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:	29.07.2019
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 bewährte und neue Unterrichtsmethoden sinnvoll durch digitale Technik unterstützt werden und der finanzielle Rahmen der zur Verfügung stehenden Mittel, gerade im Zuge des Digitalpakts Schule optimal genutzt werden? Ist Datenschutz gewährleistet?

Viele Anbieter setzen auf Cloudlösungen, welche eine Internetanbindung zur Nutzung obligatorisch macht. Schulen mit noch ausbaufähiger digitaler Infrastruktur bedarf es jedoch an skalierbaren Hard- und Softwarelösungen, die zur Not auch offline einsetzbar sind, um auch jetzt schon kosteneffizient digitale Technik im Unterrichtsalltag zu integrieren.

Im Rahmen dieser Arbeit soll die gegenwärtige Situation in Sachen Digitalisierung hinsichtlich Soft- und Hardware sowie dem damit verbundenen Einsatz von Softwareanwendungen zu Unterstützung von interaktiven Unterrichtsmethoden analysiert werden. Anschließend soll darauf aufbauend eine Client-Server Web-Anwendung zur Lösung implementiert werden, 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
DARPA	Defense Advanced Research Projects Agency
ARPANET	Advanced Research Projects
WWW	World Wide Web
LAN	Local Area Network
WLAN	Wireless Local Area Network
WAF	Web-Application-Framework

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.

1.1.1 Besuch 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.

1.2 Problemstellung

Am 04.04.2019 trat die Änderung des Art. 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 auch 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 würde das eine Förderungssumme von ca. 500 Euro pro Schülerin bzw. Schüler bedeuten. 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 digitale Unterrichtsmedien wie interaktive Whiteboards.

Das Bundesministerium für Bildung und Forschung (BMBF) gegenargumentiert damit, dass kein digitales Medium alleine gute Bildung fördert, sondern immer dahinterstehende pädagogische Konzepte aus einer Vielfalt von Angeboten entscheidend sind. [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.

Der populäre Tablet Computer 'iPad' der Firma Apple inc. kostet in der günstigsten Variante bereits mindestens 449€ [5] (Stand April 2019), was schon knapp 90% des Förderungsvolumens pro Schülerin und Schüler ausmachen würde. Als ein Gegensatz wäre hier der Einplatinencomputer Raspberry Pi zu nennen, welcher bereits für 33 Euro erwerblich ist (Stand April 2019) und genug Rechenkapazitäten bereitstelle um zahlreiche Projekte im Bildungsbereich durchzuführen. Mit Touchscreenmodul und Schutzhülle liegt der Preis insgesamt bei ca. 150 Euro, was immer noch weniger als die Hälfte des Fördervolumens beträgt.

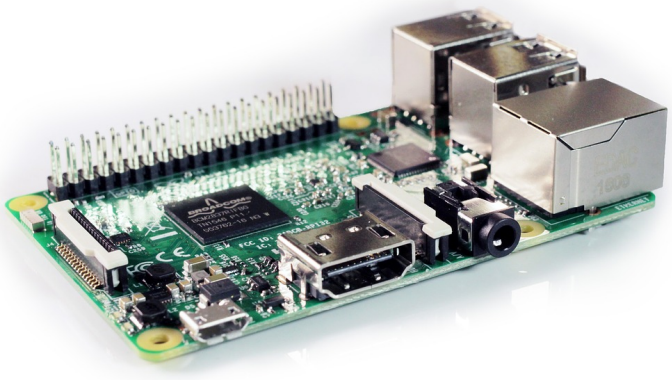

Abb. 1.1: Der Raspberry Pi 3 - Einplantinencomputer [6]

1.3 Zielsetzung

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[7]. 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 quasi Industriestandard 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.
Ein Raspberry Pi Einplantinencomputer bietet bereits genügend Leistung für Web-

[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

technologien und ein günstigen Anschaffungspreis. Auch besitzen bereits 67% der 10-11 jährigen Jugendlichen ein Smartphone [8] welches ebenfalls genug Leistung für Webanwendungen aufweisen.

Eine Softwarelösung zur Unterstützung von interaktiven Unterrichtsmethoden, welche auf Webtechnologien basiert, könnte den Rahmen der im Digitalpakt Schule fließenden Gelder optimierter ausschöpfen und Schulen finanzielle Flexibilität einräumen.

Diese Arbeit wird sich der Thematik von pädagogischen digitalen Konzepten und Varianten von interaktiven Unterrichtsmethoden nur im Rahmen der Softwareentwicklung widmen und ihre forschungsrelevante Tiefe nicht gänzlich erfassen, da dies den Rahmen der Zielsetzung überschreiten würde.

1.4 Aufbau der Arbeit

Im Anschluss an dieses Kapitel 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 nicht-kommerziellen 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.

2 Grundlagen

Dieses Kapitel soll einen grundlegenden Überblick über die zwei wichtigsten Themengebiet dieser Arbeit bieten, Schule und IT und der damit verbundene Einsatz von digitalen Unterrichtsmethodiken

2.1 Digitalisierung an Schulen

Die folgenden zwei 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.

2.1.1 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 1.3 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 Lehrender oder sogar die Hausmeisterin/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.

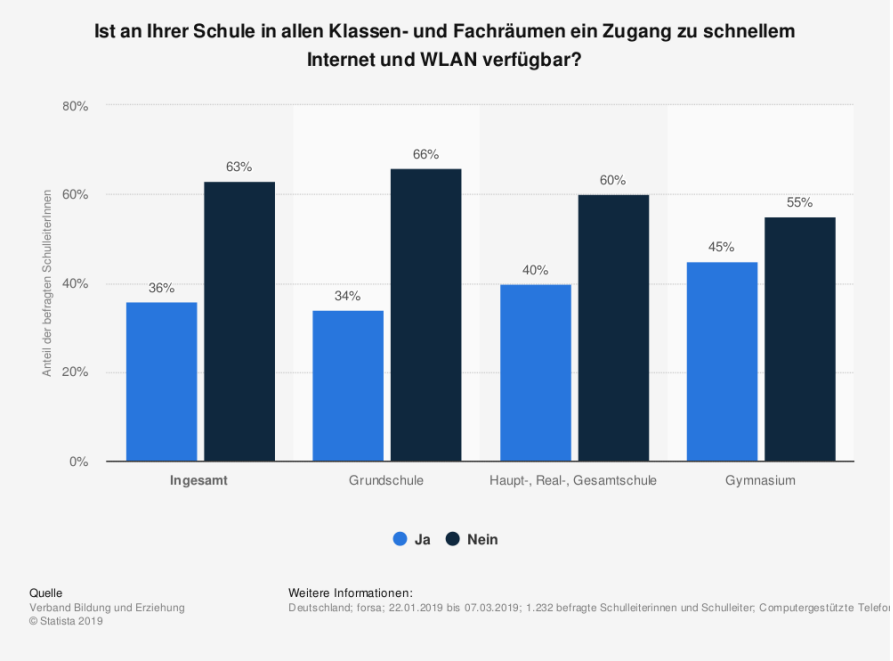

Abb. 2.1: Verfügbarkeit von schnellem Internet und WLAN in Klassenräumen: Bundesweiten Erhebung zum Thema Digitalisierung an allgemeinbildenden Schulen in Deutschland. [11]

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) scheint 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.

2.1.2 Ausblick digital gestützte interaktive Unterrichtsmethoden

Die erwähnte Technik im Abschnitt 2.1 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 eine umfangreiche Liste in der Sektion 'Interaktion' finden. Zum Beispiel lassen sich verschiedene Aufgabenformen ausmachen, die dann an einer digitalen Tafel von den Schülerinnen und Schülern gelöst werden sollen. Dies umfasst z.B. das Markieren, Sortieren, Zuordnen (Paare finden) von Bildern, Multiple Choice Aufgaben, Quiz Anwendungen, App-gestützte Spiele wie interaktive Lern-Rallyes (Rätsel, Herausforderungen und Medieninhalte können vielfältig miteinander verbunden werden), Brainstorming, u.v.m. Generell lässt sich feststellen, dass die Palette von Anwendungsmöglichkeiten enorm ist und viele klassische Konzepte, die sich bereits analog interaktiv durchführen lassen konnten, auch in einer digitalen Version bereitstehen. Beispielsweise können Unterrichtsmethoden wie ein Lern-Quiz sich auch mit Papier und Stiften durchführen, digitale Technik kann hier jedoch viel Arbeit abnehmen und lässt die Ausführung der Unterrichtsmethodik deutlich immersiver und medial interaktiver zu. So kann das Medium Film eingebettet werden, was analog nur sehr aufwändig möglich wäre. Abseits spielerischer Szenarien lässt die Mathematik Software GeoGebra das visualisieren von mathematischen Zusammenhängen zu. So können diese visualisiert und die Auswirkung von anderen Werten gezeigt werden. Das Konzept Bring-your-own-device, welches vorsieht das zu Unterrichtende eigene Geräte mitbringen, wie z.B. ein Smartphone, steigert das Maß von Interaktivität zusätzlich. So ist es möglich, dass eine ganze Lerngruppe oder gänzliche Schulklasse simultan einer interaktiven Unterrichtsmethode partizipiert. Ebenso kann ein Dozierender in einer Prüfungssituation auf Software zurückgreifen, welche die Prüfung des Kandidat unterstützt und eine anschließende Auswertung der Antworten wesentlich automatisiert und vereinfacht. An vielen deutschen Universitäten, wie auch der HTW Berlin, 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 Möglichkeiten der Anwendungen enorm ist. Eine bewusste Einbettung in den Unterricht kann ebendiesen positiv unterstütze und den Lehrenden entlasten. Als positiven Nebeneffekt lässt sich das Steigern von digitalen Kompetenzen seitens der Schülerinnen und Schülern vermerken, welche im Zuge der immer fortschreitenden Digitalisierung weltweit eine nicht zu unterschätzende Fähigkeit ist.

2.1.3 Datenschutz an Schulen

Am 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. Dies ist sofern nichts neues, da die Datenverarbeitung und Auskunftsrechte in §64 des SchulG (Schulgesetz) im Falle des Bundeslands Berlin geregelt sind und dieser weiterhin anzuwenden ist. Neu ist allerdings, dass die Verantwortlichen für die Verarbeitung von personenbezogenen Daten weitere Aspekte 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 2.1.1 genannte Auslagerung in eine Cloud könnte hier ebenfalls helfen, wenn der Cloud-Anbieter EU-Datenschutzverordnung konform arbeitet. Es ist auf jeden Fall ratsam, wenn Cloud-Anbieter und Server ihren Standort in Deutschland haben.

2.2 Überblick Webtechnologie

Diese Sektion soll einen grundlegenden über im Kontext dieser Arbeit wichtigen Begrifflichkeiten bieten. Die folgenden Untersektion 2.2.1 ff. werden die Thematiken nur grob umreißen, da eine detaillierte Betrachtung der genannten Begriffe den Rahmen dieser Arbeit weit überschreiten würde.

2.2.1 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 [17].
Der 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 Informationskultur.

Das Intranet beschreibt analog dazu ein lokal abgeschlossenes Netzwerk von Computern, bspw. innerhalb eines Unternehmens. Dabei endet ein Intranet klar an seinen Grenzen und ein Gateway fungiert als Übergabepunkt ins Internet. 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. [15].

2.2.2 Client-Server Modell

Das Client-Server Modell beschreibt das Prinzip der Kommunikation zwischen zwei Teilnehmer 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 Clientsoftware mit der die Verbindung zum Server aufgebaut wird. Im Fall des WWW (siehe 2.2.4) ist dies in den meisten Anwendungsszenarien ein Webbrowser. Der Client fordert dabei eine Resource an, welche auf dem Server vorliegt oder dort speziell für die Anfrage des Clients generiert wird (siehe auch Sektion 2.2.5). Das Client-Server Modell sieht vor, dass immer der Client die Verbindung aufbaut, nie andersherum [16]. 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.

2.2.3 Kommunikation

Die Kommunikation im Internet und Intranet erfolgt über Protokolle. Ein Protokoll kann als ein Satz von Kommunikationsregelvorschriften verstanden werden [17], 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. (Siehe Tabelle)
Das 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. Das OSI-Modell 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.

2.2.4 World Wide Web

Das World Wide Web (WWW) ist die wohl populärste Anwendung des Internets [17] 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 2.2.2) 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 2.2.3)ist das Hypertext Transfer Protocol (HTTP) bzw. in verschlüsselter Form Hypertext Transfer Protocol Secure, 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 2.2.2), 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.

2.2.5 Webanwendungen und Webservices

Im Laufe der Entwicklung des WWW (2.2.4) 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 [17]. 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. Die Fokus hier liegt auf das 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 Protocoll ausgetauscht (SOAP) und deren Beschreibung über die Web Services Description Language (WSDL) definiert. Anfrage- und Antwortformat der Daten ist XML (Extensible Markup Language), eine Auszeichnungssprache, welche HTML sehr ähnelt aber deutlich allgemeiner ist. XML kann als mehr als Regelwerk verstanden werden, mitdessen Hilfe Entwickler ihre eigene hierarische Beschreibung einer Datenstruktur vornehmen können. XML und HTML leiten sich bei der von der SGML (Standard Generalized Markup Language) ab, welches ihre Ähnlichkeit zusätzlich herleitet [18].

REST: (Representational State Tranfer) Hier kann jede einzelne Funktion des Webservices über eine jeweils zugeordnete URL abgerufen (Uniform Resource Locator) werden, umgangssprachlich als Webadresse bekannt. Das WWW selbst kann als REST-Webservice verstanden werden [19].

2.3 Websockets

Bezugnehmend auf die Problematik, welche durch die Kommunikationsstrategie über das http-Protokoll entsteht (siehe Sektion 2.2.4), 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 [20], 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.
Das folgende Bild zeigt einen Performanz Vergleich in Anbetracht des zusätzlichen

Payloads von REST- und WebSocket Nachrichten.

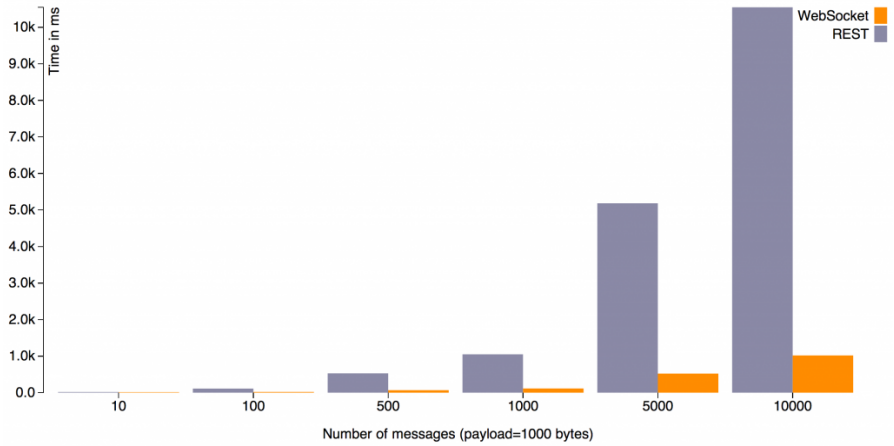

Abb. 2.2: Performanzvergleich REST versus WebSockets - Benötigte Zeit um N Nachrichten einer konstanten Nachrichtengröße zu verarbeiten. [21]

2.4 Webapplikationsentwicklung

Dieses Kapitel soll den wesentlicheren Bestandteil dieser Arbeit grundlegend beleuchten, der Entwicklung von Webapplikationen. Webanwendungen und Webservices können unter diesem Begriff zusammengefasst werden.

2.4.1 Web-Application-Frameworks

Bei der Entwicklung von Webapplikationen wird oftmals auf Frameworks (z.Dt. Rahmengerüste), spezifischer Web-Application-Framework (WAF) zurückgegriffen. Ein WAF bezeichnet damit ein Programmgrundgerüst, welches als Grundlage zum Einsatz kommt [22]. Dies erleichtert die Entwicklung ungemein, da auf bereits vorgefertigte Ansätze und Programmbausteine zugegriffen werden kann und diese nicht selbst von Hand implementiert werden müssen. Diese WAFs reflektieren zumeist auch eine Modelle und Prinzipien, welche, falls dem Entwickelnden bekannt, den Einstieg erleichtern. Ein für Frameworks bekanntes Paradigma stellt das Umsetzungsparadigma

Inversion of Control (IoC), z.Dt. Umkehrung der Steuerung da, welches u.a. auch in der objektorientieren 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 Programmfluss-

steuerung bei. Ein Nachteil, der durch dein Einsatz von einem WAF bedingt ist, stellt die Einschränkung der Freiheit während des Implementierungsprozesses da, dieser wird jedoch billigend in Kauf genommen, da sich ein Reduktion des Zeit- und Kostenaufwands erhofft wird. Die Wahl des richtigen WAF ist ein wichtiger Entschcheidungsprozess, bei dem mehrere Faktoren beachtet werden müssen, wie z.B. benötigte Einarbeitungszeit und Lizenzen.

2.4.2 Serverseitiger Ansatz

Anknüpfend an Sektion 2.2.5, 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 2.4.3 beschrieben.
Serverseitige Webapplikationen verfolgen oftmals den Multi-Page Ansatz, das heiß pro Anfrage (Request) wird eine anderes Dokument dem Client (Webbrowser) übergeben. Wichtige Programmiersprachen für den Ansatz sind php, Ruby, Python, Java und auch JavaScript, was vorher zunächst nur auf der Clientseite zur Anwendung kam.
Die **Model - View - Controller** Architektur (MVC) ist 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 beschreibt eine Datenstruktur an sich. In einem Webshop wären dies z.B. die Produkte und deren Eigenschaften wie ID, Name, Preis usw.
View: Diese beschreibt die reine Ansicht eines Dokuments. In einem Webshop wäre dies z.B. die Detailseite eines Produkts. Dabei sollte so wenig wie nötig Logik selbst im Code der View vorkommen.
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.
Es folgt eine Tabelle, die einen groben Überblick über bekannte WAFs, welche den serverseitigen Multi-Page Ansatz verfolgen [23].

Tab. 2.1: Überblick 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, die den Rahmen der Tabelle überschreiten würden. 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 4 ausführlich behandelt.

2.4.3 Clientseitiger Ansatz

Das Programmiermodell des WWW, welches durch die Architektur des Hypertext Transfer Protocol (HTTP) geprägt ist, wird bei der Entwicklung von Webapplikationen übernommen. Dies sieht eine Anfrage immer seitens des Clients vor (siehe auch Sektion 2.2.4). 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 zutrage. Hierbei wird o.g. Problem damit umgangen, indem bei Aufruf einer Internetseite die gesamte HTML Benutzeroberfläche inklusive Programmlogik in Form von JavaScript 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 [**safran2011webtechnologien:article**]. Jeder moderne Webbrowser hat einen JavaScript 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 Unternehmen

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. Es hat vergleichsweise eine steile Lernkurve und setzt etwas Einarbeitungszeit voraus.
React ist streng genommen kein WAF, sondern lediglich eine JavaScript Bibliothek. Es bietet aber über Erweiterungen die Möglichkeit, wie ein WAF genutzt zu werden, was seine Flexibilität noch erhöht. Entwickelt und Betrieben wird React von der Firma Facebook inc.
Vue ist ein clientseitiges JavaScript WAF, ursprünglich entwickelt von Evan You. Es gilt als einfacher zu erlenen als Angular und ist sehr flexibel.

Das Entwickeln von clientzentrischen JavaScript 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 1.3 erwähnte Google Docs, welches eine voll umfassende Textverarbeitungslösung im Browser bietet. Derartige Applikationen werden Rich Internet Application (RIA) genannt.

2.4.4 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].

2.4.5 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.

Tab. 2.2: Webapplikationsentwicklung im Vergleich [7]

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

3 Analyse

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

3.1 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 [24] 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 als lokale Installation genutzt werden kann. Positiv hervorzuheben ist, dass bei der Cloud Variante der Nutzende vorab seine Server Region festlegt. 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[25]. 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, welches 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, Lehrender, Schülerinnen und Schüler nutzen das gleiche Gerät. Andere Anwendungen erfordern zusätzliche Clients, sprich Geräte wie Smartphones oder Computer, laufen also im Mehrbenutzerbetrieb. Ebenso können Lehrende Prüfungsaufgaben erstellen und diese dann Abfragen und Auswerten. Eine strikte Unterscheidung zwischen Lehrer-, Klassen-, und Studierendenansicht findet nicht

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

statt. Ein großer Anspruch der Software ist, dass ein Lehrender ein ganzes Set an Aktivitäten für den Unterricht erstellen kann und dieses schrittweise durchlaufen wird. Es kann bspw. mit einem Test begonnen werden, anschließend erfolgt ein Folie mit einem Begriff und darauffolgend wird ein interaktive Unterrichtsmethode ausgeführt usw.

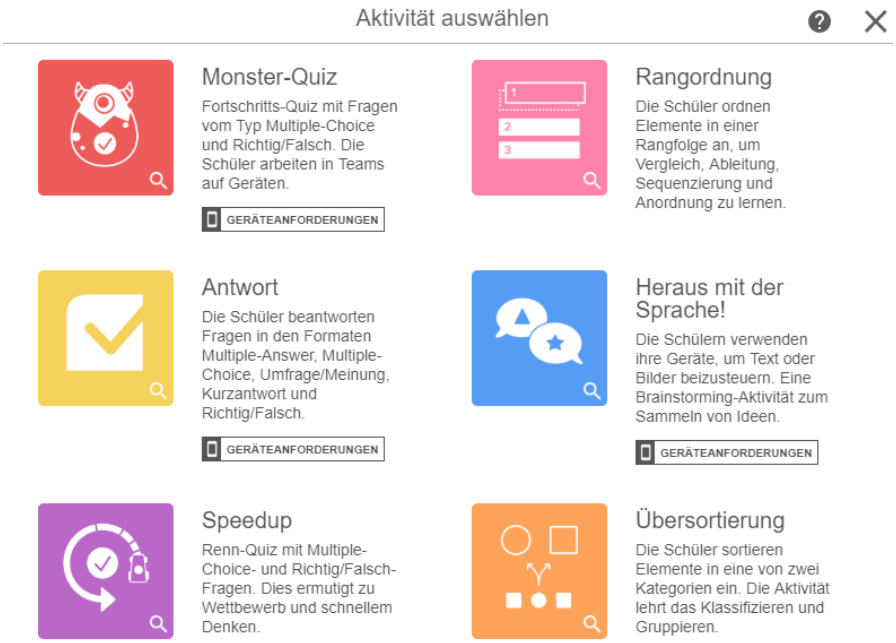

Abb. 3.1: Screenshot SMART Learning Suite Online:
Im Bild ist die Maske zur Aktivitätserstellung zu sehen.
Benötigen die Schüler ein Endgerät, so wird dies gekennzeichnet [24]

ClassFlow Ähnlich zu SMART Learning Suite Online ist ClassFlow eine Software, welche das Durchführen von interaktivem Unterricht ermöglicht. Der 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 der Anbieter Promethean Limited nicht[26]. Es können auch die meisten interaktiven Unterrichtsmethoden, in ClassFlow Aktivitäten genannt, 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. 3.2: Screenshot ClassFlow:
Aktivitäten können i.d.R. am gleichen Gerät ausgeführt sowie an Endgeräte der Schüler geschickt werden. [27]

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. Man kann Google Classroom allerdings auch privat mit einem Google Konto nutzen, wenn explizt angegeben wird, dass die Software nicht in einer Bildungseinrichtung genutzt wird. Bildungsreinrichtungen 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. Eine analoges Softwareangebot besteht auch für Unternehme namens G Suite. 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 eigenen 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 beschränke, hierbei jedoch oftmals auch interessante Ansätze zu finden sind. Gerade wenn Dozierende unkompliziert eine bestimmte interaktive Unterrichtsmehthode im Unterricht einsetzen möchte, bietet es sich an auf einen kleineren Anbieter zurückzugreifen. Zu nennen wäre hier z.B. die Software **Plickers**, welche das Prinzip bring-your-own-device etwas abändert. Die Schülerinnen und 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**, hier ist allerdings ein Endgerät pro Schülerin bzw. Schüler notwending. Wer nur teilnimmt muss jedoch keine Applikation installieren, hier reicht ein spezieller Link der in einem Webbrowser aufgerufen wird.

3.1.1 Gegenüberstellung

Nach der Präsentation der ausgewählten Angebote in Abschnitt 3.1 werden diese in folgenden Hauptkriterien miteinander verglichen:

1. **Serverstandort**: Werden die Server des Anbieters innerhalb der Europäischen Union betrieben, sodass datenschutzrechtliche Regelungen dieser 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 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? (Single)

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 lassen sich nutzen? Bei mehr als zwei wird hier die reine Zahl genannt.

Tab. 3.1: Tabellarischer Vergleich existierender Plattformen

Produkt	SMART LSO	ClassFlow	Google Classroom	Plickers	Poll Everywhere
Serverstandort EU	Ja*	Nein	Nein	Nein	Unbekannt
Online Nutztung	Ja	Ja	Ja	Ja	Ja
Offline Nutzung	Ja	Nein	Nein	Nein	Nein
Betriebsmodus	Single/Set	Single/Set	Single	Single	Single
Clients	2 (Lehrer/Schüler)	2 (Lehrer/Schüler)	2 (Lehrer/Schüler)	2 (Lehrer/Schüler)**	2 (Lehrer/Schüler)
Registrierung	Ja/Nein	Ja/Ja	Ja/Ja	Ja/Nein	Ja/Nein
Unterrichtsmethoden	12	10	Frage/Quiz	Quiz	Umfragen

* Option ist innerhalb der Software wählbar.
** Der 'Client' für die Schülerinnen und Schüler stellt in diesem Fall Papier mit aufgedruckten QR Codes da.

Aus der Tabelle lässt sich erschließen, 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 den Betrieb als Software vor, die eine gesamte Unterrichtseinheit als solche unterstützt und in ihr eine leitende Funktion einnimmt. Es können einzelne Methoden gestartet werden, doch die hauptsächliche Stärke findet sich in dem Erstellen von Sets, welche dass iterative Abarbeiten von Folien und interaktiven Unterrichtsmethoden vorsieht. Die zwei größten Vorteil von SMART Learning Suite Online sind einerseits der verfügbare Einsatz von Offline Applikationen, welche allerdings installiert werden müssen, was die genannten Nachteile aus Sektion 2.4.5 mit sich bringt, und dem Serverstandort innerhalb der EU, welcher explizit festgelegt werden kann. ClassFlow bietet dafür noch mehr Interaktion auch außerhalb der Ausführung von interaktiven Unterrichtsmethoden, da der Lehrende z.B. jederzeit das aktuelle Tafelbild an die angeschlossenen Clients der Schülerinnen und Schüler schicken kann. Dabei kann auch zwischen Gruppen unterschieden werden. Wird das Tafelbild von einem Client verändert, kann der Lehrende dies akzeptieren und z.B. als neue Grundlage für den weiteren Unterricht verwenden. Schülerinnen und Schüler können die SMART Learning Suite Online auch anonym nutzen, während dies bei ClassFlow nicht möglich ist, wobei dieses vorgehen ohnehin der Betriebsart hinderlich sein könnte. Die Anbieter Plickers und Pull Everywhere sind fokussierter auf das Anbieten einer interaktiven Unterrichtsmethode. Beide Angebote können auch gut von Unternehmen und anderen Nutzenden während Präsentationen und Vorträgen genutzt werden, die eindeutige Nutzung innerhalb einer Bildungseinrichtung ist nicht gegeben aber auch bewusst nicht gewünscht. 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.

Dieser Punkt soll in der zu entwickelnden Software gelöst werden. Keine 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.

Der Vergleich zeigt folgende wünschenswerte Eigenschaften für die im Rahmen dieses Projektes zu implementierende Software auf:

- 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 ermöglicht werden, welcher selbstständig ein Netzwerk aufbaut und Clients die Verbindung ermöglicht. Ein Betrieb im Intranet würde ggf. das Aufkommen datenschutzrechtliche Fragen deutlich eindämmen.

- Ein zusätzlicher Präsenter Client, welche unabhängig und 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 interaktive Tafel angeschlossenen ist zur reinen Anzeige der relevanten Inhalte nutzen. Diese Anzeige kann zusätzlich auf die visuellen Ansprüche eine Großbildanzeige optimiert sein, sodass ein Betrachten auch aus weiterer Entfernung problemlos möglich ist. Je nach Situation können Server und Clients aber auch einfach auf derselben Maschine ausgeführt werden.

- 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 auch 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.

- Das Anbieten mehrere interaktiver Unterrichtsmethoden innerhalb der Software ist wünschenswert, um die Nutzung und Inbetriebnahme attraktiver zu machen und die Fragmentierung der Inanspruchnahme mehrere Anbieter seitens der Nutzenden einzudämmen.

3.2 Systembeschreibung

Aus Gründen der Lesbarkeit wurde im folgenden Abschnitt die männliche Form gewählt, nichtsdestoweniger beziehen sich die Angaben auf Angehörige beider Geschlechter.

Die Software soll als Web Server-Applikation implementiert sein. Diese soll skalierbar sein, d.h. ein Betrieb rein im Intranet ohne großen Installationsaufwand soll möglich sein als auch der Betrieb auf einem entfernten, via Intranet oder Internet angebundenen Server. Die Software soll auch ohne aktive Internetanbindung im Intranet (resp. LAN) nutzbar sein. Über zwei Web-Client Lösungen kann mit dem Server interagiert werden, eine für Lehrende, eine für Schüler. Ein dritter Client, welche unabhängig fungiert, 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 (auch Set genannt) zu erstellen, diese zu starten sowie zu beenden. Während einer aktiven Lehreinheit, ist es Dozierenden möglich, diese zu leiten. (Fortschritt, Verbundene Schüler/Studenten verwalten, Speichern, je nach Typ der Lehreinheit.) Eine Lehreinheit besitzt eine oder mehrere interaktive Unterrichtsmethodiken softwareseitig umgesetzt. Als erste Umsetzung erfolgt in diesem Projekt die Implementierung eines Brainstorming 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 die zur Laufzeit einer aktiven Lehreinheit und seiner interaktiven Unterrichtsmethode relevanten Informationen an. Dieser ist zur Anzeige auf einem Großflächenanzeigegerät ausgelegt (Projektor, Fernseher, Smart Board). Es soll problemlos möglich sein, mehrere Präsenter Clients anzukoppeln.

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.

3.3 Zielgruppe

Das Software-Projekt soll sich in erster Linie an Bildungseinrichtungen jeglicher Art und deren Dozierenden richten, welche eine lokal ausgeführte Softwarelösung für das Durchführen von interaktiven Unterrichtsmethoden bevorzugen. Darüber hinaus auch an jegliche, die digital gestützte interaktive Lern- und kompetitive Kleinstspiele nutzen möchten. Dabei ist eine flexible Skalierbarkeit des Servers gegeben (siehe Abschnitt 3.5). Des weiteren ist das Software-Projekt attraktiv für die Open-Source Community, welche das Projekt weiter ausbauen kann sowie neue Typen von Lehreinheiten (interaktive Unterrichtsmethode softwareseitig umgesetzt / 'Spiel'-Art) hinzufügen kann.

3.4 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 derselbe Maschine ausgeführt) sowie im Intranet (LAN, Server und Clients auf unterschiedlichen Maschinen 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. Ebenso wird bei dem Prototyp vermindert Augenmerk auf das Design der Client Anwendungen gelegt, worunter die Usability jedoch nicht leiden soll. Das Design wird leicht anpassbar sein. Wie in Abschnitt 3.2 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 Datenbank, Management verbundener Clients etc.) wird bereitgestellt. Zunächst wird pro Lehreinheit (Set) nur eine interaktive Unterrichtsmethode zugewiesen werden können.

3.5 Systemanforderungen

Aufbauend auf das Analysekapitel dieser Arbeit (Abschnitt 3 ff.) sowie dem Abschnitten 2.1 und 2.1.3 lassen sich funktionale und nicht-funktionale Anforderungen

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

an das System ausformulieren, welche in den folgenden zwei Abschnitten gelistet werden.

3.5.1 Nicht Funktional

Die Erstellung der nicht funktionalen Anforderungen an das System wurde aufbauend auf dem Vergleich existierender Plattformen aus Abschnitt 2.4.5 angefertigt, wobei noch weitere für das Projekt als wichtig erachtete Aspekte mit eingeflossen sind.

Tab. 3.2: 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

3.5.2 Funktional

Aus Gründen der Übersicht wurden die Funktionale Anforderungen in den Anhang dieser Ausarbeitung verschoben. Diese wurden aufbauend auf den Erkenntnissen des Vergleiches existierender Plattformen aus Abschnitt 2.4.5 gebildet und ausformuliert. Hierbei wurde auch der Implementierungsstatus mit vermerkt, welcher chronologisch erst nach **Kapitel 5**, der eigentlichen Implementierung, vorgenommen wurde.

3.6 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.

3.6.1 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 (Querverweis zu Abschnitt 1) soll hierbei als Mindestanforderung definiert sein. Durch die Nutzung des Serverwebframeworks '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.

3.6.2 Client

Der im Rahmen dieses Projekts zu entwickelnde Client Software kann wie in Abschnitt 3.2 erläutert, in drei Parts eingeteilt werden. Die Verwaltung im Backendbereichs der Server Software soll eine besonders niedrige Hardwareanforderung aufweisen, da hier gänzlich auf den Einsatz von JavaScript verzichtet werden wird. Dies soll eine Server-Verwaltung auch bei deaktiviertem JavaScript gewährleisten. Die restlichen Software Module sollen in jedem modernen Webbrowser auf jedem

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

Endgerät lauffähig sein. Der 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)[28].

4 Konzept

Das folgende Kapitel soll Gedanken zur Realisierung des Projekts widerspiegeln. Dies umfasst den allgemeinen Systemaufbau, sowie Entwürfe und Architekturvorstellungen hinsichtlich der Server- und Clientseite und die damit verbunden eingesetzten Softwaremodule.

4.1 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 eventuelle Wartungsaufgaben und Aufgaben wie das Starten des Servers, soll die Steuerung über eine Kommandozeile möglich sein. Alle anderen Interaktionen werden über den Client ausgeführt.

4.2 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 2.2.4). 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.

Schlussfolgernd aus Systemaufbau und Netzwerkaufbau lässt sich folgende Abbildung skizzieren:

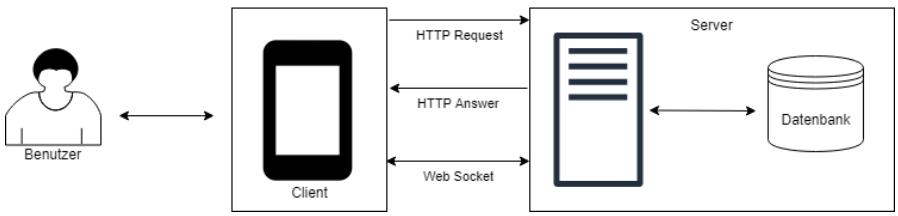

Abb. 4.1: Kommunikationsaufbau des zu entwickelnden Systems

4.3 Entwurf des Servers

Aufbauend auf das Grundlagen und Analysekapitel sollen in diesem Abschnitt die Entwurfsgedanken hinsichtlich der Serverkomponente des Projekts widergespiegelt werden.

4.3.1 Laufzeitumgebung: Node.js

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

1. Node.js nutzt als Paketmanager und Projektverwaltungstool **NPM** (Node Paket Manager). Mit dieser Software ist der Zugang zu über 350.000 Paketmodulen (Stand 13. Januar 2017) gegeben und diese können das Projekt Modular erweitern. 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 4.4.2). Eine gute Skalierbarkeit ist ebenfalls gegeben. Dies wird in folgenden Abschnitten genauer erläutert. Ist ein besonders hohen Ressourcenbedarf von Nöten (z.B. eine Bildungseinrichtung möchte einen zentralen Server installieren, welche viele Klassen/Kurse bedienen soll) können mehrere Serverinstanzen auf einer Maschine parallel laufen und vorab mit einem Lastenverteiler (Load Balancer) Server, wie z.B. NGINX verwaltet werden. Durch die genannte Argumentation soll das Projekt mit NodeJS realisiert werden und nicht mit einem php-Framework.
Da Node.js grundlegend sehr offen ist was seinen Einsatzzweck betrifft, soll als Webserver Modul das Node-Paket **Express.js** genutzt werden, welches im nächsten Abschnitt genauer erläutert wird.

4.3.2 Webserver: Express

Um mit Node.js komfortabel eine Webapplikation zu implementieren, soll das bekannte Web-Framework Express eingesetzt werden, welches viele HTTP-Dienstprogrammmethoden und den Einsatz von Middlerwarefunktionen gestattet. Hierbei wird jeder eingehende HTTP Request 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 soll für den gesamten administrative Teil des Lehrer-Login zum Einsatz kommen. Ebenso soll durch Express das Anlegen und Editieren von Lehreinheiten möglich sein (vgl. 3.2). Da für den gesamten Lehrer-Backendbereich Express zum Einsatz kommen soll und hier mit einfachen HTTP-Requests gearbeitet wird, kann auf der Einsatz von JavaScript auf der Client-Seite auf ein Minimum reduziert werden, was den Einsatz auf Servereinheiten mit nicht modernem Webbrowser entgegenkommt (Sollte Client und Server auf der gleichen Maschine ausgeführt werden).

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

4.3.3 SocketIO

Die JavaScript-Bibliothek Socket.IO ermöglicht bidirektionale Echtzeit-Kommunikation zwischen Webclient und Server, wobei dabei Bibliothek sowohl auf Server- wie auch 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. Für das Anmelden von Schülerinnen und Schülern, das Durchführen von interaktiven Unterrichtsmethoden soll SocketIO zum Einsatz kommen. Hierzu soll Express die entsprechenden Client Daten auf einer festgelegten Route senden und anschließend die Kommunikation von SocketIO kontrolliert werden. Da die Nutzung der Software rein im Intranet nutzbar ist und Nutzende über ein WLAN Zugriff erhalten können, ist der im Abschnitt 2.3 genannte Nachteil von erhöhtem Datenverkehr zu vernachlässigen.

4.3.4 Sonstige Module

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

- **Body-Parser**: Diese Modul ermöglicht das einfach Auslesen von HTTP-Requests möglich. Schickt ein Client bspw. Formulardaten können diese

einfach gelesen und ausgewertet werden. Dies soll im Backendbereich der Lehrenden oftmals die Praxis sein.

- **express-session**: Da Lehrende und Administratoren zur Nutzung der Software einen gültigen Zugang besitzen müssen, ist zur Authentifizierung des Nutzenden der Einsatz von Sessions vorgesehen (Querverweis Abschnitt2.2.4). 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 Wahl der Datenbank).

- **Pug**: Die Template Engine Pug besitzt seine eigene Syntax und macht das Entwerfen und Schreiben von HTML Templates, welche serverseitig übermittelt werden, sehr komfortabel. Zusätzlich werden Funktionalitäten wie Vererbung und Mixins unterstützt. Ein Einsatzbeschreibung erfolgt in Kapitel Implementierung. Pug soll für sämtliche zu übertragende HTML Dokumente zum Einsatz kommen.

4.3.5 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. Grundlegen können Datenbanksysteme in zwei Kategorien unterteilt werden:
SQL und **noSQL** Systeme.
SQL Systeme speichern ihren Daten in sogenannten Relationsmodellen, welche als Tabelle visualisiert werden können. Hierbei beschreibt der Tabellenkopf den Datensatz und seine Datentypus (jede Spalte für sich) während Zeilen eine Entität (Eintrag) in der Datenbank beschreiben. Vorteil hierbei ist, dass die Daten konform sind, d.h. bei Zugriff liefert immer einen Rückgabewert [20]. 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 potentere Server gekauft werden müssen. Mehrere Relationsmodelle können über Fremd-Schlüsse (Querverweise) miteinander verbunden werden, um auch komplexere Sachverhalte abbilden zu können.

NoSQL lassen sich in verschiedene Subkategorien je nach Arbeitsweise beim Speichern der Datensätze einteilen[20]. Am populärsten sind 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 definiert im Vergleich zu SQL Systemen. Dies räumt eine große Flexibilität beim Speichern von Daten ein, da Datensätze auch unvollständig gespeichert werden müssen. 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 Modell Teil von zu bereitstellenden Daten auch wie diese über welche Funktionalität aus der Datenbank geladen werden sollen. Den Controller soll nur die vom Modell bereitgestellten Funktionalitäten nutzen und keine direkten Datenbankzugriffe selbst ausführen. Damit die Software im hohem Maße skalierbar bleibt, ist der Einsatz eines sogenannter Object-Relationship-Mapper, kurz ORM, (Objektrelationale Abbildung) vorgesehen, der an verschiedenste Datenbanksysteme angebunden werden kann. Da das Projekt in seiner kleinsten Skalierung lokal auf einem Einplantinencomputer wie dem Raspberry Pi 3 und im lokal im Intranet laufen können soll, ist für den Anfang die Verwendung eines Datenbanksystems, welches vollständig durch eine Programmbibliothek lauffähig ist, vorgesehen. 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 gemeinfreie 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 mit einhergeht.
Als ORM fiel die Wahl auf das Node.js Modul Sequelize, welches neben SQLite mit viele andere bekannte SQL Datenbanksystemen zusammenarbeiten kann, u.A. Postgres, MariaDB und Microsoft SQL Server. 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 eingeloggten Lehrenden in das System. Dazu werden entsprechende Tabellen zur Verwaltung der Sessions und deren Lebenszeit automatisch in der Datenbank via Sequelize angelegt. Zuvor sollen die Passwörter sicher gespeichert werden, d.h. nicht im Klar-Text, 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 bereites NoSQL Äquivalent gefunden werden konnte, fiel die Wahl 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 einhergeht, da dann auch der ORM gewechselt und die Modelle entsprechend angepasst werden müssten.

4.3.6 Server Architekturdiagramm

Schlussfolgernd lässt sich der finale Entwurf des Servers folgend visualisieren und wird in Kapitel **5 Implementierung** umgesetzt.

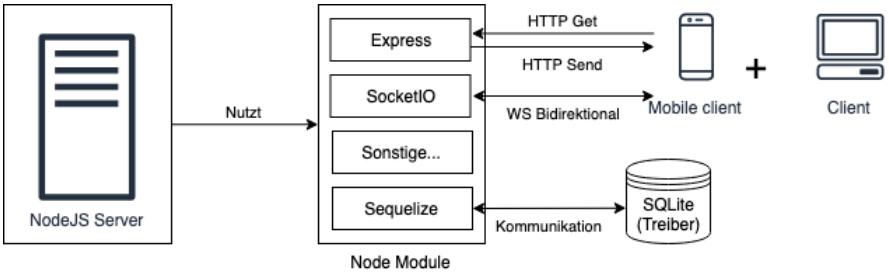

Abb. 4.2: Aufbau der geplanten Serverarchitektur

Hinweis: NodeJS Module aus der Sektion **Sonstige Module** wurden aus Gründen der Visualisierung in dieser Grafik nicht näher betrachtet.

4.4 Entwurf des Clients

Wie zuvor erörtert, ist es vorgesehen drei verschiedene Clients zu implementieren, welche alle auf dem gleichen Prototyp basieren sollen, allerdings verschiedene Zwecke verfolgen. Dies sieht 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

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

darauffolgendem Kapitel genannt.

4.4.1 Gedanken zu UI

Da das Projekt als Web-Applikation implementiert werden soll, wird die UI gänzlich durch die Stylesheet-Sprache CSS beschrieben. Im Jahr 2018 wurden erstmals häufiger mobile Endgeräte wie Smartphones zur Internetnutzung herangezogen als der klassische Computer oder Laptop[29]. Viele Design-Frameworks setzen daher schon länger auf das Prinzip 'Mobile First'. Da die Applikation letztlich sowohl auf Computern und Smartphones aber auch Großbildgeräten angezeigt werden soll, bildet ein flexibles Web-Design-Framework als Basis, welches bereits viele relevante Web-Design Standards wie das o.g. 'Mobile First' berücksichtigt, ein solides Fundament in Sachen Usability und Design. Eines der populärsten dieser Art ist **Bootstrap**, welches zusammen mit einem frei erhältlichen Design-Theme von `https://freehtml5.co/` genutzt werden soll. Das gewählte Theme soll an die Applikation angepasst und teils für die Großbild Anzeige des Presenter Clients optimiert werden.

4.4.2 Browserify

Um das Nutzen von Node.js Packages sowie damit verbundene `require()` Funktionalität auch auf der Clientseite zu ermöglichen, soll die JavaScript Bibliothek **Browserify** zum Einsatz kommen. Mit ihr können alle Node.js Packages, 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 JavaScript Datei zur Verfügung, die nur noch im HTML-Dokument eingebunden werden muss. Mithilfe der `require()` Funktionalität, welche von Node.js gestellt wird, kann der Code auch übersichtlicher in mehrere Dateien/Module aufgeteilt werden. Dies war ohne Aufwand auf der Browserseite nicht ohne weiteres möglich, ist aber durch die Einführung von Modulen seit ECMAScript[6] in Version 6 nun möglich. Oftmals wird aber aus Kompatibliltätsgründen zu älteren Webbrowsern auf Lösungen wie Browserify oder auch Babel gesetzt. Letztere übersetzt den geschriebenen JavaScript Code so, dass er auch von älteren Webbrowsern interpretiert wird (auch Transpiler genannt). Sämtliche folglich genannten JavaScript Module resp. Bibliotheken sollen via Browserify in eine

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

JavaScript Datei zusammengefasst werden und anschließend pro Client eingebunden werden. Das bringt den zusätzlichen Vorteil dass für jegliches, auf Browserseite eingesetztes JavaScript nur einen einzigen HTTP-Get Request benötigt, da wie erwähnt nur eine JavaScript Datei pro Client angefordert werden muss.

4.4.3 JavaScript Lösungen

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

- **VueJS**: Um das Anzeige, Editieren und Anpassen dynamischer Inhalte zu erleichtern, soll das JavaScript-Webframework VueJS zum Einsatz kommen, da dieses gut skalierbar ist und alle benötigten Funktionalitäten mit sich bringt. Im Vergleich zu AngularJS und React (siehe auch Abschnitt 2.4.3), 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[30]. VueJS ist auch gemessen an der Dateigröße von nur 80 KB deutlich kleiner im Vergleich zu Angular mit 500 KB.
- **SocketIO**: Das bereits in Abschnitt 4.3.3 erwähnte SocketIO besitzt ein Gegenstück, welches auf der Clientseite 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 (WordCloud), welche bei der interaktiven Unterrichtsmethode Brainstorming zum Einsatz kommt, und sonstigen Diagrammen soll die auf diese Szenarien spezialisierte JavaScript Bibliothek ZingChart genutzt werden. Die Software ist in einer freien Version erhältlich, wobei lediglich ein kleines ZingChart Logo stets angezeigt werden muss[31].

4.4.4 Client Architekturdiagramm

Nachfolgend die umzusetzende Architektur der Client-Anwendung.

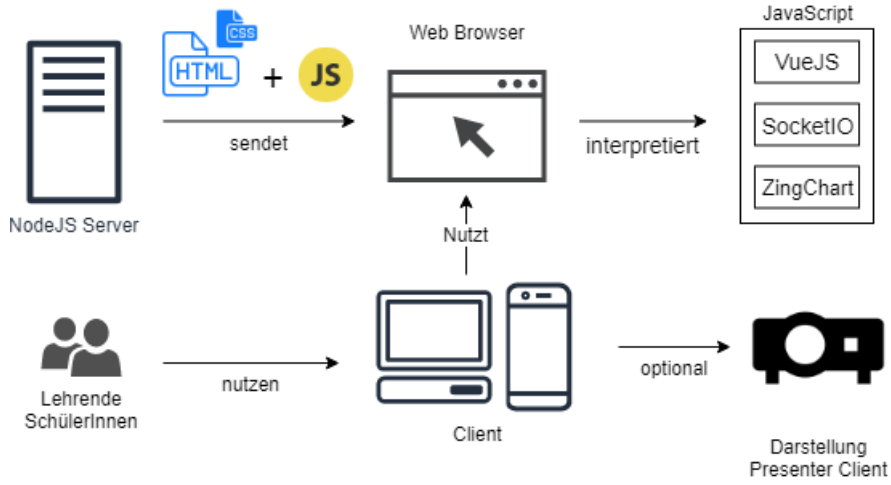

Abb. 4.3: Aufbau der geplanten Client Architektur

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 der gleichen Maschine ausgeführt werden (z.B. in unterschiedlichen Tabs eine Webbrowsers) als auch auf entfernten.

5 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.

5.0.1 Implementierung der Server-Software

Aufbauend auf das Entwurf Kapitel soll die Server-Software mit NodeJS und ExpressJS als Hauptkomponente entwickelt werden. Dazu wird zunächst im folgenden Abschnitt 5.0.2 der HTTP Server grundlegend konfiguriert und anschließend dessen Routen im darauffolgenden Kapitel 5.0.5 angelegt.

5.0.2 ExpressJS Setup

Nachdem das Projekt grundlegend mit dem Befehl `npm init` initialisiert wurde, kann ExpressJS einfach über den Node Packet Manager (nachfolgend NPM genannt) hinzugefügt werden. Zusätzlich wird das NPM Paket IP genutzt um die aktuelle IP-Adresse der Maschine zu automatisch ermitteln und den ExpressJS Server auf dieser lauschen zu lassen. Dies ist mit wenigen Zeilen Code erledigt:

```
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 Adapters der Maschine auf dem er ausgeführt wird, zusätzlich auf Port 3000, dies sollte 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.

5.0.3 Autarkes WLAN

Um einen Stand-Alone Betrieb[7] der Software zu ermöglichen, wird als Lösung der Betrieb eines unabhängigen kabellosen Netzwerkes (WLAN/Wifi) angestrebt.

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

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 Zugangspunkt für andere Geräte zu. Der Server wird mit diesem Netzwerk verbunden, sowie alle Clients. Dies 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 entwickeln. Diese könnte automatisch einen WLAN-Hotspot erstellen und den Netzwerkverkehr eventuell limitieren. Gleiches gilt analog für windows- oder unixbasierte Computer, ist aber problematischer, siehe dazu nächsten 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+ gibt es mit den NPM Paket `node-hotspot` auch die Möglichkeit, dies direkt mit NodeJS zu erledigen. Diese 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 5.0.5).

   Wird die Applikation in einer vorhandenen Netzwerkinfrastruktur betrieben, ist ein Betrieb in jedem Falle gewährleistet. Ein mobiler WLAN-Router könnte ebenfalls genutzt werden, sollte keine ausreichende Infrastruktur vorhanden sein. Dieser müsste einmalig konfiguriert werden und ist anschließend in der Lage die Serverapplikation für Clients ansprechbar zu machen. Ein solches Gerät gibt es bereits ab ca. 10 Euro zu erwerben.

5.0.4 Verschlüsselung

Um eine verschlüsselte Kommunikation zwischen Server und Clients zu gewährleisten, ist der Datenaustausch über das HTTPS Protokoll vorzuziehen. Um HTTPS allerdings sinnvoll nutzen zu können, ist ein Zertifikat von einer Zertifizierungsstelle

(CA) notwendig, was meist mit Kosten verbunden ist. Allerdings bietet der Anbieter **Let's Encrypt**[32] kostenlose Zertifikate an, welche sich problemlos bei vorhandenem SSH-Zugang installieren lassen. Für den Intranet Betrieb können relativ einfach eigens ausgestellte Zertifikate genutzt werden, welche mit Shell-Programmen wie **openssl** generierbar sind[33]. Allerdings warnen moderne Webbrowser den Nutzenden recht auffällig, dass die genutzte Verbindung dennoch nicht sicher ist, da dem Zertifikat nicht vertraut werden kann. 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.

5.0.5 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 spezial Routen für das Error-Handling (z.B. 404 - Seite nicht gefunden) sowie besondere für die WebSocket Kommunikation, welche aber im Hintergrund genutzt werden und im Abschnitt **??** beleuchtet 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 gilt: Die weiterführend Routen werden in Dateien ausgelagert, um die Übersicht des Quelltextes zu wahren. Ebenso sind auch diese Routen sog.

Middleware-Funktionen. Dies wird im nächsten Abschnitt genauer beleuchtet. Daher ist auch die Reihenfolge wichtig, würde die `'/'` Route als erstes angelegt werden, so würde diese alle folgenden, spezifischeren 'abfangen'.

5.0.6 Reflexion des MVC Schemas

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

Tab. 5.1: 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

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.

Folglich müssen die entsprechenden Controller-Funktionen an Routen gebunden werden. Dabei wurde sich teils am RESTful Design orientiert, die Umsetzung erhebt jedoch kein Anspruch vollkommen 'RESTful' zu sein. Zunächst müssen entsprechende Unterrouten zu den aus Abschnitt 5.0.5 bereits angelegten hinzugefügt werden. Zwecks der Übersicht wird hierzu ein Routes Ordner angelegt, welche die entsprechenden Router enthalten soll. Folgende Router werden angelegt:

routes/client.js: Legt alle Student Client und Presenter Client relevanten Routen fest.
routes/teacher.js: Alle für das Backend resp. Lehrerbereich relevanten Routes werden hier angelegt.
Es folgt ein Codebeispiel 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 Funktion `getStudent` des Controllers 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

5.0.7 Einrichtung der Datenbank

Die SQL Datenbank 'SQLite' und der Object-Relationship-Mapper 'Sequelize' können einfach über den 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. Es muss der Dialekt 'sqlite' angegeben werden und 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-Beispiel wird die Tabelle bzw. Sequelize Model 'student' im Modul tables konfiguriert.

```
exports.student = (sequelize, Sequelize) => {
  return sequelize.define('student', {
    id: {
      type: Sequelize.INTEGER,
      autoIncrement: true,
      allowNull: false,
      primaryKey: true
    },
    name: {
      type: Sequelize.STRING,
      allowNull: false
    },
  })
};
```

Listing 5: Anlegen einer Tabelle und deren Beziehungen

Im Datenbank Utility Modul wird nun die Konfiguration geladen und anschließend dessen Beziehung zu anderen Entitäten eingestellt. Durch Sequelize kann hier ein relativ humanes Sprachbild verwendet werden. Das folgende Beispiel zeigt die Beziehungen zwischen EduSession und Student an:

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

Listing 6: Konfiguration von Entitätsbeziehungen

Nach dem selben 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 JavaScript Funktionen. Jedes innerhalb Sequelize definierte Model bietet diese automatisch an. Als Beispiel könnte nun über `const studentToFind = Student.findByPk(1);` die Entität mit der ID 1 geladen werden. All diese bereitgestellten Funktionen sind JavaScript Promises, d.h. sie werden asynchron ausgeführt und es Bedarf entsprechendes Handeln im Fehlerfall. 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. Die in Tabelle 5.1 gelisteten Modelle werden gemäß der Arbeitsteilung des MVC-Schemas hauptsächlich direkt mit Sequelize arbeiten und den Controllern entsprechende Funktionen bereitstellen.

5.0.8 Implementierung des LehrerInnen Bereiches

Nachdem die aus Abschnitt 5.0.6 genannten Model-Module 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 exportieren soll. 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 benannten (GET, POST, etc.).

View Rendering mit PUG Für alle Views soll die Template Engine Pug zum Einsatz kommen (siehe auch Kapitel 4 Konzept). Dazu wird PUG für das Rendern aller nicht statischen Routen im Hauptmodul der Software registriert. PUG macht das Schreiben von HTML Dokumenten sehr komfortable und unterstützt Vererbung (Layouting). So wird zunächst für den LehrerInnen Bereich ein Main 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 von Views gefüllt werden, welche vom Main Layout erben. Die Controller können Daten an die Views übergeben, welche von diesen dann dargestellt werden. Dazu bietet PUG wie die meisten Template Engines Funktionen an, wie das Iterieren über Datensätze mittels Schleifen ermöglicht oder dynamisch generierte Inhalte abhängig von konditionalen Ausdrücken. Sämtliche an

die Clients ausgelieferte HTML Dokumente sollen von PUG on-demand generiert werden. PUG benutzt seine eigene Templating-Language, welche sich deutlich von dem bekannten 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.
Es werden anschließen für alle Modelle Views angelegt, damit Lehrende sich anmelden und einloggen sowie neue Benutzer anlegen und editieren können. Ebenso für das erstellen von Lehreinheiten vom Typ Brainstorming und Quiz. Dabei werden bei Dateneingabe seitens der Nutzende Formulare verwendet, welche anschließend via POST Request an den Server geschickt und dort von Controllern und ihren jeweiligen Modellen ausgewertet. Das tatsächliche Ausführen und dessen Entwicklung wird im später folgenden Abschnitt **?? Implementierung der Lehreinheiten** beschrieben.
HIER NOCH EIN LISTING?

UI Design Aufbauend auf den Abschnitt 4.4.1 des Entwurf-Kapitels wird zur Designumsetzung das Frontend-CSS-Webframework **Bootstrap** genutzt. Dieses kann ebenfalls einfach über NPM dem Projekt hinzugefügt werden. Als Design-Theme soll das frei erhältliche Bootstrap Theme `Neat` von **freehtml5.co** die Grundlage bilden. Dieses wird gänzlich im Backend-Bereich des Lehrkraftzugangs genutzt, angepasste Teile anknüpfend für den Student Client und den Presenter Client, welche insbesondere noch durch eigenen CSS Code für die Nutzung auf Großbildgeräten wie Fernsehern und Projektoren optimiert wird. Es wird sichergestellt dass sich die Software angenehm auf stationären wie mobilen Endgeräten nutzen lässt.

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 **Konzept** Kapitel genannte `express-session` NPM-Modul genutzt, welches bereits vollständig mit der von Sequelize verwalteten SQLite Datenbank einsatzbereit ist. Nach der Einrichtung im Hauptmodul wird eine eigene Middleware geschrieben, welche bei allen abgesicherten Routen als erstes aufgerufen wird und überprüft, ob die vom Client übertragene Session noch gültig ist.

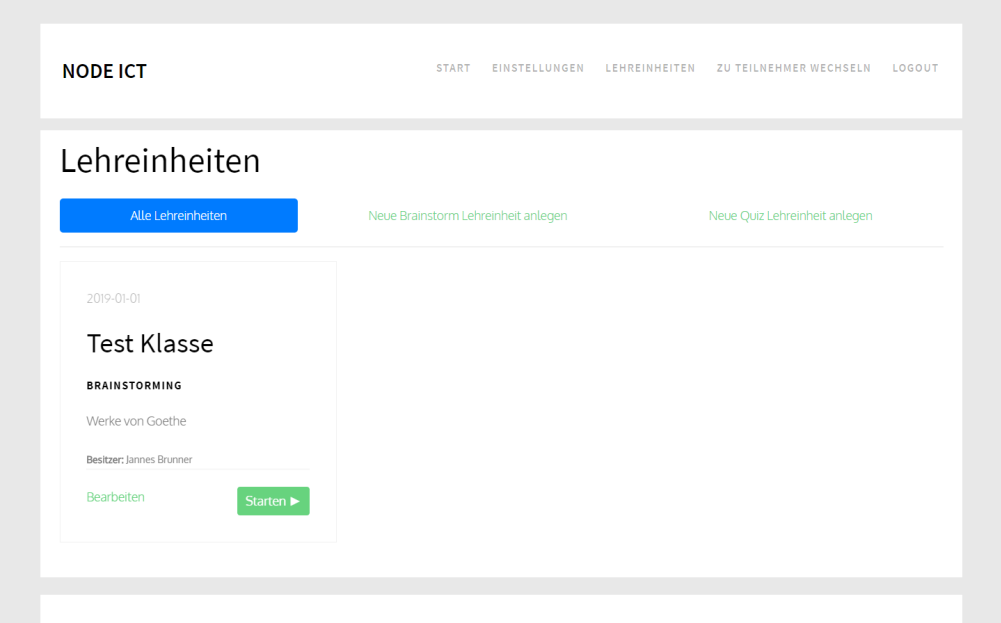

Abb. 5.1: UI Design der Applikation beispielhaft illustriert durch ein Screenshot des Lehreinheiten Bereiches im Backend Zugang des Teacher Clients.

```
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 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.

5.0.9 Umsetzung des Client Softwareanteile

Nachdem die Funktionalitäten Erste Initialisierung der Software, Login/Logout und 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 Hauptfunktionalität der Software und verlangt mehr Interaktion auf der Client Seite.

Die **Browserify** Software wird einfach über den NPM der Projekt hinzugefügt und ist sofort einsatzbereit. Alle im Abschnitt 4.4.3 des Konzeptkapitels erwähnten JavaScript Lösungen sind als NPM Packet verfügbar und werden ebenfalls dem Projekt hinzugefügt. Pro Client (Teacher Client, Student Client, Presenter Client)

wird ein Development-Modul angelegt. In diesem können alle benötigten JavaScript Bibliotheken normal importiert und genutzt werden. Via Browserify wird anschließend pro Client ein Production Modul generiert, 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 JavaScript 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 VueJS die angezeigten Informationen auf dem UI aktuell. VueJS ist dabei pro Client als einfaches JavaScript Objekt auch von außen ansprechbar, was die Schnittstelle für andere Bibliotheken, insbesondere SocketIO, bildet. **SocketIO** auf der Client-Seite ist für den gesamten Datenverkehr zwischen Server und Client verantwortlich. Sowohl auf Server- wie auch Client-Seite können 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 eintreffende Daten kümmert. Die ausgetauschten Daten müssen hierbei nicht zwangsläufig zuvor indas JSON-Format umgewandelt werden, wie dies sonst bei REST-Apis üblich ist.
Auf der Server-Seite kümmert sich das Modul `ioSocketHandler` 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 Lehreinheittyp (Brainstorming und 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 gehalten. Tritt nun eine Schülerin oder 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 (Session) 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` Modul der jeweiligen Lehreinheiten Klasseninstanz zugeordnet.

Es folgt ein Codeauszug welcher den Datenaustausch zwischen Server und Client zeigt.

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

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.

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

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.

5.0.10 Problemstellen der Implementierung

Während der Implementierung stellte sich anfangs das Verbindungsmanagement der verbundenen Clients über SocketIO als instabil heraus, da Sockets bei jedem Verbindungsabbruch sich zwar selbständig erneut verbinden, jedoch immer unter einer neuen Session ID. Dies führte zunächst zu unerwartetem Verhalten während einer ausgeführten Lehreinheiten Ausführung. Dem konnte aber durch zusätzliche Authentifizierungsdaten entgegengewirkt werden. Bei mobilen Geräten wie Smartphones scheint dies auch geräteabhängig zu sein, da manche hier die Verbindung z.B. beim Ausschalten des Bildschirms sofort unterbrechen, während andere diese im Hintergrund weiter aufrecht erhalten.
Ebenso war es schwierig ein gut funktionales Word-Cloud / Wörterwolke Modul zu finden, welches sich ohne nennenswerte Probleme mit VueJS im Einklang nutzen lassen konnte. Generell wird VueJS in diesem Projekt recht rudimentär eingesetzt, was seine Funktionsweise zwar nicht einschränkt, jedoch das volle Potential dieser Web-Frontend-Engine nicht gänzlich nutzt. Eine tiefgreifendere Einarbeitung in VueJS ist aus Zeitmanagement Gründen nicht erfolgt.

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. Alternative Module erwiesen sich als ungenügend.

6 Auswertung

In diesem vorletzten Kapitel dieser Ausarbeitung soll die implementierte Software als Endresultat mit den ursprünglichen Plänen und Vorstellungen verglichen werden. Ebenso wird über einzelne Bestandteile des Projektes hinsichtlich Optimierung gesondert diskutiert und anschließend gesammelte Erfahrungen, die mit der Umsetzung des Projekts einhergingen, reflektiert. Im letzten Abschnitt **6.0.4 Ausblick** wird auf die Zukunft des Projektes näher eingegangen.

6.0.1 Umsetzung versus Planung

Im Rückblick auf die ersten Vorstellungen und Entwürfe hinsichtlich des Projekts, wurden diese im Bezug auf Grundfunktionalität und Design erfüllt. Die entwickelte Serversoftware ist flexibel auf verschiedenen Betriebssystemen und unterschiedlicher Netzwerk-Infrastruktur einsetzbar. Dabei ist ein Offline im Intranet gänzlich möglich und somit keine Ressourcen, die aus dem Internet geladen werden müssen, notwendig. Lehrende oder Verwalter können die Software mit wenig Aufwand installieren und bis auf die NodeJS Umgebung mitsamt NPM ist kein tiefgreifender Systemeingriff notwendig. Neue Nutzer können sich einfach nach der Installation registrieren, Lehreinheiten anlegen und diese ausführen. Die zwei Unterrichtsmethoden Brainstorming und Quiz wurden erfolgreich umgesetzt. Die angestrebte Drei-Client Lösung wurde implementiert und die Clients können je nach Situation und Anforderung auf unterschiedlichen Geräten ausgeführt werden. Die Anwendung läuft stabil und weist nur selten Fehler auf. Insgesamt kann der Code jedoch an vielen Stellen noch mittels Refactoring optimiert werden.

6.0.2 Verbesserungsvorschläge

Während des Entwicklungsprozesses sind einige Problemstellen zum Vorschein getreten, die im Nachhinein Verbesserungspotential hinsichtlich Usability, Performanz oder Software-Design aufweisen.

Der Nachrichtenaustausch welcher über das WebSocket Protokoll via SocketIO realisiert ist, sollten 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 gestalten. Gerade der Ausführungscode der interaktiven Unterrichtsmethoden könnte noch besser gekapselt

und noch sinnvoller aufgeteilt werden, auch in Hinsicht der Daten, welche zwischen Server und Client ausgetauscht werden. Der Installationsprozess könnte ggf. noch automatisierter erfolgen und so wenig Technik affinen Nutzenden die Installation erleichtern. Wie im vorangegangen Abschnitt bereits erwähnt, kann der Code der Clients, welche im Webbrowser ausgeführt wird, mit mehr Kenntnissen über die Entwicklung mit VueJS und SocketIO maßgeblich optimiert werden.

Zum Zeitpunkt des Abschlusses des Projekts kann eine Lehrkraft ein Lehreinheit anlegen, welche eine Unterrichtsmethode (in diesem Falle Brainstorming oder Quiz) beinhalten kann. Eine Verkettung von mehreren Unterrichtsmethoden wäre wünschenswert (Set Betrieb, siehe auch Abschnitt 3.1.1).

6.0.3 Erfahrungsauswertung

Aufgrund der Arbeit an diesem Projekt wurden vielerlei Erfahrungen hinsichtlich der Entwicklung eines verteilten Systems in Form eine Webanwendung gesammelt. Insbesondere die gewonnen Kenntnisse im Umgang mit den Frameworks NodeJS, Express und SocketIO 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 von VueJS interessant. Das Vorwissen über die Konkurrenten Angular und React war hilfreich wurde jedoch definitiv um neue Ansätze ergänzt. Die in der neusten Iteration des JavaScript hinzugefügten Funktionalitäten wurde verinnerlicht und das Wissen um die Sprache intensiviert.

6.0.4 Ausblick

Nach Beendigung des Bachelorprojekts soll dieses nicht verworfen werden sondern weiterhin privat ausgebaut werden. Während der Entwicklung kamen immer wieder Ideen für weitere Funktionalitäten auf, welche aber aus zeitlichen Gründen nicht implementiert worden sind oder nur als Konzept vorlagen. Dies wären z.B. Funktionen die den Dozent noch weiter während des Unterrichts unterstützen könnten oder das schreiben einer API, um die Software an andere Systeme anbinden zu können. Ein mögliches Vorhaben wäre es, die Software in einem Fork nach dem REST-Design aufzubauen und die Kommunikation anders zu gestalten, um anschließend zu vergleichen, welche Vorgehensweise entsprechende Vor- und Nachteile mit sich bringt. Auch könnte Dank der NodeJS Basis des Projektes sich unabhängig eine 'richtige' Desktop Applikation entwickelt werden, was mit dem Electron Framework realisierbar wäre.

Literaturverzeichnis

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

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

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

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

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

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

[8] 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/ (besucht am 17. 04. 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/.

[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/.

[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/ (besucht am 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/.

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

[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/.

[15] Wikipedia.org. *Intranet*. 30. Apr. 2019. URL: https://de.wikipedia.org/wiki/Intranet.

[16] Elektronik-Kompendium.de. *Client-Server-Architektur*. URL: https://www.elektronik-kompendium.de/sites/net/2101151.htm.

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

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

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

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

[21] 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 (besucht am 23. 07. 2019).

[22] 1&1 Ionis. *Webframeworks – Überblick und Klassifizierung*. 30. Jan. 2019. URL: https://www.ionos.de/digitalguide/websites/web-entwicklung/webframeworks-ein-ueberblick/.

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

[24] SMART. *SMART Learning Suite Online*. 12. Juli 2019. URL: https://suite.smarttech.com/login.

[25] 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.

[26] Promethan Limited. *ClassFlow Website Privacy Policy*. 2017. URL: https://classflow.com/privacy-policy/.

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

[28] Juriy Zaytsev. *ECMAScript 6 compatibility table*. 3. Juli 2019. URL: https://kangax.github.io/compat-table/es6/.

[29] 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/.

[30] 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/.

[31] ZingChart. *Commercial JavaScript Charts - Licensing Options | ZingChart*. 14. Juli 2019. URL: https://www.zingchart.com/pricing.

[32] Lets-Encrypt.org. *Erste Schritte - Let's Encrypt - Freie SSL/TLS Zertifikate*. URL: https://letsencrypt.org/de/getting-started/.

[33] 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/.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Anhang

Abschnitt: Funktionale Anforderungen

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
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, 27. Juli 2019,

(Ort, Datum, Unterschrift)