HOME
HOCHSCHULE
MERSEBURG FH
University of
Applied Sciences

# Webbasiertes E-Learning-Managementsystem für die Grundlagenausbildung an Hochschulen

**Bachelorarbeit**

**von**

**Martin Fligge**

Matrikelnummer: 20349

Fachbereich: IKS

Studiengang: Angewandte Informatik

27. April 2017

Betreuer:

Prof. Dr. E. Liebscher

Prof. Dr. M. Schenke

## Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig und ohne fremde Hilfe angefertigt worden ist. Alle Textstellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Alle Quellen, die dem World Wide Web entnommen oder in einer digitalen Form verwendet wurden, sind der Arbeit beigefügt. Der Durchführung einer elektronischen Plagiatsprüfung stimme ich hiermit zu. Die eingereichte elektronische Fassung der Arbeit entspricht der eingereichten schriftlichen Fassung exakt. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht. Ich bin mir bewusst, dass eine unwahre Erklärung rechtliche Folgen haben kann.

Ort, Datum, Unterschrift

# Inhaltsverzeichnis

# 1 Einleitung

> „Eigentlich lernen wir nur von Büchern, die wir nicht beurteilen können. Der Autor eines Buches, das wir beurteilen könnten, müßte von uns lernen.“
>
> Johann Wolfgang von Goethe

E-Learning Systeme verbreiten sich immer mehr. Ständig steigt die Zahl der im Netz verfügbaren Tutorials und Anleitungen in den verschiedensten Präsentationsmöglichkeiten. Ganze Websites die sich ausführlich mit Webdevelopment befassen, erfahrene Privatanwender welche eine komplette Videoreihe zur Textverarbeitung *LaTeX* auf *YouTube* teilen oder die Hochschule, welche seinen Studenten eine gewisse Lernunterstützung auf einer internen Website anbieten möchte. Bei so viel Vielfalt und dem Mangel an Normen und Regeln ist es nicht verwunderlich, dass einige Lernsysteme deutlich besser geeignet sind als andere. Wenige Fälle sind sogar gänzlich ungeeignet. Von falschen Fakten, über verwirrende Grammatik bis hin zur undursichtigen Präsentationsebene. Es gibt viele Dinge, die der Autor oder das Autorenteam einer E-Learning-Plattform verkehrt machen kann.

Einer der bereits erwähnten Punkte, die ein gutes Lernsystem ausmachen ist die Darstellung der Lerninhalte für den Lernenden. Ein zielführender Leitfaden oder ein klar definierter Lernpfad muss für den Lernenden von Anfang an erkennbar sein. Nützliche Hilfsmittel hierfür sind unter anderem anschauliche Oberflächen und Fortschrittsgraphen, bei denen der Nutzer augenblicklich erkennt, was er tun muss um sein Ziel zu erreichen, was er bereits erreicht hat und welches die nächsten Schritte auf seinem Weg sind.

Dieses Ziel zu erreichen ist Teil dieser Arbeit. Das Projekt widmet sich dabei im speziellen dem bereits existierenden E-Learning-System des Prof. Dr. Eckhard Liebscher zur Mathematikausbildung an der Hochschule Merseburg auf der Basis von *ILIAS*[1]. Die Informationen dieser Wissenssammlung wurden in Kooperation mit studentischen Hilfskräften zusammengetragen und den Vorlesungsinhalten angepasst. Bisher lagen die Informationen allerdings in trockener Text- und Bildform als einfache Liste im *ILIAS* vor. Der Lerner musste sich hierbei selbstständig einen geeigneten Lernpfad suchen oder stur von „oben nach unten“ die Inhalte abarbeiten. Zwar sind die Informationen eine gute Grundlage zur Wissensvermittlung, reichen aber alleine nicht aus um das maximale Potential dieser Plattform auszuschöpfen. Im Kern dieser Arbeit wird also versucht eine grafische Oberfläche in Form einer Webpage zu erschaffen, die dem Lerner als Unterstützung im Lernprozess dient, sowie eine Veranschaulichung des erreichten Lernfortschritts beinhaltet.

Im Abschnitt 2 werden zuerst die technischen Grundlagen und die verwendeten Technologien erläutert. Im darauf folgenden Abschnitt 3 wird das Konzept und der theoretische Aufbau des Projekts aufgeschlüsselt und im 4. Abschnitt wird auf die eigentliche Implementierung der Anforderungen und Funktionen eingegangen. Anschließend wird mit einem erklärenden Beispiel, Abschnitt 5, das ganze System getestet und alle Funktionen einmal erläutert und im Abschnitt 6 wird ein konkreter Anwendungsfall erstellt. Zu guter Letzt erfolgt im Abschnitt 7 eine Zusammenfassung des erreichten Fortschritts und ein kurzer Ausblick in dem einige Zukunftsideen für die Lernplattform zusammengetragen werden.

Zur Erstellung von Graphen, Diagrammen und ERM's benutzte ich das webbasierte Tool draw.io ([draw.io]) sowie die Programme Dia ([Dia]) und MySQLWorkbench ([MSQLW]).

[1] [ILIAS] - „ILIAS ist eine freie Software zum Betreiben einer Lernplattform, [...].“

# 2 Technische Grundlagen

## 2.1 HTML

*HTML* steht für **H**yper **T**ext **M**arkup **L**anguage welche für die Erstellung von Websites verwendet wird. Sie setzt sich aus sogenannten Tags[2] zusammen und geht heute Hand in Hand mit *JavaScript* und CSS (siehe: 2.4 CSS) (**C**ascade **S**tyle **S**heet).

Der erste Prototyp von *HTML* wurde 1991/1992 von Tim Berners-Lee entwickelt und beruhte auf der bereits existierenden *SGML* (**S**tandard **G**eneralized **M**arkup **L**anguage). Die Idee war es, eine Sprache zu entwickeln welche maschinenunabhängig ist und somit das Resultat auf allen Endgeräten gleich aussehen lässt. Damals bestand *HTML* allerhöchstens aus einer Hand voll möglicher Tags wie zum Beispiel die Tags zum bilden von Überschriften „<h1>“ bis „<h6>“ oder das Paragraphen-Tag „<p>“. Die ersten Prototypen und Versionen unterlagen keinerlei Vorschriften oder anderen Standardisierungsbemühungen, was den Erfolg von *HTML* zunächst eindämmte. Erst mit *HTML* 3.2 wurde am 14. Januar 1997, also 5 Jahre nach den ersten Prototypen, eine einheitliche Version vom „*World Wide Web Consortium*“ (kurz: *W3C*) vorgeschlagen. Die sogenannt „W3C Recommendation“.

Während die *IETF* (**I**nternet **E**ngineering **T**ask **F**orce) und das *W3C* sich um weitere Vereinheitlichungen bemühte, schritt die Entwicklung des World Wide Webs unglaublich schnell voran. Besonders 1995 überschlugen sich die Ereignisse regelrecht. *Microsoft* (siehe 8. Glossar) gab Version 1.0 des *Internet Explorers* heraus. *HTML* 3.0 wurde veröffentlicht und von der *IETF* als Standard vorgeschlagen, um 5 Monate später wieder verworfen zu werden. Das *W3C* begann Ihren eigenen Browser, den „*Arena Browser*“, zu entwickeln. Es brach damit ein regelrechter Wettkampf zwischen den Browserentwicklern, insbesondere *Microsoft* und *Netscape* (siehe 8. Glossar), aus der bis heute in ungebremster Form vorherrscht. *Netscape* ist mittlerweile jedoch ein Unternehmen welches nicht mehr als solches exisitiert, stattdessen sind Mozilla, Google und Apple als Hauptkonkurrenten in den Wettkampf eingetreten.

[2] [Tag] - „Ein Tag [tæg] [...] ist eine Auszeichnung eines Datenbestandes mit zusätzlichen Informationen.“

Am 18. Dezember 1997 wurde dann *HTML* 4.0 wurde veröffentlicht. Erstmals mit Stylesheets (siehe: 2.4 CSS), Frames[3] und Scripts[4]. *HTML* 4.01 erschien am 24. Dezember 1999, etwa einen Monat später dann die erste Version von *XHTML* (**X** für (englisch) **Ex**tensible → (deutsch) Erweiterbar) und im Juli 2006 die 2. Version. Die aktuelle Version zum Zeitpunkt dieser Arbeit ist *HTML5*, welche auch *XHTML* ablöste, und erschien im 4. Quartal 2014.

## 2.2 PHP

Ursprünglich als „**P**ersonal **H**ome **P**age Tool“ bekannt steht *PHP* heute für „*PHP: Hypertext Preprocessor*“, wobei die Abkürzung ein rekursives Akronym und Backronym darstellen soll. Es handelt sich um eine an *C* und *Perl* angelehnte, serverseitige Scriptsprache für Websites und Webanwendungen mit Datenbanksupport und Internetprotokollverwaltung. Weil es eine Open-Source[5] Software ist, stehen unzählige Bibliotheken zur Verfügung, mit der man auf nahezu alle Eventualitäten vorbereitet ist.

[3] [Frame] - Ein Frame ist ein Bereich einer HTML-Seite, in dem andere HTML-Seiten aufgerufen werden können.
[4] [Script] - Ein Script ist eine vordefinierte Ereignissequenz mit einem oder mehreren festgelegten Auslösern.
[5] Die Besonderheit einer Open-Source-Software ist der öffentlich zugängliche Quellcode.

## 2.3 JavaScript

Hand in Hand mit *HTML* geht *JavaScript* oder kurz „JS“. Es handelt sich hierbei um eine clientseitige Scriptsprache, mit welcher man Programmlogik, *HTML*-Modifikationen und Animationen auf einer Website integrieren und dynamisch erzeugen kann. Es ist beispielsweise möglich *HTML*-Elemente zu verändern, zu erstellen, zu löschen oder Formulareingaben eines Nutzers auf Gültigkeit zu überprüfen ohne dabei den eigentlichen *HTML*-Code zu bearbeiten. Hauptsächlich verwendet man *JavaScript* auf Webseiten um diese individueller zu gestalten und einen größeren Funktionsumfang zu ermöglichen. Während man bei HTML auf die vorgefertigten Funktionen angewiesen ist, hat man in *JavaScript* die Freiheit einer eigenständigen Programmiersprache. Von Spielen über Kalkulationsprogramme bis hin zu animierten *HTML*-Elementen ist alles möglich.

Die Sprache *JavaScript* wurde 2 Jahre nach Veröffentlichung als *ECMA* Script standardisiert (ECMA 262) und gilt als objektorientiert, wobei es allerdings keine Klassen gibt. JS lässt sowohl objektorientiertes, funktionales und prozedurales Programmieren zu.

Der Vorgänger zu der heutig bekannten Sprach hieß *LiveScript*, welche im September 1995 vom Unternehmen Netscape als eingebettete Scriptsprache in einer Vorversion des *Navigators* 2.0 veröffentlicht wurde. Entwickelt wurde *LiveScript* vom damals 34 jährigen Brendan Eich. Da sich *LiveScript* sehr an *Java* anlehnt und *Java* bereits eine verbreitete Programmiersprache war, nannte man *LiveScript* später aus marketingtechnischen Gründen zu *JavaScript* um. Ähnlich wie *HTML* unterlag auch *JavaScript*, bzw. *LiveScript* dem Wettrennen von *Microsoft* und *Netscape*. Der ewige Kampf um den technischen Vorsprung führte einerseits zu einer raschen Entwicklung und Erweiterung der Scriptsprache, auf der anderen Seite aber auch zu Kompatibilitätsproblemen. Während die *Navigator*-Nutzer Probleme mit der Darstellung von *Internet Explorer*-Scripten hatten, hatten auch die Internet-Explorer-Nutzer Schwierigkeiten mit Scripten die für den *Navigator* entwickelt wurden waren. Daher blieb praktisch für beide Unternehmen jeweils die Hälfte aller Kunden unerreichbar. Somit musste ein schmaler Grad aus Kompatibilität und Standardisierung eingehalten werden, gleichzeitig musste jedes Unternehmen aber selber seine eigenen Features ausarbeiten um sich von der Konkurrenz abzuheben.

In so einem Wettkampf kommt es vor, dass die Browser mit Fehlern oder Sicherheitslücken veröffentlicht werden. Besonders *JavaScript* half Angreifern in so einem Fall relativ leicht an empfindliche Daten zu gelangen. Zwar wurden solche Lücken immer sofort mit Patches geschlossen, der Ruf von *JavaScript* und dem entsprechenden Browser ist aber nach wie vor geschädigt. So kommt es immer wieder zu Berichten, welche *JavaScript* als großes Sicherheitsproblem hinstellen. Einige davon sachlich, aber überzogen:

> „*So können Angreifer beispielsweise Nutzerkennung mit Passwörtern oder auch lokal gespeicherte Daten von privaten und kommerziellen Internetnutzern ausspionieren. [...] Das BSI empfiehlt deswegen den Betreibern von WWW-Servern dringend, vollständig auf JavaScript zu verzichten oder zumindest für ihre sicherheitsbewussten Kunden Alternativangebote bereitzustellen, die ohne aktive Inhalte dargestellt werden können.*“
> - Bundesamt für Sicherheit in der Informationstechnik [BSI]

und einige offensichtlich nur um dem Leser etwas darbieten zu können:

> „*Vorsicht ist geboten, wenn Unternehmen für ihre Web-Seiten das Gestaltungssystem JavaScript verwenden. Dahinter verbergen sich kleine Programme, die sich unbemerkt auf dem Kunden-PC einnisten und ihn angreifbar für fremde Zugriffe machen.*“
> - Bild am Sonntag [BamS]

Dennoch kam es offiziell nie zu erwähnenswerten Schäden durch *JavaScript* und auch der Bericht der *BSI* wurde in Fachkreisen scharf kritisiert und als übertriebene „Panikmache“ abgestempelt.

## 2.4 CSS

Das **C**ascade **S**tyle **S**heet (wörtlich übersetzt: „kaskadierender Stil Zettel“), dient dem formatieren von *HTML* Inhalten und ist meist eine eigenständige Datei im Dateisystem der Webplattform mit der Endung .css, kann aber auch direkt in die .html-Dateien eingebunden werden. Letztere Variante reduziert aber drastisch die Übersichtlichkeit Quellcodes und sollte nur für einzeln auftretende Formatierungen oder für Ausnahmen benutzt werden.
Bevor *CSS* (ursprünglich *CHSS* $\rightarrow$ **C**ascade **H**TML **S**tyle **S**heet) entwickelt worden war, wurden sämtliche Formatierungen mit HTML vorgenommen, welches allerdings von jedem Browser verschieden interpretiert und dargestellt wurde. Die Hauptvorteile von *CSS* sind also:

- einheitliches Aussehen in allen üblichen Webbrowsern
- große Palette an Funktionen und Designmöglichkeiten
- zentrale Designverwaltung durch ausgelagerte .css-Datei

## 2.5 Learning-Managementsysteme

Learning-Managementsysteme oder Lern-Managementsysteme, kurz auch LMS, sind in den meisten Fällen webbasierte Sammlungen von Lernmitteln. Die Besonderheit besteht jedoch darin, dass es sich nicht um bloße Text- und Hyperlinksammlungen mit eventuellen Grafiken handelt, sondern den Lernenden auch im Prozess des eigentlichen Lernens unterstützen und leiten. Dazu wird der komplette Inhalt des LMS in passende Teilbereiche unterteilt und Stück für Stück dem Lernenden unterbreitet um gezielt Lernstoff für beispielsweise eine Klausur oder eine Führerscheinprüfung bereitzustellen.
Ein LMS besteht aus verschiedensten Elementen, welche alle einen eigenen Zweck erfüllen:

### Kurse

Kurse sind die Grundlage eines jeden LMS. Ein Lernsystem ohne Kurs, also ohne Inhalt den es vermitteln will, ist sprichwörtlich wie ein Auto ohne Teile. Über einen Kurs wird der gesamte Inhalt den ein LMS zu bieten hat in Teilabschnitte begrenzt und thematisch kategorisiert. Der Vorteil besteht darin, dass jeder Kurs nur Wissen enthält, das auch wirklich für das entsprechende Thema benötigt wird. Es werden somit keine irrelevanten Fakten vermittelt und der Lernaufwand des Anwenders sinkt. Im späteren Beispiel wird auch der Begriff Lernmodul anstelle von Kurs als synonym verwendet.

### Benutzersystem

Um die Eindeutigkeit jedes Benutzers zu Gewährleisten ist es notwendig, dass sich ein Benutzer am System verschlüsselt Anmelden kann und dort ein eigenes, eindeutig zuweisbares Profil besitzt. Dies ist im weiteren auch für das nächste Element von entscheidender Bedeutung.

### Rollen- und Rechtesytem

Ein Lernsystem hat verschiedenste Anwender wie Administratoren, Editoren und Lernende, beziehungsweise auch Lernende mit erweiterten Rechten. So kann ein Administrator und Editor beispielsweise neue Kurse und Lerngruppierungen anlegen und verwalten, aber nur der Administrator kann die Rechte von anderen Nutzern verändern. Ein Beispiel für erweiterte Nutzerrechte wären die Editorrechte für einzelne Kurse oder Exklusivzugang zu etwaigen kostenpflichtigen Kursen. Könnte man die Nutzer nicht eindeutig zuordnen, bzw. es könnte sich jeder Nutzer mit den selben Zugangsdaten anmelden, könnte man nicht gewährleisten, dass jeder Nutzer ausschließlich die für Ihn vorgesehene Rolle einnimmt und, viel schlimmer noch, würde ein großes Sicherheitsproblem auftreten, da jeder jederzeit alle persönlichen Daten einsehen oder Lernmaterialen löschen/verfälschen könnte.

### Kommunikation

Egal ob Forum, Chat, Notizseite oder Kalender. Ein Medium zum Austausch von Informationen erleichtert das Lernen, die Wartung und Verbesserung der gesamten Plattform. Sollten Inhalte unklar oder unvollständig sein, hat man einen Anhaltspunkt um Fragen zu stellen und Probleme zu lösen so wie technische Fehler bekannt zu geben. Durch aus kann sich auch eine sogenannte Community (eine Gemeinschaft aus Anwendern welche die Plattform regelmäßig verwenden) bilden, in denen erfahrene Langzeitnutzer die Fragen von Anfängern oder auch Fortgeschrittenen beantworten und somit eine wertvolle Erweiterung für die gesamte Lernerfahrung darstellen. Kein Medium der Welt ist so flexibel wie ein Mensch und somit kann Lern- und Informationsgehalt durch eine aktive Community von keiner anderen Methode übertroffen werden. Nicht ohne Grund gibt es Websites im Internet, die ausschließlich dieses Format der Wissensübermittlung anbieten. Beispiele hierfür sind *„www.stackoverflow.com“*, *„de.answers.yahoo.com“* sowie etwaige Live-Chat Supports.

**Medien**

Eine lange „Textwand“ durchzulesen ist nicht sehr motivierend. Der Anwender verliert schnell die Konzentration und somit auch die Lust am weiterlernen. Um den Lernalltag aufzulockern, empfiehlt es sich möglichst viele verschiedene Darstellungsarten zu verwenden. Jedoch ist zu beachten, dass bei einer Überladung an Grafiken schnell das wesentliche verloren geht. Auch kann keine Grafik alleinig ein Thema erklären, sondern dient stets als unterstützendes Element zum leichteren Verständnis. Obwohl die meisten LMS als Grundlage eine textbasierte Wissensbibliothek vorweisen, da dort der Erschaffungs- und Verwaltungsaufwand der Inhalte relativ einfach und sachlich ist, ist auch das Videomedium eine immer gefragtere Quelle von Informationen. Durch stilistische Mittel, Individualität, so wie eventuelle Mimik und Gestik kann der Autor eines solchen Videos genauestens auf Kniffe und Stolperfallen im Stoff hinweisen und visuell Lösungswege aufzeigen, welche in einem einfachen Text verloren gehen können. Hinzu kommt, dass ein Video mehr Informationen in einem kürzen Zeitraum enthalten kann, da es sowohl Bild- und Tonmaterial gleichzeitig vermittelt. Von hohem Wert sind auch Abfragen des bereits Gelernten durch Tests und Quiz wie etwas Multiple-Choice Fragen oder Lückentexten. Dadurch wird der Anwender gezwungen zuvor verarbeitete Thematiken selbstständig zu kombinieren und anzuwenden was den Lernstoff besonders einprägsam macht. Eine geeignete Kombination aus verschiedenen Medien kann den Lernfortschritt deutlich beschleunigen.

Die verbreitetsten Lern-Managementsysteme sind *„Moodle“*, *„stud.ip“* und *„ILIAS“*. Der Inhalt dieser Arbeit ist für eine auf *„ILIAS“* basierendes Wissenssammlung konzipiert.

## 2.6 Datenbanken

Unter einer Datenbank versteht man ein Zusammenspiel aus Software und Hardware um elektronisch große Mengen von Daten effektiv verwalten und speichern zu können. (Vermerk: aus juristischer Sicht sind auch nicht-elektronische Datensammlungen Datenbanken) Man unterscheidet zwischen verschiedenen Typen von **D**aten**b**anks**s**ystemen (DBS, manchmal auch **D**aten**b**ank**m**anagement**s**ystemen → DBMS) welche alle ihre eigenen spezifischen Eigenschaften besitzen und je nach Anwendungsfall individuell gewählt werden müssen. Die am häufigsten auftretenden Systeme sind relationale DBS und objektorientierte DBS. Andere Beispiele wären hierarchische DBS, netzwerkartige DBS, dokumentenorientierte DBS sowie diverse Mischformen der genannten Modelle.

#### Relationales DBS

Die Datenverwaltung findet in tabellarischer Form statt. Hierbei steht jede neue Zeile für einen Datensatz und die einzelnen Attribute bilden in der Regel die Tabellenspalten. Es können beliebige Beziehungen zwischen den Tabellen vorgenommen werden.

#### Objektorientiertes DBS

Das Datenbanksystem legt selbstständig die Beziehungen zwischen den Datenobjekten fest. Objekte können Eigenschaften und Attribute von anderen Objekten erben.

#### Hierarchische DBS

Die einzig mögliche Beziehung der Datenobjekte stellt eine Eltern-Kind-Beziehung dar.

#### Netzwerkartiges DBS

Die Datenobjekte werden durch Netzbeziehungen miteinander verknüpft.

#### Dokumentenorientiertes DBS

Die Datenobjekte können über verschiedene Attribute und Datenstrukturen verfügen.

## 2.7 MySQL

Spricht man von *MySQL* handelt es sich um eine Open-Source Software für relationale Datenbanksysteme, welche vom schwedischen Unternehmen *„MySQL AB“* entwickelt worden ist. Das Unternehmen wurde 2008 von *„Sun Microsystem“* übernommen. Heute verwaltet die *„Oracle Corporation“* die freie Software und stellt eine freie Variante zur Verfügung und eine Variante für die eine kommerzielle Lizenz ausgehändigt wird. Der Unterschied liegt im Support. Die freie Variante erhält keinen Support oder etwaige Garantien, wohin gegen die kommerzielle Lizenz mit umfangreichen Garantieleistungen gedeckt ist. Für den Zugriff auf eine relationale Datenbank, um Tabellen zu erstellen oder zu bearbeiten, wird die Sprache SQL (**S**tructured **Q**uery **L**anguage) verwendet.

## 2.8 XAMPP Entwicklertool

Eines der fundamentalen Entwicklertools im Entstehungsprozess dieses Projektes. *XAMPP* dient als Bündel von einem *Apache Webserver*, einem Datenbanksystem beruhend auf *MariaDB*, sowie einer *PHP*- und *Perl*-Version. Einmal heruntergeladen und installiert arbeiten alle Komponenten vorkonfiguriert und auf einander abgestimmt. Da der Kernfokus der Arbeit nicht auf der Einrichtung der Laufzeitumgebung liegen sollte, sondern sich zu $100\%$ um die eigentliche Software drehen soll, ist dieses Softwarepaket ideal geeignet. Ein weiterer kleiner aber feiner Vorteil besteht darin, dass alle Daten, die Datenbank und die *HTML*- und *PHP*-Dateien lokal auf dem Rechner vorliegen und ausschließlich über „localhost“ abgerufen werden können und zwar auch nur dann, wenn *XAMPP* im Hintergrund aktiv ist. Somit muss man sich in der Entwicklung noch keinerlei Gedanken um Sicherheit und böswillige Zugriffe von Außen machen und man kann im Hand umdrehen alles was Nötig ist nach seinen Vorstellungen konfigurieren. Selbst ohne dauerhafte Internetverbindung.

# 3 Konzept

Für das Konzept dieses Projektes muss man 2 Standpunkte betrachten: einmal den Studierenden, also den Endanwender und Nutzer und zum anderen den Autor und Editor der Inhalte, welcher den Hintergrundkontent erstellt, editiert und in geeigneten Paketen bereitstellt.

Jede dieser beiden Parteien hat andere Aufgaben und Ansprüche an das System, welche im folgenden Abschnitt behandelt werden. Am Ende dieses Abschnitts wird außerdem auf die verwendeten Technologien eingegangen und die Vor- und Nachteile abgewogen.

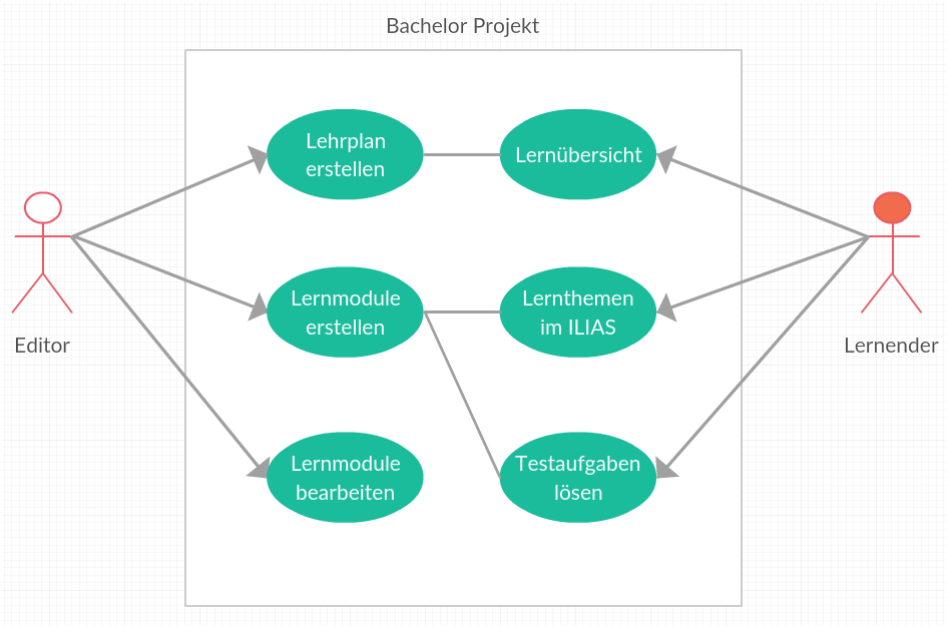


Abbildung 1: Use-Case Diagramm des Bachelorprojektes

## 3.1 Autor

Der Ersteller der Inhalte ist die wichtigste Person für alle Nutzer der Plattform. Er partitioniert nicht nur vorhandene Inhalte in passende Blöcke, sondern erstellt auch einen Ablaufplan und Übungsaufgaben um den Lernenden optimal im Lernprozess zu unterstützen. Des weiteren ist der Autor verantwortlich für die Qualität und die Richtigkeit aller Inhalte. Daraus ergeben sich besondere Ansprüche, die unbedingt erfüllt werden müssen um ein hochwertiges Lernsystem zu erschaffen.

Um Inhalte für die Lernplattform bereit zu stellen benötigt der Autor zunächst eine Möglichkeit auf die Datenbank zu zugreifen. Dort muss auf einen Blick zu erkennen sein, welche Lernblöcke bereits online sind und welche noch fehlen. Das Editieren der Lernblöcke kann entweder über eine Eingabemaske direkt auf der Website geschehen, welche auf die Datenbank aufgelegt ist oder direkt per Nutzerkonto in der Datenbank.

Eine Eingabemaske hätte den Vorteil, dass Sie individuell angepasst werden kann und je nach Umfang der Möglichkeiten die leichtere Methode für den Contentcreator ist um Lernblöcke online zu stellen. Auf der anderen Seite bietet die Maske aber eben nur genau so viel Flexibilität wie man ihr im Programmcode zugewiesen hat. Möchte man sicher gehen das wirklich alle Funktionen abgedeckt sind, bietet sich der Direktzugang zur Datenbank an, da man dort auf alle Daten in jeder erdenklichen Form zugreifen kann. Allerdings setzt dies besonderes Vorwissen in beispielweise SQL voraus, was wiederum die Anwenderfreundlichkeit stark einschränkt. Außerdem kann es zu Sicherheitsproblemen kommen, wenn man die Editierenden direkt in der Datenbank arbeiten lässt, besonders wenn es ein ganzes Team aus Autoren gibt. Aus diesen Gründen besteht im Falle dieses Projektes sowohl eine Eingabemaske welche die leichte und schnelle Verarbeitung von neuen Inhalten oder dem löschen alter Inhalte ermöglicht, als auch der Zugang zur Datenbank mit einem eingeschränkten Nutzerkonto um bestehende Module zu verändern oder andere, ungeplante Aufgaben zu vollziehen.

## 3.2 Lernender

Der Nutzer der Lernplattform verfolgt das Ziel der Wissensbereicherung und erwartet einen klaren Leitfaden sowie eine selbsterklärende Oberfläche. Er setzt außerdem die Relevanz der Themen für seinen weiteren Lernfortschritt, als auch die Richtigkeit aller Inhalte voraus. Man kann von einem Endnutzer einer E-Learningplattform nicht erwarten, dass er Quellen vergleicht oder sich andere Referenzen besorgt. Erst recht nicht, wenn die Plattform semesterbegleitenden Lerninhalt für einen oder mehrere bestimmte Studienkurse darstellt. Obwohl die Vielfalt der Anforderungen für den Lernenden nur gering ist, ist die Einhaltung dieser Kriterien von entscheidender Bedeutung wenn man das Ziel der E-Learningplattform zufriedenstellend erreichen will.

Das erste was dem Lernenden auf der Webseite ins Auge springt muss selbsterklärend sein. Ein einheitliches Erscheinungsbild der Webseiten fördert den Einstieg in das System. Ein unnötig komplizierter Einstieg mindert die Interesse am Lernen und gefährdet so den gesamten Lernprozess von Beginn an. Ein einfacher visueller Effekt, welcher sowohl Fortschritt als auch Möglichkeiten für den nächsten Schritt aufzeigt und gleichzeitig eine gute Gesamtpräsentation aller Inhalte darstellt ist optimal. Somit erkennt der Lernende sofort, wo er bei der letzten Lernsession aufgehört hat und welche Möglichkeiten ihm mit dem bereits gelernten zur Verfügung stehen.

Wertvoll für eine studienkursbegleitende Lernplattform ist es auch, wenn Online- und Kursinhalte gemeinsame Begriffe wie etwa Überschriften und Definitionen benutzen. Dies suggeriert dem Studenten/Schüler, dass er das richtige Lernmodul bearbeitet und gibt Bestätigung, dass das momentan zu Lernende wirklich Relevant ist, wodurch die Motivation des Lernenden und das Vertrauen in das Onlinesystem erhöht wird.

## 3.3 verwendete Technologien

Ein entscheidendes Kriterium für dieses Projektes ist, dass es immer und überall zur Verfügung stehen muss. In Zeiten von Smartphone und Internet-To-Go bietet sich hierfür ein webbasierter Dienst an, wodurch alle clientseitigen Anwendungen wegfallen. Ein Desktopprogramm ist nicht nur schwer zu warten, auch ist die Synchronisation der Lerninhalte ein Problem. Den Vorteil, dass eine Desktopanwendung auch ohne Internet funktionieren würde ist nichtig, da stets dafür gesorgt werden muss, dass die Lerninhalte aktuell sind, weshalb ohnehin stets eine funktionierende Internetanbindung gewährleistet werden muss.

Als Grundtechnologie für die Lernplattform wird eine einfache, auf *HTML* basierende Website verwendet. Dadurch wird sicher gestellt, dass man mit gängigen Mitteln Zugang zum E-Learningsystem hat, denn ein Internetbrowser ist auf jedem Smartphone und Tablett vorinstalliert und wird vom Lernenden ohnehin täglich benutzt. Die Erreichbarkeit ist somit also keine Hürde für den Lernenden, lediglich die Internetadresse muss er sich merken oder als Lesezeichen abspeichern. Um alle benötigten Anforderungen bereitzustellen ist der Kern der Website eine Struktur aus *JavaScript*-Befehlen und -Funktionen. Die gegebene Kompatibilität zwischen *HTML* und *JavaScript* prädestiniert diese Technologiekombination gerade zu als Herzstück für die Website. Sämtliche Logik (ausgenommen sind wenige Stellen, an denen der Nutzer direkt mit der Datenbank interagiert) ist in *JavaScript* formuliert und sorgt dafür, dass genau das geschieht was bei Interaktion X geschehen soll.

Für das Design der Website wird ein für *HTML* übliches „**C**ascade **S**tyle **S**heet“
(*CSS*) verwendet, welches problemlos in *HTML* integrierbar ist und somit auch ohne weiteres von *JavaScript*-Befehlen beeinflusst werden kann.

Die verwendeten Sprachen sind in einer deutlichen Monopolstellung. Alternative Technologien exisitieren zwar, kommen aber in Sachen Kompatibilität und Verbreitungsgrad der unterstützenden Browser nicht an die im Projekt verwendeten Sprachen *JavaScript*, *PHP*, *HTML* und *SQL* heran. Als Alternative zu reinem *JavaScript* stehen zum einen verschiedene Implementationen von *JS* (zum Beispiel: *CoffeeScript* oder *JSX*) als auch eigenständige Sprachen wie *GoogleDart* oder *Script#* zur Verfügung.

Des weiteren wird für Zugriffe auf die Datenbank die serverseitige Sprache *PHP* verwendet. Sie ist einzig für diesen Zweck entwickelt worden und bietet sich daher an um ohne Umweg eine Verbindung zur Datenbank herzustellen und unkompliziert die benötigten Daten abzurufen. Alternativen für *PHP* wären *ASP*, *Ruby on Rails* oder *Perl*, aber auch hier ist es einfach der Verbreitungsgrad und die Menge des Funktionsumfang, welche *PHP* an die Spitze der Kontrahenden setzt, auch wenn hier der Vorsprung nicht so gravierend ist, wie es bei *JavaScript* der Fall ist. Außerdem ist zu beachten, dass besonders *ASP* und *Ruby on Rails* vielversprechende Zukunftstechnologien sind und *PHP* mit Sicherheit in naher Zukunft in Sachen Umfang und damit zwangsweise auch im Verbreitungsgrad einholenwerden. Was die Einfachheit der Implementierung und Anwendung betrifft, haben die beiden genannten Alternativen bereits die Nase vorn.
An einigen Stellen, wie etwa der Eingabemaske des Editors liegt eine *PHP*, bzw. *SQL*-Logik direkt unter der *HTML*-Oberfläche. Diese Stellen sind die Einzigen, in denen kein *JavaScript* zur Datenverarbeitung benötigt wird.
Ein großer Vorteil des gewählten Technologiebündels ist, dass es eine relativ einfache Möglichkeit ist, dass Projekt umzusetzen, da alle Sprachen untereinander kompatibel sind und sich bereits jahrelang bewährt haben. Sie gelten als Standardtechnologien im Gebiet des Webdevelopments und sind sehr ausgereift, wenn auch aufgrund ihres Alters teilweise unnötig kompliziert durch mit veraltete Syntax und Programmierweise.

Eine eventuelle Alternative, oder viel mehr eine mögliche Ergänzung zur Web-Site Variante wäre eine Smartphone-App geschrieben in *Java* bzw. *Swift*[6] um die Zugänglichkeit und Verfügbarkeit für den Lernenden noch weiter zu erhöhen. So könnte er beispielsweise im öffentlichen Nahverkehr oder kurz vor einer Klausur bequem vom Handy aus die Lerninhalte abrufen.

[6]Eine eigens für *Apple*-Produkte entwickelte Programmiersprache.

## 3.4 Anforderungen

Die Anforderungen für dieses Projekt sind:

**Erstellung einer GUI für den Autor/Editor**

Um das System mit Inhalten und Kontext zu füllen und erstellte Inhalte bearbeiten zu können, ist es zwingend notwendig eine angepasste Oberfläche für diese Aufgabe zu erstellen.

**Erstellung einer GUI für den Lernenden**

Damit das eigentliche Ziel dieser E-Learningplattform erfüllt werden kann, muss eine optisch ansprechende und verständliche Oberfläche geschaffen werden, welche den Lernenden Schritt für Schritt an das Lernziel heranführt.

**Anzeige des Lernfortschritts**

Um Verwirrung zu vermeiden und die Langzeitmotivation zu fördern ist es ratsam eine grafische Fortschrittsübersicht in die Plattform einzubauen.

**Ablage der Testaufgaben**

Die Testaufgaben müssen einen einheitlich definierten Platz, etwa in Form eines Dateiordners besitzen um den Verwaltungsaufwand zu minimieren.

**Datenbankanbindung**

Als zentrales Informationszentrum soll eine Datenbank existieren, welche in direkter Wechselwirkung zur E-Learningplattform steht.

**Check zur Kreisfreiheit des Graphen**

Es muss ein Kontrollsystem implementiert werden, welche die Vorgängerverknüpfungen der einzelnen Lernmodule auf Lücken, Widersprüche und Kreisfreiheit überprüft um unlösbare Lernsysteme und Endlosschleifen zu verhindern.

# 4 Implementierung

Herz und Nieren der Website sind *JavaScript*-Funktionen. Sie Regeln so gut wie alles, was mit Benutzerinteraktion zu tun hat. Vergleichbar mit einem Ameisenbau in dem die Bewohner alle unterschiedliche Funktionen haben, ist der Hintergrund der Website in Funktionen eingeteilt die alle eine ganz spezielle Aufgabe erfüllen. Dennoch kann eine Funktion nicht ohne die andere existieren und arbeiten. Der Vorteil dieser modularen Verfahrensweise ist die extrem hohe Wart- und Modifizierbarkeit. Des weiteren erspart man sich als Entwickler viel Schreibarbeit, wenn man wiederkehrenden Quellcode in Funktionen verpackt und an gegebener Stelle einfach die Funktion aufruft, anstatt den benötigten Code doppelt erneut zu schreiben.

Würde man keine Funktionen verwenden und den gesamten Funktionsinhalt jedes Mal aufs neue in den Quellcode schreiben, würde das nicht nur die Übersichtlichkeit verringern, sondern müsste man bei Änderungen alle betroffenen Stellen einzeln suchen und korrigieren, was den Zeitaufwand enorm in die Höhe treiben würde. Verwendet man hingegen eine Funktion, muss man lediglich innerhalb dieser alle Änderungen einmalig vornehmen und sie gelten für das gesamte Programm.

Im Nachfolgenden werden nun alle Hintergrundtechnologien beschrieben und deren Zusammenhänge erläutert. Es wird erklärt wie genau jede Funktion funktioniert und in wie fern sie mit anderen Funktionen zusammenspielen. Die Erläuterungen finden ungefähr in der logischen Reihenfolge statt, wie Sie auch auf der eigentlich Website zum Einsatz kommen. Da es sich aber um eine Plattform mit menschlicher Eingabe handelt und niemand vorhersagen kann, was Nutzer XY als nächstes tun wird, mag die Reihenfolge nicht immer 100% die Reihenfolge sein, die tatsächlich auftreten wird.

Um einen groben Überblick über den internen Aufbau des Projektes zu übermitteln folgt eine Übersicht, wie die Website-Dateien untereinander zusammenarbeiten. Die style.css-Datei wird hierbei nicht berücksichtigt, da sie keine Informationen zum Gesamtsystem beisteuert und somit für die Informationsverarbeitung nicht relevant ist.

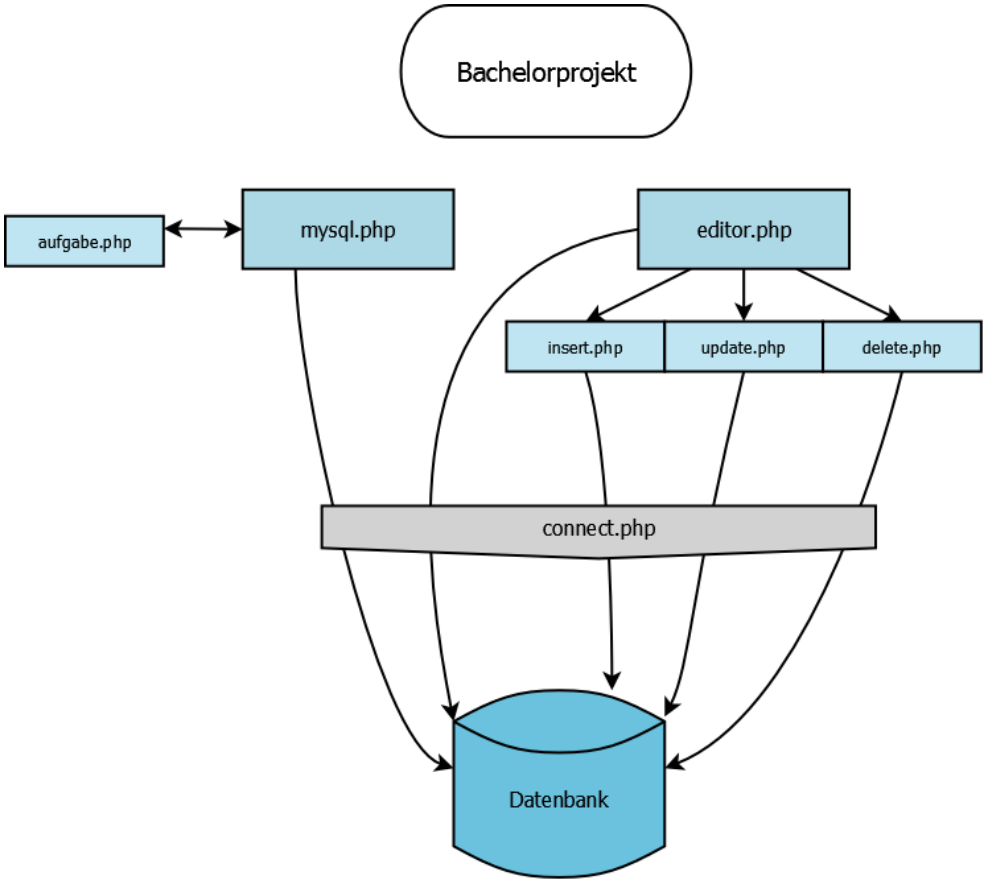


Abbildung 2: Zusammenhang der Website-Files und der Datenbank

## 4.1 Die Datenbank

Die Informationen zu den Inhalten der Webseite, also die Lernmodule, ihre Beschreibungen oder die dazu gehörigen Testaufgaben liegen in einer *MariaDB*-Datenbank. Sie ist eine relationale Datenbank, das heißt dass Daten in Form von Tabellen vorliegen und teilweise voneinander abhängig sind. Es gibt also Relationen (lateinisch *„relatio“* → „Beziehung“, „Verhältnis“) zwischen den Tabellen. In dem konkret vorliegenden Fall wäre es zum Beispiel der Zusammenhang zwischen den einzelnen Lernmodulen, welches sich in einer Vorgänger/Nachfolger-Beziehung äußert. Dazu später mehr.

### 4.1.1 Aufbau und Struktur

Als grundlegendes Datenbankverwaltungssystem dient *MariaDB*. Ausschlaggebend hierfür war, dass *MariaDB* im *XAMPP*-Entwicklertool enthalten war und die Entwicklung von Anfang an auf diesem Tool beruhte. Die Datenstruktur innerhalb der Datenbank besteht aus Tabellen. Eine der Tabellen ist die Wertetabelle. In ihr sind reihenweise alle Datensätze[7] eingetragen und dazu spaltenweise die entsprechenden Attribute, wie etwa die Bezeichnung, die Beschreibung, oder die Lösung der Testaufgaben. Die 2. Tabelle ist eine Zuordnungstabelle und dient der Aufschlüsselung der Vorgänger/Nachfolger-Problematik. Hier werden pro Reihe 2 unterschiedliche ID‘s aus der ersten Tabelle miteinander verknüpft um dadurch eine Abhängigkeit erschaffen.

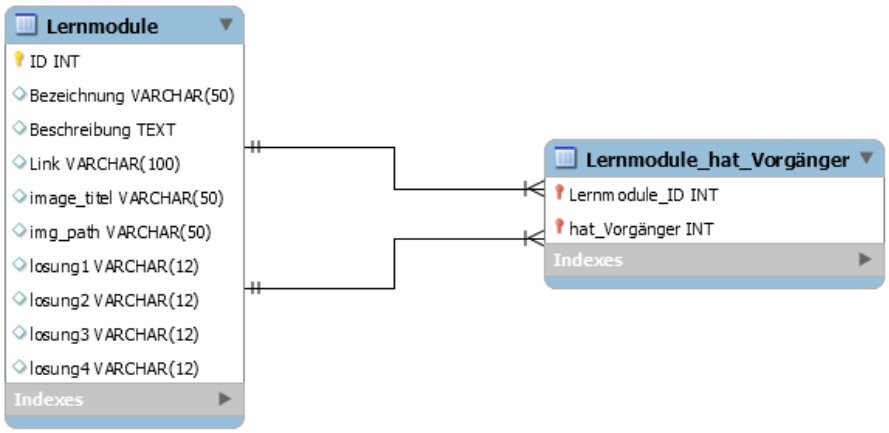


Abbildung 3: Entity-Relationship-Model

[7] Als einen Datensatz bezeichnet man alle zusammengehörigen Informationen eines Datenbankeintrages. Beispiel: *Angestellter Nr. 28, Mustermann, Max, Personalabteilung* - wäre ein Datensatz

| Bezeichnung | Datentyp | Beschreibung |
|---|---|---|
| ID | Integer | Eine wählbare Nummer welche allerdings nie doppelt auftreten darf. |
| Bezeichnung | varChar(100) | Speichert die Bezeichnung der Lerneinheit. |
| Beschreibung | Text | Beinhaltet eine kurze Erläuterung zum Inhalt der Lerneinheit. |
| Link | varChar(200) | Beinhaltet den Link, welcher zu der Lernsammlung im *ILIAS* führt. |
| image_titel | varChar(100) | Speichert einen kurze Betitelung der Testaufgabenbilder. |
| img_path | varCar(50) | Beinhaltet den Dateipfad zum Bild, welches sich außerhalb der Datenbankstruktur befindet. |
| losung1-4 | varChar(20) | Enthält die Lösung(en) für die Testaufgabe. Es werden maximal 4 Lösungen pro Aufgabe unterstützt (losung1 bis losung4) und kann aus Zeichen, Symbolen und Ziffern bestehen. |

Legende zu dem Datentypen:

**Integer** Ein Integerwert in *MariaDB* besteht immer aus ganzen Zahlen zwischen $0$ bis $4.294.967.295$ (unsigned), bzw. von $-2.147.483.648$ bis $2.147.483.647$ (signed).

**Text** Der Datentyp *TEXT* in *MariaDB* beschreibt einen Text aus Buchstaben, Ziffern und Sonderzeichen mit maximaler Länge von 65.535 Zeichen.

**varChar(x)** *VarChar* (kurz für variabler Character) dient ähnlich wie *TEXT* der Speicherung von verketteten Zeichen. Der eingeklammerte Wert $X$ steht dabei für die Menge von Zeichen, welche verwendet werden dürfen. Diese genauere Restriktion der Länge der Kette dient dem Speichermanagement und wird besonders bei Datenbanken mit mehreren tausenden Einträgen relevant.

## 4.2 PHP-Zugriff auf die Datenbank

*PHP* dient unter anderem dem Abrufen von Datenbankinhalten auf externen Plattformen. Diese serverseitige Sprache enthält viele vorgefertigte Funktionen mit denen sich Datenbankzugriff vollziehen lassen.

Um auf die Datenbank zugreifen zu können, muss als erstes eine Verbindung aufgebaut werden. Da es im Projekt mehrere Stellen gibt, an denen eine Datenbankverbindung aufgebaut wird und man bei Änderung der Zugangsdaten nicht jedesmal alle relevanten Stellen manuell ändern möchte, liegt die Routine welche die Verbindung aufbaut, in einer extra .php-Datei und wird einfach überall dort eingefügt (im Fachjargon nennt man es auch „includen“, englisch → „einfügen“) wo eine entsprechende Neuverbindung notwendig ist.

Listing 1: connect.php

```
<?php
function connectdb(){
        $servername = "localhost";
        $username = "root";
        $password = "";
        $dbname = "bachelor";

        // Create connection
        global $conn;
        $conn=new mysqli($servername, $username, $password,
                        $dbname);

        // Check connection
        if ($conn->connect_error) {
                die("Connection failed: ".
                        $conn->connect_error);
        }
        echo "Connected successfully";
        return $conn;
} ?>
```

Legende zur Farbcodierung für folgende Quellcode-Listings:

| | |
|---|---|
| Olivegrün | PHP-Keywords |
| Violet | Schlüsselbegriffe der Datenbank |
| Blau | SQL-Variablen |
| Rot | JavaScript-Funktionen |

Möchte der Systemadministrator eine neue Datenbank als Grundlage wählen, oder hat er lediglich die Zugangsdaten geändert, so muss er nur in der „connect.php“-Datei die relevanten Variablen *$servername*, *$username*, *$password* und *$dbname* ändern. Da auf dem Test- und Entwicklungssystem die Datenbank nur zeitweise online und lediglich über die Adresse „localhost“ (also 127.0.0.1) erreichbar war, war die Vergabe eines Passwortes nicht zwingend notwendig weshalb das Passwortfeld aus einer leeren Zeichenkette besteht.

Ist die Verbindung zur Datenbank einmal aufgebaut, hat man Zugriff auf alle Tabelle in der selektierten Datenbank. In unserem Fall liegen 2 Tabellen vor, eine Datentabelle und eine Beziehungstabelle. (siehe Abbildung 3: ER-Model)
Auch hier bietet *PHP* wieder vorgefertigte Funktionen, mit deren Hilfe *SQL*-Befehle an die Datenbank gesendet werden und das Resultat empfangen wird. Insgesamt findet der Zugriff an 5 Stellen statt: Im Hauptfenster des Lernenden („mysql.php“) und jeweils einmal in einer Datei namens „insert.php“ , „delete.php“ und „update.php“, welche alle Ihre Ursprungsdaten aus einer Action-Form in der „editor.php“ beziehen. Der letzte Zugriff findet in der „editor.php“-Datei selber statt. (siehe Abbildung 2: Zusammenhang der Unterdateien)

## 4.3 Editorseite

In der„editor.php“-Datei befindet sich der Quellcode für eine Reihe von Eingabemasken und Kontrollübersichten, welche dem Nutzer ein überschaubares Editieren der Datenbankeinträge ermöglichen soll, ohne direkt auf die Datenbank zugreifen zu müssen. Lädt man die Editor-Seite, so sieht man 3 Masken, eine Übersichtsliste und eine Kontrollanzeige. Es existiert eine Maske zum Anlegen neuer Datensätze, eine Maske zum Editieren vorhandener Einträge und eine kleine Maske zum Löschen von Einträgen anhand ihrer ID, eine Übersichtliste in welcher alle Einträge aufgelistet werden, sowie einer Anzeige, ob im Lehrplan ungewünschte Kreise exisitieren.

Die Übersichtliste enthält eine absolute Nummerierung beginnend bei „1.“ fortlaufend bis zum letzten Datenbankeintrag gefolgt von der ID und der Bezeichnung jeder Spalte der Datenbank. In den Abschnitten 5: *Beispiel* und Abschnitt 6: *Anwendungsbeispiel* werden die Elemente der Editor-Seite genauer erläutert.

Die eingegeben Daten und Werte der 3 Eingabemasken werden mittels einer *PHP*-Post Methode an 3 untergeordnete Dateien übermittelt wobei jeder Maske einer der 3 Dateien zu geordnet ist:

- „insert.php“ zum Einfügen neuer Datensätze
- „update.php“ zum Editieren bestehender Datensätze
- „delete.php“ zum Löschen vorhandener Datensätze anhand ihrer ID

Erst in den untergeordneten Dateien werden die jeweilig benötigten *SQL*-Befehle ausgeführt.

Listing 2: PHP post-Methode

```
<form action="insert.php" method="post"
2                 autocomplete="off" target="_blank">
```

Da in der Datei zum Löschen einfach nur die ID des gewünschten, bzw. unerwünschten Datensatzes benötigt wird, kommen wir hier auch mit einer einfachen Routine aus. Nachdem die Verbindung über die „connect.php“-Datei aufgebaut wurde wird einfach der *SQL*-Befehl

Listing 3: SQL-Befehl zum löschen

```
$sql = "DELETE FROM `lernmodule`
                        WHERE `lernmodule`.`ID` = $id;";
```

ausgeführt. Ins deutsche Übersetzt wird der Datenbank gesagt: *„Lösche aus* (der Tabelle) *'lernmodule', wo die ID des Lernmoduls gleich der übermittelten ID ist.“* Die übermittelte ID „$id“ ist hierbei die, welche zuvor in die Eingabemaske eingegeben wurde. Ist der Befehl abgeschickt worden, kann es genau zu 2 Ergebnissen kommen. Entweder der Datensatz wurde gefunden und gelöscht, oder es konnte kein Datensatz mit der entsprechenden ID gefunden und somit auch nicht gelöscht werden.
In der Datei zum einfügen eines neuen bzw. zum editieren eines bestehenden Datensatzes findet grundsätzlich die gleiche Prozedur statt, lediglich ist der *SQL*-Befehl etwas komplexer. Dennoch funktioniert er auf die gleiche Art und Weise. Hier als Beispiel der Befehl zum Einfügen:

Listing 4: SQL-Befehl zum einfügen

```
$sql = "INSERT INTO `lernmodule`(`ID`, `Bezeichnung`,
                `Beschreibung`, `Link`, `image_titel`,
                `img_path`, `losung1`, `losung2`, `losung3`,
                `losung4`)
                VALUES ('$id', '$bezeichnung',
                '$beschreibung', '$link', '$image_titel',
                '$img_path', '$losung1', '$losung2',
                '$losung3', '$losung4')";
```

Um auch hier wieder eine verständlichere Umschreibung des Befehls zu nennen: *„Füge ein in* (die Tabelle) *'lernmodule'* (in die Spalten) *ID, Magenta Bezeichnung, Beschreibung, Link, image_titel, img_path, losung1, losung2, losung3, losung4 die Werte ID, Bezeichnung, Beschreibung, Link, image_titel, img_path, losung1, losung2, losung3, losung4“* Wobei auch hier die Werte wieder aus der Eingabemaske stammen.

Eine kritische Sicherheitslücke bei Eingaben von Nutzern ist die sogenannte „*SQL-Injection*“. Hierbei schreibt der böswillige Nutzer direkte *SQL*-Befehle in die Eingabemaske um beispielsweise Tabellenspalten zu verändern, ganze Tabellen zu löschen oder sich selber ein Adminkonto anzulegen. Dies kann auf 2 Arten verhindert werden:

- Zum einen durch sehr strenge Rechteeinschränkungen des Datenbankkontos mit welchem die Zugriffe erfolgen. Dies erfolgt durch den Systemadministrator, welcher die Datenbank verwaltet und die Rechte der Nutzerkonten festlegt.
- Zum anderen bietet *PHP* auch hier eine Funktion, mit welcher Eingaben direkt auf *SQL*-Befehle überprüft und korrekt formatiert werden.

Übergibt man an die *PHP*-Funktion „*mysqli_real_escape_string*“ eine beliebige Serie von Zeichen und Zeichenketten erkennt sie automatisch böswillige Eingaben mithilfe von umfangreicher Syntaxüberprüfung und wandelt unschädliche Eingaben direkt in *SQL*-taugliche Werte um. Sowohl in der „delete.php“-Datei, als auch in der „insert.php“-Datei kommt diese Funktion zum Einsatz und verwehrt somit Angreifern die Möglichkeit einer *SQL*-Injektion. Genauere Information, über die Funktionsweise von „*mysqli_real_escape_string*“ findet man in der *PHP*-Dokumentation. [PHP Doku]

### 4.3.1 TopSort und Kreisfreiheit

Eine weitere wichtige Funktionalität der Editorseite ist das Kontrollkästchen zur Überprüfung der Kreisfreiheit des Lehrplans. Mit Hilfe dieser einfachen Anzeige kann der Ersteller von neuen Lernplänen sofort erkennen, ob der erstellte Lehrplan überhaupt von Anfang bis Ende durchgearbeitet werden kann, oder ob es Lerneinheiten gibt, welche gar nicht erreicht werden können. Diese Anzeige hat dabei lediglich 3 Zustände:

- gelber Signalkreis: es wurde keine Überprüfung durchgeführt
- roter Signalkreis: Lehrplan ist nicht azyklisch und kann nicht topologisch sortiert werden
- grüner Signalkreis: der Lehrplan ist azyklisch und kann topologisch sortiert werden

Die Grundlage dieses Überprüfung ist ein TopSort-Algorithmus, genauer gesagt eine Variante der Breitensuche (engl.: *breadth-first search*). Als Ausgangspunkt dient stets ein Element des Lehrplans welches keine Vorgänger besitzt, somit ist dies nahezu immer das Element mit der ID 1. Von dort aus werden alle Nachbarknoten des aktuellen Knotens in die Warteschlange übertragen, wenn die Anzahl der Vorgänger (auch Eingangsgrad oder indegree genannt) des zu übertragenden Knotens minus 1 = Null entspricht ($nachbar^{indegree} - 1 = 0$), andernfalls wird nur der Eingangsgrad um eins verringert. ($nachbar^{indegree} - 1 = nachbar^{indegree}$) Jedesmal wenn ein neuer Knoten aus der Warteschlang entfernt und behandelt wird, wird eine Zählvariable um 1 erhöht. Der Algorithmus endet wenn die Warteschlange leer ist wobei der Wert des Zählers verrät, ob der durchlaufene Graph azyklisch ist oder nicht.

Damit der Algorithmus problemlos funktionieren kann, muss zunächst einmal eine passende Datenstruktur generiert werden. Hierfür wird ein Array aus Objekten mit folgender Struktur erstellt:

- ModulID: Die ID des Lernthemas
- anzVorg: Ein numerischer Wert, welcher den Eingangsgrad angibt
- nachfolger: Ein Array mit der ID aller Nachfolgerthemen

Nun kann der Algorithmus auf die Datenstruktur angewendet werden. Als erstes müssen Startknoten anhand der Eingangsgrade bestimmt werden:

Listing 5: Bestimmung der Startknoten

```
        for (var obj of filteredModules){
2       if (obj.anzVorg == 0)
        queue.push(obj);}
```

Wurde nun mindestens ein Knoten ohne Vorgänger ermittelt und in die Queue gestellt, so beginnt der eigentlich Algorithmus und arbeitet bis nichts mehr in der Queue ist.
Als erstes wird das erste Queue-Element auf einer extra Variable gespeichert und aus der Queue gelöscht. Alle anderen Elemente welche sich eventuell noch in der Queue befinden rücken dabei einen Platz weiter nach vorne und schließen die Lücke von Position 1. Zusätzlich wird die Zählvariable, welche mit dem Wert 0 beginnt, um 1 inkrementiert.

Listing 6: Queue shift und extra Variable

```
1       while(queue.length != 0){
                var v = queue[0];
3               queue.shift();
                zaehler++;
```

Im nächsten Schritt wird der Eingangsgrad jedes Nachfolgerknotens um 1 reduziert und bei einem Eingangsgrad von 0 der Queue hinzugefügt

Listing 7: Verringerung des Eingangsgrades und Queue push

```
        for (var adjacent of v.nachfolger){
2               if (--filteredModules[adjacent].anzVorg == 0{
                        queue.push(filteredModules[adjacent]);
4               }
        }
```

Am Ende des Algorithmus, also wenn die Queue leer ist, wird der Zählerwert mit der Gesamtanzahl der Knoten des Graphes verglichen. Weicht der Zähler hier von der Anzahl der Knoten ab, so ist der Graph entweder nicht azyklisch, oder verfügt über keine topologische Sortierung. In beiden Fällen gibt der Algorithmus den boolischen Wert „*false*“ zurück. Wenn Zähler und Anzahl übereinstimmen wird „*true*“ zurückgegeben.

## 4.4 Die Lernerseite

Die Lernerseite ist die komplexeste Datei des gesamten Projektes. Sie beinhaltet die komplette Programmlogik zur Determinierung der Aktivität der Lernmodule und deren Aufbau und Anordnung. Wie genau der technische Hintergrundablauf stattfindet, wird im folgenden Abschnitt erläutert. Um eine bessere Übersicht über das Geschehen zu bekommen, ist der grobe Zusammenhang in einem Diagramm dargestellt:

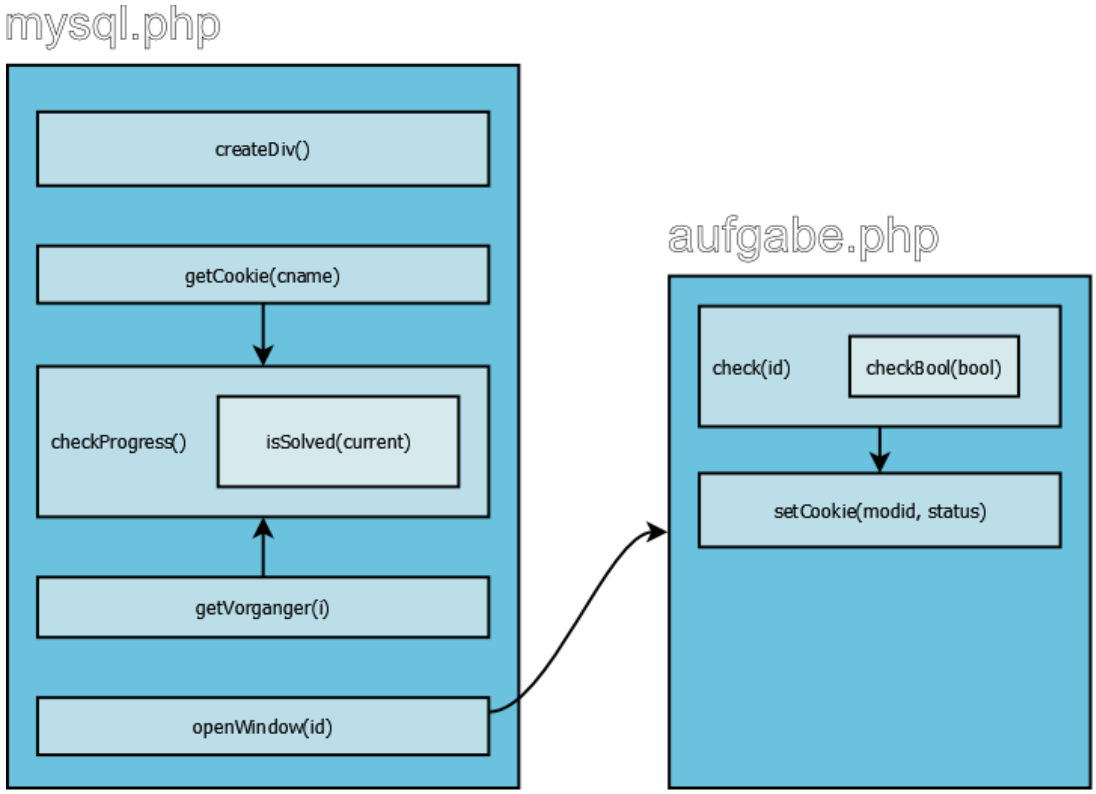


Abbildung 4: Funktionenübersicht der Lernerseite

### 4.4.1 PHP und Variablendeklaration

Das erste was auch in dieser Datei wieder passieren muss, ist die Verbindung zur Datenbank. Genau wie in der „insert.php“ und der „delete.php“ wird hier die „connect.php“-Datei included und die Funktion zum Verbinden aufgerufen.

War die Verbindung erfolgreich wird mit den richtigen *SQL*-Befehlen jeder Datensatz der Datenbank einzeln abgerufen und in einem Array[8] gespeichert. Diese Prozedur geschieht 4 mal in Folge. Jeweils einmal für die einzelnen Lernmodule („ID“, „Bezeichnung“, „Beschreibung“ und „Link“ aus der Tabelle „Lernmodule“), jeweils einmal um via *SQL*-Logik sowohl alle Nachfolger ein Moduls, als auch deren Vorgänger zu ermitteln und zu guter Letzt werden die Lösungen für jedes Lernmodul separat aus der Datenbank ausgelesen und abgespeichert um die Handhabung der Variablen später im Programm etwas flexibler zu gestalten. Wurden alle Datenbankzugriffe ausgeführt wird die Verbindung zur Datenbank wieder geschlossen. Es muss also nicht mehr auf die Datenbank zugegriffen werden, solange die Website offen bleibt oder bis sie aktualisiert und neu geladen wird. Bis hier hin liegen alle benötigten Daten in den angelegten *PHP*-Arrays vor. Da der Rest der Website aber mit *JavaScript* gesteuert werden soll, muss noch eine Konvertierung von *PHP* nach *JavaScript* vorgenommen werden. Mit der *PHP*-Funktion „*json_encode*“ werden *PHP*-Variablen korrekt in *JavaScript* taugliche Formate umgewandelt. Um nun die konvertierte *PHP*-Variable auf einer *JavaScript*-Variable zu speichern muss man in der *JS*-Variablendeklaration „*json_encode*“ aufrufen und per „echo“-Befehl die PHP-Variable ausgeben lassen. Diese Ausgabe wird dann auf der *JS*-Variable abgespeichert:

Listing 8: Übertragen eines PHP-Arrays in ein Javascript-Array

```
1 var modules = <?php echo json_encode($dbarray);?>;
```

Was auf den ersten Blick ein wenig seltsam erscheinen mag, taucht in der Praxis an Schnittstellen zwischen *PHP* und *JavaScript* öfter auf. Mitten in einer *JavaScript* Befehlszeile wird ein *PHP*-Befehl ausgeführt und dessen Ergebnis wird von *JS* interpretiert als wäre es eine normale Variable.

[8] Array - Ein Array ist ein Datenstruktur, in welcher viele Daten mit gleicher/ähnlicher Strukturierung abgespeichert werden können.

Allerdings hat diese Methode der Variablenkonvertierung durch vorgefertigte Funktionen auch eine Schattenseite. Da es sich bei der *PHP*-Variable um ein Array aus Objekten handelt (zur Erinnerung: jeder Datensatz ist ein Objekt bestehend aus den jeweiligen Spalten der Datenbank, jedes dieser Objekte wird der Reihe nach in einem Array gespeichert) erzeugt die *json_encode*-Methode ein etwas verworrenes Object-Array dessen innere Struktur zunächst undurchsichtig erscheint. Dies verkompliziert für ausstehende Personen den Einarbeitungsprozess in den Quellcode. Vorteile der verwendeten Methode verglichen zu einer manuellen Konvertierung sind eine deutlich einfachere Implementierung, Vermeidung von Logikfehler in der Programmierung und Reduzierung des Quellcodes wodurch letztendlich ein zuverlässigeres Endprodukt entsteht.

### 4.4.2 createDiv-Funktion

Das erste was passiert, wenn die Website geladen wird, wird in der Funktion *createDiv()* definiert. Diese Funktion wird nur einmal ausgeführt und baut die Lernmodulübersicht auf. Wie der Name schon andeutet, ist jede Box technisch gesehen ein eigener *HTML* Div-Container. Div-Container werden in *HTML* gerne als logische oder stilistische Gruppierung betrachtet und können auch ineinander verschachtelt sein. Der Vorteil besteht darin, dass man jeder Div eine eigene Klasse zuweisen und so exakt bestimmen kann, wie sie aussehen und wie sie sich gegenüber anderen Elementen der Website verhalten soll. So kann man die Div-Elemente zum Beispiel nebeneinander in einer Reihe anordnen, ihre konkrete Größe, Form und Farbe bestimmen, oder sie gänzlich ohne stilistische Mittel ausstatten um sie für den Betrachter unsichtbar zu gestalten. Eine Besonderheit der Modulübersicht beruht auf dem Fakt, dass nichts „*hard coded*“ ist. Das bedeutet, dass die Anzahl, das Erscheinungsbild und die Beschriftung der einzelnen Elemente nicht explizit im Quellcode angegeben ist und statt dessen dynamisch generiert werden, je nachdem welche Informationen sich in der Datenbank befinden und welchen Lernfortschritt der Lernende bereits erreicht hat. Das ist notwendig da sich die Werte innerhalb der Datenbank, welche als einzige Bezugsquelle dienen, ständig ändern können. Es ist nicht möglich die Beschriftung aller Elemente von Hand im Code zu verankern, denn niemand kann wissen, welche Module morgen benötigt werden. Dennoch muss gewährleistet sein, dass jedes einzelne Lernmodul stets mit den korrekten, leserlichen Informationen gefüllt ist. Hierfür ist es wichtig, dass die Formatierung der Datenbank stets erhalten bleibt. Eine etwaige Änderung an der Spaltenzahl der Datenbank kann katastrophale Folgen haben und dazu führen, dass gar nichts mehr

funktioniert. Ähnlich wie Zahnräder in einem Uhrwerk sind alle Rohdaten im Hintergrund des Projektes genau aufeinander abgestimmt und strikt definiert um die Einhaltung der besprochenen Problematik zu garantieren. In der folgenden Grafik wird der Aufbau eines Div-Container veranschaulicht.

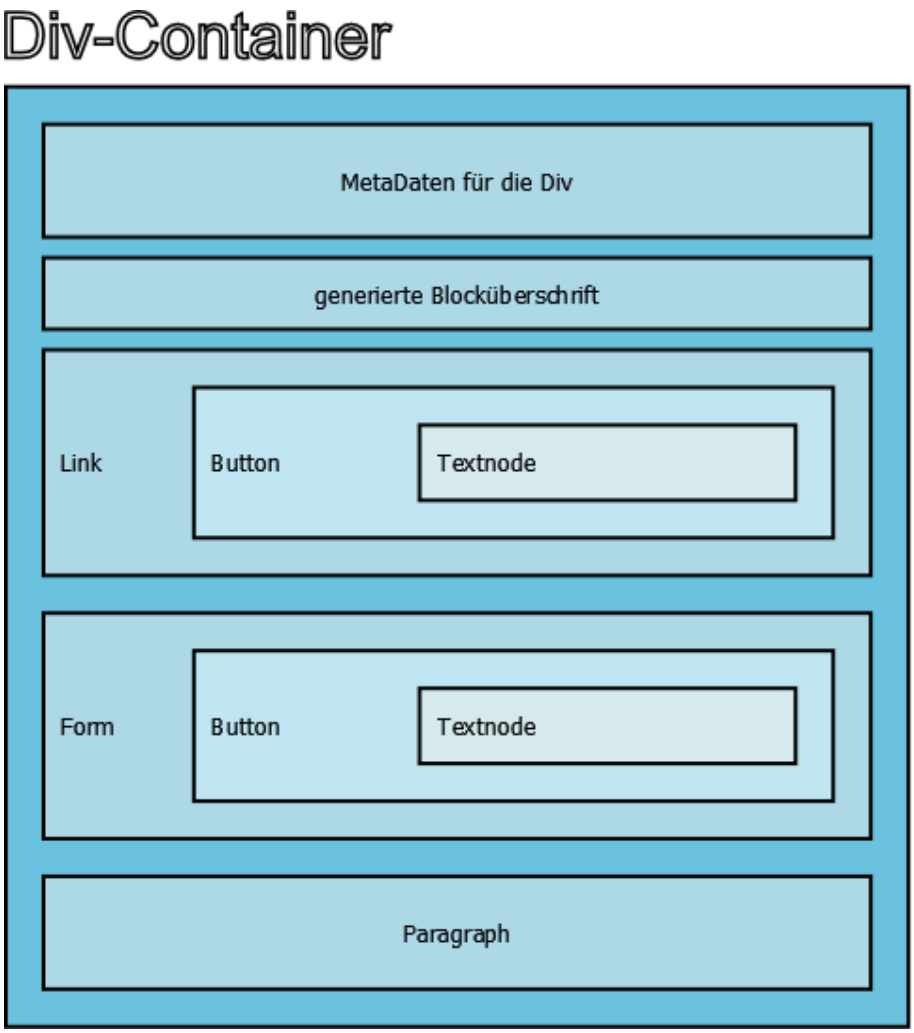


Abbildung 5: theoretischer Aufbau eines Div-Containers

Die MetaDaten setzen sich zusammen aus der ID, welche aus der Datenbank ausgelesen wird, der Klassenbezeichnung, damit mit der „formate.css“-Datei das Erscheinungsbild festgelegt werden kann (dazu später mehr) und der Hintergrundfarbe. In der *createDiv* wird zunächst jeder Container grau gestaltet, die eigentlich Hintergrundfärbung passiert später in der *checkProgress*-Funktion.

Die generierte Blocküberschrift ist ein einfacher Text, welcher mit *HTML* „*<h4>*“-Tags umschlossen ist (Überschrift 4. Grades) und sich aus der ID, der Bezeichnung, einem Zeilenumbruch, dem Wort „Bezeichnung“, der Modulbeschreibung und diversen Formatierungszeichen zusammensetzt.

Betrachtet man Abbildung 5 oben, stellt man fest, dass es 2 Knöpfe (Buttons) gibt, welche sich aber in verschiedenen Elternelementen befinden. Der erste Knopf wird mit einem Link verknüpft, der zweite Knopf ruft eine Funktion auf. Um die etwas seltsame Hierarchie des ersten Knopfes zu verstehen, muss man wissen welche Möglichkeiten *HTML* bietet um Links dynamisch zu generieren. Aus menschlicher Sicht, würde man zunächst einen Knopf erstellen, anschließend einen bestimmten Text auf den Knopf legen und zu guter Letzt definieren was beim Drücken des Knopfes geschehen soll. Die *HTML*-Logik funktioniert aber verschieden, wenn man dynamisch einem Knopf eine URL zuweisen möchte. So erstellt man zunächst einen Link, der die URL des Ziels beinhaltet und weist anschließend diesem Konstrukt einen Knopf zu.

Auch beim zweiten Knopf muss man *HTML* etwas austricksen, wenn man dynamische Ergebnisse erzielen möchte. Normalerweise taucht ein Knopf immer als Abschluss eines *Form*-Elementes auf, um beispielsweise Nutzereingaben zu bestätigen und zur weiteren Verarbeitung vorzubereiten. Da man aber nun einen Knopf haben möchte, welcher eine ganz andere Aufgabe erfüllen soll als bloßen Text weiter zu verarbeiten, legt man zunächst eine *Form*-Element an um den Knopf zu erhalten und weist zuletzt explizit eine selbstgeschriebene Funktion zu. So kommt es, dass 2 scheinbar gleiche Elemente in unterschiedlichen Elternumgebungen eingebettet sind.

Als letztes Element ist ein einfaches *Paragraph*-Element in der Div zu finden. Ein Paragraph ist ein einfaches Text-Element und dient der Ausschrift von Zusatzinformationen wie die benötigten Vorgänger oder ob das Modul bereits gelöste wurde.

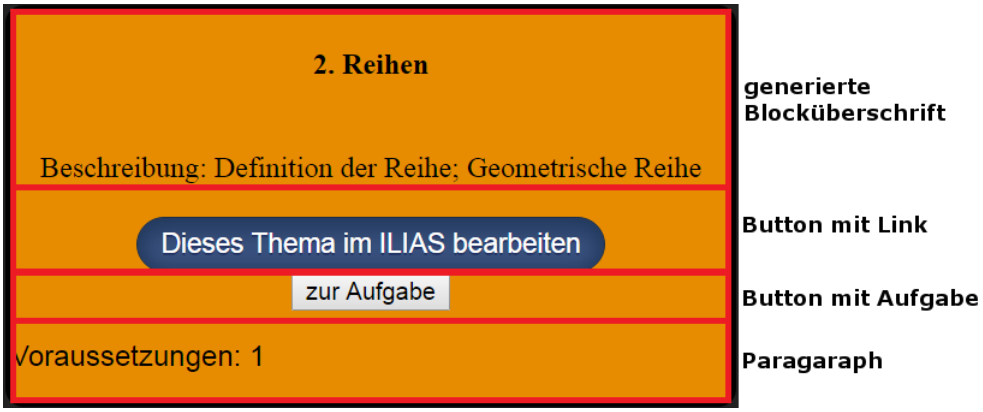


Abbildung 6: Beispiel für einen Div-Containers

### 4.4.3 checkProgress-Funktion

Nachdem alle Div's mit grauen Hintergründen und deaktivierten Aufgabenknöpfen erstellt wurden wird die *checkProgress*-Funktion aufgerufen. Wie der Name schon vermuten lässt, überprüft diese Funktion mithilfe von den erstellten Cookies, wie weit der Lernende im Lernstoff vorangeschritten ist und markiert alle Lerneinheiten dem entsprechenden.
Es gibt 3 Zustände, die eine Lerneinheit annehmen kann:

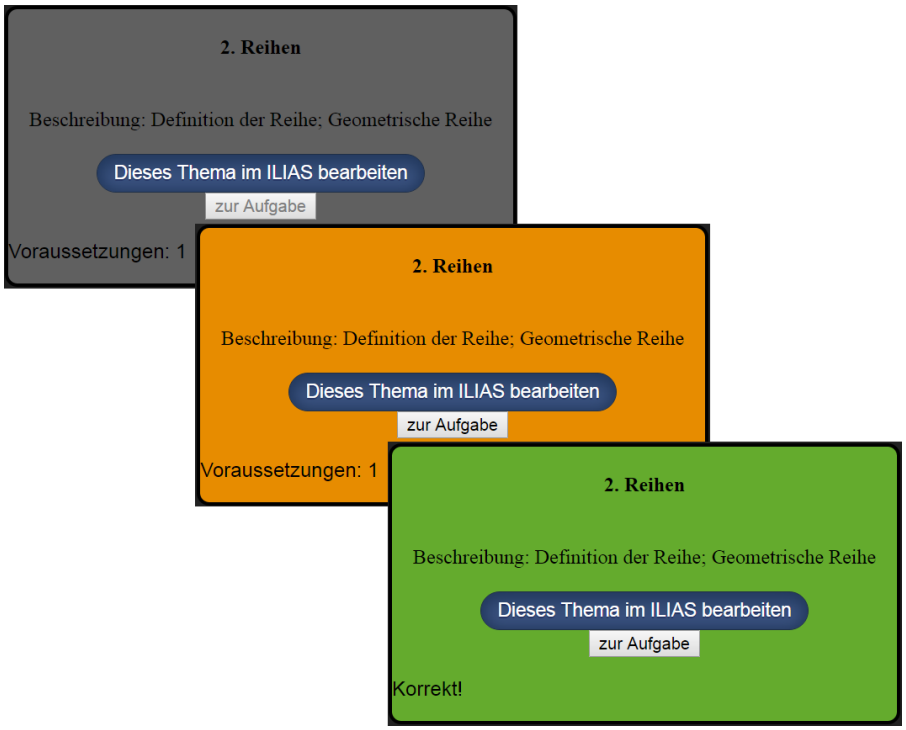


Abbildung 7: Alle Zustände eines Lernmoduls

Ist die Box grau und verfügt über einen ausgegrauten „zur Aufgabe"-Knopf, symbolisiert das dem Lernenden, dass dieser Themenbereich noch nicht für Ihn freigeschaltet wurde. Wurden alle Vorbedingungen für einen Themenbereich erfüllt, in dem die entsprechenden Testaufgaben aller Vorgänger gelöst wurden, ändert sich der Modi der Lerneinheit.

Ist die Box orange, wurden alle benötigten Vorgängermodule gelöst, noch nicht aber das Modul selbst. Klickt der Lernende auf den „Zur Aufgabe"-Knopf und löst die Testaufgabe korrekt, ändert sich die der Zustand der Lerneinheit erneut.

Ist die Box grün bedeutet das, dass die hinterlegte Testaufgabe erfolgreich gelöst wurde. Sind alle Vorgänger einer grauen (also gesperrten) Lerneinheit mit grün markiert, ändert sich der Status der Einheit von grau zu orange und kann vom Lernenden bearbeitet werden.

Der Fortschritt des Lernendens wird ausschließlich über sogenannte Cookies erfasst. Cookies sind Informationsschnipsel, welche dem Browser dienen um wiederkehrende Informationen zu speichern. So muss man beim besuchen der Lieblingswebsite beispielsweise das Passwort nur einmal eingeben und kann es anschließend in einem Cookie speichern lassen. Beim nächsten Besuch wird dann automatisch das Passwort in das dafür vorgesehene Eingabefeld geschrieben. Im Falle dieses Projektes beinhalten die Cookies, ob eine Lerneinheit gelöst wurde. Um Nutzerfortschritte effektiv zu speichern wird normalerweise ein Accountsystem benötigt, bei dem sich jeder Nutzer registrieren muss. Der erreichte Fortschritt kann dann direkt einem bestimmten Nutzer zu gewiesen und jederzeit wieder ausgelesen werden. Das Cookiesystem ist dies bezüglich nicht optimal, da der Fortschritt nicht Personen- sondern Endgerätbezogen ist. Wollen 2 Lernende den selben PC benutzen um die Aufgaben zu lösen, kann der 2. Lernende nur den Fortschritt des 1. Lernendens einsehen und darauf basierend weiter lernen, oder er kann alle Cookies löschen und somit den Fortschritt zurücksetzen. Auch sind Cookies leicht manipulierbar, so verfügt jeder moderne Internetbrowser über die Möglichkeit gespeicherte Cookies anzusehen, zu editieren und neue Cookies zu erstellen.

Der Grund für die Nutzung dieses Systems beruht auf dem begrenzten Zeitrahmen dieser Arbeit. Da ein Nutzerkontosystem zu komplex wäre, um es in das Projekt zu integrieren, hat man sich für die weniger professionelle Methode der Cookies entschieden, welche wesentlich Zeiteffizienter ist. Ein weiterer Grund ist, dass die Erstellung eines Nutzeraccounts immer eine gewisse Hürde für den Lernenden darstellt, da hierbei persönliche Daten preisgegeben werden müssen. Denn für eine seriöse Nutzerkontenerfassung ist zwangsweise eine E-Mail-Verifizierung des neu angelegten Nutzers nötig und nicht jeder trägt gerne seine E-Mail-Adresse oder andere persönliche Daten auf einer fremden, gänzlich unbekannten Website ein.

Wird eine der Testaufgabe korrekt gelöst, wird für den entsprechenden Lernabschnitt ein Cookie generiert. Dieser Cookie enthält einen einfachen Textstring, welcher im spezifischen Fall hier „*solved*“ lautet. Die Funktion überprüft nun für jedes Lernmodul einzeln, ob ein Cookie mit genanntem Text existiert. Wird kein Cookie gefunden, unternimmt das Programm nichts und sucht weiter. Lediglich bei dem aller ersten Lernmodul ist eine Sonderbehandlung nötig. Zur Erinnerung: Ein Lernmodul wird immer in grauem, deaktivierten Zustand generiert und wird erst freigeschaltet, wenn alle Vorgängerbedingungen erfüllt wurden. Nun verfügt das aller erste Lernmodule jedoch nicht über einen Vorgänger und würde somit in der Theorie niemals freigeschalten werden. Dies wiederrum hätte zur Folge, dass der Lernende alle Lernmodule welche das erste Modul als Vorbedingung haben niemals bearbeiten könnte. Um dies zu verhindern setzt die *checkProgress*-Funktion das erste Modul immer auf den „Verfügbar“-Status (orangener Hintergrund) und gegebenenfalls später auf „gelöst“ (grüner Hintergrund).

Eine weitere Aufgabe der Funktion ist es, alle verfügbaren Lernmodule auf den orangenen Status zu setzen. Um dieser Aufgabe gerecht zu werden ist es notwendig, alle Nachfolger des gelösten Moduls zu ermitteln und zu überprüfen, ob alle deren Vorgänger gelöst wurden. Tatsächlich ist hier weniger die Logik ein Problem gewesen. Viel mehr wurde die Sache durch die Art der Variablenspeicherung erschwert. Wie bereits in Abschnitt 4.4.1 erwähnt liegen alle Lernmodule und deren Informationen in einem Object-Array vor. Das heißt, dass jedes Lernmodul ein Objekt ist, welche alle Nacheinander in einem Array, also einer Liste, gespeichert wurden. Der Umgang und das Auslesen von Informationen aus Objekten ist etwas schwieriger als das Managen von Arrays, da man durch ein Objekt nicht einfach vom ersten bis zum letzten Element iterieren kann. Um Informationen aus einem Objekt zu bekommen muss man das gewünschte Objekt suchen (man iteriert also durch das gesamte Objektarray und sieht sich jedes Objekt einzeln an) und dann die gewünschten Informationen aus den „*Key*-*Value*-Paaren“ des Objektes auslesen. *Key*-*Value*-Paare sind Datenpaare welche aus einem Schlagwort (dem *Key*) und den dazu gehörigen Werten (*Value*) zusammengesetzt sind.

So kann man sich beispielsweise nur die Vorgänger eines Elementes ausgeben lassen:

Listing 9: Auslesen aller Vorgänger der Lerneinheit i

```
var result = nachfolger.findIndex(function (obj){
        return obj.hat_vorganger == i;
        });
parseInt(result);
```

Zur Erläuterung: Wir durchsuchen die Objektliste mit dem Namen „nachfolger“ und wollen den Index des Objektes herausfinden, bei dem das Wertepaar mit dem Key hat_nachfolger gleich dem Iterationsschritt $i$ der Schleife ist. Haben wir ein solches Objekt gefunden, wird der Index auf der Variable result gespeichert und rein vorsorglich nochmal mit Hilfe von *parseInt* zu einem reinen Integer umgewandelt.

Auch in dieser Funktion bestand wieder die Herausforderung darin alles so dynamisch zu gestalten, dass sie für alle möglichen Eventualitäten gewappnet ist. So existieren zum Beispiel Testaufgaben mit mehreren Lösungen, die der Lernende angeben muss. Erst wenn alle Lösungen einer Aufgabe korrekt sind, soll der Themenbereich als gelöst markiert werden. Hierfür ist die „*isSolved*“-Funktion zuständig (siehe Abbildung 5). Für jedes korrekt gelöste Lösungsfeld einer Testaufgabe wird der boolische Wert „true“ in ein Array geschrieben. Ist das Lösungsfeld falsch beschrieben, setzt die Funktion ein „false“ an die entsprechende Stelle. Erst wenn alle Werte des so erstellten Arrays auf „true“ stehen, werden die Nachfolger-Module freigeschaltet. In der aktuellen Version zum Zeitpunkt dieses Projektes werden bis zu vier simultane Lösungsfelder pro Aufgabe unterstützt.

### 4.4.4 getVorganger-Funktion

Wie bereits mehrfach verdeutlicht wurde, ist die Ermittlung aller Vorgänger und Nachfolger jedes einzelnen Lernmoduls essentiell für die Funktionsweise der Website. Während die Nachfolger bereits in der *checkProgress*-Funktion ermittelt wurden, hat die Vorgängeranalyse eine eigene Funktion bekommen, da ihre Rolle wesentlich tragender ist. Im Kern funktioniert das Suchen der Vorgänger wie das Suchen der Nachfolger (siehe: Listing 9), der einzige Unterschied beruht hier auf der Tatsache, dass es sich um eine eigenständige Funktion, welche die ID des aktuellen Lernmoduls übermittelt bekommt und als Ergebnis ein Array zurückliefert, in dem alle Vorgänger des aktuellen Lernmoduls notiert sind um sie später an verschiedenen Stellen im Programmcode zu verwenden. Beispielsweise in der *checkProgress*-Funktion.

### 4.4.5 getCookie-Funktion

Mit der *getCookie*-Funktion werden die erstellten Cookies abgefragt und auf deren Inhalt überprüft. Hierfür muss eine Reihe von Formatierungsbefehlen erfolgen, um den gewünschten String von anderen, nicht gebrauchten Informationen, wie etwa dem Ablaufdatum des Cookies, zu isolieren. Die Funktion überprüft also ob an der jeweiligen Stelle des Cookie der Textstring „*i*=*solved*“ vorliegt, oder nicht. Hierbei steht „*i*“ für das jeweilige Lernmodule. Ein kurzes Beispiel soll die vorliegende Funktionsweise und Zusammenhänge besser erläutern: Student $X$ löst das Lernmodul mit der ID $5$. Nach dem Student $X$ nun die abschließende Testaufgabe richtig gelöst hat, wird ein Cookie gesetzt. Das Cookie hat als Titel die ID des gelösten Lernmoduls, also in diesem Fall die Nummer $5$, sowie ein Ablaufdatum von $6$ Monaten. Damit sich die restliche Website nun korrekt an den Lernfortschritt anpassen kann, übermittelt die *getCookie*-Funktion für jedes vorhandene Cookie den Inhalt und übergibt in dieser Beispielsituation den String „*5*=*solved*“ an die *checkProgress*-Funktion, in welcher der Aufruf von *getCookie()* statt fand. Dies ermöglicht der *checkProgress*-Funktion das korrekte Darstellen der Farbcodierung der Lerneinheiten.

# 5 Beispiel

## 5.1 Der Editor

In diesem Abschnitt wird ein einfaches Beispiel durchgesprochen werden. Beginnend mit einem leeren System und endend mit dem Lösen aller Testaufgaben von einem Lernenden.

Als Ausgangssituationen ist eine leere Datenbank sowie die Website gegeben. In der Datenbank befinden sich lediglich zwei leere Tabellen: die Tabelle *Lernmodule* und die entsprechende Zuordnungstabelle für Vorgänger und Nachfolger. Das Wissenslexikon existiert bereits und wird per Linkverweis zugänglich gemacht. In diesem Beispiel liegen die Lerninhalte in einem fiktiven *ILIAS*-System mit der Adresse „ILIAS.com“. Diese Adresse ist nicht repräsentativ für eine eventuell tatsächlich existierende Website mit dieser Adresse und dient lediglich der Verdeutlichung dass es sich um ein *ILIAS*-System handelt.
Damit auf der Website die Lernmodule angezeigt werden, muss der Systemadministrator oder eine andere zugriffsberechtigte Person nun Datensätze in die Datenbank einspeisen. Dies kann direkt über das integrierte Webinterface der Datenbank geschehen oder über die Editorfunktion der Website. Wählt man den Weg über das Webinterface der Datenbank hat man vollen Zugriff auf alle Einstellungen, Rechte und Werte der Datenbank.

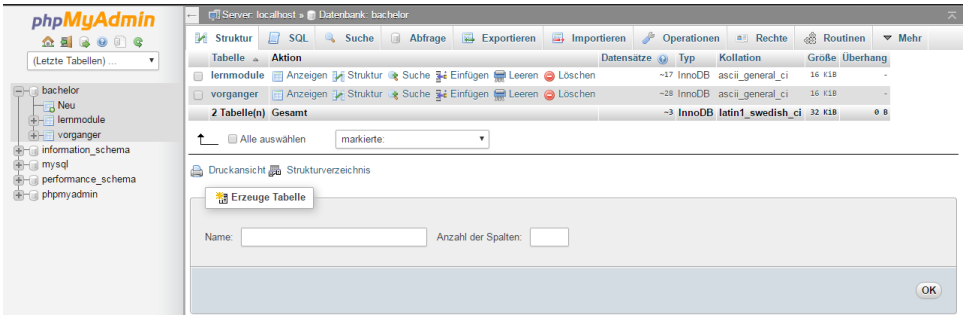


Abbildung 8: Webinterface der Datenbank mit den beiden Tabellen lernmodule und vorganger

Ein Leihe könnte sich aber von der Vielfalt an Optionen durch aus überfordert fühlen. In diesem Fall ist es ratsam die Eingabemasken der Website zu verwenden.

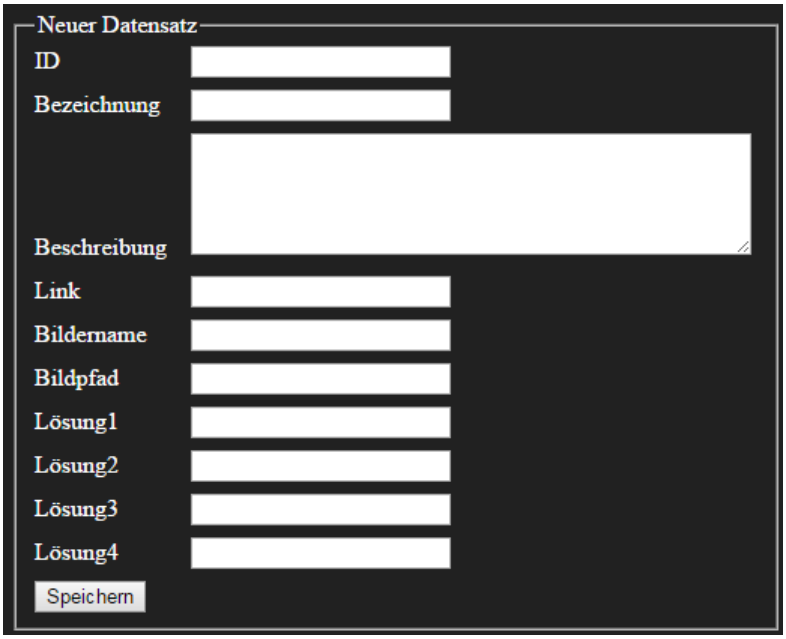


Abbildung 9: Eingabemaske „Neuer Datensatz“ der Webseite

In diesem Beispiel wollen wir 5 Datensätze einfügen und erledigen das mit Hilfe der Eingabemasken. Die 5 Beispieldatensätzen haben folgende Werte:

| **ID** | **Bezeichnung** | **Beschreibung** | **ILIAS-Link** | **Bildpfad** | **Lösung** |
|---|---|---|---|---|---|
| 01 | Addition | Addieren von Zahlen | ILIAS.com /addition | ../aufg /addition.png | 5 |
| 02 | Subtraktion | Subtrahieren von Zahlen | ILIAS.com /subtraktion | ../aufg /subtraktion.png | 2 |
| 03 | Multiplikation | Das kleine 1 mal 1 | ILIAS.com /multiplikation | ../aufg /multiplikation.png | 9 |
| 04 | Division | Dividieren von Zahlen | ILIAS.com /division | ../aufg /multiplikation.png | 4 |
| 05 | Grammatik | Lehre der Rechtschreibung | Duden.de | ../aufg /division.png | Xylophon |

Man beachte, dass die wirkliche Tabelle zusätzliche Spalten bietet. Darunter sind 3 weitere Spalten für Lösungen, falls eine Aufgabe mehr als nur eine Lösung erfordert, sowie eine Spalte um für das Aufgabenbild eine Bezeichnung abzuspeichern, damit im Falle eines Fehlers leichter ausgemacht werden kann, welches Bild fehlt oder beschädigt ist. Des weiteren sind bewusst fehlerhafte und nicht passende Einträge gemacht wurden, um im weiteren Verlauf des Beispiels alle Eventualitäten durchspielen zu können.

Die Zuordnungstabelle hat nur 2 Spalten. Beide Spalten beinhalten hierbei jeweils eine unterschiedliche ID aus der 1. Tabelle. In unserem Beispiel sieht die Tabelle wie folgt aus:

| **ID** | **hat_vorganger** |
|---|---|
| 2 | 1 |
| 3 | 1 |
| 3 | 2 |
| 4 | 3 |

Hierbei fällt sofort auf, dass Werte mehrfach vorkommen und beiden Seiten der Tabelle zugeordnet werden können. Damit man besser versteht was die Tabelle aussagt, folgt die Tabellenlogik jetzt in ausgeschriebener Form und als Verlaufsgraph.

- Das Modul mit der ID 02 hat als Vorgänger das Modul 01
- Das Modul mit der ID 03 hat als Vorgänger die Module 01 und 02
- Das Modul mit der ID 04 hat als Vorgänger das Modul 03

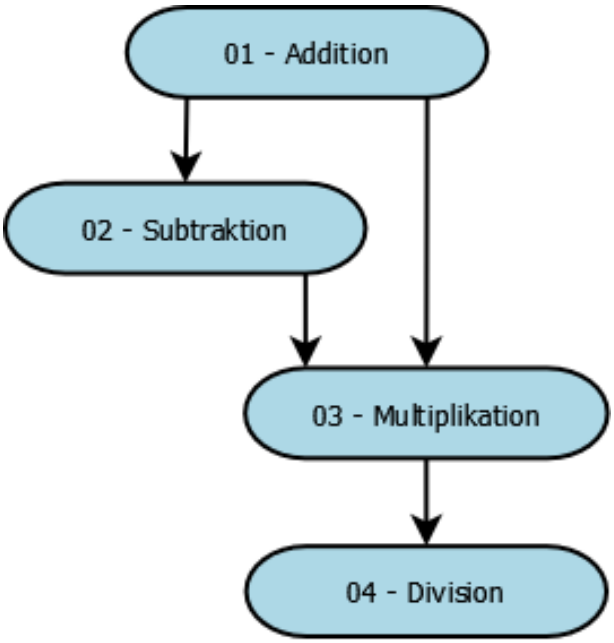


Abbildung 10: Abhängigkeitsrelation der Beispielmodule

Nach dem alle Werte korrekt in die Eingabemaske eingegeben wurden, kann mit Hilfe der Übersichtsliste im Editorfenster noch einmal überprüft werden, ob alle Lernmodule eingetragen wurden oder ob ein Fehler bei der Nummerierung bzw. Bezeichnung unterlaufen ist.

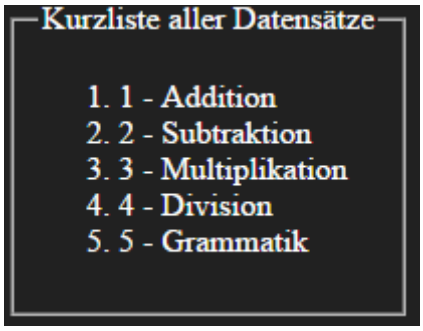


Abbildung 11: Liste der Beispielmodule

Nun haben wir hier 2 Fehler: eines der Lernmodule passt nicht zum Lernstoff und in einem anderen Datensatz wird die falsche Testaufgabe angezeigt. Kümmern wir uns zunächst um den falschen Datensatz. Da wir keinen Nutzen für ihn haben, können wir ihn einfach vollständig aus der Datenbank entfernen. Hier haben wir wieder die Möglichkeit das Datenbankinterface oder die entsprechende Maske auf der Editor-Seite zu verwenden. Wir benutzen hier im Beispiel die Eingabemaske und tragen dort einfach die ID des Lernmoduls ein, dass gelöscht werden soll. Wie wir anhand der Übersichtsliste erkennen können, wollen wir den Datensatz mit der ID „5“ löschen.

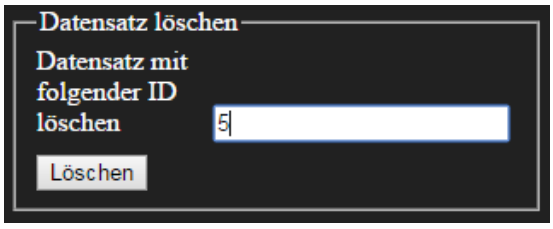


Abbildung 12: Eingabemaske zum Löschen eines Datensatzes

Drückt man abschließend auf „Löschen“ wird der komplette Datensatz aus der Datenbank entfernt und verschwindet beim nächsten Aktualisieren der Seite aus der Übersichtsliste.

Um das zweite Problem zu beheben nutzen wir die Maske zum Ändern von Datensätzen. Dazu wählen wir im Dropdown-Menü[9] den gewünschten Datensatz aus, in unserem Fall Nummer 4 - Divison.

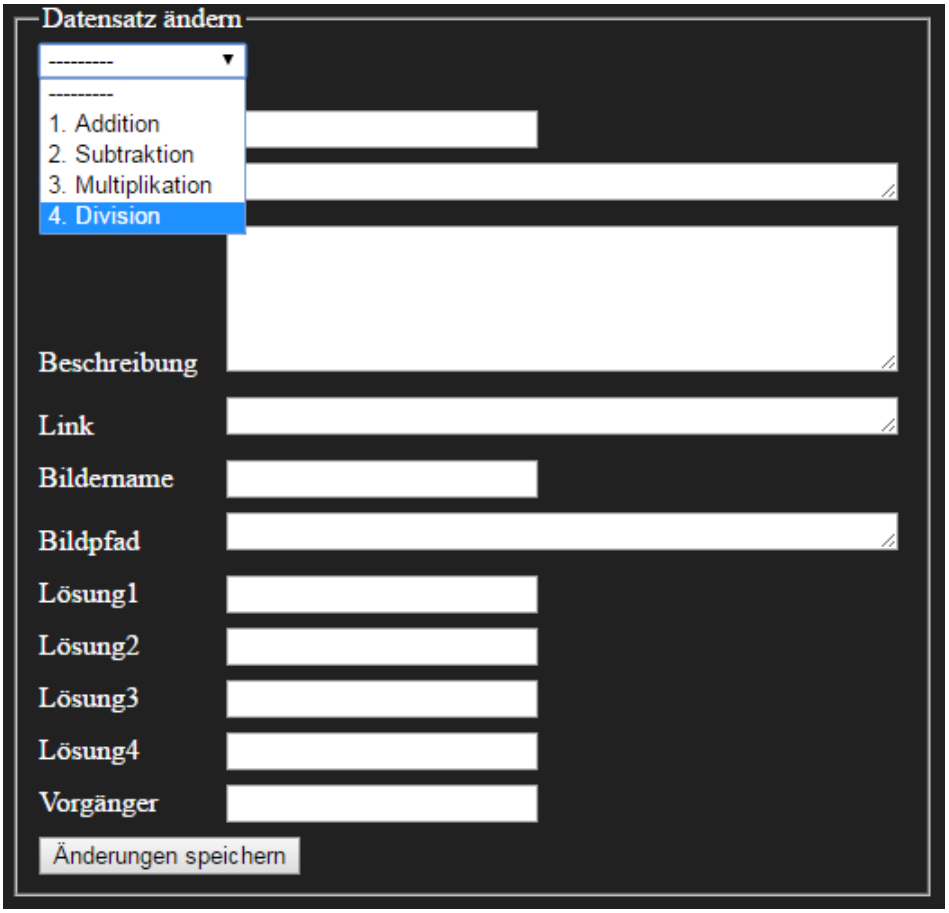


Abbildung 13: Auswahl des zu ändernden Datensatzes

[9] [Dropdown]-Menü „[...], ist ein Steuerelement einer grafischen Benutzeroberfläche. Dabei wird über einen Mausklick auf eine Schaltfläche einer Menüleiste oder Symbolleiste ein Untermenü angezeigt.“

Es werden automatisch alle Eingabefelder, welche in der Datenbank hinterlegt wurden, mit den aktuellen Werten ausgefüllt. Nun ändern wir die fehlerhafte Stelle ab und drücken auf „Änderungen speichern“.

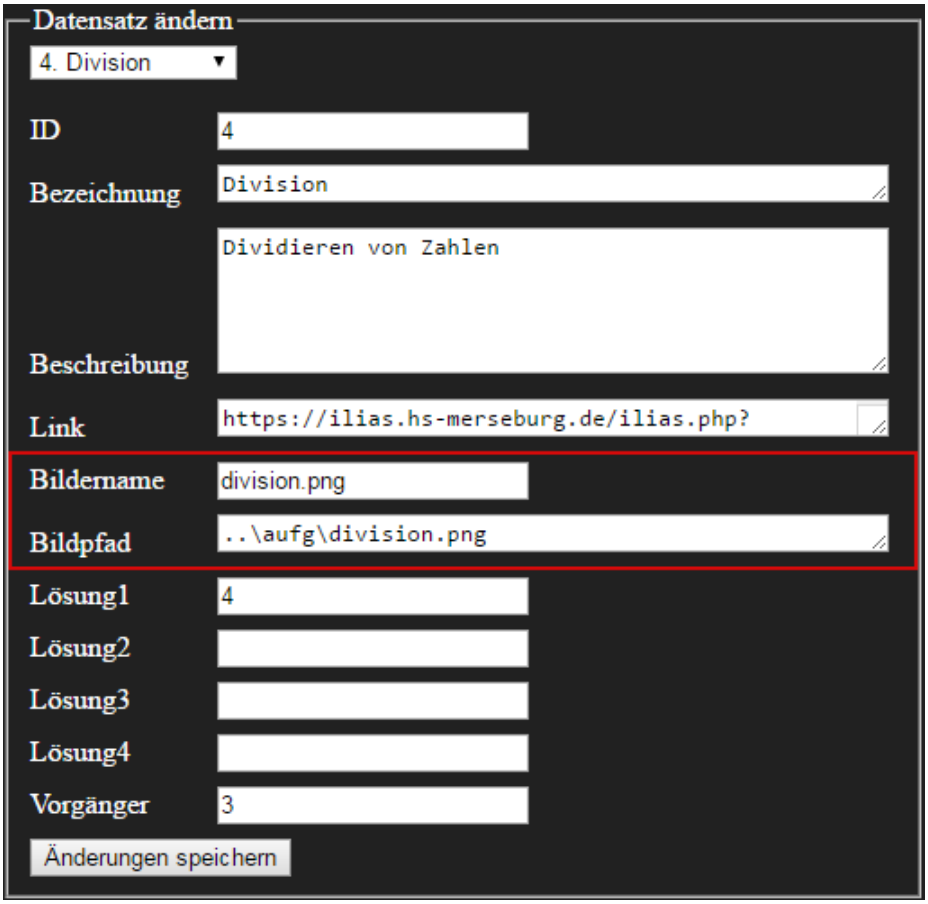


Abbildung 14: Korrektur des falschen Datensatzes

Wir haben jetzt alle Fehler korrigiert und können uns der letzten Aufgabe des Editors zu wenden: den Bildern für die Testaufgabe.

Um die Testaufgaben, welche in Bildform parallel zur Datenbank im gleichen Dateisystem liegen, verfügbar zu machen müssen zunächst die Bilddateien erstellt werden. Dies kann auf mehreren Wege erfolgen. Eine Möglichkeit wäre zum Beispiel das Erstellen einer *PDF*-Datei mit Hilfe von *LaTeX* und einem anschließenden Ausschneiden der entsprechenden Aufgaben mit Hilfe eines Screencapture-Tools[10]. Zu guter Letzt müssen die Bilddateien noch in den entsprechenden Ordner im Dateisystem abgelegt werden. Dabei ist zu beachten, dass der Bildpfad in der Datenbank relativ zum Speicherort der Datenbank liegt. Das bedeutet, dass als Pfadursprung der Ordner angenommen wird in dem sich die Datenbank befindet. Angenommen man hätte folgende Datenstruktur:

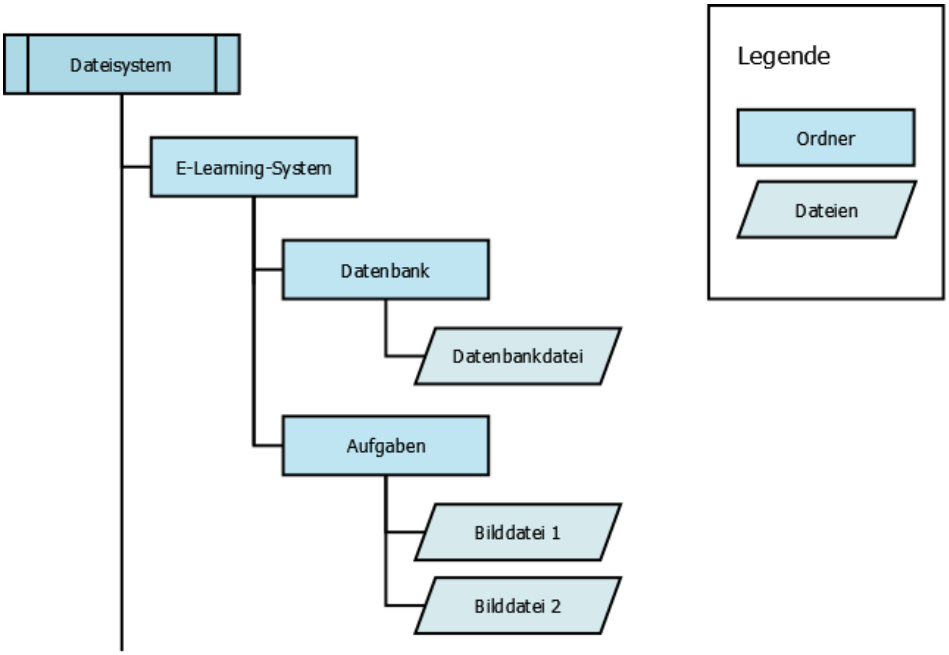


Abbildung 15: Schema des Dateisystems

So müsste der Dateipfad für „*Bilddatei1*“ lauten: „ ..\Aufgaben\Bilddatei1“, da die Datenbank immer von sich selbst als Ausgangspunkt ausgeht. Wurden alle Bilddateien angelegt und mit dem korrekten Pfad in die Datenbank eingetragen hat der Autor seinen Aufgabenbereich abgeschlossen.

[10]eine Software mit deren Hilfe man den gesamten Bildschirminhalt, oder Teile davon, in Video- oder Bildformat festhalten und abspeichern kann. (Windows: Snipping Tool - mitgelieferte Software; IOS: Bildschirmfoto - [CMD]+[Shift]+[4])

## 5.2 Der Lerner

Als Lerner hat man eine Hauptansicht, in welcher auf einen Blick alle Lernmodule einsehbar sind. Außerdem erkennt man anhand der Farbkodierung der einzelnen Module den Lernstatus. Grün steht für gelöste Module, Orange steht für freigeschaltete Module und grau steht für gesperrte Module. Am Anfang ist immer nur das erste Modul freigeschalten, also orange hinterlegt.

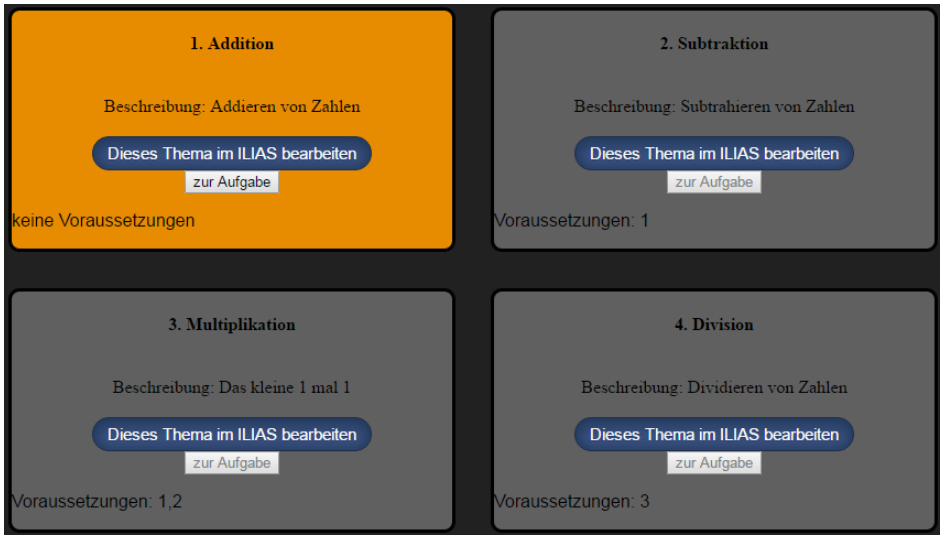


Abbildung 16: Startansicht des Lerners

Als erstes muss sich ein Lernender also das Wissen für die erste Lerneinheit aneignen. Um zur Wissenssammlung zu gelangen, muss der blaue „dieses Thema im ILIAS bearbeiten“-Knopf gedrückt werden. Es öffnet sich ein neuer Tab mit der Zieladresse welche zuvor in die Datenbank eingetragen wurde. Im Beispielfall wäre das also „ILIAS.com/Addition“, falls man den Knopf im ersten Modul gedrückt hat. Hat man sich vorher bereits im *ILIAS* mit seinen persönlichen Daten angemeldet, so landet man nun direkt auf der entsprechenden Informationsseite des angeklickten Themas. Hat man sich jedoch noch nicht eingeloggt, landet man auf der Loginseite des *ILIAS*. Dort muss man sich zunächst einloggen und anschließend den blauen Knopf erneut anklicken um zur Wissenssammlung zu gelangen.

Nun kann der Lernende nach eigenem Ermessen den angebotenen Lernstoff verinnerlichen. Fühlt sich der Lernende informiert genug um die Testaufgabe zu lösen, kann er zur E-Learning-Website zurückkehren und den Knopf „zur Aufgabe“ drücken. Hierdurch öffnet sich erneut ein neuer Tab mit dem Bild der Aufgabe, welche zuvor im Dateisystem der Datenbank abgelegt wurde.

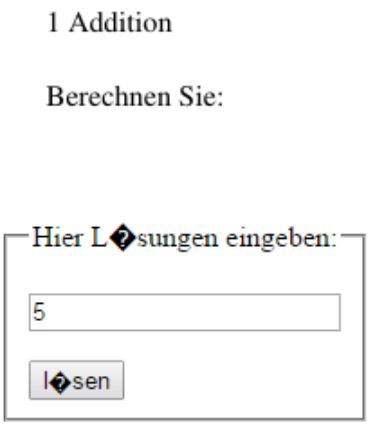


Abbildung 17: Ansicht der Beispielaufgabe zur Addition

Links unter der Aufgabe befindet sich ein Kästchen in dem man die Lösung (gegebenenfalls bis zu 4 Eingabefelder pro Aufgabe) eingeben kann. Ist die Lösung falsch (das heißt, weicht die eingetragene Antwort von der in der Datenbank hinterlegten Lösung ab) bekommt man eine Meldung, dass die Lösung nicht korrekt war.

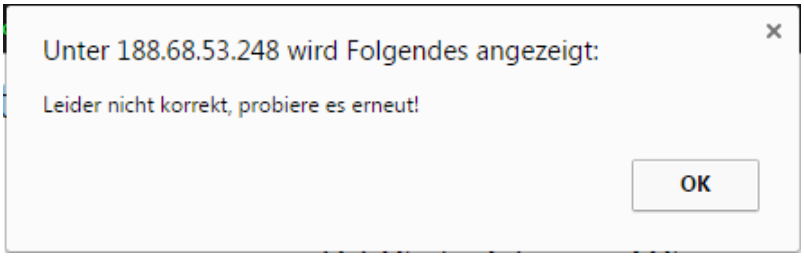


Abbildung 18: Browserbenachrichtigung bei inkorrekter Lösung, die angezeigte IP-Adresse ist hierbei die Adresse der Website.

Es gibt hierbei kein Limit wie oft man die Testaufgaben lösen darf. Man unterliegt keinem Leistungsdruck und kann die Aufgabe selbst nach dem man sie korrekt gelöst hat noch einmal aufrufen und es erneut versuchen. (Bemerkung: zum Zeitpunkt dieser Arbeit gibt es einen Fehler an dieser Stelle: gibt man ein falsches Ergebnis ein und drückt in der auftauchenden Meldung auf „OK“ verschwindet der komplette Inhalt des Aufgabenfenster. Man muss den Tab schließen und die Aufgabe wieder per Druck auf den „zur Aufgabe“-Button aufrufen um sie erneut zu lösen. Es reicht nicht aus den Tab nur zu aktualisieren.)

Gibt man die korrekte Lösung ein erhält man folgende Nachricht:

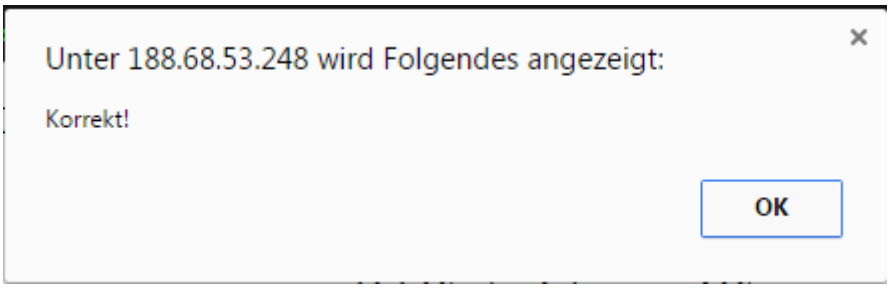


Abbildung 19: Browserbenachrichtigung bei korrekter Lösung

und man landet per Klick auf „OK“ auf der nun aktualisierten Hauptansicht. Das soeben gelöste Thema ist nun grün hinterlegt und alle weiteren Module, welche das erste als Voraussetzung hatten sind nun orange. Dieses Muster wendet der Lerner fortlaufend an, bis er entweder sein persönliches Tagesziel erreicht oder alle Module gelöst. Insofern der Lerner seine Cookies nicht löscht oder das Endgerät ändert, kann er problemlos zu einem beliebigen Zeitpunkt weiterlernen. Jedoch verfallen die Cookies nach einem Semester (einem halben Jahr nach Erstellung) automatisch und jeglicher Fortschritt wird zurückgesetzt.

# 6 Anwendungsbeispiel

Da mit dem vorherigen Abschnitt nun alle Arbeitsschritte einmal angewendet wurden, wird in diesem Abschnitt ein komplexeres Beispiel erstellt um auch die letzten Unklarheiten zu beseitigen.

In diesem Beispiel wird der Inhalt der Plattform von einem Professor an einer Universität erstellt, welcher eine semesterbegleitende Lernhilfe für seinen Mathematikkurs zum Thema Differentialrechnung erstellen will. Die theoretischen Grundlagen sind bereits alle online in einer Wissenssammlung im universitätseigenen *ILIAS*-System zu finden und müssen nur noch verlinkt werden. Hauptanwender dieser Lernhilfe sollen Studenten der Betriebswirtschaftslehre sein.

## 6.1 Aufgaben des Professors

Der Professor hält diese Vorlesung bereits seit einigen Jahren, weshalb sein Kurs bereits eindeutig strukturiert und bewährt ist. Die einzelnen Vorlesungskapitel sind alle bereits bekannt. Die Kursunterteilung sieht also folgendermaßen aus:

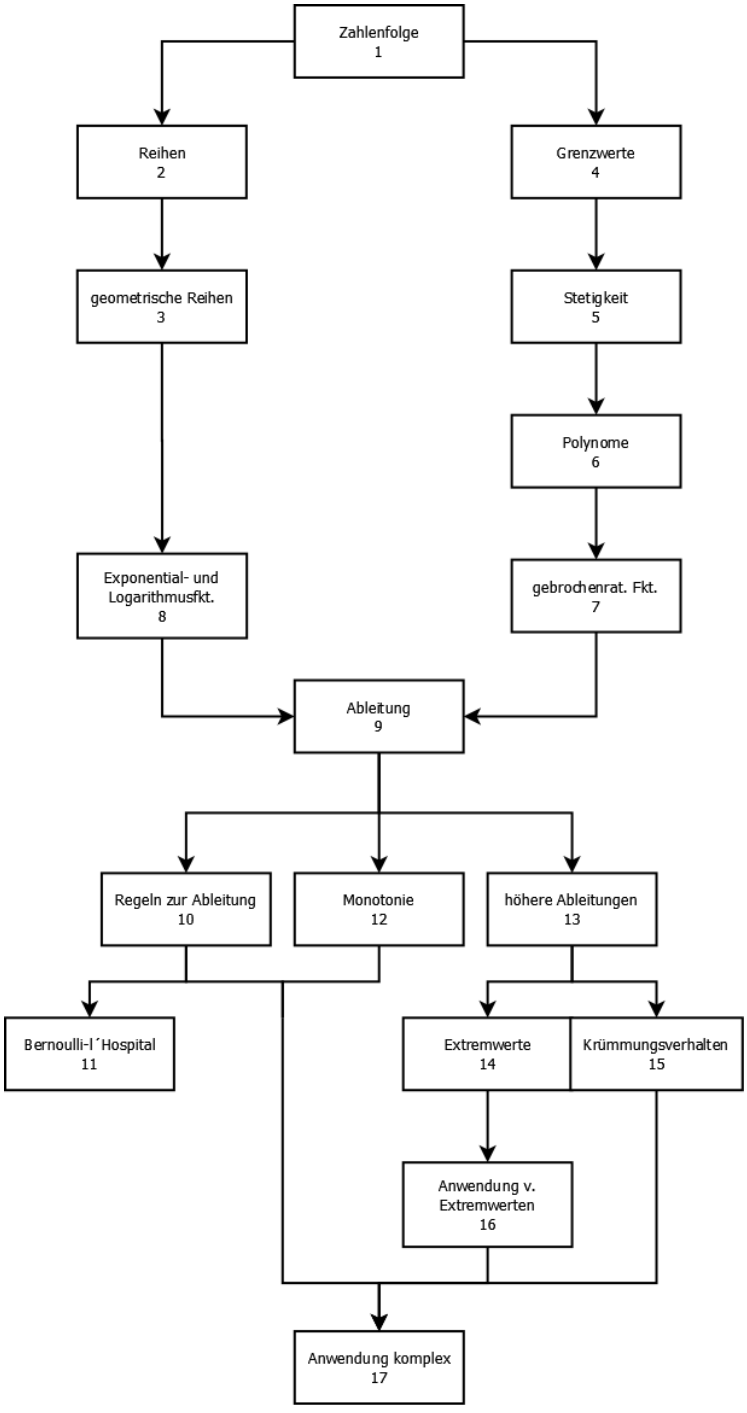


Abbildung 20: Übersicht der erstellten Lernmodule

Jede Lernmodul verfügt über einen Titel, eine Beschreibung, einen Link zum ILIAS, eine Beispielaufgabe als .png-Datei, eine Bezeichnung für das Bild der Aufgabe sowie 1 bis 4 Lösungen pro Aufgabe. Desweiteren ist genau definiert, welche Module Y benötigt werden um Modul X freizuschalten. (Siehe Abbildung 20)
Wie genau die Inhalte in die Datenbank eingetragen werden, wird in diesem Kapitel nicht weiter erläutert, da dazu das allgemeine Beispiel (Abschnitt 5) dient.

Mit einem Blick auf die Editor-Seite erkennt der Professor anhand der Kurzliste nun augenblicklich ob alle Lernmodule vorhanden sind:


Abbildung 21: Kurzliste aller Datensätze

Man beachte die doppelte Nummerierung am Anfang jeder Zeile. Die erste Nummerierung ist eine fortlaufende Nummerierung beginnend bei 1 , in einer-Schritten zählend bis zum letzten Lernmodul. Die zweite Nummerierung ist die ID des entsprechenden Lernmoduls und hängt von der Eingabe des Professors ab.
Durch die 2 verschiedenen Zählweisen, welche auf den ersten Blick redundant wirken, lässt sich schnell ermitteln, ob die richtige Menge der Lernmodule erstellt wurde und ob die manuelle ID-Vergabe der Module fehlerfrei ist.

Um den Unterschied der beiden Nummerierungen nochmal zu veranschaulichen wurde in der folgenden Abbildung ein Extrakapitel mit der ID 39 zu der Liste der Datensätze hinzugefügt. Es befinden sich nun insgesamt 18 Lernmodule in der Datenbank, 17 der 18 Module besitzen eine fortlaufende ID von 1 - 17 und das 18. Modul hat die ID 39.

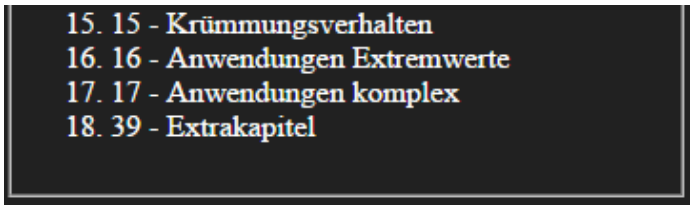


Abbildung 22: Ausschnitt der Kurzliste mit einem Extrakapitel

Zu guter Letzt kann mit Hilfe der Kreisfreiheit-Kontrolle überprüft werden, ob der erstellte Lernplan einen Widerspruch besitzt oder ob er problemlos abgearbeitet werden kann.


Abbildung 23: Kontrollanzeige zur Überprüfung der Kreisfreiheit

## 6.2 Nutzung durch den Studenten

Ein Student, welcher nun zu beginn des Semester die IP-Adresse/den Link der Plattform von seinem Dozenten erhalten hat und zum ersten Mal die Lernerseite aufruft sieht folgendes:

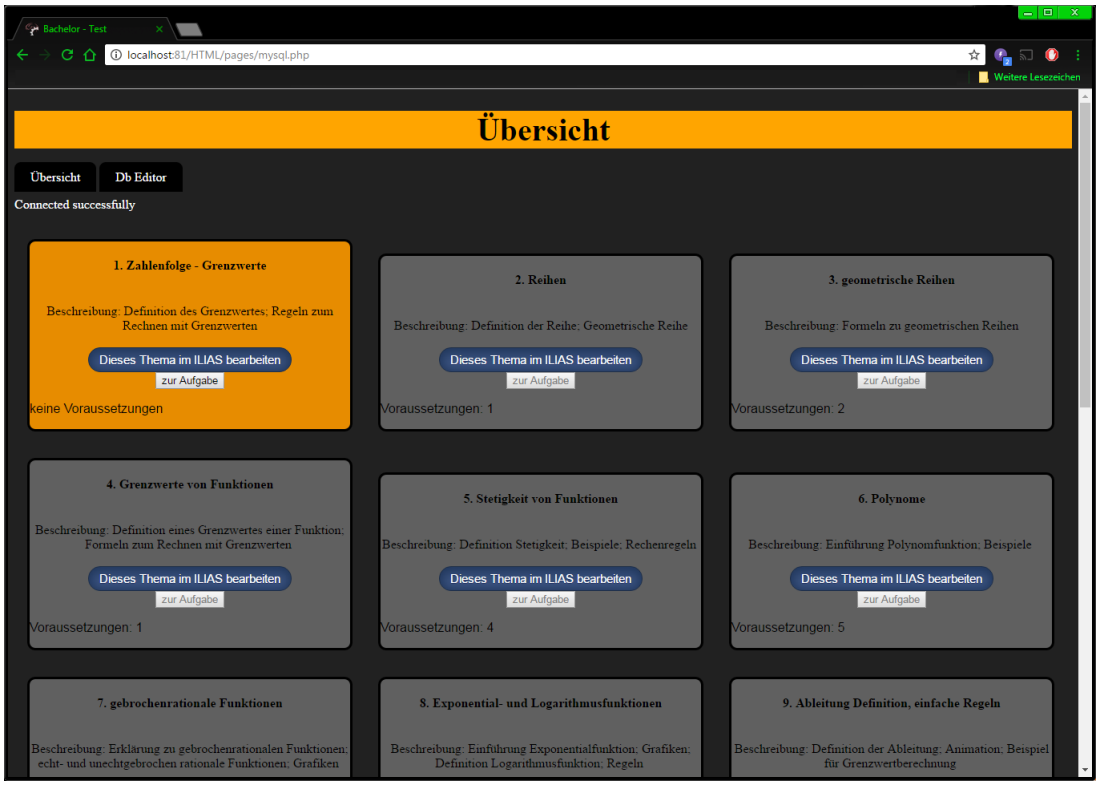


Abbildung 24: Ausgangssituation der Website

Es sind alle Module ausgegraut, bis auf das Erste. Von hier aus hat der Student nun die Möglichkeit nach belieben sich im ILIAS zu einem Thema zu belesen oder die Informationen der Vorlesungen zu festigen. Fühlt er sich in der Lage, eine Rechenaufgabe selbstständig zu lösen, so kann er sein Wissen durch das Lösen der Testaufgabe unter Beweis stellen. Mit einem Klick auf den „zur Aufgabe“-Button öffnet sich ein neuer Tab in welchem die Aufgabe gelöst werden kann. Wie schon im Abschnitt 5 Beispiel erwähnt, stehen dem Studenten beliebig viele Lösungsversuche zur Verfügung, selbst wenn die Aufgabe bereits einmal richtig gelöst wurde, kann er es jederzeit erneut versuchen.

Nachdem die erste Aufgabe durch Eingabe der korrekten Lösung erfolgreich gelöst wurde:

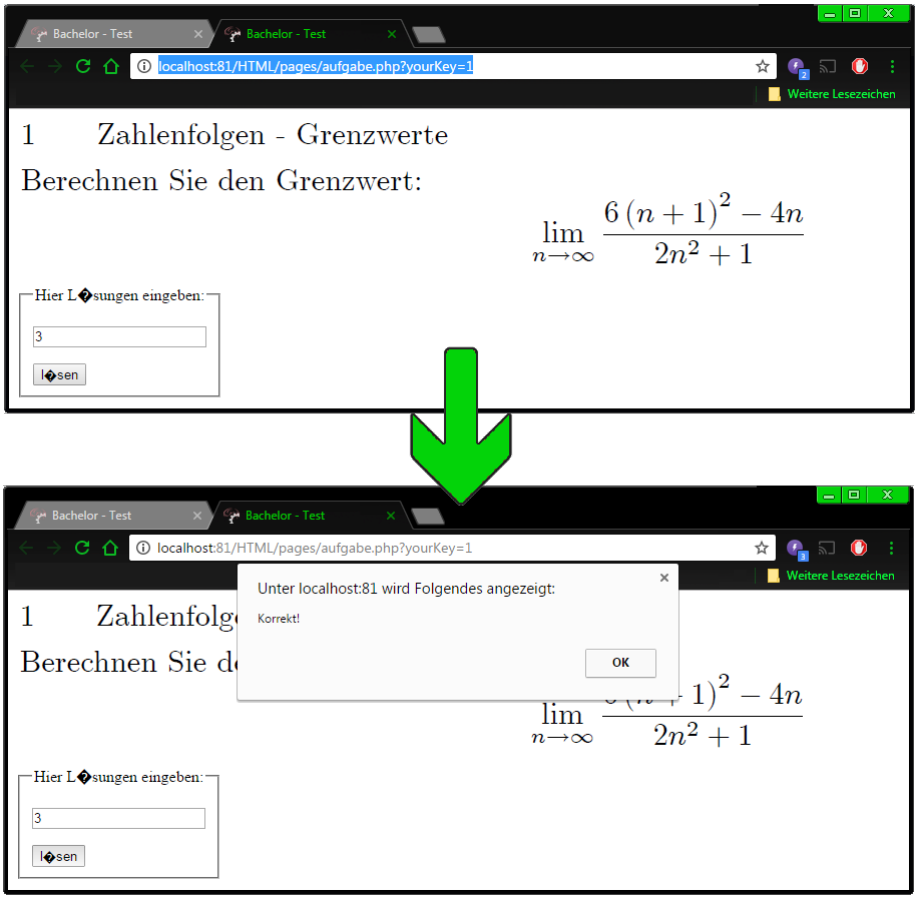


Abbildung 25: Aufgabe 1 wurde korrekt gelöst

Sieht der Student auf der Lernerseite, dass für ihn die Nachfolgenden Module 2 und 4 freigeschaltet wurden. (wie in Abbildung 20: „Übersicht der erstellten Lernmodule“ zu erkennen)

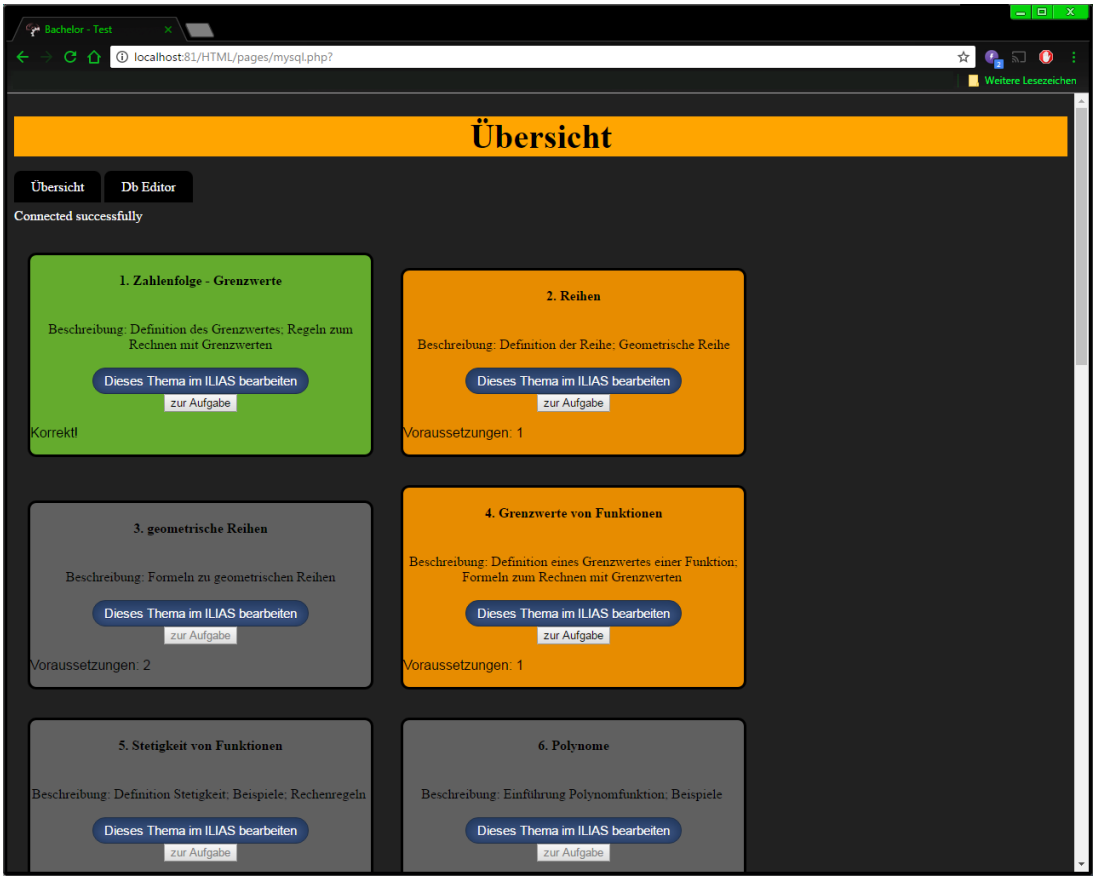


Abbildung 26: Lernerseite nach dem Aufgabe 1 gelöst wurde

Von hier an kann der Lerner nun beliebig das Planspiel wiederholen und mit jeder richtigen Antwort wird ein weiteres Modul als „Absolviert“ markiert und andere Module freigeschaltet. Dies kann solange fortgeführt werden, bis alle Module mindestens einmal korrekt gelöst wurden. Ab diesem Zeitpunkt hat der Lernende nur noch die Möglichkeit bereits absolvierte Module aufzufrischen und die selben Aufgaben nochmal zu lösen.

# 7 Zusammenfassung und Ausblick

E-Learning-Systeme entwickeln sich ständig weiter und werden immer umfangreicher. Verantwortlich dafür sind die Programmiersprachen, welche ständig aktualisiert und mit neuen Funktionen und Möglichkeiten gespickt werden. Die einfache Integration von *JavaScript* auf *HTML*- und *PHP*-Webseiten verhindert zusätzlich ein Aussterben der Sprache. Es existieren bereits jetzt so viele Anwendungsmöglichkeiten, sowohl für neue als auch für bereits bestehende Systeme und Plattformen. Auch für dieses Projekt gibt es noch reichlich Ideen die umgesetzt werden können. Im Zuge dieser Bachelorarbeit entstand ein solides Gerüst, mit reichlich Verbesserungsmöglichkeiten.

Ein erster, und sehr essentieller Schritt ist das Integrieren eines Nutzerkontensystems um Lernfortschritte und Statistiken individuell speichern und auswerte zu können und um endgerätunabhängig zu werden. Das Cookiesystem bietet keinerlei Vorteile außer der einfachen Implementierung und ist momentan eines der größten Schwachpunkte des Systems. Ein Nutzerkonto wäre außerdem ein großer Schritt in Richtung Markt- und Alltagsfähigkeit dieses Projektes.
Außerdem könnte man pro Lernmodul mehrere Aufgaben hinterlegen, welche dann per Zufall ausgewählt werden, damit selbst bei mehrfacher Wiederholung des selben Lernmoduls ein bestimmter Übungsfaktor eintritt.
Eine weitere Möglichkeit zur Verbesserung besteht im Editorteil. Einfache, steife Eingabemasken erfüllen den Zweck und sind auch praktikabel, aber bieten nicht die Individualität und Bequemlichkeit, welche mit heutigen Software-Standards erreichbar wären. So könnte man beispielsweise eine Drag& Drop[11]-Oberfläche erstellen, in der man Mittels grafischer Blöcke ein Lernsystem erstellt und Abhängigkeit unter den Lernblöcken einfach per grafischer Anordnung erschafft. Diese Methode ist nicht nur wesentlich schneller und bequemer als jeden Lernblock einzeln zu erstellen und die Abhängigkeiten manuell festzulegen, es eliminiert auch potentielle Fehlerquellen wie Endlosschleifen im Vorgängersystem oder nicht erreichbare Lernblöcke. Hinzu kommt, dass diese Variante für einen Menschen wesentlich intuitiver ist, da hier sofort ein Überblick über den gesamten Inhalt entsteht. Somit ist es leichter greif- und planbar.

[11] [DragDrop] - „[...] deutsch „Ziehen und Ablegen“, [...] ist eine Methode zur Bedienung grafischer Benutzeroberflächen von Rechnern durch das Bewegen grafischer Elemente mittels eines Zeigegerätes.“

Mit Sicherheit gibt es noch viele weitere Verbesserungsmöglichkeiten. Insbesondere wenn man ein ganzes Nutzersystem um die Anwendung baut. Die Anzahl an Funktionen, welche solch ein System bieten kann ist endlos. Jedoch bietet der Prototyp, welcher als Endergebnis dieser Arbeit vorliegt eine solide Grundlage und einen genügend großen Funktionsumfang um bereits als fundamentales System eingesetzt werden zu können. Aufgrund des rohen *JavaScript*‘s und *HTML*‘s ist auch die Integrität und Umstellung auf andere Systeme kein Problem, da man sich nicht um Kompatibilität sorgen muss. Der Verzicht auf etwaige Add-Ons oder extra Bibliotheken während der Entwicklung sorgt außerdem auch für eine leichte Modifizierbarkeit von bereits bestehendem Quellcode.

Abschließend ist zu sagen, dass das Projekt eine große Bereicherung für mich war und meinen Horizont und meine Fähigkeiten deutlich vergrößert hat.

# Literatur


[ILIAS] „ILIAS ist eine freie Software zum Betreiben einer Lernplattform, [...].“
https://de.wikipedia.org/wiki/ILIAS_(Software),
Stand 06.03.2017

[Tag] „Ein Tag [tæg] [...] ist eine Auszeichnung eines Datenbestandes mit zusätzlichen Informationen.“
https://de.wikipedia.org/wiki/Tag_(Informatik),
Stand: 23.03.2017

[BSI] Pressemitteilung des Bundesamtes für Sicherheit in der Informationstechnik, BSI aus dem Jahre 1999,
offizielle Quellen von der BSI-Webseite sind offline, inoffizielle Quellen von privaten Archiven sind u.a: http://inetbib.de/listenarchiv/msg14936.html,
Stand 06.03.2017

[BamS] BILD am SONNTAG, 21. Mai 2000, Zitat nach Christian Wenz: Handbuch JavaScript, Galileo Computing 2003
http://openbook.rheinwerk-verlag.de/javascript/javascript_01.htm

[PHP Doku] http://php.net/manual/de/mysqli.real-escape-string.php

[Dropdown] „[...], ist ein Steuerelement [...] von grafischen Benutzeroberfläche. Dabei wird über einen Mausklick auf eine Schaltfläche einer Menüleiste oder Symbolleiste [...] ein Untermenü angezeigt.“
https://de.wikipedia.org/wiki/Dropout-Menü ,
Stand 23.11.2016

[DragDrop] „[...] deutsch „Ziehen und Ablegen“, [...] ist eine Methode zur Bedienung grafischer Benutzeroberflächen von Rechnern durch das Bewegen grafischer Elemente mittels eines Zeigegerätes.“
https://de.wikipedia.org/wiki/Drag_and_Drop ,
Stand 01.12.2016

[draw.io] Online-Tool zum erstellen von Graphen und Diagrammen wie Use-Case Diagrammen und Verlaufsgraphen
https://www.draw.io/,
Stand 26.04.2017

[Dia] Schlichtes Programm zum einfachen und freien erstellen von Graphen und Übersichten aller Art.
http://dia-installer.de/index.html.de,
Stand 26.04.2017

[MSQLW] Programm zum Planen und Bearbeiten von Datenbankstrukturen.
https://www.mysql.de/products/workbench/,
Stand 26.04.2017

[MySQL1] George Reese, Randy Jay Yarger, Tim King - My SQL: Einsatz und Programmierung, O'Reilly Verlag
https://books.google.de/books?id=9ij6KNXVB3kC&dq=MySQL+Geschicht&hl=de&source=gbs_navlinks_s,
Stand 26.04.2017

## Weitere Quellen

**cookie-script.com** Generator für Cookiehinweis-Scripts
cookie-script.com

### HTML

- http://www.yourhtmlsource.com/starthere/whatishtml.html
- https://www.w3.org/People/Raggett/book4/ch02.html
- http://www.martinrinehart.com/frontend-engineering/engineers/html/html-tag-history.html
- https://en.wikipedia.org/wiki/HTML
- https://de.wikipedia.org/wiki/Hypertext_Markup_Language

### JavaScript

Stand 18.08.2016

- http://openbook.rheinwerk-verlag.de/javascript_ajax/
- https://wiki.selfhtml.org/wiki/JavaScript/Entstehung_und_Standardisierung
- https://de.wikipedia.org/wiki/JavaScript
- https://de.wikipedia.org/wiki/Brendan_Eich
- http://www.itwissen.info/definition/lexikon/Scriptsprache-script-language.html
- http://www.itwissen.info/definition/lexikon/Interpreter-interpreter.html

### E-Learning

Stand 19.08.2016

- https://www.e-teaching.org/technik/distribution/lernmanagementsysteme

### MySQL

Stand 22.08.2016

- https://de.wikipedia.org/wiki/MySQL
- https://de.wikipedia.org/wiki/MySQL_AB

### PHP

Stand 22.08.2016

- https://de.wikipedia.org/wiki/PHP

### XAMPP Entwicklertool

Stand 26.08.2016

- https://www.apachefriends.org/de/about.html

### Apache Webserver

Stand 26.08.2016

- https://de.wikipedia.org/wiki/Apache_HTTP_Server

### Datenbanken

Stand 27.10.2016

- https://de.wikipedia.org/wiki/Datenbank

### CSS

Stand 22.08.2016

- http://de.html.net/tutorials/css/lesson1.php

# 8 Glossar

## Apache Webserver

Ein Webserver ist ein Server (lat. servire - „dienen“, engl. server „Diener“) welcher dazu dient Dokumente (*HTML*-Dateien, Bilder, Nutzerprofile einer Website, etc.) an Clients zu übertragen. Die sogenannten Clients sind in den aller meisten Fällen Webbrowser wie *Mozilla Firefox* oder *Google Chrome*. Hierbei wird als Webserver im eigentlichen Sinne entweder direkt die ausführende Hardware oder nur die dienliche Software bezeichnet, welche als Host fungiert. Die Hauptanwendungsbereiche von Webservern sind auf lokaler Ebene in Büros von mittelgroßen und großen Uternehmen, vereinzelt auch auf Heimrechnern und kleineren Büros so wie im *World Wide Web*.

## ECMA

Die *ECMA International* ist eine internationale Normungsorganisationen für Informations- und Kommunikationssystem sowie Unterhaltungselektronik und wurde bereits 1961 gegründet. *ECMA* steht hierbei für **E**uropean **C**omputer **M**anufacturers **A**ssociation. Die Ziele der *ECMA* sind unter anderem die Entwicklung von Standards, Förderung der Einhaltung der Standards und Veröffentlichung von frei zugänglichen Standards und technischen Dokumentationen.

## Netscape Communcations

*Netscape* war ein US-amerikanisches Softwareunternehmen und der Pionier der Browserentwicklung. Der erste Browser aus dem Hause *Netscape* war der *Netscape Navigator*, welcher als harte Konkurrenz zu *Microsofts Internet Explorer* in die Geschichte einging. Auch war der *Navigator* 2.0 der weltweit erste Browser, der eine Scriptsprache integriert hatte. Die Entwicklung des Unternehmens hatte schnellwechselnde Höhen und Tiefen und so kam es, dass 1998 *Netscape Communications* für $4,2$ Milliarden US-Dollar vom Unternehmen *AOL* aufgekauft und am 15. Juli 2003 schließlich komplett aufgelöst wurde. Ein verbliebenes Enkelkind des Unternehmens ist das Opensource-Projekt *Mozilla Firefox*.

## Microsoft Corporation

Die Ursprünge von *Microsoft* reichen zurück ins Jahr 1975. Der damalige Student Bill Gates entwickelte zusammen mit Allan Paul und Monte Davidoff eine Programmiersprache namens *„Altair Basic“* 2.0 für den *ALTAIR 8800* Computer. Bill Gates und Paul Allan schlossen später einen Vetrag mit dem Hersteller des *ALTAIR 8800*, *MITS* (**M**icro **I**nstrumentation **T**elemetry **S**ystems), ab was unter anderem dazu führte, dass Gates sein Studium abbrechen musste. Monte Davidoff bekam eine einmalige Auszahlung von 2400$. Als Gates und Allan sich im selben Jahr um eine Werbekampagne kümmern mussten, wurde der Name *Microsoft* geboren.

Im laufe der Zeit entwickelte sich *Microsoft* weiter und arbeitete an neuen Projekten, wie zum Beispiel *Altair Basic* 3.0 oder *MS-DOS* für das Unternehmen *IBM*. Dabei unterschätzte Microsoft zunächst das neue Internet, weßhalb das Unternehmen unter Zeitdruck den *Internet Explorer* (IE) entwickelte. Mit *Windows 95* wurde dann *MSN* (**M**icro**s**oft **N**etwork) als direkte Antwort zu *AOL* und *CompuServe* veröffentlicht. Um den verpassten Start des *Internet Explorer* auszugleichen versuchte *Microsoft* seinen Browser an das Betriebssystem *Windows* zu binden und entzog dem Unternehmen *Compaq*, welches ihre PC's mit der Software des Rivalen *Netscape Communications* austattete, die Vertriebslizenz von *Windows 95*. Tatsächlich gelang es *Microsoft* seinen Rückstand aufzuholen, wenn auch nicht mit fairen Mitteln.

## Scriptsprachen

Scriptsprachen sind Programmiersprachen welche eigentlich nur für kleinere Anwendungen und Programme gedacht sind. Zur Laufzeit (Zeitraum, in dem der Script logisch aktiv ist) werden sie von einem Interpreter (im Prinzip ein Übersetzer von menschlich geschriebenem Quellcode zu maschinenkompatiblen Befehlen) ausgeführt. Aus der Natur der Scriptsprachen entstammt der oft sehr spezifische Einsatz für genau definierte Aufgaben und der daraus resultierenden überschaubaren Syntax.