

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

Bachelorarbeit

zur Erlangung des akademischen Grades

Bachelor

(B.-Sc.)

an der HTW Berlin

Hochschule für Technik und Wirtschaft Berlin
Fachbereich Informatik, Kommunikation und Wirtschaft
Studiengang internationale 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

Inhaltsverzeichnis

Fachbegriffe und Formelzeichen	2
Abkürzungsverzeichnis	2
1 Einführung	3
1.1 Motivation	3
1.1.1 Besuch Grundschule am Rüdesheimer Platz Berlin	3
1.2 Problemstellung	4
1.3 Zielsetzung	5
1.4 Aufbau der Arbeit	6
2 Grundlagen	7
2.1 Digitalisierung an Schulen	7
2.1.1 Digitale Technik im Unterricht	7
2.1.2 Ausblick Interaktive Unterrichtsmethoden	7
2.1.3 Datenschutz an Schulen	7
2.2 Überblick Webtechnologie	7
2.2.1 Intranet und Internet	7
2.2.2 Client-Server Modell	8
2.2.3 Kommunikation	8
2.2.4 World Wide Web	9
2.2.5 Webanwendungen und Webservices	9
2.3 Websockets	10
2.4 Webapplikationsentwicklung	11
2.4.1 Web-Application-Frameworks	11
2.4.2 Serverseitiger Ansatz	12
2.4.3 Clientseitiger Ansatz	13
2.4.4 Hardware Anforderungen	14
2.4.5 Vergleich zu anderen Entwicklungsansätzen	14
3 Analyse	16
3.1 Vergleich mit existierenden Plattformen	16
3.1.1 Kommerziell	16
3.1.2 Nicht-Kommerziell	16
3.2 Systembeschreibung	16
3.3 Zielgruppe	16
3.4 Abgrenzung	16

3.5	Systemanforderungen	16
3.5.1	Funktional	16
3.5.2	Nicht Funktional	16
3.6	Technische Anforderungen	16
3.6.1	Server	16
3.6.2	Client	16
4	Konzept	17
4.1	Systemaufbau	17
4.2	Server	17
4.2.1	Entwicklungsumgebung	17
4.2.2	Node.js	17
4.2.3	Express.js	17
4.2.4	Datenbank	17
4.2.5	Serverarchitekturdiagramm	17
4.3	Client	17
4.3.1	UI Entwurf	17
4.3.2	Vue.js	17
4.3.3	Lehrer - Bereich	17
4.3.4	Schüler - Bereich	17
5	Implementierung	18
6	Erprobung	19
7	Fazit	20
8	Ausblick	21
	Literaturverzeichnis	22
	Abbildungsverzeichnis	24
	Tabellenverzeichnis	25
	Anhang	25

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, 16. Mai 2019,

(Ort, Datum, Unterschrift)

Fachbegriffe und Formelzeichen

Parameter	Formel- zeichen	Einheit/Beschreibung
Abschirmkonstante	S_n	Gibt die Abschirmung von Elektronen auf äußeren Energieniveaus in Atomen mit mehreren Elektronen an
Absolute Temperatur	T	in Kelvin (K)

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 Marie 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. Die Erprobung der im Rahmen dieser Arbeit implementierten Softwarelösung wurde ebenfalls an der Grundschule am Rüdesheimer Platz durchgeführt und wird im Kapitel 6 erläutert.

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 Hauptförderungsgebiete 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 mangle. 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.

1.3 Zielsetzung

Seit dem Erfolgskurs des Web 2.0¹ 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[6]. 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 Webtechnologien und ein günstigen Anschaffungspreis. Auch besitzen bereits 67% der 10-11 jährigen Jugendlichen ein Smartphone [7] 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.

¹Web 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

1.4 Aufbau der Arbeit

2 Grundlagen

2.1 Digitalisierung an Schulen

2.1.1 Digitale Technik im Unterricht

2.1.2 Ausblick Interaktive Unterrichtsmethoden

2.1.3 Datenschutz an Schulen

2.2 Überblick Webtechnologie

Diese Sektion soll einen grundlegenden Überblick 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 [8].

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 unserer 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. [9].

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 [10]. 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 [8], 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 [8] 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 [8]. 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 [11].

REST: (Representational State Transfer) 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 [12].

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 [13], was insbesondere bei mobilen Applikationen kritisch sein kann.

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 [14]. 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 objektorientierten Programmierung Anwendung findet. Hierbei wird eine Funktion/Unterprogramm bei der Hauptprogrammbibliothek registriert und von dieser zu einem späteren Zeitpunkt aufgerufen. Dies ist umgangssprachlich auch als 'Hollywood'-Prinzip bekannt ('Don't call us! We call you' z.Dt. 'Ruf nicht uns an! Wir rufen dich an!'). Das Framework behält also die Programmflusssteuerung bei. Ein Nachteil, der durch den Einsatz von 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 Entscheidungsprozess, 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 Nutzens 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 heißt pro Anfrage (Request) wird ein 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 [15].

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 unzählige 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 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 [8]. 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 erlernen 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 [6]

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

3.1 Vergleich mit existierenden Plattformen

3.1.1 Kommerziell

3.1.2 Nicht-Kommerziell

3.2 Systembeschreibung

3.3 Zielgruppe

3.4 Abgrenzung

3.5 Systemanforderungen

3.5.1 Funktional

3.5.2 Nicht Funktional

3.6 Technische Anforderungen

3.6.1 Server

3.6.2 Client

4 Konzept

4.1 Systemaufbau

4.2 Server

4.2.1 Entwicklungsumgebung

4.2.2 Node.js

4.2.3 Express.js

4.2.4 Datenbank

4.2.5 Serverarchitekturdiagramm

4.3 Client

4.3.1 UI Entwurf

4.3.2 Vue.js

4.3.3 Lehrer - Bereich

4.3.4 Schüler - Bereich

5 Implementierung

6 Erprobung

7 Fazit

8 Ausblick

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] TecArt-GmbH. *Vorteile browserbasierter Software - Webbasiert vs. Desktop*. 2019. URL: <https://www.tecart.de/browserbasierte-software> (besucht am 16.04.2019).
- [7] 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).
- [8] Christian Safran, Anja Lorenz und Martin Ebner. „Webtechnologien-Technische Anforderungen an Informationssysteme“. In: *Lehrbuch für Lernen und Lehren mit Technologien* (2011).
- [9] Wikipedia.org. *Intranet*. 30. Apr. 2019. URL: <https://de.wikipedia.org/wiki/Intranet>.
- [10] Elektronik-Kompendium.de. *Client-Server-Architektur*. URL: <https://www.elektronik-kompendium.de/sites/net/2101151.htm>.
- [11] 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).
- [12] Thomas Bayer. *REST Web Services*. 2002. URL: <https://www.oio.de/public/xml/rest-webservices.htm> (besucht am 07.05.2019).

-
- [13] Michél Neumann. „Entwicklung eines Cloud-Service und einer Client-Anwendung unter iOS“. Diss. Hochschule Für Technik und Wirtschaft Berlin, 29. Juli 2015.
 - [14] 1&1 Ionis. *Webframeworks – Überblick und Klassifizierung*. 30. Jan. 2019. URL: <https://www.ionos.de/digitalguide/websites/web-entwicklung/webframeworks-ein-ueberblick/>.
 - [15] Livivity. *Top Web Development Frameworks in 2019*. 15. Jan. 2019. URL: <https://lvivity.com/top-web-development-frameworks> (besucht am 11.05.2019).

Abbildungsverzeichnis

Tabellenverzeichnis

2.1	Überblick serverseitiger Web-Application-Frameworks	12
2.2	Webapplikationsentwicklung im Vergleich [6]	15