möglich sein
NFA06	Usability	Die Anzeige und die damit einhergehende Usability soll auf das verwendete Gerät angepasst sein
NFA07	Sicherheit	Die Verbindung zwischen Client und Server soll verschlüsselt sein. Daten wie Passwörter sollen verschlüsselt gespeichert werden

4.6.2 Funktional

Aus Gründen der Übersicht wurden die Funktionale Anforderungen in den Anhang verschoben. Sie wurden aufbauend auf den Erkenntnissen des Vergleiches existierender Plattformen aus Abschnitt 3.4.5 gebildet und ausformuliert. Hierbei wurde auch der Implementierungsstatus mit vermerkt, welcher chronologisch erst nach **Kapitel 6**, der eigentlichen Implementierung, vorgenommen wurde.

4.7 Technische Anforderungen

In diesem Abschnitt werden die technischen Anforderungen an die Hardware erläutert, welche einen reibungslosen Betriebsablauf gewährleisten sollen. Es wird hierbei zwischen Server und Client Anforderungen unterschieden obgleich Server und Client auch auf der selben Maschine ausgeführt werden können.

4.7.1 Server

Die Server Software soll so umgesetzt werden, dass sie auch auf leistungsschwächerer Hardware problemlos mehrere Benutzer gleichzeitig ohne signifikante Performanceeinbuße bedienen kann. Die Hardwarespezifikationen eines Raspberry Pi (Version 3B) Einplantinencomputer (siehe auch Kapitel 1) werden hierbei als Mindestanforderung definiert. Durch die Nutzung der JS Laufzeitumgebung *Node.js* als Grundgerüst der Software, ist ein plattformunabhängiger Betrieb gewährleistet. Der Server soll rein im Intranet lauffähig sein und keine online Abhängigkeiten besitzen. Daher soll technisch keine (Breitband-)Internetanbindung für den Betrieb notwendig sein. Es soll ebenso möglich sein den Server „headless“, d.h. ohne angeschlossene Peripheriegeräte wie Tastatur, Maus und Bildschirm zu betreiben. Die Initialisierung der Software kann hierbei durch ein Startskript oder beispielsweise über einen SSH[4] Zugang erfolgen.

4.7.2 Client

Der im Rahmen dieses Projekts zu entwickelnde Client-Software kann wie in Abschnitt 4.3 erläutert, in drei Parts eingeteilt werden. Die Verwaltung im Backendbereich der Server-Software soll eine besonders niedrige Hardwareanforderung aufweisen, da hier der Einsatz von *JavaScript* auf ein Mindestmaß reduziert sei. Dies soll eine Server-Verwaltung auch bei deaktiviertem *JavaScript* gewährleisten. Die restlichen Software Module werden in jedem modernen Webbrowser auf jedem Endgerät lauffähig sein. Der verstärkte Einsatz von *JavaScript* ist hier unverzichtbar. Eine Kompatibilitätsabdeckung von 95% zur ECMAScript 6 (ECMAScript 2015) Spezifikation sollte vom verwendeten Webbrowser gegeben sein. Diese Anforderung erfüllen jedoch alle modernen Webbrowser (Stand 2019) [33].

[4] Mittels SSH (Secure Shell) kann eine verschlüsselte Verbindung zur Kommandozeile (Shell) auf einem Server hergestellt werden

5 Konzept

Dieses Kapitel stellt das Konzept zur Realisierung des Projekts vor. Dies umfasst den allgemeinen Systemaufbau, sowie Architekturentwürfe hinsichtlich der Server- und Clientseite inklusive den im Verbund eingesetzten Softwaremodulen.

5.1 Systemaufbau

Das System soll in eine Server- und Clientseite aufgeteilt sein. Da es sich um eine Webapplikation handeln soll, wird die Interaktion mit dem Server über einen Webbrowser stattfinden und dieser soll auch gleichzeitig der Client sein. Für Aufgaben wie das Starten des Servers, soll die Steuerung über die Kommandozeile des Host-Computers möglich sein. Alle anderen Interaktionen werden über den Web-Clients ausgeführt.

5.2 Netzwerkaufbau

Als Kommunikationsprotokoll soll das aus dem Webbereich bekannte HTTP resp. HTTPS Protokoll zum Einsatz kommen. Der Server wird als Webserver fungieren, Anfragen müssen vom Client aus initiiert werden (vgl. Abschnitt 3.2.5). Bei Parts, welche Echtzeitinteraktion benötigen, soll das Websocket (ws) Protokoll zum Einsatz kommen, welches eine bidirektionale Kommunikation zwischen Server und Client ermöglicht. Der Server soll sowohl in einem Intranet wie auch im Internet lauffähig sein. Dabei kann er, je nach infrastruktureller Realisierung über ein IPv4 oder IPv6 Adresse erreicht werden, bei Nutzung eines DNS-Servers auch über eine Domain.

Der Kommunikationsaufbau ist schematisch in Abbildung 5.1 dargestellt.

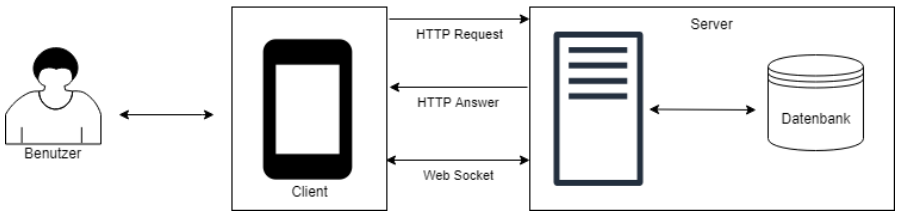

Abb. 5.1: Kommunikationsaufbau des zu entwickelnden Systems

5.3 Entwurf des Servers

Aufbauend auf das Grundlagen- und Analysekapitel sollen in diesem Abschnitt die Lösungen hinsichtlich der Serverkomponente des Projekts aufgezeigt werden.

5.3.1 Laufzeitumgebung: Node.js

Als Laufzeitumgebung und Grundbaustein des Servers wird die JavaScript-Laufzeitumgebung *Node.js* [34] genutzt werden, da dies zwei wesentliche Vorteile mit sich bringt:

1. *Node.js* nutzt als Paketmanager und Projektverwaltungstool *Node Paket Manager* (NPM). Mit dieser Software ist der Zugang zu über 350.000 Paketmodulen (Stand 13. Januar 2017) [35] gegeben und diese können das Projekt Modular erweitern. Diese können in *Node.js* gemäß dem IoC Prinzips genutzt werden (siehe auch Abschnitt 3.4.1). Ebenso können mit NPM grundlegende Start- und Installationsskripte leicht ausgeführt werden.

2. Da *Node.js* eine JavaScript-Laufzeitumgebung ist, wird zur Programmierung die Skriptsprache *JavaScript* genutzt, welche auch auf der Clientseite im Webbrowser zum Einsatz kommt. Dies erleichtert den Implementierungsprozess, da einheitlich in einer Sprache geschrieben wird.

Darüber hinaus können NPM-Pakete auch auf der Clientseite genutzt werden (siehe dazu auch Abschnitt 5.4.2). Eine gute Skalierbarkeit ist ebenfalls gegeben. Dies wird in folgenden Abschnitten genauer erläutert. Ist ein besonders hoher Ressourcenbedarf von Nöten (z.B. eine Bildungseinrichtung möchte einen zentralen Server installieren, welche viele Klassen/Kurse bedienen soll) können mehrere Serverinstanzen auf einem Computer parallel laufen und vorab mit einem Lastenverteiler (Load Balancer) Server, wie z.B. *NGINX* verwaltet werden. Um dies zu erreichen wird das Projekt mit *Node.js* realisiert werden und nicht mit einem *php*-Framework. Da *Node.js* grundlegend sehr offen ist was seinen Einsatzzweck betrifft, soll als Webserver Modul das NPM-Modul *Express* genutzt werden, welches im nächsten Abschnitt genauer erläutert wird.

5.3.2 Webserver: Express

Um mit Node.js komfortabel eine Webapplikation zu implementieren, soll das bekannte Webserver-Framework *Express* eingesetzt werden, welches viele HTTP-Dienstprogrammmethoden und den Einsatz von Middlerwarefunktionen gestattet.

Hierbei wird jeder eingehende HTTP Anfrage von Funktion zu Funktion weitergeleitet (Aufruf der Methode `next()`) oder explizit beantwortet (die Funktion besitzt ein Rückgabewert). Ebenso ist mit *Express* das Abbilden von Routen möglich. *Express* wird für den gesamten administrativen Teil des Lehrer-Login zum Einsatz kommen. Ebenso soll durch *Express* das Anlegen und Editieren von Lehreinheiten möglich sein (vgl. Sektion 4.3). Da für den gesamten Lehrer-Backendbereich *Express* zum Einsatz kommen soll und hier mit einfachen HTTP-Requests gearbeitet wird, kann der Einsatz von JavaScript auf der Client-Seite auf ein Minimum reduziert werden, was dem Einsatz auf Servereinheiten mit nicht modernen Webbrowsern entgegenkommt (sollte Client und Server auf der gleichen Maschine ausgeführt werden).

Für den interaktiven Part des Projekts werden zur Kommunikation *WebSockets* genutzt, welche mithilfe der JavaScript-Bibliothek *Socket.IO* realisiert werden sollen. Dies wird in der nächsten Sektion beschrieben.

5.3.3 Socket.IO

Die JavaScript-Bibliothek *Socket.IO* [36] ermöglicht bidirektionale Echtzeit-Kommunikation zwischen Webclient und Server, wobei jeweils ein Bibliotheksteil auf Server- und Clientseite zum Einsatz kommt. Ein großer Vorteil ist, dass beide Komponenten eine nahezu identische API aufweisen. Daten können sehr einfach von Client ereignisgetrieben (event-driven) zwischen Server und Client sowie vice versa ausgetauscht werden. Client und Server lauschen dabei gegenseitig auf Ereignisse, wie das Verbinden eines neuen Clients oder auch selbst implementierte Ereignisse. Dabei können jegliche *JavaScript* Daten hin-und hergeschickt werden. Eine händische Konvertierung in das JSON-Format ist nicht notwendig. *Express* wird zunächst Client Daten (HTML-, CSS- und JS- Daten) auf einer festgelegten Route senden. Anschließend wird die Kommunikation von *Socket.IO* gesteuert.

5.3.4 Sonstige Module

Neben *Express* ist der Einsatz von weiteren Modulen (*Node Packages*) vorgesehen, welche unterschiedliche Funktionen realisieren sollen. Diese sind:

- **Body-Parser**: Diese Modul ermöglicht das einfach Auslesen von HTTP-Requests. Schickt ein Client bspw. Formulardaten, können diese einfach gelesen und ausgewertet werden.
- **express-session**: Da Lehrende und Administratoren zur Nutzung der Software einen gültigen Zugang besitzen müssen, sind zur Authentifizierung

der Nutzenden HTTP-Sessions vorgesehen (vgl. Abschnitt3.2.5). Das Modul *Express-Session* macht das Arbeiten mit diesen sehr komfortabel. Über das Zusatzmodul *connect-session-sequelize* ist die Zusammenarbeit mit der gewählten Datenbank einfach. (Weiterführende Informationen diesbezüglich im Abschnitt 5.3.5 Wahl der Datenbank).

- **Pug**: Die Template Engine *Pug* besitzt seine eigene Syntax und macht das Entwerfen und Schreiben von HTML Templates sehr komfortabel. Zusätzlich werden Funktionalitäten wie Vererbung und Mixins unterstützt. Eine Einsatzbeschreibung erfolgt in Kapitel 6 Implementierung. Pug soll für sämtliche zu übertragende HTML Dokumente genutzt werden.

Neben den o.g. Modulen kommen noch weitere zum Einsatz, welche kleinere Funktionen realisieren. Diese wurden aus Gründen der Übersicht hier nicht gelistet, werden aber in der Datei `package.json` des Projekt-Quellcodes gelistet sein.

5.3.5 Wahl der Datenbank

Da bei dem zu entwickelnden System vielerlei Daten anfallen, wie registrierte Nutzer, angelegte Kurse, interaktive Unterrichtseinheiten und mehr, ist der Einsatz eines Datenbanksystems unerlässlich. Grundlegend können Datenbanksysteme in zwei Kategorien unterteilt werden: **SQL** und **noSQL** Systeme.

SQL Systeme speichern ihre Daten in sogenannten Relationsmodellen, welche als Tabelle visualisiert werden können. Hierbei beschreibt der Tabellenkopf den Datensatz und den Datentypus (jede Spalte für sich), während Zeilen eine Entität (Eintrag) in der Datenbank beschreiben. Vorteil hierbei ist, dass die Daten konform sind, d.h. jeder Zugriff liefert immer einen Rückgabewert [21]. Nachteil ist der erhöhte Aufwand, sollte die Definition des Relationsmodells im Nachhinein geändert werden, was das Aktualisieren sämtlicher Daten erfordern würde. Desweiteren sind SQL System schwer skalierbar, da für größere Datenbanksysteme leistungsstärkere Server gekauft werden müssen. Mehrere Relationsmodelle können über Fremd-Schlüssel (Querverweise) miteinander verbunden werden, um auch komplexere Sachverhalte abbilden zu können.

NoSQL Systeme lassen sich nach Speicher-Arbeitsweise in verschiedene Subkategorien einteilen [21]. Oft genutzt sind die Typen Dokumentenorientiert, Key-Value Pairs (Schlüssel-Wert Paare) und Graphen-basierte Systeme. Dokumentenorientierte NoSQL Datenbanken legen pro Entität ein Dokument an, in welchem die

Informationen meist im JSON Format abgespeichert werden. Key-Value Systeme verfolgen ein einfaches Zuordnungsprinzip und bilden Schlüssel-Wert Paare, ähnlich einer Dictionary Datenstruktur. Bei Graphen-basierten Systemen werden Entitäten und ihre Beziehungen untereinander an sich gespeichert. Generell sind NoSQL Systeme weniger statisch im Vergleich zu SQL Systemen. Dies räumt eine große Flexibilität beim Speichern von Daten ein, da Datensätze auch unvollständig gespeichert werden können. Dies kann auch als Nachteil interpretiert werden, ist aber generell immer vom Kontext des Projekts abhängig.

Für das zu entwickelnde System soll ein möglichst flexibler Weg gewählt werden was die Wahl der Datenbank betrifft. Da das MVC-Prinzip zum Einsatz kommen soll, beschreibt der Model-Teil von zu bereitstellenden Daten auch wie diese über welche Funktionalität aus der Datenbank geladen werden sollen. Der Controller soll nur die vom Model bereitgestellten Funktionen nutzen und keine direkten Datenbankzugriffe selbst tätigen. Damit die Software im hohem Maße skalierbar bleibt, ist der Einsatz eines sogenannter Object-Relationship-Mapper, kurz ORM, vorgesehen, welcher an verschiedenste Datenbanksysteme angebunden werden kann. Da das Projekt in seiner kleinsten Skalierung lokal auf einem Einplantinencomputer wie dem *Raspberry Pi 3* und lokal im Intranet laufen können soll, ist für den Anfang die Verwendung eines Datenbanksystems vorgesehen, welches vollständig durch eine Programmbibliothek abgebildet ist. Dies hat den Vorteil, dass kein extern laufendes Datenbanksystem installiert, gewartet und gestartet werden muss, da die komplette Datenbank in einer einzigen Datei auf dem Server gespeichert wird. Diese Anforderungen erfüllt die Programmbibliothek *SQLite* **??**. Die gesamte Datenbank kann hier sogar rein im Arbeitsspeicher gehalten werden, was jedoch den Nachteil mit sich bringt, dass bei einem Ausfall oder Abschalten des Server der kompletten Verlust sämtlicher Daten folgt.
Als ORM wird auf das NPM Modul *Sequelize*, welches neben *SQLite* mit viele andere bekannte SQL Datenbanksystemen zusammenarbeiten kann, u.A. *Postgres*, *MariaDB* und *Microsoft SQL Server* [37]. Der Wechsel auf ein anderes SQL Datenbanksystem ist somit jederzeit problemlos möglich, falls gewünscht.

Das Zusatzmodule `connect-session-sequelize` ermöglicht eine einfache Handhabung der Session-Verwaltung von Lehrenden, die in das System eingeloggten sind. Dazu werden entsprechende Tabellen zur Verwaltung der Sessions und deren Lebenszeit automatisch via *Sequelize* in der Datenbank angelegt. Zuvor sollen die Passwörter sicher gespeichert werden, d.h. nicht im Klar-Text, sondern nur als

Hashwerte, welche zusätzlich mit einem Salt verstärkt werden[5].
Da zum Zeitpunkt der Recherche kein zu *SQLite* ähnliches und für den produktiven Einsatz lauffähiges NoSQL Äquivalent gefunden werden konnte, fiel die Entscheidung auf SQLite. Die genannten Vorteile eines NoSQL Systems scheinen für die Anforderungen des zu entwickelenden Systems ohnehin nicht relevant, obgleich sogar ein Umstieg auf NoSQL Datenbkansystem möglich wäre, auch wenn dies mit einem etwas erhöhten Aufwand verbunden wäre, da dann auch der ORM gewechselt und die Models entsprechend angepasst werden müssten.

5.3.6 Server Architekturdiagramm

Zusammenfassend lässt sich der finale Entwurf der Serverarchitektur in folgender Abbildung 5.2 visualisieren. Er wird in Kapitel 6 Implementierung umgesetzt.

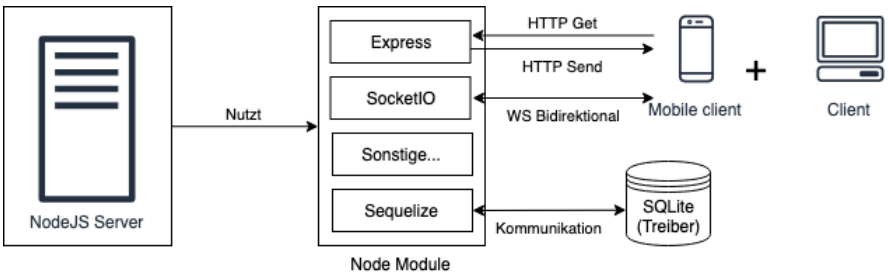

Abb. 5.2: Aufbau der geplanten Serverarchitektur

Hinweis: NodeJS Module aus der Sektion **Sonstige Module** wurden aus Gründen der Übersicht in dieser Grafik nicht abgebildet.

[5]Weiterführende Informationen unter: `https://de.wikipedia.org/wiki/Salt_(Kryptologie)`

5.4 Entwurf des Clients

Wie zuvor in Abschnitt 4.3 erörtert, ist es vorgesehen drei verschiedene Clients zu implementieren, welche alle auf dem gleichen Prototyp basieren sollen, allerdings verschiedene Zwecke verfolgen. Ein Webclient jeweils für **Lehrende/Dozierende**, **Schülerinnen und Schüler** und einen für **Großbildanzeigen** optimierten wie Projektoren o. Ä. vor. Zur Vereinfachung werden diese gemäß der vorherigen Reihenfolge **Teacher Client**, **Student Client** und **Presenter Client** in diesem und darauffolgendem Kapitel genannt.

5.4.1 User Interface

Da das Projekt als Web-Applikation implementiert werden soll, wird das UI gänzlich durch die Stylesheet-Sprache *CSS* beschrieben. Im Jahr 2018 wurden erstmals häufiger mobile Endgeräte wie Smartphones zur Internetnutzung herangezogen als klassische Computer oder Laptops [38]. Viele Design-Frameworks setzen daher schon länger auf das Prinzip „Mobile First“. Die Applikation soll letztlich sowohl auf Computern und Smartphones aber auch Großbildgeräten angezeigt werden. Ein Web-Design-Framework als Basis, welches bereits viele relevante Web-Design Standards wie das o.g. „Mobile First“ berücksichtigt, bildet ein solides Fundament in Sachen Usability (Software-Ergonomie) und UI-Design. Eines der populärsten dieser Art ist *Bootstrap*, welches zusammen mit einem frei erhältlichen Design-Theme von *FreeHTML5.co* [39] genutzt werden soll. Das gewählte Theme soll an die Applikation angepasst und teils für die Großbild Anzeige des Presenter Clients optimiert werden.

5.4.2 Browserify

Um das Nutzen von NPM Modulen und die damit verbundene `require()` Funktionalität auch auf der Clientseite zu ermöglichen, soll die Bundle-Software *Browserify* [40] zum Einsatz kommen. Mit ihr können alle Module, welche über den *Node Package Manager* in das Projekt hinzugefügt wurde im Webbrowser des Clients genutzt werden. Dazu bündelt die Bibliothek alle Module und stellt anschließend eine einzige JS-Datei zur Verfügung, die nur noch im HTML-Dokument eingebunden werden muss. Mithilfe der `require()` Funktionalität, welche eigentlich nur in der *Node.js* Umgebung genutzt werden kann, ist es möglich den Code übersichtlicher in mehrere Dateien/Module aufzuteilen. Dies war früher ohne Aufwand auf der Browser-Seite nicht möglich, ist aber durch die Einführung von Modulen seit EC-

MAScript[6] in Version 6 nun möglich. Oftmals wird aber aus Kompatibilitätsgründen zu älteren Webbrowsern auf Lösungen wie *Browserify* gesetzt. Zusätzlich kann Software wie *Babel* den geschriebenen *JavaScript* Code so übersetzen, dass er auch von älteren Webbrowsern interpretiert wird (auch Transpiler genannt). Sämtliche folglich genannten NPM Module resp. Bibliotheken sollen via *Browserify* in eine *JavaScript* Datei zusammengefasst werden und anschließend pro Client eingebunden werden. Das bringt den zusätzlichen Vorteil, dass für jegliches, auf Browser-Seite eingesetztes JavaScript nur ein einziger HTTP-Get Request für den Code benötigt wird, da wie erwähnt nur eine *JavaScript* Datei pro Client angefordert werden muss.

5.4.3 JavaScript Lösungen

Von folgenden JavaScript Lösungen soll auf der Clientseite Gebrauch gemacht werden, um den Implementierungsprozess zu optimieren:

- **VueJS**: Um das Anzeigen, Editieren und Anpassen dynamischer Inhalte zu erleichtern, soll das JS-Webframework *VueJS* [41] zum Einsatz kommen, da dieses gut skalierbar ist und alle benötigten Funktionalitäten mit sich bringt. Im Vergleich zu *AngularJS* und *React* (siehe auch Abschnitt 3.4.3), biete *VueJS* eine flachere Lernkurve und kann als ein guter Kompromiss aus seinen zwei Konkurrenten betrachtet werden. Mittlerweile hat *VueJS* seinen Konkurrent *React* in Sachen Popularität auf *GitHub* überholt [42]. *VueJS* ist auch gemessen an der Dateigröße von nur 80 KB im Vergleich deutlich kleiner als Angular mit 500 KB.

- **Socket.IO**: Das bereits in Abschnitt 5.3.3 erwähnte *Socket.IO* besitzt ein Bibliotheksteil, welcher auf der Client-Seite im Webbrowser zum Einsatz kommt. Dadurch wird die bidirektionale Kommunikation mit der Server ermöglicht und es soll in beide Richtungen Daten in Echtzeit ausgetauscht werden.

- **Zingchart**: Zur Visualisierung der Wörterwolke (Word Cloud), welche bei der interaktiven Unterrichtsmethode Brainstorming zum Einsatz kommt, soll die auf diese Szenarien spezialisierte JavaScript Bibliothek *ZingChart* genutzt werden (vgl. Abbildung 5.3). Die Software ist in einer freien Version erhältlich, wobei lediglich stets ein kleines *ZingChart* Logo stets angezeigt wird [43].

[6] Der als ECMAScript (ECMA 262) standardisierte Sprachkern von JavaScript beschreibt eine dynamisch typisierte, objektorientierte, aber klassenlose Skriptsprache. (Wikipedia.org)

Abb. 5.3: Eine Word-Cloud/Wörterwolke generiert durch ZingChart [44]

5.4.4 Client Architekturdiagramm

Nachfolgend die umzusetzende Architektur der Client-Anwendung.

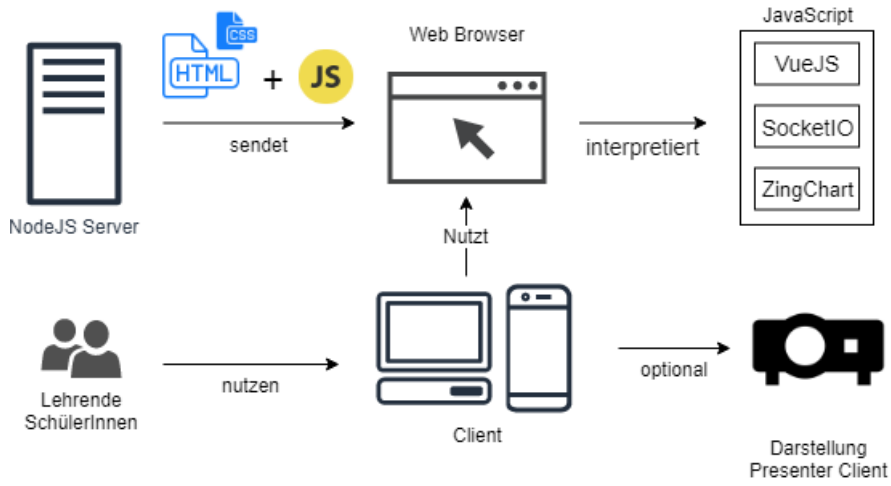

Abb. 5.4: Aufbau der geplanten Client Architektur

Hinweis: Zum besseren Verständnis wurden die Rollen des Servers und der Nutzer ebenso dargestellt. Der Presenter Client nutzt in der Grafik ein Projektor. Dies ist eine Option und nicht zwingend von Nöten. Alle Clients können auf dem gleichem Server-Host Computer ausgeführt werden (z.B. in unterschiedlichen Tabs eines Webbrowsers) als auch auf separaten Endgeräten.

6 Implementierung

Das folgende Kapitel wird Einblicke in die Entwicklung der Software im Rahmen des Projekts geben. Dabei wird iterativ chronologisch die Vorgehensweise schriftlich reflektiert und an mehreren Stellen zum besseren Verständnis auch ein Einblicke in den Sourcecode gegeben. Anknüpfend werden etwaige Probleme bei der Implementierung aufgezeigt sowie mögliche Lösungen diskutiert.

6.1 Implementierung der Server-Software

Aufbauend auf das Entwurfs Kapitel soll die Server-Software mit *Node.js* und *Express* als Hauptkomponente entwickelt werden. Dazu wird zunächst im folgenden Abschnitt 6.2 der HTTP Server grundlegend konfiguriert und anschließend dessen Routen im darauffolgenden Abschnitt 6.5 angelegt.

6.2 ExpressJS Setup

Nachdem das Projekt grundlegend mit dem Befehl `npm init` initialisiert wurde, kann ExpressJS einfach über den *Node Packet Manager* (NPM) hinzugefügt werden. Zusätzlich wird das NPM Modul *IP* genutzt um die aktuelle IP-Adresse des Host-Computers automatisch zu ermitteln und den *Express*-Server auf dieser lauschen zu lassen. Dies wird im folgendem Code-Listing getan:

```
1 const app = express();
2 const server = app.listen(3000, server_ip, function () {
3 logger.log({ level: 'info', message: `Hello! The Server is running
      on ${server_ip}!`});
4 });
```

Listing 1: Errichtung des Webservers

Der Server lauscht auf der IP Adresse des Netzwerk-Adapters der Maschine auf dem er ausgeführt wird. Zusätzlich wurde der Port 3000 spezifiziert. Dies kann je nach Konfiguration auf den Standard HTTP Port 80 resp. 443 geändert werden, sollte Verschlüsselung eingerichtet sein (HTTPS).
Anschließend können nun die Routen eingerichtet werden.

6.3 Autarkes WLAN

Um einen Stand-Alone Betrieb[7] der Software zu ermöglichen, wird als Lösung der Betrieb eines eigens aufgebautem kabellosen Netzwerkes (WLAN/Wifi) angestrebt. Folgende mögliche Lösungsszenarien wurden ausgearbeitet.

1. **Einfach**: Als einfachste Lösung hat sich der Betrieb eines lokalen WLANs über ein Smartphone oder Laptop herausgestellt. Nahezu jedes Smartphone bzw. jeder Laptop lässt das Generieren eines WLAN Zugangspunktes für andere Geräte zu. Alle Clients und der Server selbst werden mit diesem Netzwerk verbunden. Dies Verbindung wurde getestet und ein Betrieb war möglich. Da hierbei aber auch die Internetverbindung des Zugangspunktes freigegeben wird, was ggf. unerwünscht sein kann, würde es sich anbieten eine spezielle „Companion App“ zu entwickeln und für *Android / iOS* basierte Smartphones zu anzubieten. Diese App könnte automatisch einen WLAN-Hotspot erstellen und den Netzwerkverkehr ins Internet eventuell limitieren. Gleiches gilt analog für windows- oder unixbasierte Computer, ist aber problematischer, siehe dazu nächster Listenpunkt.

2. **Speziell**: Um auf einem Computer vollautomatisch einen WLAN Hotspot zu genieren, bedarf es Administrator Privilegien sowie genauere Kenntnisse über den verwendeten Netzwerkadapter. NodeJS selbst bietet nur eingeschränkt Möglichkeiten an, diese Aufgabe autark zu übernehmen, könnte aber ggf. eventuelle Shell-Skripte triggern. Unter *Microsoft Windows 7* und höheren Versionen, gibt es mit den NPM Paket `node-hotspot` auch die Möglichkeit, dies direkt mit *Node.js* zu erledigen. Dieses Paket wird in die zu entwickelnde Software integriert und soll anschließend unter einer *Microsoft Windows* Umgebung das generieren eines WLAN Hotspots / Zugangspunktes ermöglichen. Zur Verwendung werden entsprechende Steuerungsoptionen in Einstellungsbereich im Lehrkraft Backend integriert. Die interne Steuerung erfolgt über Routen (siehe auch Abschnitt 6.5).

   Wird die Applikation in einer vorhandenen Netzwerkinfrastruktur betrieben, ist ein Betrieb in jedem Falle gewährleistet. Ist diese Vorraumsetzung nicht gegeben, könnte ein mobiler WLAN-Router genutzt werden, sollte keine ausreichende Infrastruktur vorhanden sein. Dieser müsste einmalig konfiguriert werden und ist anschließend in der Lage die Serverapplikation für Clients

[7] Stand-Alone meint einen Betrieb unabhänging von ggf. vorhandenen Netzwerkinfrastruktur am Einsatzort

ansprechbar zu machen. Ein solches Gerät gibt es bereits ab ca. 10 Euro zu erwerben.

6.4 Verschlüsselung

Um eine verschlüsselte Kommunikation zwischen Server und Clients zu gewährleisten, sollte der Datenaustausch nicht über das HTTP Protokoll, sondern über das HTTPS Protokoll erfolgen. Um HTTPS allerdings sinnvoll nutzen zu können, ist ein Zertifikat von einer Zertifizierungsstelle (CA) notwendig, was, mit Kosten verbunden ist. Das Zertifikat gewährleistet die Identität des Servers. Allerdings bietet der Anbieter *Let's Encrypt* [45] kostenlose Zertifikate an, welche sich problemlos bei vorhandenem SSH-Zugang installieren lassen. Hier wird aber nur die Domain zertifiziert. Bezahlte Zertifikate von anderen CAs bieten im Vergleich höheren Support und es wird nicht nur die Domain an sich zertifiziert, sondern das ganzen Unternehmen an sich. Außerdem besteht ein Gewährleistungs- und Garantieanspruch gegenüber der CA [46].
Für den Intranet Betrieb können auch eigens ausgestellte Zertifikate genutzt werden, welche mit Shell-Programmen wie `openssl` generierbar sind [47]. Allerdings warnen moderne Webbrowser den Nutzenden recht auffällig, dass die genutzte Verbindung dennoch nicht sicher ist, da dem Zertifikat nicht vertraut werden kann. Soll die Applikation rein im Intranet laufen, könnte die Schule alternativ eine Sub-Domain anlegen, welche nur aus dem Intranet erreichbar ist und ebenfalls durch ein signiertes Zertifikat geschützt ist.
Der HTTPS Betrieb wurde erfolgreich getestet. Die Implementierung ist nicht aufwendig allerdings, birgt aber den o.g. Nachteil. Da das Intranet ein an sich abgeschlossenes Netzwerk ist, scheint der verschlüsselte Betrieb in diesem zunächst nicht wichtig, kann aber jederzeit realisiert werden und ist bei Betrieb im Internet als obligatorisch zu betrachten.

6.5 Anlegen der Routen

Grundlegend soll es folgende Routen auf dem Server geben:

- „/“: Die Haupt Route, sie wird angesteuert, wenn der Server einfach unter seiner IP (oder eingerichteter Domain) kontaktiert wird. Hier wird anschließend die Rolle des Nutzers (Lehrender oder Schülerin/Schüler) abgefragt und dementsprechend weitergeleitet.

- **„/teacher“**: Diese Route verweist auf den Lehrerbereich der Anwendung,

man kann sie auch als Backendbereich bezeichnen. Alle hinter dieser Route liegende Routen bedürfen einer Authentifizierung seitens des Nutzenden.

- **„/client“**: Diese Route verweist grundlegend auf den Student Client. Aber auch der Presenter Client wird über diese Weiche aufgerufen.

Neben diesen drei Hauptrouten existieren noch weitere Routen für das Error-Handling (z.B. „404 - Seite nicht gefunden“) sowie besondere für die *WebSocket*-Kommunikation, welche aber im Hintergrund genutzt werden. Das Anlege der Routen ist mit folgendem Codeausschnitt durchgeführt:

```
1 app.use('/teacher', teacherRoutes);
2 app.use('/client', clientRoutes);
3 app.use('/', mainRoutes);
```

Listing 2: Anlegen der Routen

Die Route Module werden im Hauptmodul (app.js) referenziert und deren Zuständigkeit festgelegt.

Zu beachten ist: Die weiterführenden Routen werden in Dateien ausgelagert, um die Übersicht des Quelltextes zu wahren. Zudem sind auch diese Routen sog. Middleware-Funktionen. Dies wird im nächsten Abschnitt genauer beleuchtet. Daher ist auch die Reihenfolge wichtig, wie in Listing 2 dargestellt. Würde die „/" Route als erstes angelegt werden, so würde diese alle folgenden, spezifischeren „abfangen“.

6.6 Reflexion des MVC Schemas

Da der Server grundlegend nach dem *Model-View-Controller* Muster arbeiten soll, gilt es dieses zu implementieren. Die folgende Tabelle 6.1 gibt einen Überblick über die anfallende Struktur:

Tab. 6.1: MVC Struktur der Implementierung

Betrifft	Model	View(s)	Controller
Lehrende	user.js	teacher/new.pug teacher/signup.pug teacher/user-edit.pug	teacher.js
Schülerinnen/ Schüler	student.js	student.pug	client.js
Lehreinheiten	eduSession.js	edusessions/*/*.pug edusessions/index.pug edusessions/running.pug	session.js quizzing.js brainstorming.js

Hinweis: Zum Zwecke der Übersicht wurden einige interne Controller-Dateien nicht gelistet, wie z.B. der Error-Controller, welcher zwar eine View besitzt, jedoch kein Model.

Nun sollen die entsprechenden Controller-Funktionen an die Routen gebunden werden. Entsprechende Unterrouten werden zu den aus Abschnitt 6.5 bereits angelegten hinzugefügt. Im Sinne der Übersicht wird hierzu ein Routen-Ordner angelegt, welcher Routen-Module enthalten soll. Folgende Routen-Module werden angelegt:

routes/main.js: Generelle Routen (Startseite etc.).
routes/client.js: Student Client / Presenter Client relevanten Routen.
routes/teacher.js: Alle für das Backend resp. Lehrerbereich relevanten Routen.

Es folgt ein Codebeispiel (Listing 3) aus der Datei `routes/client.js`.

```
1  // 3rd Party Imports
2  const express = require('express');
3  const router = express.Router();
4  // App Imports
5  const clientController = require('../controllers/client');
6  const isAuth = require('../middleware/is-auth');
7  // Presenter & Student Clients
8  router.get('/presenter/:sessionId', isAuth, clientController.
       getPresenter);
9  router.get('/student', clientController.getStudent);
10 router.get('/', clientController.getStudent);
11 module.exports = router;
```

Listing 3: Unterrouten und Controlleranbindung

Zur Verdeutlichung wird anknüpfend die in Zeile 10 des vorangegangen Listings 3, die Funktion `getStudent` des Controllers in Listing 4 gezeigt:

```
1 // GET => /client/student
2 exports.getStudent = (req, res, next) => {
3
4 return res.render('client/student',
5   {
6     docTitle: 'Student | Node ICT',
7     ipAdd: ip.address(),
8   });
9 };
```

Listing 4: GET Funktion des Student Controllers

6.7 Einrichtung der Datenbank

Die SQL Datenbank *SQLite* und der Object-Relationship-Mapper *Sequelize* können unkompliziert über NPM dem Projekt hinzugefügt werden. Nach diesem Schritt kann die Anbindung und Einpflegung folgen. Hierzu wird ein Datenbank Utility Modul angelegt, dieses soll die grundlegende Konfiguration der Datenbank enthalten und ausführen. All dies kann direkt über *Sequelize* erfolgen, welches im Hintergrund die notwendigen Schritte vornimmt. Der Dialekt `sqlite` muss in der *Sequelize* Konfiguration angegeben werden sowie der Pfad unter welchem die Datenbank als Datei gespeichert werden soll. Gemäß dem logischen Aufbau einer SQL Datenbank folgt nun das Konfigurieren und Anlegen der Tabellen und deren Beziehung untereinander. Dieser Arbeitsschritt erfolgt relativ intuitiv. Im folgenden Code-Listing 5 wird die Tabelle bzw. das *Sequelize*-Model „student“ im Modul `tables` konfiguriert.

```
1  exports.student = (sequelize, Sequelize) => {
2    return sequelize.define('student', {
3      id: {
4        type: Sequelize.INTEGER,
5        autoIncrement: true,
6        allowNull: false,
7        primaryKey: true
8      },
9      name: {
10       type: Sequelize.STRING,
11       allowNull: false
12     },
13   })};
```

Listing 5: Anlegen einer Tabelle und deren Beziehungen

Im Datenbank Modul `util/database.js` wird nun die Konfiguration geladen und anschließend dessen Beziehung zu anderen Entitäten eingestellt. Durch *Sequelize* kann hier ein relativ natürliches Sprachbild verwendet werden. Das folgende Code-Listing 6 zeigt die Beziehungen zwischen `EduSession` und `Student` an:

```
1 EduSession.hasMany(Student, { onDelete: 'cascade' });
2 Student.belongsTo(EduSession);
```

Listing 6: Konfiguration von Entitätsbeziehungen

Nach demselben Schema werden nun für alle Modelle entsprechende Tabelle angelegt und deren Beziehungen untereinander festgelegt.

Datenbank Interaktion Um Datenbank Anfragen (Queries) zu stellen, bietet *Sequelize* viele Optionen an. Diese müssen nicht in SQL geschrieben werden, sondern sind normale JS Funktionen. Jedes innerhalb *Sequelize* definierte Model bietet diese automatisch an. Als Beispiel könnte nun über den JS Code `const studentToFind = await Student.findByPk(1);` die Entität mit der ID 1 aus der Datenbank geladen werden. All diese bereitgestellten Funktionen sind *JavaScript Promises*, d.h. sie werden asynchron ausgeführt auch nicht erfüllen des *Promise* gilt es zu berücksichtigen. Im Erfolgsfall befindet sich in der Variabel nun das Objekt der Entität und dieses bietet wiederum Funktionen zur Interaktion an. Eine ausführliche Dokumentation findet sich auf der Webpräsenz von *Sequelize* [37]. Die in Tabelle 6.1 gelisteten Models werden gemäß der Arbeitsteilung des MVC-Schemas hauptsächlich direkt mit *Sequelize* arbeiten und den Controllern entsprechende Funktionen bereitstellen.

6.8 Implementierung des Lehrer-Bereiches

Nachdem die aus Abschnitt 6.6 genannten Models angelegt wurden, welche direkt mit der Datenbank interagieren, müssen anschließend Controller für die verschiedenen Abschnitte des Webservers implementiert werden. Jedes Model hat dabei einen zugehörigen Controller, welcher entsprechende Funktionen für die Routen exportiert. Wie in Listing 3 beispielhaft zu sehen, wird die GET-Route „/student“ mit der nach außen hin exportierten Funktion `getStudent` assoziiert. Um eine einheitliche Struktur zu gewährleisten, werden exportierte Funktionen eines Controller Moduls immer nach der jeweilig bedienten Request-Methode benannt
(GET, POST, etc.).

View Rendering mit Pug Für alle Views soll die Template-Engine *Pug* zum Einsatz kommen (siehe auch Kapitel 5 Konzept). Dazu wird *Pug* für das Rendern aller HTML-Dokumenten, welche nicht statischen Routen zugehörig sind, im Hauptmodul der Software registriert. *Pug* macht das Schreiben von HTML Dokumenten sehr komfortable und unterstützt Vererbung (Layouting). So wird zunächst für den Lehrer-Bereich ein Haupt-Layout angelegt, von dem später alle anderen, diesem Bereich zugehörigen Views, erben. Die Layouts können zusätzlich so genannte Blöcke enthalten, welche später dann dynamisch mit Inhalten gefüllt werden. Die Controller können Daten an die Views übergeben, welche von diesen dann dargestellt werden.
Dazu bietet *Pug* Funktionen wie das Iterieren über Datensätze mittels Schleifen an. Auch dynamisch generierte Inhalte, abhängig von konditionalen Ausdrücken, sind möglich. Sämtliche an die Clients ausgelieferte HTML Dokumente sollen von *Pug* „on-demand“ generiert werden. *Pug* benutzt seine eigene *Templating-Language*, welche sich deutlich von HTML unterscheidet, aber sehr intuitiv und schnell zu lernen ist. So wird die Hierarchie der HTML Elemente allein durch Einrückung bestimmt. Somit entfällt das Schließen dieser komplett. Siehe hierzu Abbildung 6.1.

```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) {
         bar(1 + 5)
      }
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      if (foo) {
         bar(1 + 5)
      }
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>
        Jade is a terse and simple
        templating language with a
        strong focus on performance
        and powerful features.
      </p>
    </div>
  </body>
</html>
```

Abb. 6.1: Pug im Vergleich zu HTML [48]:
Rechts abgebildet `Pug`, Links `HTLM`. Durch die Einrückung und dem Wegfall von schließenden Tags ist der Code deutlich übersichtlicher.

Es werden anschließend für alle Models Views angelegt, damit Lehrende sich anmelden und einloggen, sowie neue Benutzer anlegen und editieren können. Ebenso für das Erstellen von Lehreinheiten mit den Unterrichtsmethoden Brainstorming und Quiz. Dabei werden bei Dateneingabe HTML-Formulare verwendet, welche an-

schließend via POST-Request an den Server geschickt werden. Controller validieren und werten diese folglich aus. Die Implementierung für das tatsächliche Ausführen der Unterrichtsmethoden wird in Abschnitt 6.10 beschrieben.

6.9 UI Design

Aufbauend auf den Abschnitt 5.4.1 des Entwurf-Kapitels wird zur Designumsetzung das Frontend-CSS-Webframework *Bootstrap* [49] genutzt. Dieses kann ebenfalls einfach über NPM dem Projekt hinzugefügt werden. Als Design-Theme bildet das frei erhältliche *Bootstrap*-Theme `Neat` von *freehtml5.co* die Grundlage (siehe Abbildung 6.2). Dieses wird hauptsächlich im Lehrerbereiches genutzt und in angepasster Version für den Student Client und den Presenter Client. Letzterer wird insbesondere für die Nutzung auf Großbildgeräten wie Fernsehern und Projektoren optimiert. Es wird sichergestellt, dass sich die Software angenehm auf stationären wie mobilen Endgeräten nutzen lässt.

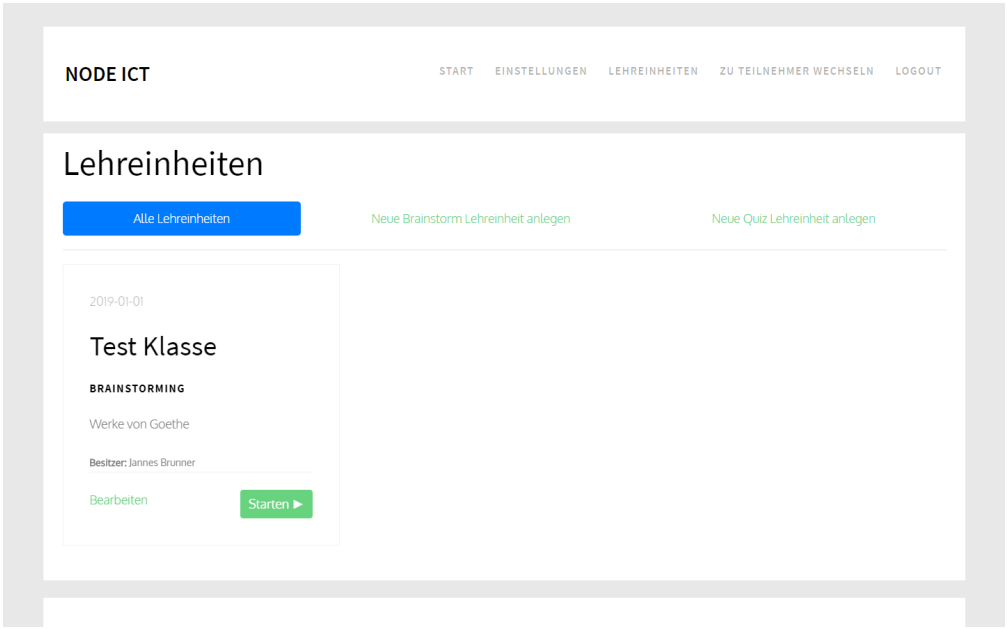

Abb. 6.2: UI Design der Applikation beispielhaft illustriert durch ein Screenshot des Lehreinheiten Bereiches im Backend Zugang des Teacher Clients.

Absicherung des Bereiches Bestimmte Bereiche der Applikation sollen nur registrierten und freigeschalteten Benutzenden zugänglich sein. Beim erstmaligen Initialisieren wird der Server mit einem Super-Administrator Account eingerichtet. Nur dieser soll neue Nutzer freischalten, andere Lehrende zu Administratoren ernennen und die Applikation auf Werkseinstellungen zurücksetzen können. Eine

entsprechende Einrichtungsmaske soll beim ersten Serverstart automatisch erscheinen. Um dies technisch zu realisieren, wird das aus dem Kapitel 5 genannte `express-session` NPM-Modul genutzt, welches bereits vollständig mit der von *Sequelize* verwalteten *SQLite* Datenbank kompatibel ist. Nach der Einrichtung im Hauptmodul wird eine eigene Middleware-Funktion geschrieben, welche bei allen abgesicherten Routen als erstes aufgerufen wird und überprüft, ob die vom Client übertragene Session noch gültig ist.

```
1 module.exports = (req, res, next) => {
2   if(!req.session.isLoggedIn) return res.redirect('/teacher/login'
        );
3   next();
4 }
```

Listing 7: Code der Authentifizierungs Middleware

Falls dies nicht der Fall ist, wird auf die Login Seite verwiesen. Das Verwalten der Sessions und das Generieren von Cookies für die Client-Seite wird automatisch von dem Modul übernommen.

6.10 Umsetzung des Client Softwareanteile

Nachdem die Funktionen „Erste Initialisierung der Software“, „Login/Logout“ inkl. Benutzerverwaltung sowie das Anlegen und Verwalten von Lehreinheiten des Typs Brainstorming und Quiz implementiert sind, sollen die Lehreinheiten auch aktiv ausgeführt werden können. Dies bildet die Kernfunktion der Software und verlangt mehr Interaktion auf der Client Seite.

Browserify wird einfach über NPM dem Projekt hinzugefügt und ist anschließend einsatzbereit. Alle im Abschnitt 5.4.3 des Kapitels Konzept erwähnten JS-Lösungen sind als NPM Modul verfügbar und werden ebenfalls dem Projekt hinzugefügt. Pro Client (Teacher Client, Student Client, Presenter Client) wird ein Development-Modul angelegt (`dev.js`). In diesem können alle benötigten JS-Bibliotheken normal importiert und genutzt werden. Via *Browserify* wird anschließend pro Client ein Production-Modul generiert (index.js), welches alle notwendigen Importe bündelt. Um diesen Prozess zu automatisieren, wird ein Skript erstellt, welches vom NPM ausgeführt werden kann. Nur das Production-Modul muss via `script`-tag in das jeweilige *Pug* Template pro Client eingebunden werden. Dies reduziert gleichzeitig die Anzahl notwendiger GET-Requests auf der Client-Seite.
Pro Client wird die UI mit dem JS-Framework *Vue.js* kontrolliert und verwaltet. Auf dem HTML Layout wird ein `div`-Element als Ankerpunkt definiert und alle

unterliegenden Elemente stehen fortan zur dynamischen Anpassung bereit. Mittels Datenbindung (Data-Binding) hält *Vue.js* die angezeigten Informationen auf dem UI aktuell. *Vue.js* ist dabei pro Client als einfaches JS Objekt auch von außen ansprechbar, was die Schnittstelle für andere Bibliotheken, insbesondere *Socket.IO*, bildet. *Socket.IO* auf der Client-Seite ist für den gesamten Datenverkehr zwischen Server und Client verantwortlich. Sowohl auf Server- wie auch Client-Seite können sog. „Listener“ programmiert werden, die auf bestimmte Ereignisse (Events) von der, jeweils anderen Seite ausgelöst, lauschen. Im Ereignisfall wird eine anonyme Funktion aufgerufen, welche sich um die eintreffenden Daten kümmert. Die ausgetauschten Daten müssen hierbei nicht zwangsläufig zuvor in das JSON-Format umgewandelt werden, wie dies sonst bei REST-Apis üblich ist.

Auf der Server-Seite kümmert sich das Modul `ioSocketHandler.js` um alle eingehend Web-Socket Verbindungen und ordnet diesen zunächst einem Namensraum (Namespace) zu. Ein Student Client wird dabei immer dem Namensraum für Student Clients zugeordnet und steht als Socket Objekt zur Verfügung, übliche Clients diesem Schema folgend. Nach erfolgreicher Verbindung wird dem Student Client eine Liste verfügbarer Lehreinheiten geschickt und diesem auf der Client Seite dargestellt. Pro Lehreinheit (Typ Brainstorming oder Quiz) gibt es einen „Session Handler“, der als JavaScript Klasse implementiert ist. Startet eine Lehrkraft eine Lehreinheit wird abhängig vom Typ eine neuen Klasseninstanz angelegt und eine Referenz in einem Speicher gehalten. Tritt nun ein Schüler der Lehreinheit bei, übergibt das übergeordnete „‘Socket Handler“-Modul das Socket-Objekt der Klasseninstanz der Lehreinheit. Die gesamte Logik und Kommunikation der auszuführenden Lehreinheit wird von der jeweiligen Klasse übernommen. Beendet eine Lehrkraft die Session, wird diese aus dem Speicher entfernt und steht nicht mehr zum Beitreten zur Verfügung. Pro gestartete Session kann über einen speziellen Link der passende Presenter Client aufgerufen werden. Dieser wird ebenfalls über das `ioSocketHandler.js` Modul der jeweiligen Klasseninstanz zugeordnet. Im Listing 8 wird ein Codeauszug gezeigt, welcher den Datenaustausch zwischen Server und Client zeigt.

```
1 /// TEACHER :::::::
2 updateSessionT() {
3   this.socketT.emit("updateSession", this.session);
4 }
```

Listing 8: Server Socket Event-Emitierung

Der Server schickt das Event `updateSession` an den Teacher Client. Als Inhalt

der Nachricht wird das Session Objekt (`this.session`) übermittelt.

```
1 // Sever tells client to update the session object
2 socket.on("updateSession", function (newSession) {
3   console.log("getting fresh session from server...", newSession)
4   if (newSession && newSession.id == vue.session.id) {
5     vue.session = newSession;
6   }});
```

Listing 9: Client Socket Event-Listener

Der Teacher Client lauscht auf das Event `updateSession`. Trifft dieses ein, wird eine anonyme Funktion aufgerufen, welche sich dem Inhalt der Nachricht annimmt. Diese überprüft in diesem Fall zunächst, ob sich die ID des zu aktualisierenden Session Objekts mit dem ursprünglichen deckt. Anschließend wird das alte Session Objekt auf das neue referenziert.

6.11 Herausforderungen der Implementierung

Während der Implementierung stellte sich anfangs das Verbindungsmanagement der verbundenen Clients über *Socket.IO* als instabil heraus, da Sockets sich bei Verbindungsabbruch zwar selbständig erneut verbinden, jedoch immer unter einer neuen Session ID. Dies führte zunächst zu unerwartetem Verhalten während einer Lehreinheiten Ausführung. Dem konnte aber durch zusätzliche Authentifizierungsdaten entgegengewirkt werden. Bei mobilen Geräten wie Smartphones erwies sich der Verbindungsabbruch zusätzlich als geräteabhängig. Manchen Geräten unterbrechen die Verbindung z.B. beim Ausschalten des Bildschirms sofort, während andere diese im Hintergrund weiterhin aufrecht erhalten.
Außerdem war es schwierig ein gut funktionales „Word-Cloud / Wörterwolke“-Modul zu finden, welches sich ohne nennenswerte Probleme mit *Vue.js* im Einklang nutzen lassen konnte. Generell wird *Vue.js* in diesem Projekt recht rudimentär eingesetzt, was seine Funktionsweise zwar nicht einschränkt, jedoch das volle Potential dieser Web-Frontend-Engine nicht gänzlich nutzt.

Das NPM-Modul `node-hotspot` wurde zwar gemäß der Instruktionen der Entwickler implementiert, allerdings konnte auf mehreren *Microsoft Windows* Testsystemen nicht selbstständig ein WLAN-Hotspot aktiviert werden. Leider war `node-hotspot` zum Entwicklungszeitpunkt (Stand Juli 2019) das einzige verfügbare NPM-Modul, welches die Generierung eines WLAN-Hotspots bezweckte.

7 Fazit

In diesem vorletzten Kapitel der Arbeit wird die implementierte Software als Ergebnis mit der ursprünglichen Zielsetzung verglichen. Anschließend werden gesammelte Erfahrungen, die mit der Umsetzung des Projekts einhergingen, reflektiert. Im letzten Kapitel 8 Ausblick wird auf die Zukunft des Projektes näher eingegangen sowie über einzelne Bestandteile des Projekts hinsichtlich Optimierung gesondert diskutiert.

7.1 Zusammenfassung der Ergebnisse

An Schulen bedarf es trotz der künftigen finanziellen Unterstützung durch den *DigitalPakt Schule* an kostengünstigen und einfache digitalen Lösungen, damit das Geld an den richtigen Stellen sinnvoll eingesetzt werden kann.
Das Konzept einer kostengünstigen, auf Webtechnologien basierenden Softwarelösung zur Unterstützung von interaktiven Unterrichtsmethoden konnte erfolgreich umgesetzt werden.

Die entwickelte Serversoftware ist flexibel auf verschiedenen Betriebssystemen und unterschiedlicher Netzwerk-Infrastruktur einsetzbar. Dabei ist ein Offline-Verwendung im Intranet möglich und es müssen keinerlei Ressourcen aus dem Internet geladen werden. Ein Betrieb im Internet mit verschlüsselter Kommunikation ist im Bedarfsfall möglich. Diese kann mit wenigen Schritten eingerichtet werden.

Der Betrieb auf dem Einplantinencomputer *Raspberry Pi 3* konnte erfolgreich getestet werden und die Hardwareanforderungen an das Server-System gering gehalten werden, was einem niedrigen Anschaffungspreis zur Folge hat.
Nutzende können die Server-Software mit wenig Aufwand installieren, es ist lediglich die *JavaScript*-Laufzeitumgebung *Node.js* einzurichten. Als Client kann jeder moderne Webbrowser auf mobilen und stationären Geräten genutzt werden.
Es konnten erfolgreich drei Webbrowser-Client Lösungen für Lehrende, Teilnehmende und Anzeigeräte in Unterrichtsräumen (Projektoren, digitale Tafeln, etc.) implementiert werden. Diese können je nach Situation und Anforderung auf unterschiedlichen Geräten ausgeführt werden, wobei sich die Benutzeroberfläche auf die verwendete Bildschirmgröße adaptiert.

Eine Lehrkraft kann sich registrieren, Lehreinheiten anlegen, diese ausführen und den Server grundlegend administrieren. Die zwei Unterrichtsmethoden Brainstor-

ming und Quiz wurden erfolgreich umgesetzt und teilnehmende Schülerinnen und Schüler können an diesen partizipieren. Anschließend besteht die Möglichkeit Ergebnisse auszuwerten. Der Server kann mehrere Nutzer und ausgeführte Unterrichtseinheiten gleichzeitig bedienen. Die Benutzeroberfläche wurde simple gehalten, damit auch weniger technisch versierte Nutzende die Software einsetzen können. Eine Lösung zur Generierung eines autarken WLANs konnte nicht zufriedenstellend gefunden werden, dafür wurden alternative Lösungen, wie das Nutzen eines Smartphones als WLAN-Zugangspunkt, ausgearbeitet.

Die Anwendung läuft stabil und neuste Spracheigenschaften der Programmiersprache *JavaScript* nach Spezifikation *ECMAScript* 2015, 2016 und 2017 wurden erfolgreich im Quellcode eingesetzt.

7.2 Erfahrungsauswertung

Die Auseinandersetzung mit Unterrichtsmethoden und der Digitalisierung an Schulen war interessant und es konnte ein besseres Verständnis für damit verbundene Probleme entwickelt werden.

Durch den Implementierungsteil der Arbeit wurden vielerlei Erfahrungen hinsichtlich der Entwicklung eines verteilten Systems in Form eine Webanwendung gesammelt. Insbesondere die gewonnen Kenntnisse im Umgang mit den Frameworks *Node.js*, *Express* und *Socket.IO* waren lehrreich und der erlangte Wissensstand kann als Basiswissen hinsichtlich vielerlei Arten von Projekten in Zukunft genutzt werden. Auf der Client Seite war der Einblick in die Arbeitsweise des Frontend-Webframework *Vue.js* interessant. Das Vorwissen über die Konkurrenten *Angular* und *React* war hilfreich, wurde jedoch um vielerlei neue Ansätze ergänzt. Die Arbeitsweise des *WebSocket* Protokolls wurde gelernt und mit der *JavaScript* Bibliothek *Socket.IO* erfolgreich eingesetzt. Die neuen Funktionen nach Spezifikation *ECMAScript* 2015, 2016 und 2017 des Sprachkerns der Programmiersprache *JavaScript* wurde verinnerlicht und das Wissen um die Sprache intensiviert.

8 Ausblick

8.1 Optimierungspunkte der Software

Der Nachrichtenaustausch, welcher über das *WebSocket* Protokoll via *Socket.IO* realisiert ist, sollte mehr vereinheitlicht werden. Grundsätzlich lassen sich jede Art von *JavaScript* Daten übertragen; ein strengeres Konzept kann hier das Verständnis für andere Entwickler fördern und den Code sauberer halten. Gerade der Ausführungscode der interaktiven Unterrichtsmethoden könnte noch besser gekapselt und noch sinnvoller aufgeteilt werden, auch in Hinblick auf die Daten, welche zwischen Server und Client ausgetauscht werden. Der Installationsprozess könnte ggf. noch automatisierter erfolgen und so wenig Technik affinen Nutzenden die Installation erleichtern. Der Code der Web-Clients, welcher im Webbrowser ausgeführt wird, kann mit mehr Kenntnissen über die Entwicklung mit *Vue.js* und *Socket.IO* optimiert werden.

8.2 Anknüpfende Ansätze

Das Projekt soll zukünftig weiterhin ausgebaut werden. Während der Entwicklung kamen immer wieder Ideen für weitere Funktionen auf, welche aber aus Gründen der Priorität nicht implementiert worden sind oder nur als Konzept vorlagen. Dies wären z.B. Funktionen, die den Dozierenden noch intensiver während des Unterrichts unterstützen oder das Schreiben einer API, um die Software an andere Systeme anbinden zu können. Zum Zeitpunkt des Abschlusses des Projekts kann eine Lehrkraft ein Lehreinheit anlegen, welche eine Unterrichtsmethode (Brainstorming oder Quiz) beinhalten kann. Die Umsetzung weiterer Unterrichtsmethoden wäre wünschenswert. Ebenso die Option, dass eine Lehreinheit mehrere Unterrichtsmethoden beinhalten kann. Ein weiteres Vorhaben wäre es, die Software in einem *Fork* nach dem REST-Design aufzubauen und die Kommunikation dementsprechend umzugestalten, um anschließend zu vergleichen, welche Vorgehensweise entsprechende Vor- und Nachteile mit sich bringt.
Die Entwicklung einer Desktop-Applikation wäre dank der *Node.js* Basis des Projekts mit dem Framework *Electron* realisierbar.

Literaturverzeichnis

[1] weitblicker.org. *Warum ist Bildung so ein wichtiges Thema?* 2019. URL: https://weitblicker.org/Warum-Bildung (Abruf vom 17. 04. 2019).

[2] dejure.org. *Art. 104c GG*. 2019. URL: https://dejure.org/gesetze/GG/104c.html (Abruf vom 15. 04. 2019).

[3] BMBF. *Wissenswertes zum DigitalPakt Schule*. 1. Jan. 2019. URL: https://www.bmbf.de/de/wissenswertes-zum-digitalpakt-schule-6496.html (Abruf vom 11. 04. 2019).

[4] Dennis Horn. *Digitalpakt Schule: Computer und Breitband allein helfen auch nicht › Digitalistan*. 26. Nov. 2018. URL: https://blog.wdr.de/digitalistan/digitalpakt-schule-computer-und-breitband-allein-helfen-auch-nicht/ (Abruf vom 11. 04. 2019).

[5] Maximilian Hett und Christina Sticht. *Tablet statt Tafel – zu Besuch in einer iPad-Schule | heise online*. 1. März 2019. URL: https://www.heise.de/mac-and-i/meldung/Tablet-statt-Tafel-zu-Besuch-in-einer-iPad-Schule-4337793.html (Abruf vom 14. 07. 2019).

[6] Apple inc. *iPad mini kaufen - Apple (DE)*. 12. Apr. 2019. URL: https://www.apple.com/de/shop/buy-ipad/ipad-mini (Abruf vom 12. 04. 2019).

[7] Pixabay. *Raspberry Pi 3*. 30. Sep. 2016. URL: https://cdn.pixabay.com/photo/2016/10/06/14/51/raspberry-pi-1719218_1280.jpg (Abruf vom 18. 07. 2019).

[8] Netzwerk Digitale Bildung. *Methodenpool*. 29. Juli 2019. URL: https://www.netzwerk-digitale-bildung.de/information/methoden/methodenpool/ (Abruf vom 30. 07. 2019).

[9] Ira Zahorsky. *Technische Ausstattung an deutschen Schulen ist mangelhaft*. 3. Jan. 2019. URL: https://www.egovernment-computing.de/technische-ausstattung-an-deutschen-schulen-ist-mangelhaft-a-787040/ (Abruf vom 04. 07. 2019).

[10] Ralf Koenzen und Susanne Ehneß. *Von der Kreidezeit ins digitale Zeitalter*. 3. Dez. 2018. URL: https://www.egovernment-computing.de/von-der-kreidezeit-ins-digitale-zeitalter-a-781172/ (Abruf vom 04. 07. 2019).

[11] Verband Bildung und Erziehung. *Ist an Ihrer Schule in allen Klassen- und Fachräumen ein Zugang zu schnellem Internet und WLAN verfügbar? [Chart]*. Hrsg. von STATISTA. 29. März 2019. URL: https://de.statista.com/statistik/daten/studie/1004594/umfrage/umfrage-zur-verfuegbarkeit-von-schnellem-internet-und-wlan-in-klassenzimmern/ (Abruf vom 26. 07. 2019).

[12] Sofatutor. *Digitaler Werkzeugkasten – Apps und Tools für den Unterricht*. 2. Mai 2016. URL: https://magazin.sofatutor.com/lehrer/digitaler-werkzeugkasten-apps-und-tools-fuer-den-unterricht/ (Abruf vom 07. 07. 2019).

[13] Klaudia Kachelrieß. *Datenschutz in der Schule | GEW-Berlin*. 7. Juli 2019. URL: https://www.gew-berlin.de/21168.php (Abruf vom 07. 07. 2019).

[14] Elke Witmer-Goßner. *Die Cloud als Lösung für DSGVO-geplagte Lehrer*. 30. Juli 2018. URL: https://www.cloudcomputing-insider.de/die-cloud-als-loesung-fuer-dsgvo-geplagte-lehrer-a-736737/ (Abruf vom 07. 07. 2019).

[15] TecArt-GmbH. *Vorteile browserbasierter Software - Webbasiert vs. Desktop*. 2019. URL: https://www.tecart.de/browserbasierte-software (Abruf vom 16. 04. 2019).

[16] Christian Safran, Anja Lorenz und Martin Ebner. „Webtechnologien-Technische Anforderungen an Informationssysteme“. In: *Lehrbuch für Lernen und Lehren mit Technologien* (2013).

[17] Wikipedia.org. *Intranet*. 30. Apr. 2019. URL: https://de.wikipedia.org/wiki/Intranet (Abruf vom 06. 05. 2019).

[18] Elektronik-Kompendium.de. *Client-Server-Architektur*. 2019. URL: https://www.elektronik-kompendium.de/sites/net/2101151.htm (Abruf vom 06. 05. 2019).

[19] AS-Computertraining GbR. *XML & HTML Unterschiede - Wissenswertes zu Syntax & Deklarationen*. 23. Jan. 2018. URL: https://www.as-computer.de/wissen/unterschiede-html-und-xml/ (Abruf vom 07. 05. 2019).

[20] Thomas Bayer. *REST Web Services*. 2002. URL: https://www.oio.de/public/xml/rest-webservices.htm (Abruf vom 07. 05. 2019).

[21] Michél Neumann. „Entwicklung eines Cloud-Service und einer Client-Anwendung unter iOS“. Diss. Hochschule Für Technik und Wirtschaft Berlin, 29. Juli 2015.

[22] Arun Gupta. *REST vs WebSocket Comparison and Benchmarks*. 24. Feb. 2014. URL: http://blog.arungupta.me/wp-content/uploads/2014/02/websocket-rest-messages-1024x517.png (Abruf vom 23. 07. 2019).

[23] 1&1 Ionis. *Webframeworks – Überblick und Klassifizierung*. 30. Jan. 2019. URL: https://www.ionos.de/digitalguide/websites/web-entwicklung/webframeworks-ein-ueberblick/ (Abruf vom 16. 04. 2019).

[24] Livivity. *Top Web Development Frameworks in 2019*. 15. Jan. 2019. URL: https://lvivity.com/top-web-development-frameworks (Abruf vom 11. 05. 2019).

[25] Statista. *Smartphone-Besitz bei Kindern und Jugendlichen in Deutschland im Jahr 2017 nach Altersgruppe*. 2017. URL: https://de.statista.com/statistik/daten/studie/1106/umfrage/handybesitz-bei-jugendlichen-nach-altersgruppen/ (Abruf vom 17. 04. 2019).

[26] Moritz Stückler. *Web-Bytecode: So sieht die Zukunft für Webassembly aus - Golem.de*. 1. Nov. 2018. URL: https://www.golem.de/news/web-bytecode-so-sieht-die-zukunft-fuer-webassembly-aus-1811-137389.html (Abruf vom 02. 06. 2019).

[27] SMART. *SMART Learning Suite Online*. 12. Juli 2019. URL: https://suite.smarttech.com/login (Abruf vom 12. 07. 2019).

[28] Smart Technologies. *SMART Service Region*. 2019. URL: https://support.smarttech.com/docs/software/smart-learning-suite-online/en/smart-service-region/default.cshtml?cshid=service-region (Abruf vom 09. 07. 2019).

[29] Promethan Limited. *ClassFlow Website Privacy Policy*. 2017. URL: https://classflow.com/privacy-policy/ (Abruf vom 09. 07. 2019).

[30] Promethean Limeted. *Kollaboratives Lernen, jetzt in der Cloud | ClassFlow*. 2019. URL: https://classflow.com/de/ (Abruf vom 27. 07. 2019).

[31] Plickers Inc. *Plickers*. 28. Juni 2019. URL: https://get.plickers.com/ (Abruf vom 03. 07. 2019).

[32] Poll Everywhere Inc. *Live interactive audience participation | Poll Everywhere*. 23. Feb. 2018. URL: https://www.polleverywhere.com/ (Abruf vom 03. 07. 2019).

[33] Juriy Zaytsev. *ECMAScript 6 compatibility table*. 3. Juli 2019. URL: https://kangax.github.io/compat-table/es6/ (Abruf vom 03. 07. 2019).

[34] Node.js foundation. *Node.js - a JavaScript runtime built on Chrome's V8 JavaScript engine.* 2019. URL: https://nodejs.org/en/ (Abruf vom 07. 06. 2019).

[35] npm inc. *npm | build amazing things.* 13. Jan. 2017. URL: https://www.npmjs.com/ (Abruf vom 04. 05. 2019).

[36] Socket.IO. *Socket.IO - real-time bidirectional event-based communication.* 2019. URL: https://socket.io/ (Abruf vom 03. 07. 2019).

[37] Sascha Depold. *Sequelize promise-based Node.js ORM.* 13. März 2019. URL: https://sequelize.org/master/ (Abruf vom 01. 08. 2019).

[38] L. Rabe. *Mobile Internetnutzer - Anteil in Deutschland 2018 | Statista.* 17. Juni 2019. URL: https://de.statista.com/statistik/daten/studie/633698/umfrage/anteil-der-mobilen-internetnutzer-in-deutschland/ (Abruf vom 23. 07. 2019).

[39] FreeHTML5.co. *Free Website Templates, Free HTML5 Templates Using Bootstrap Framework.* 2019. URL: https://freehtml5.co/ (Abruf vom 03. 06. 2019).

[40] Browserify. *Browserify - require('modules') in the browser.* 2019. URL: http://browserify.org/ (Abruf vom 03. 07. 2019).

[41] Evan You. *Vue.js - The Progressive JavaScript Framework.* 2019. URL: https://vuejs.org/ (Abruf vom 02. 05. 2019).

[42] Shaumik Daityari. *Angular vs React vs Vue: Which Framework to Choose in 2019.* 13. Juni 2019. URL: https://www.codeinwp.com/blog/angular-vs-vue-vs-react/ (Abruf vom 14. 07. 2019).

[43] ZingSoft inc. *Commercial JavaScript Charts - Licensing Options | ZingChart.* 14. Juli 2019. URL: https://www.zingchart.com/pricing (Abruf vom 14. 07. 2019).

[44] ZingSoft inc. *Wordcloud | ZingChart.* 2019. URL: https://www.zingchart.com/docs/chart-types/wordcloud (Abruf vom 06. 06. 2019).

[45] Lets-Encrypt.org. *Erste Schritte - Let's Encrypt - Freie SSL/TLS Zertifikate.* 2019. URL: https://letsencrypt.org/de/getting-started/ (Abruf vom 19. 07. 2019).

[46] Alex Ali. *Let's Encrypt vs Paid SSLs: What's the Difference (And Which Should You Use)?* 7. Jan. 2019. URL: https://www.a2hosting.com/blog/lets-encrypt-vs-paid-ssls/?aid=1656214 (Abruf vom 26. 07. 2019).

[47] Flavio Copes. *How to create a self-signed HTTPS certificate for Node.js.* 8. Sep. 2018. URL: https://flaviocopes.com/express-https-self-signed-certificate/ (Abruf vom 19. 07. 2019).

[48] Marlo Janschltz. *jade | t3n – digital pioneers.* 8. Sep. 2015. URL: https://t3n.de/news/jade-638027/jade-2/ (Abruf vom 01. 08. 2019).

[49] Twitter inc. *Bootstrap · The most popular HTML, CSS, and JS library in the world.* 1. Jan. 2019. URL: https://getbootstrap.com/ (Abruf vom 01. 08. 2019).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Anhang

Tab. 1: Funktionale Anforderungen an die Projektsoftware

ID	Name	Beschreibung	Status
F01	Datenbankanbindung	Es soll erfolgreich eine Anbindung an die Datenbank erfolgen	OK
F02	Datenbank Generierung	Alle nötigen Modelle und Tabellen soll in der Datenbank abgebildet werden und bei bedarf gänzlich neu generiert werden	OK
F03	GET Requests	Der Server soll in der Lage sein GET Requests entgegenzunehmen und zu beantworten. Im Fehlerfall soll auch eine Antwort erfolgen.	OK
F04	POST Requests	Der Server soll in der Lage sein POST Requests entgegenzunehmen und zu beantworten. Im Fehlerfall soll auch eine Antwort erfolgen.	OK
F05	Login System	Nutzende der Software in der Rolle als Lehrkraft oder Administrator sollen in der Lage sein sich bei dem System mittels Authentifizierung an- und abzumelden mittels HTTP-Session und Session-Cookies	OK
F06	MVC - Pattern	Das Model-View-Controller Muster soll beim Datenfluss im Backend/-Lehrerbereich des Servers reflektiert und zum Einsatz kommen	OK
F07	WebSockets	Bei der Ausführung von Lehreineheiten und den damit verbundenen Unterrichtsmethoden soll die Kommunikation zwischen Server und Web-Client über das WebSocket Protokoll erfolgen	OK
F08	Drei Client Implementierung	Um die Software größtmöglichst flexibel einsetzen zu können, sollen drei verschieden optimierte Web-Clients implementiert werden, genauer einen für die ausführende Lehrkraft, einen für Studierende, eine für Anzeigemedien, auf größere Darstellung optimiert im Unterrichtsraum	OK

Fortsetzung der Tabelle Funktionale Anforderungen an die Projektsoftware

ID	Name	Beschreibung	Status
F09	Administration	Authentifizierte Nutzer wie Dozierende und Administratoren sollen die Software verwalten können, dies umfasst u.A. das Anlegen und Freischalten von Nutzenden und das Setzen der Werkseinstellungen	OK
F10	Absicherung	Alle Routen die höhere Privilegien verlangen, sollen nur authentifizierten Nutzenden zugänglich gemacht werden, dies betrifft vor allem F09 und Besitzer abhängig erstellte Ressourcen	OK
F11	Ausführung von Lehreinheiten	Lehrkräften bzw. Dozierende soll es möglich sein zwei Typen von interaktiven Unterrichtsmethoden mit Studierenden durchzuführen, dies umfasst zunächst die Typen Brainstorming und Quiz	OK
F12	Einfacher Studierendenzugang	Studierende bzw. Schülerinnen und Schüler sollen über einen QR Code vereinfacht Zugriff auf den Server erhalten, vorausgesetzt sie sind mit dem gleichem Netzwerk wie der Server verbunden	OK
F13	Wifi/WLAN Hotspot	Der Server soll ein eigenes WLAN Netzwerk bereitstellen können in dem der Host-Computer des Servers als WLAN Access Point fungiert, und verbundenen Clients automatisch eine IP-Adresse zuweist (DHCP)	N.I.*

* nicht implementiert.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und dazu keine anderen als die angeführten Behelfe verwendet, die Autorenschaft eines Textes nicht angemaßt und wissenschaftliche Texte oder Daten nicht unbefugt verwertet habe. Die elektronische Kopie ist mit den gedruckten Exemplaren identisch.

Berlin, 05. August 2019,

(Ort, Datum, Unterschrift)