



Hochschule für Technik,
Wirtschaft und Kultur Leipzig

**Entwicklung und Konzeption eines E-Learning
C++ Kurses auf Basis einer C-Turtle-Umgebung
für das Modul „Grundlagen der Programmierung“
an der HTWK-Leipzig**

**ABSCHLUSSARBEIT ZUR ERLANGUNG DES AKADEMISCHEN GRADES
BACHELOR OF SCIENCE**

IM STUDIENGANG INFORMATIK BACHELOR DER FAKULTÄT INFORMATIK UND MEDIEN

Mirella Maria Willems

Erstprüfer: Prof. Dr. rer. nat. Mario Hlawitschka

Zweitprüfer: M.S Felix Stolze

Leipzig, den 28. September 2023

Kurzfassung

E-Learning spielt in der heutigen Hochschulbildung eine entscheidende Rolle und wird auch in den kommenden Jahren weiterhin von Bedeutung sein. Daher wurde im Rahmen dieser Bachelorarbeit eine E-Learning Plattform mit verschiedenen C++ Kursen für das Modul “Grundlagen der Programmierung“ an der HTWK Leipzig entwickelt. Diese Arbeit beschäftigt sich mit der Entwicklung der Plattform von Anfang bis Ende. Im Fokus lag die Entwicklung und Konzipierung verschiedener C++ Kurse, die auf einer C-Turtle Umgebung basieren. C-Turtle ist eine grafische Darstellung einer Schildkröte und eignet sich sehr für den Einstieg in die Programmierung ([1]). Dabei wurde stets darauf geachtet, dass dieses Projekt einen Beitrag zur Weiterentwicklung des E-Learning Angebots der HTWK Leipzig leistet.

Abstract

E-learning plays a crucial role in today's higher education and will continue to be important in years to come. Therefore, within the scope of this bachelor thesis, an e-learning platform with various C++ courses was developed for the module “Fundamentals of Programming“ at the HTWK Leipzig. This thesis deals with the development of the platform from start to finish. The focus was on the development and design of different C++ courses based on a C-Turtle environment. C-Turtle is a graphical representation of a turtle and is very suitable for the introduction to programming ([1]). It has always been a priority that this project contributes to the further development of the e-learning offer of the HTWK Leipzig.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und nicht anderweitig zu Prüfungszwecken vorgelegt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Ort, Datum: Leipzig, 28.09.2023

Unterschrift: Mirella Willmann

Danksagung

Zu Beginn möchte ich meinen Eltern Valeria und Walter danken, da sie mich stets unterstützt und an mich geglaubt haben.

Ich möchte meiner Schwester Elisabeth dafür danken, die immer an meiner Seite war und mich emotional unterstützt hat.

Ein großes Dankeschön geht an meinen Betreuer Herrn Stolze, der mich während der Arbeit tatkräftig unterstützt und motiviert hat.

Ebenso möchte ich mich bei meinem Erstbetreuer Prof. Hlawitschka bedanken, der mir die Möglichkeit gegeben hat, meine Arbeit in Zusammenarbeit mit ihm und für sein Modul GDP zu verfassen und für sein Vertrauen.

Ich möchte mich auch bei Prof. Hering für seine Unterstützung und für das E-Learning Modul bedanken, das mir so gut gefallen hat, dass ich mich entschieden habe, meine Bachelorarbeit in diesem Bereich zu schreiben. Ich wünsche ihm Alles Gute.

Natürlich möchte ich mich auch bei meiner Lerngruppe bedanken, ohne euch wäre das Studium nur halb so schön gewesen!

Ich bedanke mich bei meinen Haustieren Amadeus, Geppetto und den sieben Katzen, die mich ständig beim Schreiben der Bachelorarbeit abgelenkt haben.

Zuletzt möchte ich mich bei all jenen herzlich bedanken, die mich bis hierhin begleitet und mich während meiner Bachelorarbeit auf unterschiedliche Weise unterstützt haben. Ihr wisst, wer ihr seid, und ich bin froh euch in meinem Leben zu haben!

Inhaltsverzeichnis

Inhaltsverzeichnis	III
Abkürzungsverzeichnis	V
1. Einleitung	1
1.1. Ziele der Arbeit	2
1.2. Methodisches Vorgehen	2
2. Grundlagen	5
2.1. Der Begriff E-Learning	5
2.2. E-Learning Anwendungsszenarien	5
2.2.1. Lernprogramme CBT und WBT	6
2.2.2. Lernmanagementsystem/Lernplattform	6
2.2.3. Blended Learning	7
2.3. Die verschiedenen Lerntheorien	8
2.3.1. Die Lerntheorie Behaviorismus	8
2.3.2. Die Lerntheorie Kognitivismus	9
2.3.3. Die Lerntheorie Konstruktivismus	9
2.3.4. Die Lerntheorie Konnektivismus	9
2.4. Gamification	10
2.5. Didaktik an der Hochschule	10
2.5.1. Diversität und Inklusion	10
3. Die Methodik	12
3.1. Themenfindung und Motivation	12
3.1.1. Das Modul Grundlagen der Programmierung	12
3.1.2. C-Turtle und die Turtlegrafik	13
3.2. Der Leitfaden	14
3.2.1. Rahmen	14
3.2.2. Akteure definieren	14
3.2.3. Anforderungen, Lernziele und grundsätzliche Ideen	15
3.2.3.1. C++ Einführungskurs	15
3.2.3.2. Willkommensseite	16
3.2.3.3. Erstellung eines Pseudocodes	17
3.2.3.4. Anforderung der Gamification	18
3.2.3.5. Erstellung eines Algorithmus	19

Inhaltsverzeichnis

3.2.3.6. Zukünftige Anforderung	19
3.2.4. Didaktische Methode	19
3.2.5. Lernorganisation	20
3.2.6. Vorgehen	20
3.2.7. Ähnliche Einführungskurse	20
4. E-Learning Aspekte und multimediale Inhalte	21
4.1. Interfacedesign und Screendesign	21
4.1.1. Layout	21
4.1.2. Header, Logo und Footer	23
4.1.3. Die Navigation	23
4.1.4. Farbpalette, Grafiken und Schriftart	24
4.2. Verwendete Gamification	25
4.2.1. Narration und Storytelling	25
4.2.2. Der Pebble-Simulator	26
4.2.3. Fortschrittsanzeige	27
4.2.4. Kursabschluss und Urkunden	28
4.2.5. Interaktive Auswahl und Vorgehen	29
5. Die verschiedenen Kursinhalte	31
5.1. Der C++ Einführungs Kurs	31
5.2. Der Algorithmus Kurs	32
5.2.1. Algorithmus Teil 1	33
5.2.2. Algorithmus Teil 2	33
5.2.3. Algorithmus Teil 3	34
6. Implementation	35
6.1. Technische Informationen zum Backend und Frontend	35
6.2. Frontend	36
6.2.1. Willkommensseite	36
6.2.2. C++ Einführungskurs	38
6.2.3. Algorithmus Teil 1	40
6.2.4. Algorithmus Teil 2	41
6.2.5. Algorithmus Teil 3	42
6.3. Einblick in die Programmierung und Komponenten	43
6.3.1. Code-Editor-Komponente	43
6.3.1.1. TurtleViewer-Komponente	45
6.3.2. Akkordeons	46

Inhaltsverzeichnis

6.4. Backend und Server	47
6.4.1. API-Server	47
6.4.1.1. Endpoint Task	48
6.4.1.2. Datenbank	48
6.4.2. Ausführungs-Server	50
6.5. Ausführung und Benutzung	53
6.5.1. HTWK-Server aufsetzen	53
7. Ergebnisse	54
8. Fazit und Ausblick	55
Abbildungsverzeichnis	57
Verzeichnis der Beispiele	57
A. Anhang	58
A.1. Modulbeschreibung des Moduls GdP	58
A.2. Inhaltliche Organisation GdP	60
A.3. Bash Skript - Server	62
Literaturverzeichnis	VIII

Abkürzungsverzeichnis

HTWK	Hochschule für Technik, Wirtschaft und Kultur
GdP	Grundlagen der Programmierung
CBT	Computer Based Training
WBT	Web Based Training
LMS	Learning Management System
INB	Informatik Bachelor
MIB	Medieninformatik Bachelor
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol

1. Einleitung

Es ist nichts Neues, dass wir uns im Zeitalter der Digitalisierung befinden. Das Leben ohne Smartphone, Laptop, Computer, Tablet oder Smartwatch kann sich kaum noch jemand vorstellen. Diese elektronischen Geräte sind mittlerweile ein integraler Bestandteil vieler Menschen, denn sie dienen unter anderem als Portal in das Internet. Dort findet man neben Entertainment und sozialen Netzwerken auch vor allem eins: Wissen. Dieses Wissen wird im Internet nicht nur gespeichert, sondern ständig aufbereitet, überarbeitet und auf unterschiedlichste Art und Weise zugänglich gemacht.

Die Wissensvermittlung mithilfe digitaler Medien wird schon in sämtlichen Bereichen genutzt, sei es im rein privaten Rahmen, in konventionellen Bildungseinrichtungen oder großen Unternehmen. Der individuelle Lernprozess wird dabei durch passende multimediale und internetbasierte Materialien gefördert, die speziell dafür entwickelt wurden, im E-Teaching und E-Learning angewendet zu werden ([2]).

Hochschulen stellen ebenfalls solche vielfältigen multimedialen und internetbasierten Möglichkeiten zur Verfügung. Damit Studierende diese Angebote nutzen können, wird ein Personal Computer¹ (PC) benötigt. Im Jahr 2022 besaßen etwa 92 Prozent der deutschen Haushalte einen eigenen PC ([4]). Sollte es einem Studierenden dennoch finanziell nicht möglich sein, sich einen PC oder Laptop anzuschaffen, besteht die Option, über die eingeschriebene Hochschule oder Universität, ähnlich wie es an der HTWK-Leipzig möglich ist ([5]), elektronische Geräte auszuleihen. Abgesehen davon verfügen die meisten Universitäten und Hochschulen über Computer-Arbeitsräume, die es den studierenden Personen ermöglichen, direkt von der Bildungseinrichtung aus zu arbeiten ([6]).

Obwohl das Thema E-Learning allgegenwärtig von großer Bedeutung ist, hinkt Deutschland im Vergleich zu zahlreichen anderen OECD²-Ländern sowohl beim Aufbau digitaler Infrastrukturen als auch bei der Anwendung digitaler Technologien und Dienstleistungen hinterher ([7]). Die Corona-Krise hatte in den letzten Jahren einen ungeplanten, aber signifikanten Schub in Richtung Digitalisierung ausgelöst ([8]). Es bedarf allerdings weiterer Maßnahmen, um die deutschen Hochschulen auf internationale digitale Standards auszurichten. Diese Arbeit setzt genau dort an.

¹Aus dem Englischen “personal computer”: persönlicher Computer ([3])

²OECD: Organisation für wirtschaftliche Zusammenarbeit und Entwicklung mit 38 Mitgliedsstaaten ([7])

1.1. Ziele der Arbeit

Das Ziel dieser Arbeit besteht darin, einen Beitrag zur Weiterentwicklung des E-Learning-Angebots der HTWK Leipzig zu leisten, indem ein E-Learning Kurs für das Modul Grundlagen der Programmierung (siehe 3.1.1) entwickelt wird. Hierbei war ein wichtiges Ziel, dass der Kurs auch nach Abschluss dieser Arbeit in dem GdP-Modul verwendet wird.

Entwickelt wurde ein Einführungskurs für die Programmiersprache C++, unter der Verwendung von C-Turtle (siehe 3.1.2), einer Variation der Python-Turtle ([1]). Ein zentraler Schritt auf dem Weg zu diesem Ziel war die Konzipierung und Umsetzung der E-Learning Plattform, wobei besonderes Augenmerk auf die didaktischen Aspekte der Lehre gelegt wurde. Angehenden Studierenden soll damit ein unkomplizierter Einstieg in die Welt des Programmierens ermöglicht werden. Die Verwendung von C-Turtle schien daher äußerst geeignet, da C-Turtle den Ruf hat, einen einfachen Einstieg in dieses durchaus komplexe Fachgebiet zu bieten ([9]). Das Hauptanliegen liegt somit darin, eine angenehme Herangehensweise bei der Bewältigung von Programmieraufgaben und der Konzeption von Algorithmen zu schaffen. Dieser Ansatz soll den Studierenden nicht nur Spaß am Programmieren bringen, sondern auch ihr Selbstvertrauen stärken. Passend dazu lautet das wiederholende Motto des Kurses: “Du kannst Programmieren“. Motivierende Worte können Wunder bewirken und die Studierenden dazu bringen, mehr Interesse zu zeigen und eine positive Veränderung in Bezug auf das Lernen zu entfachen ([10]). Durch das Wachsen des eigenen Selbstvertrauens ergibt sich außerdem die Möglichkeit, die Erwartung an die eigene Selbstwirksamkeit zu erfüllen. Hierbei handelt es sich um den Glauben daran, dass man zukünftig in der Lage sein wird, neue oder herausfordernde Probleme erfolgreich zu bewältigen ([11]).

1.2. Methodisches Vorgehen

Im Abschnitt 3 erfolgt eine vertiefte Auseinandersetzung mit der Methodik. Dieses Unterkapitel soll nur eine grundlegende, knappe Erklärung des methodischen Ansatzes geben.

Direkt zu Beginn wurden die ersten Pläne zur Entwicklung der einzelnen Kurselemente in enger Zusammenarbeit mit dem GDP-Professor, Professor Hlawitschka, und dem Bachelor Prüfer, Felix Stolze, erarbeitet. Da es sich hierbei nicht nur um eine reine Literaturrecherche, sondern auch um die praktische Umsetzung einer E-Learning-Website handelt, war es notwendig alle Details im Voraus abzustimmen, bevor mit der Bachelorarbeit angefangen werden konnte. Nach sorgfältiger Absprache konnte es also in die vertiefte Ideenfindung und Entwicklung des ersten Prototyps gehen. Hierbei war es von Anfang

1. Einleitung

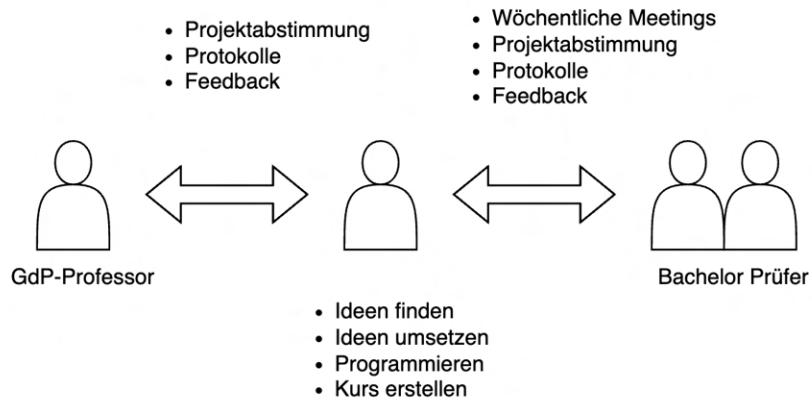


Abb. 1.1.: Projektabstimmung und Austausch (eigene Darstellung)

an wichtig, dass alle Projektbeteiligten stets über jede Entscheidung informiert waren und dass ein kontinuierlicher Austausch untereinander stattfand. Es wurde beschlossen, alle zwei Wochen Besprechungen zu halten, um sich über den Fortschritt auszutauschen. Diese Intervalle können als zweiwöchentliche Sprints im Kontext der Softwareentwicklung betrachtet werden. Dies gewährleistete ein reibungsloses Fortschreiten des Projekts und eine gemeinsame Vision für das Endziel. Das Vorgehen und die Kommunikation mit allen Projektbeteiligten wurde in Abbildung 1.1 dargestellt.

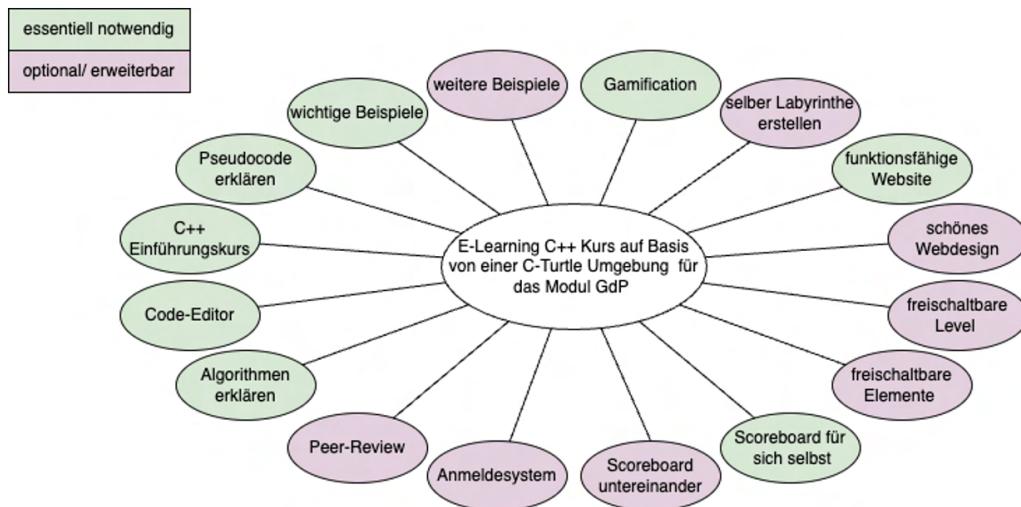


Abb. 1.2.: Essentielle und Optionale Anforderungen (eigene Darstellung)

1. Einleitung

Nach Entwicklung der Ziele und der Erstellung der ersten Konzeptideen (siehe 3.2.3), begann die Entwicklungsphase. Es wurde ein Plan erarbeitet, der vorsah, zunächst die Website mit den essentiellen Funktionen zu erstellen, die zu einem späteren Zeitpunkt weiter ausgebaut werden können. Dieser Plan ist in Abbildung 1.2 dargestellt und zeigt sowohl die essentiellen als auch die optionalen Funktionen.

2. Grundlagen

In diesem Kapitel werden die wesentlichen Begriffe und Grundlagen beleuchtet, die oft im Zusammenhang mit E-Learning auftreten.

2.1. Der Begriff E-Learning

Der Begriff E-Learning erweist sich bei genauer Betrachtung komplexer als zunächst angenommen, da es zahlreiche verschiedene Interpretationen darüber gibt, was dieser Begriff eigentlich genau umfasst und somit viele verschiedene Definitionen und Synonyme existieren ([12]). Angesichts der rasanten Veränderungen in den technischen und didaktischen Rahmenbedingungen werden Definitionen schnell überholt, manchmal auch noch bevor sie vollständig formuliert werden können. E-Learning ist so vielfältig, dass eine einheitliche Definition nicht möglich ist ([13]).

Ursprünglich kommt der Begriff aus dem Englischen “Electronical-Learning“ und heißt übersetzt “elektronisches Lernen“. Es werden aber auch Begriffe wie “technologiegestütztes Lernen“ und “digitales Lernen“ mit E-Learning in Verbindung gesetzt ([14]).

Im Allgemeinen ist zu sagen, dass E-Learning eine Vielzahl technologiegestützter Ansätze umfasst, die dazu dienen können, das Lernen durch die Verwendung elektronischer Geräte zu unterstützen und zu erleichtern ([15]). E-Learning ist der übergeordnete Begriff für sämtliche Formen der Anwendung digitaler Medien zu Bildungs- und Lernzwecken, sei es auf digitalen Speichermedien oder über das Internet ([16]).

2.2. E-Learning Anwendungsszenarien

Da der Fokus dieser Arbeit nicht auf der detaillierten Analyse der unterschiedlichen Anwendungsszenarien liegt, sondern es um die Bereitstellung eines C++ E-Learning-Kurses handelt, wird im Folgenden lediglich ein kurzer Blick auf einige der verschiedenen E-Learning Anwendungsszenarien geworfen, da diese für spätere Erläuterungen (siehe 3.2.4) von Relevanz sind.

2.2.1. Lernprogramme CBT und WBT

Lernprogramme sind spezifische Computeranwendungen, die zur Vermittlung von Lerninhalten genutzt werden ([13]). Dabei werden diese Computeranwendungen in zwei Kategorien unterteilt. Es gibt einerseits das Computer Based Training (CBT) und andererseits das Web Based Training (WBT)([13]).

CBTs werden in der heutigen Zeit als überholt wahrgenommen, da für die Benutzung insbesondere Speichermedien wie Disketten, CD-ROMs oder DVDs zum Einsatz kommen ([17]). Es handelt sich also um eine computerbasierte Bereitstellung von Lehrmaterialien, die jederzeit und ohne Internet aufrufbar sind ([13]).

WBTs hingegen sind moderne Arten von Lernprogrammen, die eine Verbindung zum Internet benötigen. Inhalte werden über das Internet aufgerufen und können daher auch laufend abgeändert und erneuert werden ([17]). WBTS ermöglichen es Studierenden, flexibel auf den gewünschten Inhalt zugreifen zu können. Sie haben die Möglichkeit, den Lehrstoff in ihrem eigenen Tempo zu erarbeiten und die Inhalte nach Belieben zu wiederholen oder nachzuschlagen ([13]). Darüber hinaus erfolgt die Gestaltung von WBTS üblicherweise in multimedialer Form ([17]), unter Einsatz von Texten, Grafiken und interaktiven Aufgaben, um das Lernen attraktiver und effektiver zu gestalten.

Bei den für diese Bachelorarbeit erstellten Kursen handelt es sich um ein WBT. Die Kurse wurden mit Hilfe multimedialer Gestaltung erstellt (siehe 4) und stellen den Studierenden über das Internet verschiedene C++ Kurse bereit.

2.2.2. Lernmanagementsystem/Lernplattform

Lernmanagementsysteme (LMS) sind webbasierte Systeme oder Plattformen, die zur Bereitstellung, Verwaltung und Organisation von Lernmaterialien in Bildungseinrichtungen genutzt werden [18]. Sie können auch für die Verwaltung von Nutzerdaten und Lernprozessen verwendet werden [18]. Die Aufgabe des LMS liegt meistens darin Lernstrukturen abzubilden, sowie Kurse und Materialien bereitzustellen [18]. Eine idealtypische Architektur eines LMS kann der Abbildung 2.1 entnommen werden.

Die erstellte Turtle-Website ([19]) ist ebenfalls ein LMS, da sie eine umfassende Lernumgebung bietet, die verschiedene Kurse (siehe 5), Personalisierungen (siehe 4.2.2), Fortschrittsanzeigen (siehe 4.2.3) und vieles mehr bereitstellt. Die Plattform kann von Studierenden verwendet werden, um auf verschiedene Übungsaufgaben zuzugreifen. Dadurch wird ihnen die Gelegenheiten gegeben, ihr Wissen aktiv unter Beweis zu stellen.

2. Grundlagen

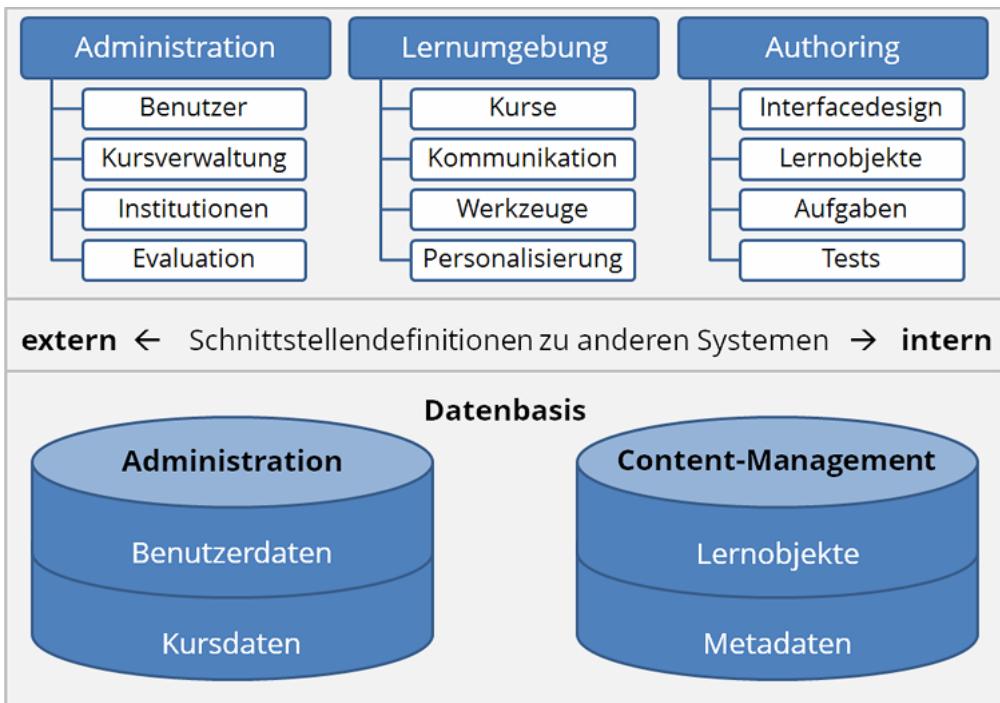


Abb. 2.1.: Idealtypische Architektur eines Learning Management Systems ([20])

2.2.3. Blended Learning

Blended Learning¹ beschreibt in der Praxis das kombinierte pädagogische Konzept, bei dem Präsenz- und Online-Unterricht abwechselnd integriert werden ([21]). Seminare und Übungen können, anders als der Unterricht, online stattfinden und dadurch eine ausgewogene Mischung von Online- und Präsenzlehre bieten ([21]). Forschungen legen nahe, dass die Integration digitaler Medien äußerst hilfreich ist, um spezifische Lernansätze zu fördern ([13]). Das Konzept des Blended Learnings wird auch bei der Turtle-Website verwendet. Die Website soll in den ersten Wochen des Moduls GdP verwendet werden (siehe 3.2.5) und den Studierenden neben der Präsenzlehre die Möglichkeit bieten, die bereitgestellten Übungsaufgaben online zu bearbeiten.

¹Übersetzt: gemischtes Lernen

2.3. Die verschiedenen Lerntheorien

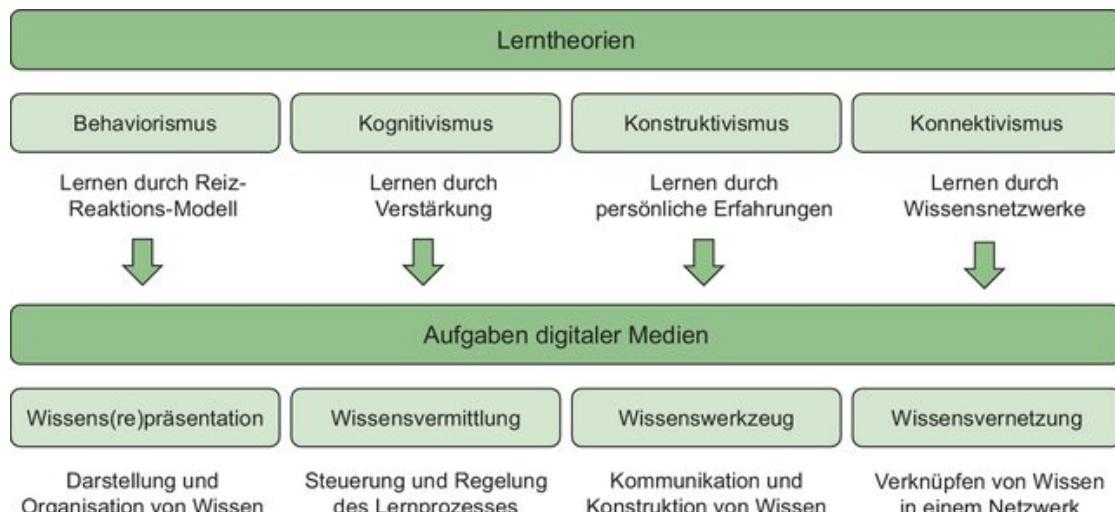


Abb. 2.2.: Lerntheorien und Aufgaben digitaler Medien ([22])

Lerntheorien erforschen Veränderungen im menschlichen Verhalten und Denken, welche nicht auf angeborene Reaktionen wie Reflexe und Instinkthandlungen zurückzuführen sind ([23]). Sie geben Antworten darauf, wie der Mensch lernt und was sich dabei bei der lernenden Person abspielt ([24]). Lerntheorien und ihre vielfältigen Erklärungsansätze helfen beim Verständnis menschlicher Lernprozesse und deren Abläufe beim Aufbau von neuen Fähigkeiten und Kompetenzen ([22]). Daraus ergeben sich Möglichkeiten, gezielte Lehrveranstaltungen und Lehrmaterialien mit einer optimierten didaktischen Ausrichtung zu konzipieren und zu gestalten ([23]). Die Abbildung 2.2 bietet einen Überblick über verschiedene Lerntheorien und den gesonderten Aufgaben von digitalen Medien beim Lernen ([22]). Die Grafik dient der Veranschaulichung der Lerntheorien, die in den einzelnen Unterkapiteln näher erleuchtet werden.

2.3.1. Die Lerntheorie Behaviorismus

Behaviorismus², auch Verhaltenstheorie genannt, beschäftigt sich mit der Verhaltensweise und den Reaktionen eines Individuums. Sie betrachtet das Lernen als Resultat von Anreizen, Belohnungen und Bestrafungen, wobei bestimmte Reize mit spezifischen Reaktionen verknüpft sind. Das äußere Verhalten steht hierbei im Mittelpunkt, während innere Gedanken und Prozesse, die das Lernen beeinflussen, vernachlässigt werden ([25]).

²Aus dem Englischen “behavior”: Verhalten

2. Grundlagen

Das Verhalten und Lernen wird beim Behaviorismus durch Reiz-Reaktions-Mustern erklärt ([26]). Positive Reaktionen können durch Belohnungen verstärkt werden, während unerwünschte Reaktionen durch Bestrafung reduziert werden. Belohnung und Bestrafung spielen eine zentrale Rolle für den Lernerfolg ([25]).

2.3.2. Die Lerntheorie Kognitivismus

Der Kognitivismus beschäftigt sich mit der Verarbeitung von Informationen und dem Aufbau von Wissen in Bezug auf den Menschen ([25]). Lernen wird als aktiver Prozess verstanden, bei dem Individuen Informationen aufnehmen, organisieren, analysieren und interpretieren, um daraus neues Wissen zu konstruieren. Anders als beim Behaviorismus rückt beim Kognitivismus die Verarbeitung von Informationen und die dadurch verbundene Erkenntnis in den Fokus. Wahrnehmung wird nicht passiv empfangen, sondern aktiv verarbeitet. Damit steht immer die Informationsaufnahme, deren Verarbeitung und die anschließende Informationsspeicherung im Vordergrund ([25]).

2.3.3. Die Lerntheorie Konstruktivismus

Beim Konstruktivismus liegt der Fokus auf der individuellen Wahrnehmung und Interpretation ([25]). Die lernende Person selbst erstellt aus eigener Wahrnehmung und Erlebnissen das interne Wissen, welches folglich nicht von einem Lehrenden vermittelt wird ([26]). Es ist ein anlaufender und intern gesteuerter Prozess ([23]), bei dem der Mensch durch eigenständiges Überlegen und Auswerten von Erfahrungen sein eigenes Wissen aufbaut. Die lernende Person setzt sich dabei selbstständig mit dem zu lernenden Inhalt auseinander ([27]). Die Auswahl des zu lernenden Inhaltes, hängt von der persönlichen Motivation und den individuellen Interessen eines jeden Einzelnen ab.

2.3.4. Die Lerntheorie Konnektivismus

Der Konnektivismus wird auch öfters als die Lerntheorie des digitalen Zeitalters bezeichnet ([28]). Diese Lerntheorie sagt aus, dass der Mensch ein vernetztes Individuum ist und daher soziale Interaktion einen wichtigen Faktor im Lernprozess darstellt. Lernen geschieht durch aktive Beteiligung und Zusammenarbeit mit einem Netzwerk, bei dem auch der Austausch von Gedanken und Ideen stattfindet ([29]).

2.4. Gamification

Die Gamification³, auch Gamifizierung genannt, beschreibt das Verwenden von spieltypischen Elementen und Vorgängen in einem spiel-fremden Kontext ([30]). Ziele der Gamification sind unter anderem die Motivationssteigerung und eine Verhaltensänderung der Personen, die mit dem Gamification-Element interagieren ([30]). Meistens handelt es sich dabei um Punktesysteme, Preise, Urkunden, Abzeichen, Levels oder Belohnungen ([31],[30]). In der Forschung, insbesondere im Bildungsbereich, existieren bereits zahlreiche Studien, die sich mit der Anwendung von Gamification befassen. Es gibt auch theoretische Ansätze, die sich den Auswirkungen von Gamification auf den Lernerfolg widmen ([32]). Diese legen nahe, dass Gamification-Elemente die Förderung lernrelevanter Aktivitäten unterstützen und dass diese Aktivitäten in einem positiven Zusammenhang mit dem Lernerfolg der Studierenden stehen ([32]). Weitere Forschungsergebnisse zeigen, dass sowohl der Einsatz von Fortschrittsanzeigen als auch von Storyelementen, selbst wenn sie einzeln verwendet werden, eine motivierende und leistungsfördernde Wirkung auf die Studierenden ausüben können ([32],[33]).

2.5. Didaktik an der Hochschule

Die Hochschuldidaktik vereint die traditionellen Prinzipien der Didaktik mit den spezifischen Anforderungen der Hochschullehre. Didaktik beschäftigt sich sowohl mit der Konzeption, Gestaltung und Umsetzung von Lehr- und Lernprozessen, als auch mit der Entwicklung von verschiedenen Lernmaterialien, die dem Lernenden dabei helfen sollen, sich Wissen anzueignen und diese neu erworbenen Kompetenzen zu fördern. In dieser Hinsicht kann die Didaktik sowohl als die Wissenschaft der Lehre, als auch die Kunst des Unterrichtens betrachtet werden ([34]). In Bezug auf die Hochschullehre können Lehrstrategien mithilfe der Hochschuldidaktik dynamisch und effektiv konzipiert werden. Im Fokus liegt die Konzeption und Entwicklung von kreativen und anpassungsfähigen Unterrichtsansätzen, die interessante Gestaltung von Lehrveranstaltungen oder die Auswahl geeigneter Lernmethoden ([35]).

2.5.1. Diversität und Inklusion

Die Begriffe Diversität und Inklusion spielen in der Hochschuldidaktik eine wesentliche Rolle. Inklusion kann als Konzept des Umgangs mit Diversität gesehen werden ([36]). Das Ziel ist es, unterschiedliche Individuen und Studierendengruppen einzubeziehen und

³Aus dem Englischen “game”: Spiel

2. Grundlagen

darauf zu achten, dass jede studierende Person, unabhängig von ihren Fähigkeiten, ethnischen Hintergründen, ihrer Religion, Behinderungen, Geschlecht oder ihrem Alter, die gleichen Bildungschancen erhält. Es ist hierbei wichtig eine inklusive Lehr- und Lernumgebung zu erstellen, die auf Bedürfnisse unterschiedlichster Studierender eingeht ([37]). Um diese Ziele zu erreichen, soll unter anderem das HEAD⁴ Wheel, welches in Abbildung 2.3 zu sehen ist, Lehrenden und Lernenden die Vielfalt von hochschulischer Diversität näherbringen.

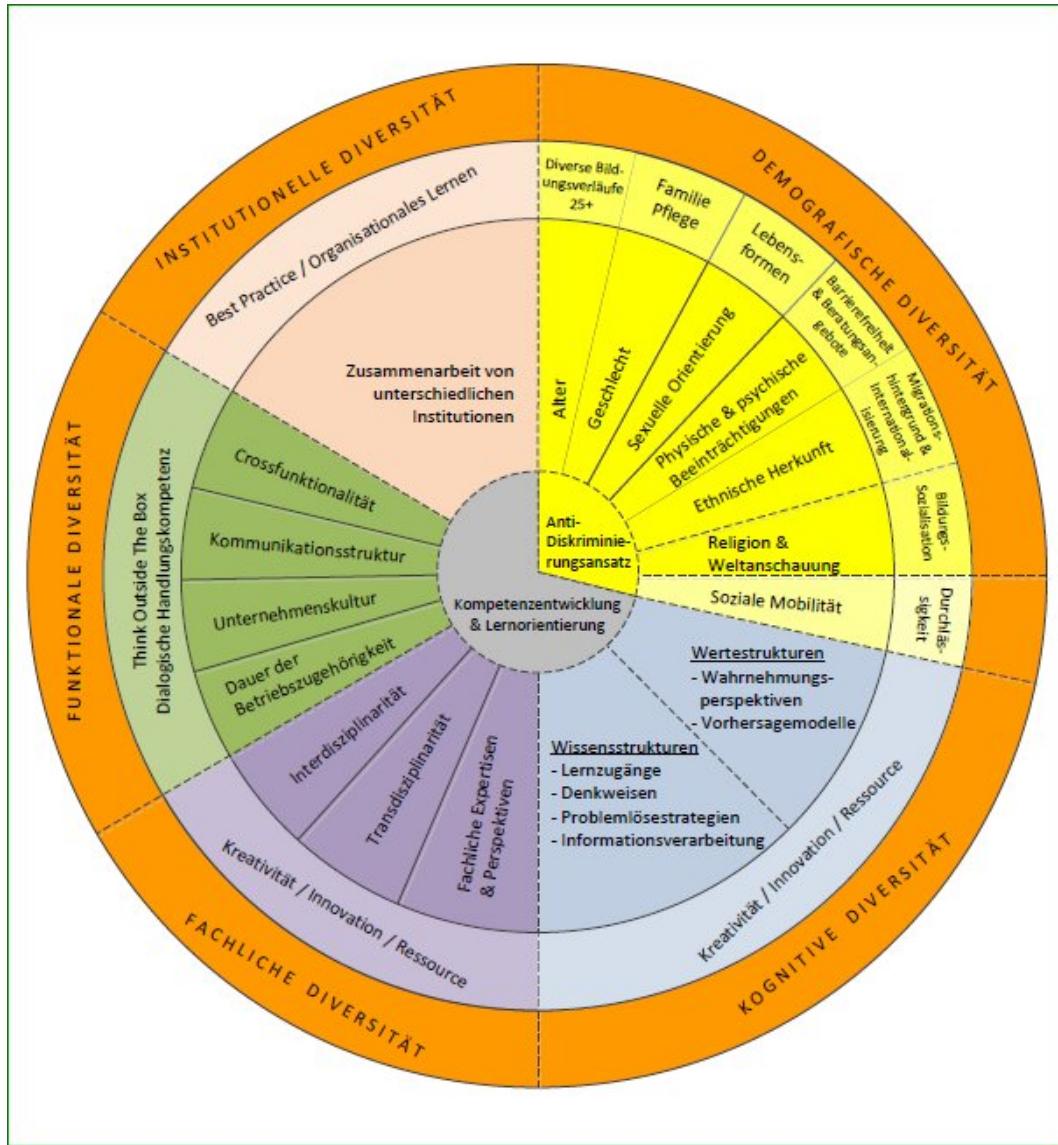


Abb. 2.3.: HEAD Wheel, Gaisch und Aichinger ([38])

⁴HEAD: Higher Education Awareness for Diversity

3. Die Methodik

In diesem Kapitel wird die Methodik betrachtet, welche den Verlauf der Arbeit dokumentiert. Hierbei wird genauer geschaut, wie der zu erstellende Einführungskurs aus einer Idee entstand und sich im Laufe der Zeit zu einer umfassenden interaktiven Lernplattform entwickelte.

Dabei ist festzuhalten, dass während der Bearbeitung wöchentliche Meetings stattfanden (siehe 1.2), die essentiell für den Prozess waren. Zusätzlich wurden sowohl die Website als auch diverse Kursinhalte kontinuierlich überarbeitet, um eine stetige Verbesserung zu gewährleisten.

3.1. Themenfindung und Motivation

Zu Beginn stand die Themenfindung an. Es gab verschiedene E-Learning Ansätze, jedoch wurde anfangs das Ziel gesetzt, dass die Arbeit einen bedeutenden Beitrag zur E-Learning-Entwicklung an der HTWK-Leipzig leisten soll. Durch eigene Erfahrungen aus dem Informatikstudium entstand die Intention, Erstis¹ zu unterstützen und den Übergang in das Hochschulleben und die Programmierung zu erleichtern. Aus diesen verschiedenen Ansätzen wurde das Thema “Entwicklung und Konzeption eines E-Learning C++ Kurses auf Basis einer C-Turtle Umgebung für das Modul “Grundlagen der Programmierung“ an der HTWK-Leipzig entwickelt. Dafür mussten verschiedene E-Learning Ansätze (siehe 2) verstanden werden, um den Kurs von Anfang an zu erstellen.

3.1.1. Das Modul Grundlagen der Programmierung

GdP ist ein Modul des ersten Semesters der HTWK-Leipzig in den beiden MINT²-Studienfächern Informatik und Medieninformatik. Es handelt sich dabei um ein Pflichtmodul, welches üblicherweise im Wintersemester stattfindet ([39]). Laut Modulplan (siehe A.1) sollen Studierende unter anderem erste Erfahrungen und Kompetenzen in der Programmiersprache C++ erhalten. Sie sollen beim Abschluss des Moduls in der Lage sein, einfache und kleine Algorithmen oder Programme zu schreiben. Die Studierenden sollen außerdem die Grundlagen der Objektorientiertheit verstehen.

¹Studierende im ersten Semester

²MINT: Abkürzung für Studiengänge Mathematik, Informatik, Naturwissenschaften und Technik

3.1.2. C-Turtle und die Turtlegrafik

Die Turtle³grafik wurde ursprünglich in der Programmiersprache Python entwickelt. Es handelt sich dabei um eine Bibliothek, die ihren Nutzern die Möglichkeit gibt, eine Schildkröte zu steuern. Diese zieht hinter sich Linien, wodurch verschiedene Bilder und Muster gezeichnet werden können ([40]). In der Abbildung 3.1 ist eine Turtlegrafik abgebildet.

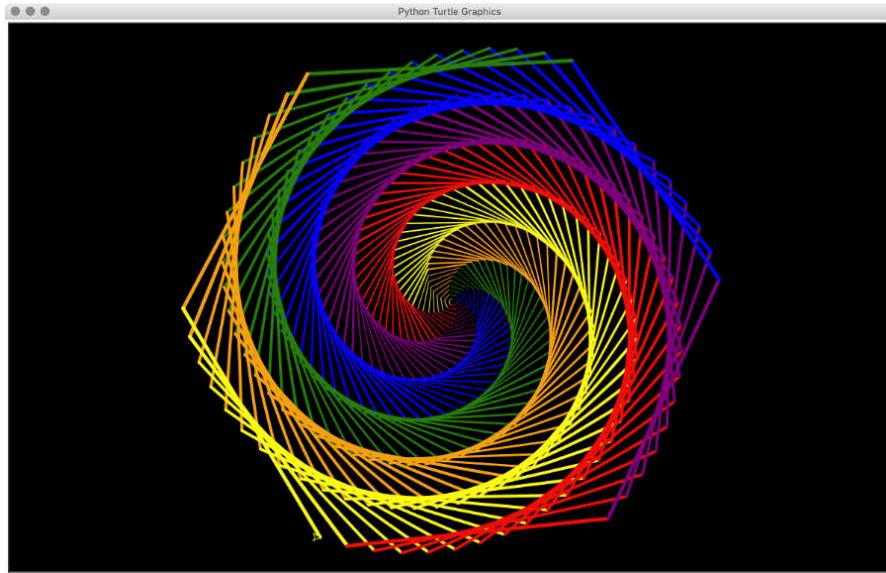


Abb. 3.1.: Mit Turtle erstellte Grafik ([41])

Es besteht die Möglichkeit der Farbänderung der Linien und der Steuerung der Schildkröte durch verschiedene Befehle und Anweisungen. Sie ist bekannt dafür, Kindern, Jugendlichen und Personen ohne Programmervorkenntnissen einen einfachen Start in die Programmierung zu ermöglichen ([9]).

Da in dem Modul GdP die Programmiersprache C++ unterrichtet wird (siehe A.1), wurde die Python Turtle in C++ Code umgewandelt ([1]) und dadurch die “C-Turtle” programmiert. Die für dieses Bachelorprojekt erstellte C-Turtle besitzt, anders als andere Implementationen, nur die folgenden Methoden:

1. Um ein Feld nach vorne bewegen
2. Nach rechts drehen (um 90 Grad)
3. Nach links drehen (um 90 Grad)

³Aus dem Englischen “turtle”: Schildkröte

3. Die Methodik

Außerdem benutzt die C-Turtle nur die Attribute:

1. Prüfe, ob bereits im Ziel
2. Prüfe, ob eine Wand vorne
3. Prüfe, ob eine Wand links
4. Prüfe, ob eine Wand rechts

In den ursprünglichen Umsetzungen von C-Turtle konnte beispielsweise der Drehwinkel der Schildkröte angegeben werden ([40]). Diese Funktion wurde jedoch nicht übernommen, denn das Ziel eines Abschnittes des Kurses ist es, einen Algorithmus zu entwickeln, der eine Schildkröte aus einem Labyrinth führt (siehe 3.2.3.5). Die Möglichkeit der Drehung in einem anderen Winkel wurde bewusst nicht integriert, da dies nicht dazu beiträgt, den Algorithmus einstiegsfreundlich zu halten.

3.2. Der Leitfaden

Für die erfolgreiche Entwicklung des Projekts wurde ein Leitfaden ([16]) erstellt, der als Richtlinie diente und den Entwicklungsprozess unterstützte. Dieser Leitfaden half bei der Erschaffung klarer Strukturen für ein effektives Projektziel.

3.2.1. Rahmen

Als erstes sollten die Rahmendaten des Projekts erfasst werden. Nach erfolgreicher Themenfindung (siehe 3.1) wurde die grobe Projektidee formuliert. Diese schreibt die Erstellung eines Kurses vor, der studierenden Personen die Möglichkeit bietet, eine Einführung in C++ zu erhalten. Anhand verschiedener Übungen sollen die Studierenden zudem das Schreiben von Algorithmen erlernen.

3.2.2. Akteure definieren

Die Akteure geben an, welche Personen im Kurs involviert sind und welche spezifischen Rollen diese besitzen ([16]). Der GdP-Professor ist in diesem Fall der Auftraggeber, der ein E-Learning Angebot für sein Modul haben möchte. Die Erstsemesterstudierenden der Studienfächer INB und MIB sind die Zielgruppe, die an dem Kurs teilnehmen sollen. Nicht nur die Anzahl der Studierenden variieren jedes Semester, sondern auch die Programmier-Vorkenntnisse der einzelnen Teilnehmenden ([16]).

3. Die Methodik

3.2.3. Anforderungen, Lernziele und grundsätzliche Ideen

In enger Abstimmung mit dem Professor des Moduls GdP, sowie den Bachelorbeauftragten wurden Anforderungen und Ideen für den praktischen Teil dieser Bachelorarbeit erstellt. In den folgenden Unterkapiteln werden die verschiedenen Mockups, Anforderungen und Ideen näher betrachtet. Im Kapitel 5 wird über die Entwicklung der einzelnen Kursinhalte und im Kapitel 6 detailliert über die Implementation und Umsetzung gesprochen.

3.2.3.1. C++ Einführungskurs

Ein wichtiger Bestandteil ist der C++ Einführungskurs (siehe 5.1). Dieser Kurs basiert auf dem Lehrmaterial des Moduls GdP und hat das Ziel, den Studierenden eine grundlegende Einführung in die Programmiersprache C++ zu vermitteln. Die Konzeptidee kann aus Abbildung 3.2 entnommen werden.

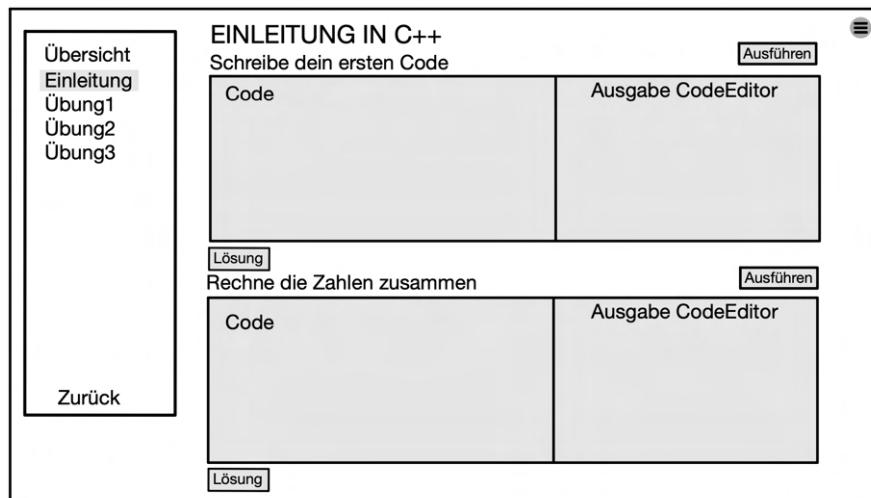


Abb. 3.2.: Konzeptidee für den C++ Einführungskurs (eigene Darstellung)

3. Die Methodik

3.2.3.2. Willkommensseite

Im Rahmen des Projekts sollte eine Willkommensseite erstellt werden, die den Studierenden sämtliche relevanten Details zu den Kursen, sowie verschiedene Anforderungen übersichtlich zur Verfügung stellt. Die ersten Konzeptideen für die Willkommensseite sind in Abbildung 3.3 und Abbildung 3.4 dargestellt.

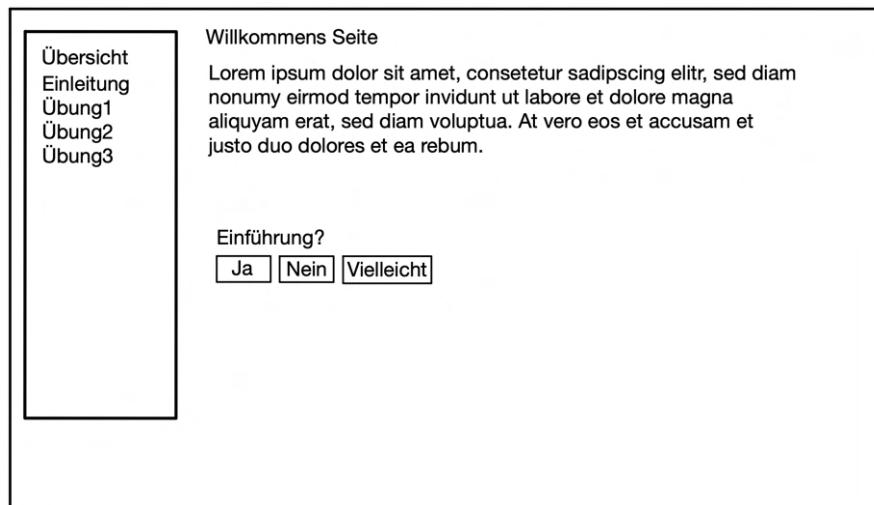


Abb. 3.3.: Konzeptidee 1 für die Willkommens-Seite (eigene Darstellung)

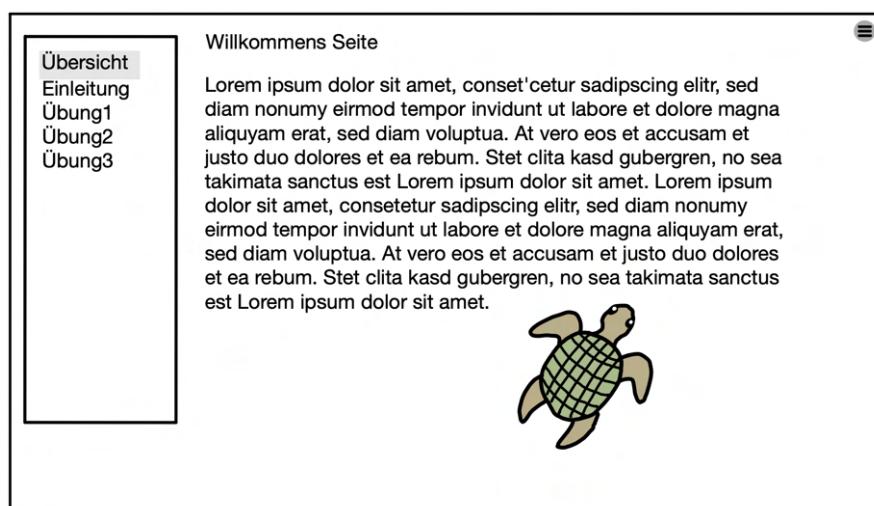
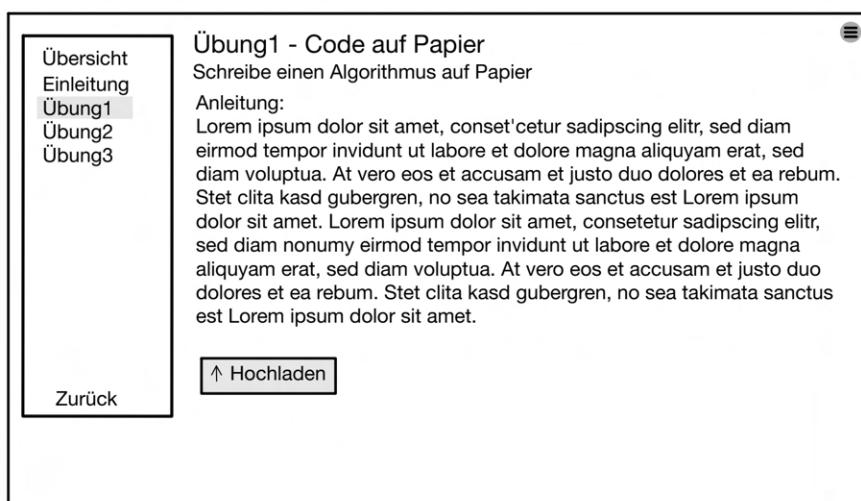


Abb. 3.4.: Konzeptidee 2 für die Willkommens-Seite (eigene Darstellung)

3. Die Methodik

3.2.3.3. Erstellung eines Pseudocodes

Ein weiterer Bestandteil umfasst die Erstellung eines Pseudocodes. Pseudocode stellt ein Hilfsmittel in der Informatik dar, bei der Algorithmen und Codeabschnitte in verständliche Sprache geschrieben werden ([42]). Diese Methode unterstützt die programmierende Person dabei, eigene Ideen oder Lösungsansätze in schriftlicher Form festzuhalten und ermöglicht ebenfalls externen Personen den Code zu verstehen ([43]). Es hilft den Studierenden, Prozesse vereinfacht darzustellen und zu veranschaulichen, ohne sich mit der technischen Umsetzung auseinander zu setzen. ([44]). Die Studierenden sollen nach Abschluss dieses Kurssegments, wie in Abbildung 3.5 gezeigt, in der Lage sein, selbständig einen Algorithmus in Pseudocode zu schreiben.



The image shows a user interface mockup for a web application. On the left, there is a sidebar with a dark header containing the text "Übung1 - Code auf Papier". Below the header, there is a list of navigation items: "Übersicht", "Einleitung", "Übung1" (which is highlighted with a gray background), "Übung2", and "Übung3". At the bottom of the sidebar is a button labeled "Zurück". The main content area has a light gray background. At the top right of the content area is a small icon consisting of three horizontal lines. The main title "Übung1 - Code auf Papier" is centered at the top of the content area. Below it is the subtitle "Schreibe einen Algorithmus auf Papier". Underneath the subtitle is the heading "Anleitung:". Following this is a large block of placeholder text in Latin: "Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet." At the bottom of the content area is a button labeled "↑ Hochladen".

Abb. 3.5.: Konzeptidee für die Erstellung von Pseudocode (eigene Darstellung)

3. Die Methodik

3.2.3.4. Anforderung der Gamification

Von Beginn an war die Integration von Gamification und E-Learning-Aspekten (siehe 4.2, 2.4) ein entscheidender und unverzichtbarer Schritt, um die Kurse auf eine ansprechende und effektive Weise zu konzipieren. Hierfür wurde unter anderem das Spiel "Pebble-Simulator" erstellt, was Abbildung 3.6 und 3.7 entnommen werden kann. Dabei sollen die Studierenden selbst eine Schildkröte steuern und sie aus dem Labyrinth führen. Dies verleiht der Lernumgebung eine spielerische Note.

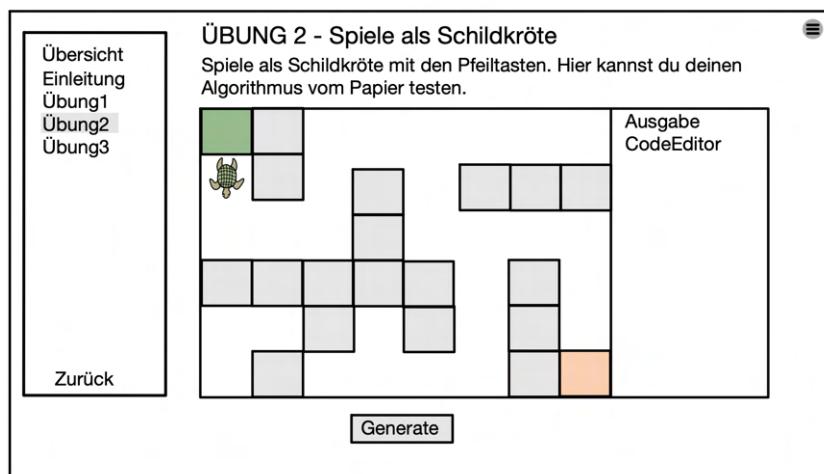


Abb. 3.6.: Konzeptidee 1 für die Gamification (eigene Darstellung)

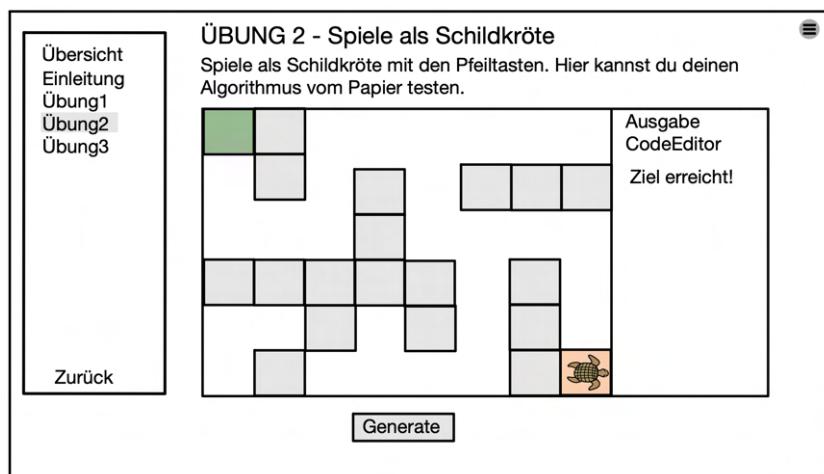


Abb. 3.7.: Konzeptidee 2 für die Gamification (eigene Darstellung)

3. Die Methodik

3.2.3.5. Erstellung eines Algorithmus

Des Weiteren soll ein Kurselement den Studierenden beibringen, einen Algorithmus zu erstellen. Die finale Aufgabe besteht darin, aus dem erstellten Pseudocode (siehe 3.2.3.3) einen Algorithmus zu formulieren, der die Schildkröte erfolgreich aus dem Labyrinth führt. Die Konzeptidee kann Abbildung 3.8 entnommen werden.

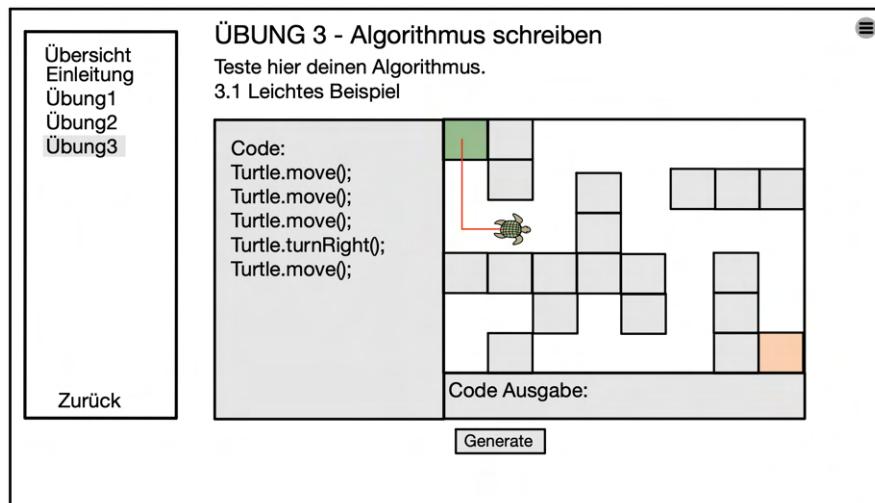


Abb. 3.8.: Konzeptidee für die Erstellung eines Algorithmus (eigene Darstellung)

3.2.3.6. Zukünftige Anforderung

Eine zukünftige Anforderung beinhaltete die Integration einer Peer-Review-Funktion (siehe 8), die es Studierenden ermöglichen soll, ihren erstellten Pseudocode zur Begutachtung an andere Mitstudierenden zu senden und Feedback einzuholen. Die gegenseitige Bewertung und Verbesserung der Lösungsansätze soll den Studierenden helfen, sich untereinander auszutauschen und kann der Lerntheorie des Konnektivismus (siehe 2.3.4) zugeordnet werden.

3.2.4. Didaktische Methode

Die gewählte pädagogische Herangehensweise wurde identifiziert und umfasst ein Lernmanagementsystem (siehe 2.2.2), das im Rahmen von Blended Learning (siehe 2.2.3) und unter Anwendung von WBTs (siehe 2.2.1) eingesetzt wird. Die Lernaufgaben setzen sich aus verschiedenen Elementen zusammen. Darunter befinden sich ausführbare Code-Editoren (siehe 6.3.1), Übungsaufgaben (siehe 5) und anklickbare Akkordeons (siehe 6.3.2).

3. Die Methodik

3.2.5. Lernorganisation

Die Lernorganisation beschreibt, wie die Durchführung des Angebots strukturiert wird ([16]). Der Kurs soll als Erweiterung des Lernmaterials für das Modul GdP dienen und ab der vierten Woche des Wintersemesters eingeführt werden. Die Aufgaben können sowohl in Seminaren als auch im individuellen Selbststudium zu Hause und im eigenem Tempo von den Studierenden bearbeitet werden.

3.2.6. Vorgehen

Es wurde besonders darauf geachtet, eine agile Vorgehensweise zu verwenden, um sicherzustellen, dass neue Anforderungen und Wünsche direkt berücksichtigt werden können. Die Website wurde so gestaltet, dass auch zukünftige Anforderungen und Wünsche (siehe 3.2.3.6) ohne Probleme abgeändert und eingebunden werden können. Die wöchentlichen Meetings (siehe 1.2) haben dabei sehr geholfen, da diese eine flexible Anpassung während des Entwicklungsprozesses ermöglichten.

3.2.7. Ähnliche Einführungskurse

Es wurde Wert auf eine intensive Auseinandersetzung des Lehrinhalts und Skripts von GdP gelegt. Zudem wurden auch andere Vorkurse und Einführungskurse untersucht. Dabei lag der Fokus vor allem auf Online-Kursen, um Einblicke in die Struktur und Gestaltung von Webkursen zu gewinnen ([45], [46], [47], [48], [49]), [50]). Hierzu fiel auf, dass der Inhalt der Kurse so gut wie immer in einfacher und verständlicher Sprache geschrieben wurde und nach einer Erklärung oftmals eine Übungsaufgabe folgte. Diese Abfolge wurde auch bei der Erstellung der Übungsaufgaben der einzelnen Kurse (siehe 5) eingehalten.

4. E-Learning Aspekte und multimediale Inhalte

Bevor die technische Umsetzung des Frontends und Backends betrachtet wird, ist es notwendig, sich mit den generellen E-Learning-Aspekten und multimedialen Gestaltungsmaßnahmen zu befassen, die für alle entwickelten Frontend-Seiten von Bedeutung waren.

4.1. Interfacedesign und Screendesign

Um sicherzustellen, dass das Interfacedesign und das Screendesign harmonisch sind, müssen Aspekte wie Gestaltung, Benutzerfreundlichkeit und Interaktionsdesign sorgfältig abgestimmt werden ([51]). Dabei sollte stets berücksichtigt werden, dass der Inhalt dynamisch nach Bedarf angepasst werden kann, um neue Übungen hinzuzufügen, während das Design jedoch vorerst unverändert bleibt ([51]).

4.1.1. Layout

Um mit dem Erstellen der Website beginnen zu können, müssen vorerst Gedanken bezüglich Displaygröße, Seitenverhältnis, Displayauflösung und die Darstellung in Hoch- oder Querformat gemacht werden ([51]). Die Website wurde mit der Intention erstellt, die Studierenden einen eigenen Code am PC, Laptop oder Tablet erstellen zu lassen. Aus diesem Grund kommt für die Gestaltung dieses Projekts das Querformat in Frage.

Gerätetyp	Größe (Inch)	Seitenverhältnis	Typische Auflösungen (Pixel)
Smartphones	4 – 6	9:16	720 x 1.280, 1.080 x 1.920
Tablet-PCs	7 – 12	4 : 3, 16:10	1.024 x 768, 1.280 x 800, 2.048 x 1.536
Notebooks	13 – 17	16 : 9	1.366 x 768, 1.600 x 900, 1.920 x 1.080
Monitore	22 – 30	16 : 9, 16:10	1.920 x 1.080, 1.920 x 1.200

Abb. 4.1.: Screendesign ([51])

Jeder Gerätetyp hat andere Auflösungen und Seitenverhältnisse (siehe Abbildung 4.1), daher ist stets zu beachten, dass die Website nicht nur für ein Gerät, Monitor oder Laptop entwickelt wird, sondern diese auch an verschiedenen Geräten getestet wird ([51]).

Es ist außerdem unklar, mit welchem Webbrowser die Studierenden die Website aufrufen, dadurch kann nicht genau gesagt werden, wie viel Platz von den Menü-/Navigationsleisten des Browsers in Anspruch genommen wird ([51]). Es kann zudem nicht vorhergesagt wer-

4. E-Learning Aspekte und multimediale Inhalte

den, ob sich die Studierenden die Website im Vollbildmodus anzeigen lassen, Windows-Taskleisten oder ein Mac-Dock vorhanden sind ([51]). Der Viewport, der die tatsächliche sichtbare Fläche der Website anzeigt, könnte daher unter verschiedenen Voraussetzungen eingeschränkt sein ([51]). Dies deutet darauf hin, dass ein adaptives Layout erforderlich ist, um sicherzustellen, dass die Website auf verschiedenen Endgeräten optisch stimmig wirkt ([51]). Allerdings genügt das allein nicht, da es unmöglich ist, für jede einzelne Gerätevariante spezifische Anpassungen vorzunehmen. Daher ist ein responsives Layout notwendig. Die Website muss also in der Lage sein, auf verschiedene Änderungen des Viewports zu reagieren ([51]). Es ist möglich, den Viewport der Website anzupassen, aber über einen bestimmten Punkt hinaus kann dies problematisch sein, da bestimmte Editoren verzerrt dargestellt werden können, was das Nutzungserlebnis erheblich beeinträchtigen würde. Daher wird jedem empfohlen, die Website im Vollbildmodus zu öffnen, um eine optimale Darstellung zu gewährleisten. Die Website soll mit einem Laptop, Rechner oder Tablet aufgerufen werden. Für Mobiltelefone ist die Website nicht ausgelegt, da für den Code-Editor (siehe 6.3.1) ausreichend Platz benötigt wird.

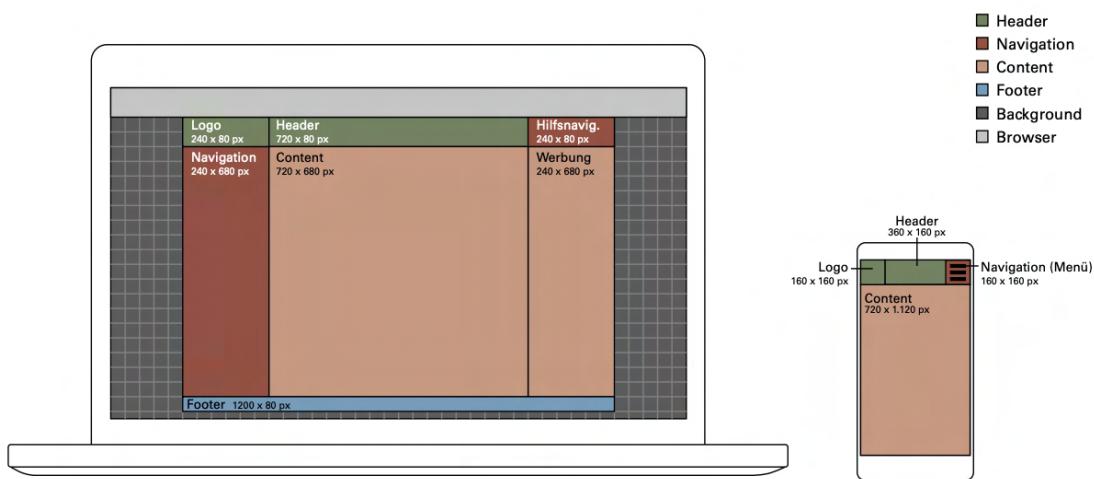


Abb. 4.2.: Standardlayout: typische Bereiche einer Website ([51])

Bei der Erstellung der Website wurde sich an dem Standardlayout (siehe Abbildung 4.2) orientiert. Dabei befindet sich die Navigation (siehe 4.1.3) im linken Bereich der Website. Auch die Position des Logos, des Footers und des Contentbereichs entsprechen der Abbildung. Jedoch wird anstelle der Hilfsnavigation der Fortschritt des Kurses angezeigt. Darüber hinaus gibt es keine Werbung, wodurch der Contentbereich größer ausfällt.

4.1.2. Header, Logo und Footer

Im oberen Kopfbereich der Website befindet sich zu jeder Zeit der sogenannte Header¹. Er dient als räumliche Abgrenzung zur Inhaltsseite. Das Logo der Website ist stets auf der oberen linken Seite platziert und ermöglicht durch einen Klick das Zurückkehren zur Startseite. Innerhalb der Kurskapitel wird im Header ein Highscore-Balken eingeblendet, der anzeigt, zu wie viel Prozent der Kurs bereits abgeschlossen wurde. Dieser Balken ist statisch positioniert, daher bleibt er beim Scrollen immer sichtbar.



Abb. 4.3.: Footer der Turtle-Website (eigene Darstellung)

Der Footer² ist ein wichtiger Bestandteil einer Website. Er befindet sich konstant am unteren Ende der Webseite, also wortwörtlich am Fuße der Seite. Der Footer ist in der Regel ein Bereich, der sich vom Rest der Website abhebt ([52]). Auf der Turtle-Website ist der Footer mit einer farblichen Abtrennung gekennzeichnet (siehe Abbildung 4.3). Der Footer enthält meistens wichtige Informationen zur Website, Links zu anderen Seiten oder bietet den Nutzern die Möglichkeit, mit den Entwicklern Kontakt aufzunehmen. Der Footer trägt zur Nutzerfreundlichkeit der Website bei und signalisiert außerdem, dass das Ende der Website erreicht wurde ([52]).

4.1.3. Die Navigation

Die Navigation, die den Studierenden bei der Orientierung hilft, wird am linken Rand der Website dargestellt. Sie erscheint erstmals auf der Startseite und gibt von oben nach unten die empfohlene Abfolge der verschiedenen Kurssegmente vor. Die Navigation ist in Ebenen unterteilt, sobald ein Navigationspunkt ausgewählt wird, hebt er sich farblich ab und führt die nutzenden Personen zur entsprechenden Seite. Darunter erscheinen zusätzliche Navigationspunkte, die die Struktur der aktuellen Seite verdeutlichen. Die Navigation besitzt in diesem Fall die sogenannte Baumstruktur und ist daher äußerst anwenderfreundlich und bietet eine intuitive Bedienung ([51]). Sie strukturiert Informationen übersichtlich und ermöglicht zudem eine mühelose Integration neuer Inhalte in die vorhandene Struktur ([51]).

¹Aus dem Englischen “head”: Kopf

²auch Fußzeile genannt

4.1.4. Farbpalette, Grafiken und Schriftart

Auf die Farbpsychologie im Webdesign wurde ebenfalls Wert gelegt. Die Auswahl von Farben im Webdesign spielt eine bedeutende Rolle, da sie spezifische Emotionen und Reaktionen in uns Menschen hervorrufen können ([53]). Menschen reagieren auf Farben aufgrund kultureller und psychologischer Faktoren unterschiedlich, wobei erlernte Assoziationen bezüglich Farben in verschiedenen Kulturen variieren können ([53]). Selbstverständlich sollte berücksichtigt werden, dass Farben von jedem Individuum unterschiedlich wahrgenommen werden und es keine festen Regeln für die Verwendung von Farben gibt ([54]). Die Website wurde mit vielen Grüntönen gestaltet (siehe Abbildung 4.4), da die Farbe Grün in der westlichen Kultur oft symbolisch für das Leben und persönlichen Wachstum steht ([53],[54]). Diese Symbolik passt gut zu den Studierenden, die im Verlauf des Kurses ebenfalls weiterwachsen und sich entwickeln.

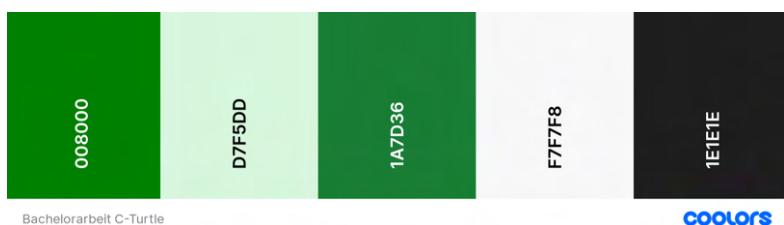


Abb. 4.4.: Verwendete Farbpalette ([55])

Durch die gezielte Verwendung von Farben können Website-Besuchende dazu angeregt werden, bestimmte Knöpfe zu betätigen oder sich generell besser auf den Inhalt zu konzentrieren. Neben den Grüntönen wurden überwiegend neutrale und kühle Farbtöne verwendet. Diese Farben ermöglichen den nutzenden Personen, die Inhalte in einer ruhigen und entspannten Umgebung zu erfassen, ohne von zu vielen Farbeinflüssen abgelenkt zu werden. Dennoch werden wichtige Aktionen oder Handlungen durch die Verwendung der grünen Farbe hervorgehoben. Die Verwendung neutraler Farbtöne entspricht auch dem aktuellen Design-Trend, da sie ein minimalistisches Erscheinungsbild vermitteln ([53]). Es wurde darauf geachtet, dass der Hintergrund etwas abgedunkelt wurde, da ein rein weißer Hintergrund auf Dauer anstrengend für die Augen sein kann ([51]). Basierend auf der vorgegebenen Farbpalette wurden die Grafiken sorgfältig ausgesucht und zeichnen sich alle durch ein äußerst schlichtes Design aus. Sämtliche Grafiken stammen aus Open-Source Quellen und können daher frei und kostenlos genutzt werden ([56],[57],[58]).

Die Standardtypische Schrift der HTWK ist Source Sans Pro [59]. Damit der Wiedererkennungswert vorhanden bleibt, wurde auch diese Schriftart bei der Gestaltung der Website verwendet.

4.2. Verwendete Gamification

Es wurden Gamification-Elemente (siehe 2.4) innerhalb der Lernumgebung verwendet, die einen großen Teil zur Entwicklung der Website beitragen. Dabei soll das Verwenden von Gamification in der Onlinelehre vor allem zur Motivationssteigerung und zu bestimmten Verhaltensänderungen der anwendenden Personen führen, damit ein finales Ziel erreicht wird ([30]). Das finale Ziel dieses Webkurses ist der erfolgreiche Abschluss der verschiedenen Kapitel. Die Motivation soll dabei so gut es geht aufrechterhalten werden, wofür es verschiedene Lösungen gibt, die in den folgenden Unterkapiteln durchleuchtet werden.

4.2.1. Narration und Storytelling

Das Erzählen von Geschichten und Abenteuern übt schon seit unserer Kindheit eine Faszination auf uns Menschen aus. Eine gelungene Geschichte vermag das Interesse der lesenden Personen zu wecken und sie gleichermaßen zu inspirieren ([60]). Mit Hilfe einer Narration können komplexe Sachverhalte spielerisch aufgeschlüsselt und aus unterschiedlichen Perspektiven betrachtet werden ([60]). Aus diesem Grund ist das Erzählen von Geschichten schon immer ein wichtiger Bestandteil von Bildungsprozessen gewesen ([61]). Wenn es in der Geschichte einen roten Faden oder eine Hauptfigur gibt, können Details und Zusammenhänge besser eingeprägt werden ([61]).

So wurde auch Pebble, die Schildkröte, ins Leben gerufen. Pebble hat sich nach einem schlimmen Sturm verirrt und findet den Weg nicht zurück zu seiner Familie. In seiner eigenen Not bietet Pebble den Studierenden seine Hilfe an, denn er ist selbst Informatiker und kann ihnen viele wertvolle Informationen zu Pseudocode und Algorithmen vermitteln. Als Gegenleistung dafür bittet er die Studierenden, ihn aus verschiedenen Labyrinthen zu helfen. Diese fortlaufende Geschichte erstreckt sich über sämtliche Kapitel, wobei Pebble die Moderation und Begleitung der Studierenden in jedem Abschnitt übernimmt. Pebble hat außerdem die Rolle eines Mentors und motiviert die Studierenden regelmäßig mit aufmunternden Worten. Er ist daher die handelnde Hauptfigur, die eine Problemstellung hat, die am Ende durch Hilfe der Studierenden gelöst wird ([60]). Nachdem alle Kapitel erfolgreich abgeschlossen sind, kehrt Pebble in die Obhut seiner Familie zurück.

4.2.2. Der Pebble-Simulator

Wie bereits in Abschnitt 3.2.3.4 im Zusammenhang mit der Konzeptidee erläutert wurde, sollte ein Spiel entworfen werden, bei dem die studierende Person in die Rolle von Pebble, der Schildkröte, schlüpft und diese mithilfe verschiedener Tasten aus dem Labyrinth steuern kann. Abbildung 4.5 zeigt einen Ausschnitt des Spiels “Pebble-Simulator”.

Die Bewegungen der Schildkröte sind auf das Vorwärtsgehen (mit Taste W), Rechtsdrehen (mit Taste D) und Linksdrehen (mit Taste A) beschränkt. Diese Einschränkung ist von Bedeutung, da die Schildkröte beim Schreiben des Algorithmus ebenfalls nur über bestimmte Methoden und Attribute verfügt (siehe 3.1.2).

Die Fähigkeiten der Schildkröte sind auf das beschränkt, was auch eine reale Schildkröte machen könnte. Daher sind Aktionen wie das Überspringen einer Mauer oder das Rückwärtslauen nicht möglich.

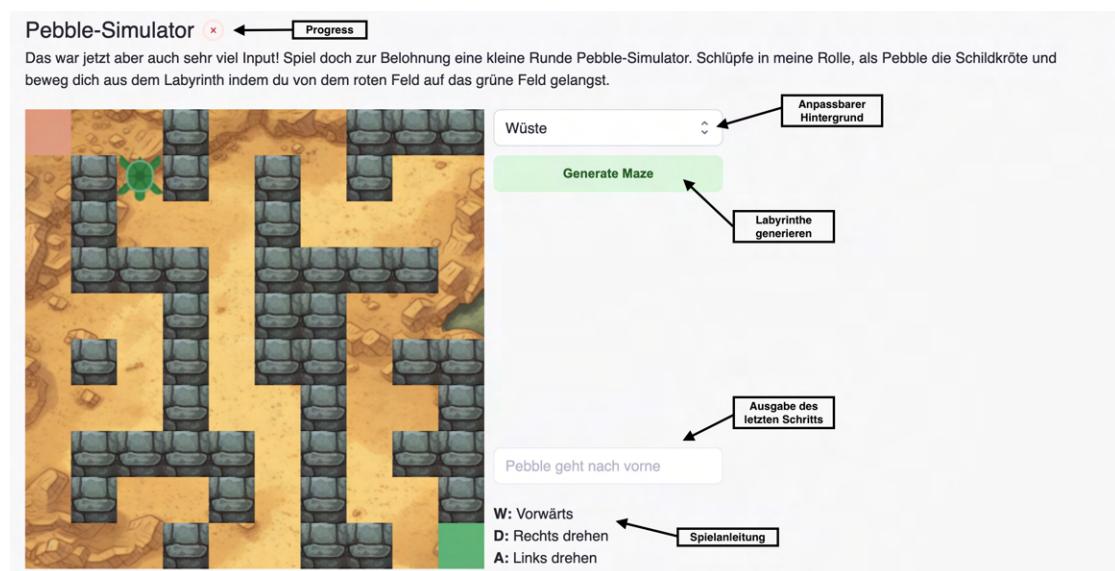


Abb. 4.5.: Das Spiel Pebble-Simulator mit Erklärungen (eigene Darstellung)

Innerhalb des Spiels haben die Studierenden die Option, die Hintergrundgrafik des Spiels sowie das Labyrinth über die entsprechend markierten Bereiche anzupassen. Zusätzlich steht ein kleines Feld zur Verfügung, das den letzten Zug des Spielers anzeigt.

4.2.3. Fortschrittsanzeige

Die Fortschrittsanzeige ist ebenfalls ein wichtiges Elemente der Gamifizierung ([62]). Sie ermöglicht den teilnehmenden Personen, jederzeit den Stand ihres Fortschritts im Kurs zu verfolgen und zu erfahren, wie viel Prozent ihnen noch bis zur vollständigen Absolvierung des Kurses fehlen. Die Fortschrittsanzeige befinden sich im Header (siehe 4.1.2) und wird vollständig angezeigt, wenn die Leiste 100% erreicht hat. Um die Leiste zu füllen, ist es erforderlich, das Kapitel abzuschließen und die bewertbaren Aufgaben erfolgreich zu bewältigen. Die bewertbaren Aufgaben sind durch ein kleines rotes Kreuz gekennzeichnet, das nach erfolgreicher Bearbeitung grün wird (siehe Abbildung 4.6). Mit jeder abgeschlossenen Aufgabe steigt auch der Fortschritt in der Anzeige.

Aufgabe ist noch nicht bearbeitet worden, Fortschrittsanzeige bei 0%, rotes Kreuz



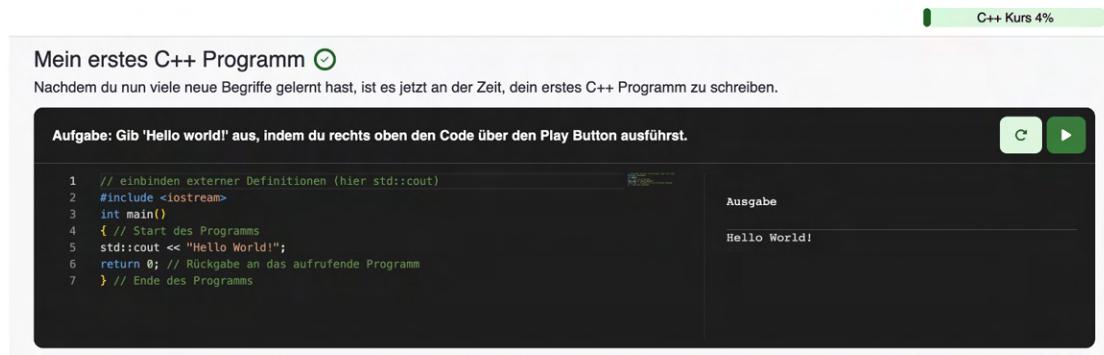
The screenshot shows a dark-themed C++ code editor. At the top, a green progress bar indicates 'C++ Kurs 0%' with a small progress indicator. Below it, the title 'Mein erstes C++ Programm' is shown with a red close button. A descriptive text below the title says: 'Nachdem du nun viele neue Begriffe gelernt hast, ist es jetzt an der Zeit, dein erstes C++ Programm zu schreiben.' A task box contains the instruction: 'Aufgabe: Gib 'Hello world!' aus, indem du rechts oben den Code über den Play Button ausführst.' To the right of the code area is a 'Ausgabe' (Output) window which is currently empty. On the far right, there are two buttons: a green one with a white play icon and a blue one with a white 'c' icon.

```

1 // einbinden externer Definitionen (hier std::cout)
2 #include <iostream>
3 int main()
4 { // Start des Programms
5     std::cout << "Hello World!";
6     return 0; // Rückgabe an das aufrufende Programm
7 } // Ende des Programms

```

Aufgabe ist bearbeitet worden, Fortschrittsanzeige bei 4%, grüner Haken



This screenshot shows the same environment after the task has been completed. The progress bar now shows 'C++ Kurs 4%' with a green progress indicator. The title 'Mein erstes C++ Programm' now includes a green circular icon with a white checkmark. The task box remains the same. The 'Ausgabe' window now displays the output 'Hello World!'. The green and blue buttons on the right are still present.

```

1 // einbinden externer Definitionen (hier std::cout)
2 #include <iostream>
3 int main()
4 { // Start des Programms
5     std::cout << "Hello World!";
6     return 0; // Rückgabe an das aufrufende Programm
7 } // Ende des Programms

```

Abb. 4.6.: Fortschrittsanzeige bevor und nach Lösung der Aufgabe (eigene Darstellung)

4.2.4. Kursabschluss und Urkunden

Am Ende des C++ Einführungskurses befindet sich ein Knopf mit der Aufschrift “Kurs abschließen”. Dieser Knopf ist erst anklickbar, sobald alle Aufgaben erfolgreich gelöst wurden und die Fortschrittsanzeige 100 % erreicht hat. Sobald der Knopf betätigt wird, erscheint eine Nachricht, die der studierenden Person zum erfolgreichen Abschluss gratuliert.

Nach Abschluss des Algorithmus Kurses erscheint ein weiterer Knopf am Ende des Kapitels, der mit der gleichen Aufschrift “Kurs abschließen“ gekennzeichnet ist. Falls auch in diesem Kurs die Fortschrittsanzeige auf 100 % steigt, lässt sich dieser Knopf drücken. Daraufhin erscheint eine Nachricht, in der Pebble den Studierenden für die Hilfe dankt. Zudem können die Studierenden eine Teilnahme-Urkunde herunterladen (siehe Abbildung 4.7), auf der alle von ihnen abgeschlossenen Kursinhalte aufgeführt sind. Dies dient als Erinnerung und als schöner Abschluss des Kurses.

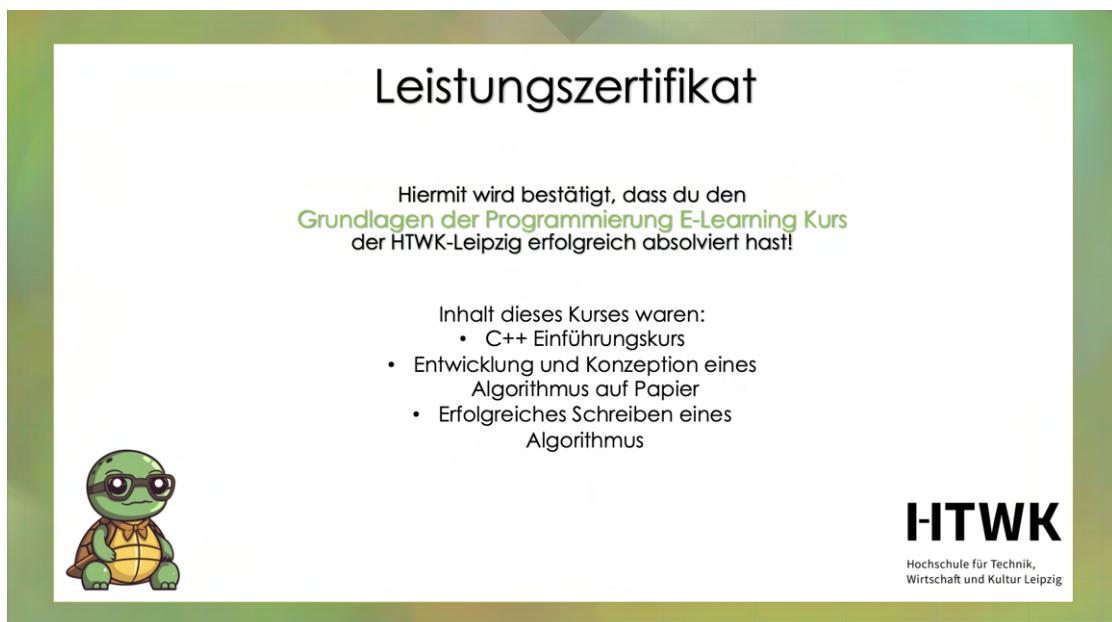


Abb. 4.7.: Zertifikat nach dem Kursabschluss (eigene Darstellung)

4.2.5. Interaktive Auswahl und Vorgehen

Die Studierenden erhalten die Möglichkeit, ihre eigenen Programmierkenntnisse an einem kleinen Quiz auf der Willkommensseite einzuschätzen. Hier stellt Pebble die Frage, ob die Studierenden bereits Programmiererfahrungen haben, und gibt je nach Auswahl der Studierenden unterschiedliche Rückmeldungen (siehe Abbildung 4.8).

Nimm gerne an dieser kleinen Umfrage teil und erzähl mir, ob du bereits Programmiererfahrungen hast.

- Anfänger (keine Erfahrung)
- Fortgeschritten (etwas Erfahrung)
- Profi (viel Erfahrung)

Nimm gerne an dieser kleinen Umfrage teil und erzähl mir, ob du bereits Programmiererfahrungen hast.

- Anfänger (keine Erfahrung)
- Fortgeschritten (etwas Erfahrung)
- Profi (viel Erfahrung)

Nimm gerne an dieser kleinen Umfrage teil und erzähl mir, ob du bereits Programmiererfahrungen hast.

- Anfänger (keine Erfahrung)
- Fortgeschritten (etwas Erfahrung)
- Profi (viel Erfahrung)

Nimm gerne an dieser kleinen Umfrage teil und erzähl mir, ob du bereits Programmiererfahrungen hast.

- Anfänger (keine Erfahrung)
- Fortgeschritten (etwas Erfahrung)
- Profi (viel Erfahrung)

Abb. 4.8.: Interaktive Umfrage (eigene Darstellung)

4. E-Learning Aspekte und multimediale Inhalte

Anschließend wird den Studierenden die Vorgehensweise des Kurses erläutert (siehe Abbildung 4.9). Für Studierende, die ihre Kenntnisse auf Anfänger- oder Fortgeschrittenenniveau angeben, ist die Teilnahme am C++ Einführungskurs vorgesehen.

Für diejenigen, die sich selbst als Profis einschätzen, ist die Teilnahme optional. Im Sinne des Konstruktivismus (siehe 2.3.3) haben die Studierenden hier die eigenständige Entscheidungsfreiheit, ob sie den Einführungskurs belegen möchten. Dennoch wird eine Nachricht angezeigt, die die Teilnahme zur Auffrischung ihrer Kenntnisse empfiehlt.

Vorgehensweise

- **Anfänger/Fortgeschrittene:**
Bearbeite zuerst den C++ Einführungskurs.
Anschließend sollst du Algorithmus Teil 1, 2 und 3 bearbeiten.
- **Profi:**
Der C++ Einführungskurs ist für dich optional, aber denk dran, dass es nicht schadet nochmal einen Einstieg zu bekommen.
Anschließend sollst du Algorithmus Teil 1, 2 und 3 bearbeiten.



Abb. 4.9.: Vorgehensweise für den Kurs (eigene Darstellung)

5. Die verschiedenen Kursinhalte

Wie bereits in Kapitel 3.2.3 erläutert, war die Entwicklung und Konzeption eines C++ Einführungskurses und die Erläuterung von Algorithmen ein zentrales Anliegen. In diesem Kapitel werden nun die konkreten Kursinhalte dieser beiden bedeutenden Kurselemente angeschaut und untersucht, wie diese umgesetzt wurden. Für die Realisierung wurden sowohl das GdP-Skript als auch andere Kurse (siehe 3.2.7) als Referenz herangezogen. Der Fokus dieses Kapitel liegt auf den Aufbau der einzelnen Kursinhalte. Die Implementierung wird im Kapitel 6 ausführlich besprochen.

5.1. Der C++ Einführungs Kurs

Die Entwicklung und Gestaltung des C++ Einführungskurses orientiert sich am Lehrplan des GdP-Moduls (siehe 3.2.3.1). Das Hauptziel besteht darin, den Studierenden eine Einführung in die Programmiersprache C++ zu bieten. Da nicht alle Studierenden zu Beginn des INB/MIB-Studiums bereits Programmierkenntnisse besitzen, sollte dieser Kurs eine Gelegenheit bieten, grundlegende Kenntnisse zu erlangen. Für Studierende mit fortgeschrittenen Programmierkenntnissen ist die Teilnahme an diesem Kurs zwar optional (siehe 4.2.5), aber dennoch empfehlenswert, da dieser grundlegende Informationen über Schleifen, Variablen, Datentypen und Operatoren bereitstellt, die für die Bewältigung des nächsten Kursabschnitts (siehe 5.2) von Bedeutung sind. Die Studierenden haben die Freiheit, selbst zu entscheiden, ob sie den Einführungskurs bearbeiten möchten, was auf die Prinzipien der Lerntheorie des Konstruktivismus (siehe 2.3.3) zurückzuführen ist. In diesem Einführungskurs haben die Studierenden die Möglichkeit, für jede Aufgabe Punkte zu sammeln, um ihre Fortschrittsanzeige zu steigern (siehe 4.2.3). Der Kurs startet wie auch das GdP Modul mit allgemeinen Definitionen (siehe A.2), die auch dem GdP-Skript entnommen werden können. Diese Definitionen sind in sogenannten Akkordeons (siehe 6.3.2) untergebracht und können durch einen einfachen Klick geöffnet werden. Bei den Aufgaben wird am Anfang betont, dass die Studierenden jederzeit für ausführlichere Definitionen und Erklärungen in dem GdP-Skript nachschlagen können. Es wird den Studierenden anschließend gezeigt, wie die Syntax eines C++ Programms aussieht und sie werden aufgefordert, den Code-Editor (siehe 6.3.1) zu benutzen, um den bekannten Einführungstext “Hello World“ auszugeben ([63]). Zusätzlich erhalten die Studierenden eine Erklärung darüber, was Kommentare sind und wie sie diese verfassen können. In Bezug auf die inhaltliche Struktur deckt der C++ Einführungskurs die Themen der GdP-Skripts der Wochen 1, 2 und 4 ab (siehe A.2). Der Inhalt wurde

5. Die verschiedenen Kursinhalte

auf das Wesentliche reduziert und einfach gehalten, wobei der Fokus auf den Elementen liegt, die später im zweiten Kursabschnitt (siehe 5.2) benötigt werden. Die Studierenden erwerben Kenntnisse über Variablen, Datentypen, Operatoren, Kontrollstrukturen, Fallunterscheidungen und Schleifen, da all diese Konzepte für die Entwicklung eines Algorithmus von wesentlicher Bedeutung sind ([45]). In diesem Kontext ist es von großer Bedeutung, die Aufgaben bewusst einfach zu gestalten, um Programmierunbefahrenen die Möglichkeit zu geben, diese erfolgreich zu bearbeiten. Das Ziel ist es, eine Lernumgebung zu schaffen, die für alle Studierenden zugänglich ist und ihnen die Möglichkeit bietet, schrittweise Vertrauen in ihre Fähigkeiten aufzubauen, während sie die Grundlagen der C++ Programmierung erlernen.

Es besteht die Möglichkeit, jederzeit neue Erklärungen und Übungen hinzuzufügen, wodurch die Gestaltung des Kurses flexibel und anpassbar bleibt (siehe 3.2.6) und daher auch zukünftig sinnvoll benutzt werden kann (siehe 8).

5.2. Der Algorithmus Kurs

Anders als beim Einführungskurs, sollen alle Studierenden den Algorithmus-Kurs absolvieren. Der Kurs ist in drei Abschnitte unterteilt, die als Ganzes betrachtet das Ziel haben, den Studierenden beizubringen, wie sie einen Algorithmus entwickeln und schreiben (siehe 3.2.3.5).

Dieser Algorithmus Kurs vereint sowohl Elemente der Lerntheorie des Behaviorismus (siehe 2.3.1), als auch der Lerntheorie des Kognitivismus (siehe 2.3.2). Die Verwendung von Fortschrittsanzeigen (siehe 4.2.3) und Zertifikaten (siehe 4.7) dient dazu, Studierende zu belohnen, sie zu motivieren und ihnen ein Ziel vor Augen zu setzen, was dem Behaviorismus zugeordnet werden kann. Gleichzeitig erhalten die Studierenden eine aktive Rolle, bei der sie eigenständige Informationen aufnehmen, verarbeiten und anhand vorgegebener Problemstellungen Lösungswege entwickeln sollen ([25]), was der Lerntheorie des Kognitivismus entspricht. Die Kombination und Verwendung dieser beiden Lerntheorien verdeutlicht, dass es möglich ist, verschiedene Theorien zu vereinen, um eine effiziente und innovative Lehr- und Lernumgebung zu erstellen ([64]).

5.2.1. Algorithmus Teil 1

Der Kurs startet mit dem Abschnitt “Algorithmus Teil 1“, der das Ziel hat, den Studierenden beizubringen, wie ein Pseudocode geschrieben wird (siehe 3.2.3.3). Zunächst werden verschiedene Entwurfstechniken wie Programmablaufpläne und Struktogramme erläutert, bevor der Schwerpunkt auf Pseudocode gelegt wird. Hierbei lernen die Studierenden, wie sie im Pseudocode Kontrollstrukturen verwenden können und wie sie einen Pseudocode schreiben. Das Schreiben des Pseudocodes wird anhand eines spielerischen Beispiels, nämlich dem Backen einer Tiefkühl-Pizza, veranschaulicht. Es soll in diesem Kurssegment nicht um das technische Schreiben von einem Programm gehen, sondern um das Schreiben eines Pseudocodes. Diese Abtrennung wurde gemacht, da es besser ist, Algorithmen zunächst unabhängig von der Programmiersprache zu notieren, da viele technische Details vorerst weggelassen werden können ([50]).

Nach dem Pizza-Beispiel können die Studierenden das Spiel “Pebble-Simulator“ (siehe 4.2.2) spielen. Dieses Spiel wird dort eingebaut, um eine klare Trennung zwischen dem Erlernten und dem darauffolgenden Schreiben des Pseudocodes zu schaffen. Anschließend erhalten die Studierenden die Aufgabe, einen eigenen Pseudocode zu erstellen, der Pebble die Schildkröte dabei hilft, aus einem Labyrinth zu entkommen. Es wird den Studierenden die Attribute und Methoden gegeben, die Pebble ausführen kann (siehe 3.1.2). Zusammen mit diesem Wissen sollen sie einen Pseudocode im vorgegebenen Textfeld schreiben. Es steht den Studierenden selbstverständlich auch der “Pebble-Simulator“ zur Verfügung, um ihren geschriebenen Pseudocode zu testen. Für die Zukunft ist an dieser Stelle geplant den Pseudocode für eine Peer-Review unter den Studierenden freizugeben (siehe 8), für eine gegenseitige Bewertung.

5.2.2. Algorithmus Teil 2

In diesem Abschnitt erhalten die Studierenden eine Einführung in Klassen und Objekte, sowie deren Verwendung von Attributen und Methoden. Diese Konzepte bilden Grundlagen für das Verständnis der Programmierung in objektorientierten Sprachen. Zusätzlich wird ihnen gezeigt, wie sie den Code-Editor mit der Labyrinthanzeige (siehe 6.3.1.1) verwenden können. Die Studierenden werden dazu aufgefordert, einfache Algorithmen und Codesegmente zu entwickeln, um Pebble aus sehr einfach gehaltenen Labyrinthen zu führen. Dieser praktische Teil des Algorithmus-Kurses dient als Einstieg in die Welt der Algorithmusentwicklung. Es legt die Grundlagen für die Studierenden, damit sie im nächsten Abschnitt des Algorithmus-Kurses (siehe 5.2.3) die gestellten Methoden und Attribute (siehe 3.1.2) eigenständig nutzen können, um einen komplexeren Algorithmus zu entwickeln und dadurch Pebble aus schwierigeren Labyrinthen zu befreien.

5.2.3. Algorithmus Teil 3

Im dritten und abschließenden Abschnitt des Algorithmus-Kurses geht es darum, den von den Studierenden erstellten Pseudocode (siehe 5.2.1) in einen ausführbaren Algorithmus zu umschreiben (siehe 3.2.3.5). Dabei wird den Studierenden anhand eines Beispiels gezeigt, wie sie einen Pseudocode in C++ Code umwandeln können. Sie erhalten nützliche Tipps, die ihnen bei dem Überführen ihres Pseudocodes in C++ Code unterstützen sollen. Der zuvor erstellte Pseudocode dient dabei als Ausgangspunkt und das Kapitel führt sie schrittweise durch die Umschreibung ihres Pseudocodes.

Das endgültige Ziel besteht darin, einen Algorithmus zu schreiben, mit dem Pebble die Schildkröte aus verschiedenen Labyrinthen herausgeführt werden kann. Dabei gibt es drei Schwierigkeitsstufen der Labyrinthe, um den Algorithmus der Studierenden unter Beweis zu stellen. Sobald die Studierenden bei allen drei Labyrinthen das Ziel erreicht haben, kann der Kurs abgeschlossen werden (siehe 4.2.4).

6. Implementation

In diesem Kapitel wird ausführlich auf die Implementierung und Umsetzung des Frontends¹ und Backends² geschaut. Dabei wird detailliert auf die technischen Schritte eingegangen, die erforderlich waren, um die Webseite funktionsfähig zu machen. Die kompletten Inhalte des Codes (siehe [19],[66]), können auf GitHub³ eingesehen werden.

6.1. Technische Informationen zum Backend und Frontend

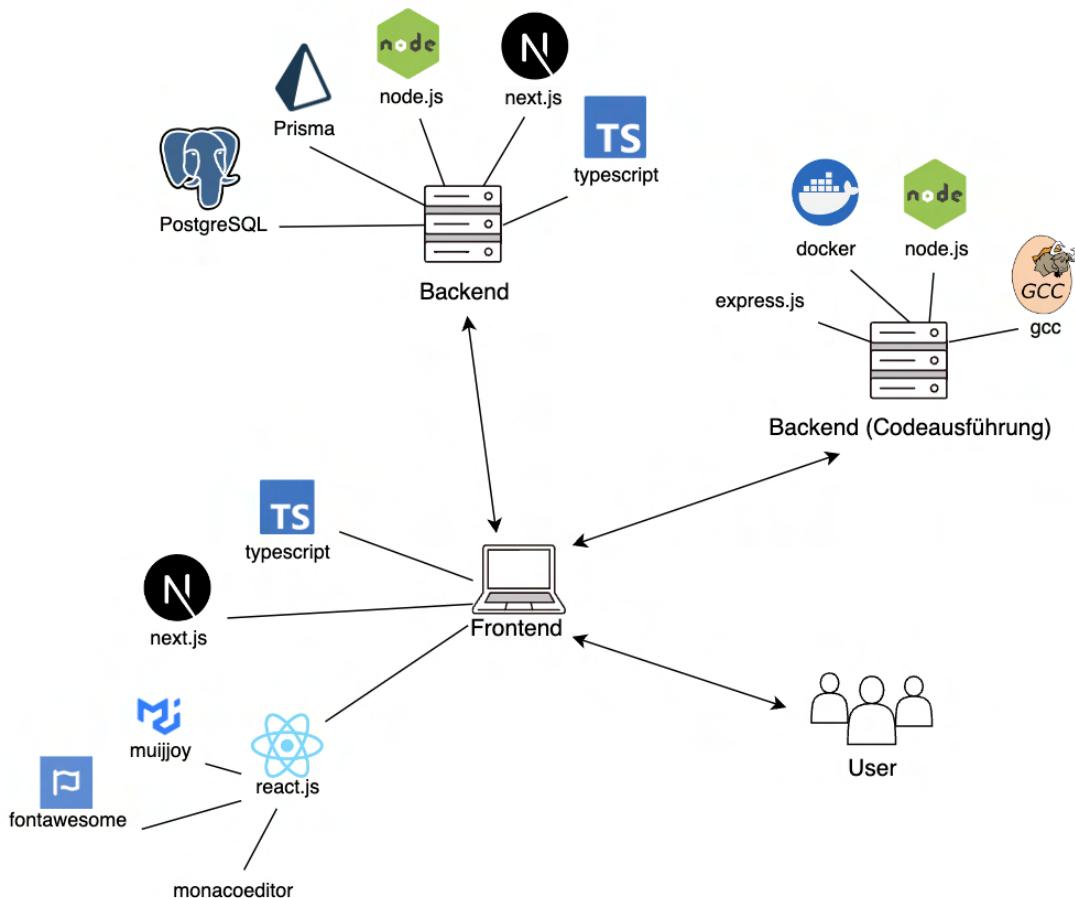


Abb. 6.1.: Anschauung von benutzten Technologien (eigene Darstellung)

¹Frontend: beschreibt den für den User sichtbaren Teil der Website [65]

²Backend: ist der technische Teil der Umsetzung, was vom Nutzer nicht eingesehen werden kann [65]

³Eine Codehostingplattform, die Versionskontrolle und Zusammenarbeit für Team oder Projektarbeiten ermöglicht ([67])

Bevor die individuellen Aspekte des Frontends und Backends betrachtet werden können, muss zunächst die Art und Weise, wie beide miteinander in Verbindung stehen berücksichtigt werden. Im Verlauf dieses Projekts wurden verschiedene Tools und Technologien eingesetzt (siehe Abbildung 6.1). Die Entwicklung fand in der Entwicklungsumgebung Visual Studio Code ([68]) statt. Dabei wurde das Frontend mithilfe von Next.js entwickelt, einer Erweiterung für React, die es ermöglicht das Frontend und Backend einer Applikation zu entwickeln ([69]). Es wurde zudem TypeScript, eine Erweiterung von JavaScript verwendet, wodurch eine moderne und strukturierte Herangehensweise an die Entwicklung ermöglicht wurde ([70]). Für die Plattform wurde ein Backend mit zwei Teilen entwickelt. Beim ersten handelt es sich um einen API-Server (siehe 6.4.1), der zuständig für die Benutzerdaten, die Kommunikation mit der Datenbank und dem Speichern des Fortschritts ist. Das zweite ist der Ausführungs-Server (siehe 6.4.2), der dafür zuständig ist, den von dem Benutzer geschriebenen C++ Code auszuführen und dessen Ergebnisse zurück an das Frontend zu schicken. Der C++ Code wird in einem isolierten Docker Container ausgeführt, um vor etwaigen Angriffen zu schützen.

6.2. Frontend

In diesem Unterkapitel wird das Frontend der Website, die verschiedenen Lernkurse und Umsetzung betrachtet. Der Code wurde unter Verwendung von React entwickelt, was die Erstellung und Wiederverwendung von einzelnen Komponenten (siehe 6.3) ermöglicht.

6.2.1. Willkommensseite

Die Willkommensseite dient dazu, den Studierenden über die verschiedene Kurse zu informieren und einen ersten Einblick zu geben (siehe 3.2.3.2). Neben einem Quiz (siehe 4.2.5) und anklickbaren Akkordeons (siehe 6.3.2) beginnt die Reise von Pebble der Schildkröte (siehe 4.2.1) mit einer Vorstellung seinerseits. Des Weiteren erhalten die Studierenden allgemeine Informationen über den Kurs und können sich mit dem Layout (siehe 4.1.1) vertraut machen. Die Umsetzung der Willkommensseite wurde in den Abbildungen 6.2, 6.3 und 6.4 dargestellt.

6. Implementation

The screenshot shows the first part of a welcome page for a C++ course. On the left, there's a sidebar with navigation links: "C++ Einführung", "Algorithmus Teil 1", "Algorithmus Teil 2", and "Algorithmus Teil 3". The main content area has a title "Einführungskurs in C++ mit C-Turtle" and a sub-section "Pebble die Schildkröte". It contains a text message from Pebble, a cartoon character of a green turtle wearing a yellow hard hat, and a poll asking about programming experience. Below that is a section titled "Kursinhalte" with two collapsed items: "Einführungskurs in C++" and "C-Turtle als Einstieg".

Abb. 6.2.: Umsetzung Willkommensseite Teil 1 (eigene Darstellung)

The screenshot shows the second part of the welcome page. It features a sidebar with the same navigation links as the first part. The main content area includes sections for "Kursinhalte" (with expanded items for "Einführungskurs in C++", "C-Turtle als Einstieg", and "Verständnis der Algorithmik"), "Pseudocode zu C++ Code" (with a detailed description), and "Vorgehensweise" (with instructions for beginners, advanced users, and experts). There's also a small illustration of a person working on a computer.

Abb. 6.3.: Umsetzung Willkommensseite Teil 2 (eigene Darstellung)

6. Implementation

Vorgehensweise

- Anfänger/Fortgeschritten: Bearbeite zuerst den C++ Einführungskurs. Anschließend sollst du Algorithmus Teil 1, 2 und 3 bearbeiten.
- Profi: Der C++ Einführungskurs ist für dich optional, aber denk dran, dass es nicht schadet nochmal einen Einstieg zu bekommen. Anschließend sollst du Algorithmus Teil 1, 2 und 3 bearbeiten.

E-Learning Funktionen

- Gamification
- Visuelle Unterstützung durch Bilder
- Interaktive Bausteine
- Highscore-Anzeigen
- Sammelbare/Freischaltbare Elemente

Ziele

- Spaß am Programmieren
- Einfacher Einstieg in C++/Algorithmen
- Förderung der Problemlösungsfähigkeiten
- Entwicklung kreativer Algorithmen

Abb. 6.4.: Umsetzung Willkommensseite Teil 3 (eigene Darstellung)

6.2.2. C++ Einführungskurs

Der C++ Einführungskurs (siehe 3.2.3.1) soll den Studierenden eine allgemeine Einführung in die Programmiersprache C++ geben. Durch den Einsatz von Code-Editoren (siehe 6.3.1) und Akkordeons (siehe 6.3.2) sollen in verschiedenen Kapiteln des Kurses grundlegende Konzepte von C++ mithilfe von Aufgaben erläutert werden (siehe 5.1). Einen Teil der Umsetzung des Kurses kann den Abbildungen 6.5, 6.6 und 6.7 entnommen werden.

C++ Kurs

In diesem Kurs erhältst du erste Einblicke in die Programmiersprache C++. Es ist nicht erforderlich, dass du bereits Programmiererfahrung hast. Dieser Kurs ist für alle gedacht und wurde aus dem Vorlesungsskript des Moduls GdP erstellt. Schließe diesen C++ Kurs ab, um dein Wissen unter Beweis zu stellen. Falls du bei einer Aufgabe Probleme hast, kannst du selbstverständlich jederzeit in dein GdP-Skript schauen oder einen Komiliton/Professor um Hilfe bitten.

Wissenswertes zu Beginn

- Was ist ein Algorithmus?
- Was ist ein Programm?
- Was ist eine Programmiersprache?
- Die Syntax...
- Die Grammatik...
- Die Semantik...

Mein erstes C++ Programm

Abb. 6.5.: Umsetzung C++ Einführungskurs Teil 1 (eigene Darstellung)

6. Implementation

The screenshot shows a C++ tutorial interface with a sidebar menu and several code editor windows.

- Sidebar Menu:**
 - C++ Einführung
 - Wissenswertes ...
 - Mein erstes C++...
 - Kommentar
 - Variablen
 - Datentypen
 - Operatoren
 - Kontrollstrukturen
 - Fallunterscheid...
 - Schleifen
 - Algorithmus Teil 1
 - Algorithmus Teil 2
 - Algorithmus Teil 3
- Mein erstes C++ Programm**: A code editor with the following code:

```

1 // einbinden externer Definitionen (hier std::cout)
2 #include <iostream>
3 int main()
4 {
5     // Ausgabe des Programms
6     std::cout << "Hello World!";
7     return 0; // Rückgabe an das aufrufende Programm
8 } // Ende des Programms

```

The output window shows "Hello World!"
- Kommentar**: A code editor with the following code:

```

1 // Kommentare werden nicht interpretiert, sondern vom Computer überlesen.
2 // Alles, was in einer Zeile nach // folgt ist ein einzeiliger-Kommentar.
3 // Um einen mehrzeiligen-Kommentar zu schreiben, verwendet man /* Kommentar */

```
- Variablen**: A code editor with the following code:

```

1 #include <iostream>

```

The output window shows nothing.

Abb. 6.6.: Umsetzung C++ Einführungskurs Teil 2 (eigene Darstellung)

The screenshot shows a C++ tutorial interface with a sidebar menu and several code editor windows.

- Sidebar Menu:**
 - C++ Einführung
 - Wissenswertes ...
 - Mein erstes C++...
 - Kommentar
 - Variablen
 - Datentypen
 - Operatoren
 - Kontrollstrukturen
 - Fallunterscheid...
 - Schleifen
 - Algorithmus Teil 1
 - Algorithmus Teil 2
 - Algorithmus Teil 3
- Datentypen**: A code editor with the following code:

```

1 #include <iostream>
2 int main()
3 {
4     bool ichKannProgrammieren = false;
5     std::cout << ichKannProgrammieren;
6     return 0;
7 }

```

The output window shows nothing.
- Datentyp - Zahlen (int, double, float)**: A code editor with the following code:

```

1 #include <iostream>
2 int a = 5;
3 double b = 27.3;
4 float c = 3.24;
5 std::cout << "a ist: " << a << ", b ist: " << b << ", c ist: " << c << std::endl;
6
7 return 4;

```

The output window shows nothing.

Abb. 6.7.: Umsetzung C++ Einführungskurs Teil 3 (eigene Darstellung)

6. Implementation

6.2.3. Algorithmus Teil 1

Der Algorithmus Teil 1 ist der erste Abschnitt des Algorithmus-Kurses (siehe 5.2). In diesem Abschnitt wird den Studierenden erklärt, wie sie Pseudocode schreiben können (siehe 5.2.1). Ein Teil der Umsetzung kann den Abbildungen 6.8 und 6.9 entnommen werden.

```

Pseudocode - TK Pizza backen
Wir schauen uns nun gemeinsam ein Pseudocode Beispiel an. Dieses Anwendungsbeispiel hast du definitiv schon mal selber gemacht! Es handelt sich natürlich um das Backen einer Tiefkühl-Pizza (TK-Pizza). Das ist ein sehr schönes Beispiel um dir zu zeigen, wie man einen Algorithmus als Pseudocode schreibt. Alle Schritte, die man beim Backen einer TK-Pizza macht werden nun als Pseudocode Algorithmus geschrieben.

1   öffne die Verpackung der TK-Pizza
2   schalte den Ofen ein
3   lege die TK-Pizza auf ein Backblech
4   schiebe das Backblech in den Ofen
5   warte die Backzeit ab
6   hole die Pizza aus dem Ofen
7   schalte den Ofen aus

Wie du siehst, ist dieser Pseudocode eine Anleitung für eine Person (Bäckerin), die einen Backofen hat und eine TK-Pizza backen möchte. Diesen Pseudocode könnte ein Computer nicht ausführen. Es gibt den Pizzabäcker eine klare, eindeutige, strukturierte und durchführbare Anweisungen. Dieser Algorithmus hat verschiedene Sequenzen, die immer nacheinander abgearbeitet werden. In diesem Pseudocode haben wir aber noch keine Kontrollstrukturen verwendet. Diese sind aber essentiell, für das schreiben eines guten Algorithmus.

Pseudocode - Fallunterscheidung
Du hast im C++ Einführungskurs bereits einiges über Kontrollstrukturen gelernt. Wir versuchen sie nun nochmal im Zusammenhang mit dem Begriff 'Pseudocode' zu betrachten. Falls du noch Schwierigkeiten hast, Kontrollstrukturen zu verstehen, kehre einfach zum Einführungskurs zurück und schau außerdem in deinem GdP Skript nach.

Pseudocode - Fallunterscheidung
Die Fallunterscheidung (Bedingte Verzweigung) sagt einem, dass anhand einer Bedingung/Fall entschieden wird, ob eine bestimmte Anweisung durchgeführt wird oder nicht. In der Programmiersprache C++ sind "if" und "else" notwendigen Schlüsselwörter dafür. Wir erweitern das TK-Pizza Beispiel und schauen uns an, wie man eine Fallunterscheidung in Pseudocode schreibt.

1   öffne die Verpackung der TK-Pizza
2   schalte den Ofen ein
3
4   WENN die TK-Pizza in einer Folie eingewickelt ist
5       Entferne die Folie

```

Abb. 6.8.: Umsetzung Algorithmus Teil 1, Bild 1 (eigene Darstellung)

Dein Pseudocode

W: Vorwärts
D: Rechts drehen
A: Links drehen

Dein Pseudocode

Du hast in diesem Kapitel viel über Algorithmen und Pseudocode gelernt. Nun bist du an der Reihe einen Pseudocode zu schreiben!
Die Aufgabe für deinen Pseudocode lautet:
Schreibe einen Pseudocode, der mir (Pebble die Schildkröte) hilft aus dem Labyrinth zu gelangen.
Denk dran, ich bin nur eine Schildkröte daher kann ich nur folgendes machen:
• nach vorne gehen
• nach links/rechts drehen
• schauen ob ich im Ziel ist
• schauen ob vor mir eine Wand ist
• schauen ob links/rechts von mir eine Wand ist
Teste deinen Pseudocode gerne mit dem Spiel "Pebble-Simulator" durch. Schreibe deinen Pseudocode in dem dafür vorgesehenen Feld.

Dein Pseudocode

Speichern

Abb. 6.9.: Umsetzung Algorithmus Teil 1, Bild 2 (eigene Darstellung)

6. Implementation

6.2.4. Algorithmus Teil 2

Im zweiten Teil des Algorithmus-Kurses lernen die Studierenden Klassen, Methoden und Attribute kennen. Ihnen wird der Code-Editor mit der Schildkröten-Anzeige (siehe 6.3.1.1) gezeigt und sie lernen, wie sie mit kleinen Code-Segmenten Pebble durch das Verwenden von Attributen und Methoden ins Ziel führen können. Ein Teil der Umsetzung wurde in Abbildungen 6.10 und 6.11 dargestellt.

Abb. 6.10.: Umsetzung Algorithmus Teil 2, Bild 1 (eigene Darstellung)

Abb. 6.11.: Umsetzung Algorithmus Teil 2, Bild 2 (eigene Darstellung)

6. Implementation

6.2.5. Algorithmus Teil 3

Der dritte und letzte Teil des Algorithmus-Kurses beschäftigt sich mit der Überführung von Pseudocode in C++ Code (siehe 5.2.3). Die Studierenden sollen außerdem einen Algorithmus schreiben, der Pebble aus verschiedenen Labyrinthen führt. Ein Teil der Umsetzung wurde in Abbildung 6.12 und 6.13 abgebildet.

Guideline

- 1. Verstehre deinen Pseudocode:**
Hast du deinen Pseudocode vollständig verstanden und ist dir die Bedeutung jeder Anweisung klar? Stelle sicher, dass du alle Variablen und ihre Bedeutung kennst.
- 2. Identifiziere Variablen und Datentypen:**
Überprüfe deinen Pseudocode auf verwendete Variablen und weise ihnen Datentypen zu.
- 3. Strukturiere deinen Code:**
Überlege wie du Schleifen, Bedingungen und andere Strukturen deines Pseudocodes in entsprechende C++ Codeblöcke überführen kannst. Schau wie du "if"-Anweisungen, "while"/"for"-Schleifen einsetzen kannst.
- 4. Überführung:**
Schreib mit Hilfe deines erlernten Wissens den Pseudocode in C++ Code um.
- 5. Teste deinen C++ Code:**
Laufe deinen C++ Code im Pebble Simulator durch. Leg deinen C++ Code daneben und schau dir an, ob du alles richtig eingebunden hast.

Dein C++ Code

```
#include "turtle.h"
#include <string.h>
```

SOLANGE Pebble noch nicht im Ziel
WENN rechts von mir keine Wand

Abb. 6.12.: Umsetzung Algorithmus Teil 3, Bild 1 (eigene Darstellung)

```
schwer
```

```
1 #include "turtle.h"
2 #include <stdio.h>
3 #include <iostream>
4
5 int main(){
6     Krome pebble; //Element pebble kann nun aufgerufen werden
7     while (!pebble.imZiel()) { //falls Pebble noch nicht im Ziel
8         // Wenn vorwärts frei ist, gehe vorwärts
9         if (!pebble.isWallRight()) {
10             pebble.turnRight();
11             pebble.moveForward();
12         }
13         // Wenn vorwärts frei ist, gehe vorwärts
14         else if (!pebble.isWallFront()) {
15             pebble.moveForward();
16         }
17         // Wenn rechts blockiert ist, drehe nach links
18         else {
19             pebble.turnLeft();
20         }
21     }
22 }
```

Abb. 6.13.: Umsetzung Algorithmus Teil 3, Bild 2 (eigene Darstellung)

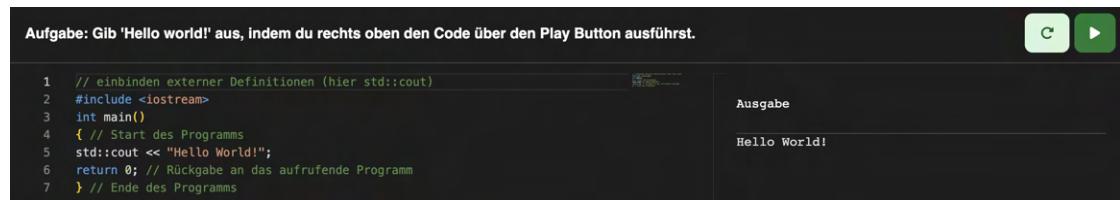
6.3. Einblick in die Programmierung und Komponenten

In diesem Kapitel findet ein Einblick in die Programmierung statt. Der komplette Code kann auf GitHub eingesehen werden([66],[19]).

Es wurden sogenannte Komponenten erstellt, welche es erlauben, die Benutzungsoberfläche in unabhängige und wiederverwendbare Teile aufzubrechen ([71]). Dabei wird jede Komponente isoliert betrachtet ([71]). Komponenten können ein beliebiges Eingabeobjekt akzeptieren und anschließend ein React-Element zurückgeben ([71]). Es werden nun zwei wichtige Komponenten betrachtet und dabei auf deren Programmierung eingegangen.

6.3.1. Code-Editor-Komponente

Der Code-Editor ist mit eines der wichtigsten Bestandteile, da dieser bei jeder Programmieraufgabe angezeigt wird. Hier können die Studierenden ihren Code direkt eingeben und ausführen lassen. Es wurde der Monaco-Editor([72]) eingebunden, einen von Microsoft entwickelter Open-Source Code-Editor, der unter anderem auch in Visual Studio Code verwendet wird. Im Header des Editors steht meistens eine Information zur Aufgabe und zwei Knöpfe am rechten Rand, mit denen die Studierenden den Code ausführen oder die Ausführung abbrechen können. In der linken Hälfte sollen die Studierenden den Code schreiben. Anschließend wird der Code an den Ausführungsserver (siehe 6.4.2) geschickt. Die Rückgabe, das sind Fehlermeldungen oder die Ausgabe, wird auf der rechten Hälfte angezeigt (siehe Abbildung 6.3.1).



Aufgabe: Gib 'Hello world!' aus, indem du rechts oben den Play Button ausführst.

```

1 // einbinden externer Definitionen (hier std::cout)
2 #include <iostream>
3 int main()
4 { // Start des Programms
5     std::cout << "Hello World!";
6     return 0; // Rückgabe an das aufrufende Programm
7 } // Ende des Programms

```

Ausgabe

Hello World!

Abb. 6.14.: Code-Editor ohne TurtleViewer (eigene Darstellung)

Es wird nun ein Codesegment, der die verwendeten Properties (Props) von der Code-Editor-Komponente besitzt genauer untersucht. In React werden Props dazu verwendet, bestimmte Eigenschaften der Komponente zu definieren. Falls die Code-Editor-Komponente verwendet wird, müssen die Props entsprechend ihrer definierten Typen übergeben werden. Die Ausnahme ist, wenn ein Fragezeichen (?) hinter ihren Namen steht. In diesem Fall ist es optional, das Prop zu verwenden([73]). Das Prop “height“, was die Höhe des Code-Editors auf der Website setzt, kann optional ein Wert zugewie-

6. Implementation

sen werden (siehe Abbildung 6.15). Falls nichts übergeben wird, ist der Wert “undefined“. Wie einzelne Komponenten damit umgehen, kann unterschiedlich sein. Im Fall von “height“ im Code-Editor wird der CSS wert “auto“ verwendet.

Es können auch Funktionen wie “onChange“ übergeben werden (siehe Abbildung 6.15). Diese Funktion dient dazu, um Eingaben des Nutzers an die übergeordnete Komponente zurückzugeben, welche sonst keinen Zugriff auf die Daten aus der Code-Editor-Komponente hätte. Im Allgemeinen ist die Verwendung von Props relevant für den Datenaustausch zwischen Komponenten.

```
interface Props {  
    title: string;  
    defaultValue: string;  
    turtle: boolean;  
    labyrinth?: Field[][];  
    height?: string;  
    codeEinAusgabe?: (  
        code_eingabe: string,  
        code_ausgabe: string,  
        imZiel?: boolean  
    ) => void;  
    onChange?: (code: string) => void;  
}
```

Abb. 6.15.: Code-Editor Properties (eigene Darstellung)

Sobald der Ausführungs-Button im Code-Editor (siehe Abbildung 6.14) betätigt wurde, wird der Inhalt des Codes, sowie das Labyrinth an den Ausführungs-Server (siehe 6.4.2) übermittelt. Falls kein Inhalt im Code-Editor vorhanden ist, wird ein leerer String übergeben und wenn kein Labyrinth vorhanden ist, wird ein leeres Array übermittelt (siehe Abbildung 6.16).

```
fetch(server, {  
    method: "POST",  
    body: JSON.stringify({  
        code: editorRef.current ? (editorRef.current as any).getValue() : "",  
        labyrinth: labyrinth ?? [],  
    }),  
})
```

Abb. 6.16.: Code-Editor method POST (eigene Darstellung)

6. Implementation

6.3.1.1. TurtleViewer-Komponente

Der TurtleViewer kann als eine Erweiterung des Code-Editors gesehen werden. Wie in der Abbildung 6.15 zu sehen ist, verfügt die Code-Editor-Komponente über ein Prop namens "turtle", was einen booleschen Wert annimmt. Im weiteren Verlauf des Codes wird festgelegt, dass die TurtleViewer-Komponente nur dann in der Code-Editor-Komponente angezeigt wird, wenn "turtle" den Wert "true" hat (siehe Abbildung 6.17).

```
{turtle && (
  <TurtleViewer
    path={path}
    field={labyrinth ?? []}
    width={400}
    height={400}
    running={isRunning}
    runningDone={() => setIsRunning(false)}
  />
)}
```

Abb. 6.17.: TurtleViewer-Komponente innerhalb Code-Editor (eigene Darstellung)

Ist das der Fall, wird auf der rechten Seite des Code-Editors (siehe 6.3.1) ein Labyrinth mit einer Schildkröte angezeigt. Diese Schildkröte verfolgt nach Ausführung den übermittelten Pfad und zeichnet für jeden durchlaufenen Schritt einen roten Strich in das Labyrinth (siehe Abbildung 6.18).

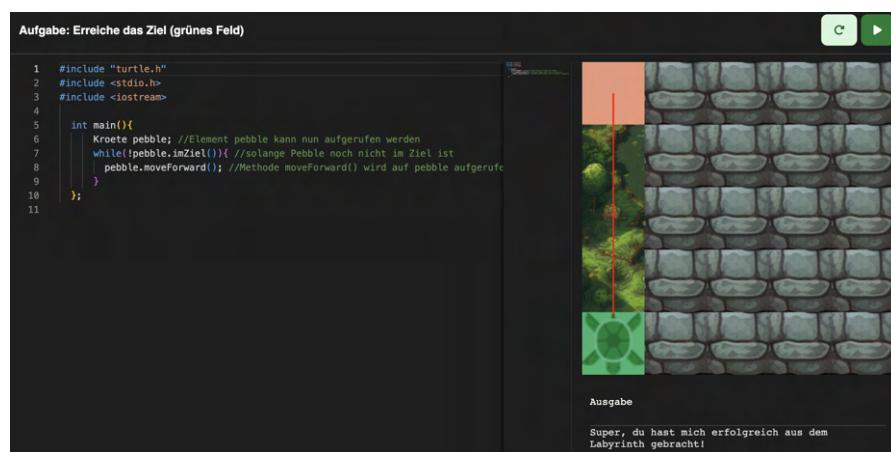


Abb. 6.18.: TurtleViewer-Komponente (eigene Darstellung)

6.3.2. Akkordeons

Die Akkordeon-Komponente bietet die Möglichkeit, ein spezielles, interaktives Element in die Website einzufügen. Wenn die studierende Person auf das Akkordeon klickt, kann es erweitert werden, um zusätzlichen Inhalt anzuzeigen. Darüber hinaus kann überprüft werden, ob das Akkordeon bereits geöffnet wurde. Gegebenenfalls wird auch die Fortschrittsanzeige (siehe 4.6) erhöht. Falls das Akkordeon geschlossen ist, wird ein Pfeil nach unten angezeigt, wenn es offen ist einen Pfeil nach oben. Das kann auch dem Code in Abbildung 6.19 entnommen werden.

```
return (
  <div className={styles.container}>
    <div
      className={styles.ChevronContainer}
      onClick={() => {
        setIsOpen(!isOpen);
        wasClicked && wasClicked();
      }}
    >
      <Typography>{titel}</Typography>
      <FontAwesomeIcon
        icon={isOpen ? faChevronUp : faChevronDown}
        color="#1A7D36"
        height={12}
      />
    </div>
    {isOpen && <Typography>{inhalt}</Typography>}
  </div>
);
```

Abb. 6.19.: Akkordeon-Komponente (eigene Darstellung)

Die Props “titel“ und “inhalt“ werden anschließend übergeben und an den Stellen eingefügt, an denen sie in der Website integriert werden sollen. Falls das Akkordeon von den benutzenden Personen geöffnet wurde, erscheint nun der Inhalt, wie in Abbildung 6.20 dargestellt.



Abb. 6.20.: Offenes und geschlossenes Akkordeon (eigene Darstellung)

6.4. Backend und Server

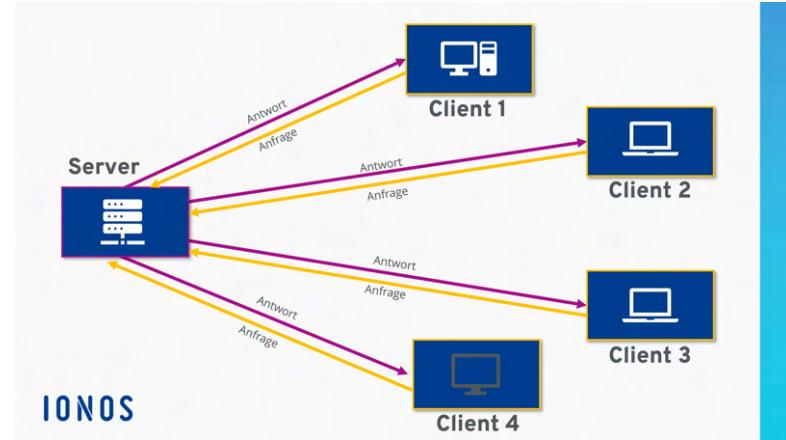


Abb. 6.21.: Server-Client Modell ([74])

In diesem Kapitel wird das Backend und seine Funktionsweise sowie die verschiedenen Server betrachtet. Dabei ist zu beachten, dass der Begriff “Server“ sich sowohl auf einen Computer beziehen kann, der über ein Netzwerk Ressourcen bereitstellt, als auch auf die Software, die auf diesem Computer ausgeführt wird ([74]).

Die Bereitstellung von Server-Diensten basiert auf dem Client-Server Modell (siehe Abbildung 6.21). Dieses Konzept ermöglicht, dass mehrere Nutzer unabhängig voneinander auf die gleichen Dienste zugreifen können ([74]).

6.4.1. API-Server

Es gibt sowohl den API-Server, als auch den Ausführungs-Server (siehe 6.4.2). Im Folgenden wird der API-Server betrachtet. Dabei steht die Abkürzung API für Application Programming Interface oder auch Programmierschnittstelle ([75]). Hierbei handelt es sich um eine Schnittstelle, die es zwei verschiedenen Systemen ermöglicht, miteinander zu kommunizieren ([76]). Wenn eine Software mit einer anderen kommunizieren möchte, erfolgt dies meist über eine API-Schnittstelle ([76]).

Der Begriff “REST⁴“ ist im Zusammenhang mit API von großer Bedeutung. Dieser Begriff wurde von Roy Fielding ([77]) entwickelt, unter der er eine Bauweise von Webanwendungen, die über HTTP kommunizieren ([76]), und zugleich eine Sammlung von Architekturbeschränkungen ([75]) beschreibt. Um im nächsten Kapitel zu verstehen, wie eine REST-API funktioniert, müssen einige der Methoden, die genutzt werden um die

⁴Representational State Transfer ([76])

6. Implementation

Kommunikation zwischen verschiedenen Systemen zu ermöglichen, betrachtet werden. Wenn der Client beispielsweise eine Anfrage mittels der “GET“-Methode an den Server sendet, erhält er die Darstellung der angefragten Ressource zurück ([76]). Alternativ kann der Client mithilfe der “POST“-Methode dem Server eine neue Ressource hinzufügen ([76]).

6.4.1.1. Endpoint Task

Der Fortschritt bei den einzelnen Aufgaben oder auch Tasks, wird in der Datenbank gespeichert. Um vom Frontend auf diese zugreifen zu können, muss ein API Endpoint für Tasks erstellt werden, von dem die API Anfragen empfangen und Antworten senden kann ([78]). Task hat zwei Endpoints. Einen GET-Endpoint “/task“, der alle Tasks des aktuellen Users zurückgibt und einen POST-Endpoint “/task“, der einen neuen Task erstellt, falls der User den Task noch nicht begonnen hat, oder einen bereits vorhanden Task aktualisiert. Die Daten für den Task werden im Request-Body übergeben.

6.4.1.2. Datenbank

```
model User {
    id      String      @id @default(uuid())
    email   String
    vorname String
    nachname String
    Task[]   Task[]
    matrikel Int
    password String      @default("")
}

model Task {
    id      String
    isReviewed Boolean
    userid   String
    user     User        @relation(fields: [userid], references: [id])
    code     String
    complete Boolean      @default(false)

    @@id([id, userid])
}
```

Abb. 6.22.: Datenbank Schema (eigene Darstellung)

Bei der erstellten Datenbank handelt es sich um eine PostgreSQL-Datenbank ([79]). Sie besitzt zwei Tabellen, einerseits Task und andererseits User, welche in einer Relation stehen, die in Abbildung 6.23 dargestellt wurden. Es handelt sich dabei um eine 1:n⁵-

⁵gesprochen: “Eins zu N“

6. Implementation

Beziehung. Ein User kann mehrere Tasks haben, aber ein Task nur jeweils einen User. Die Tabellen und ihre Relationen können in einer Schema Datei angegeben werden, aus der SQL-Statements zur Erstellung der Tabellen generiert werden können. Änderungen im Schema (siehe Abbildung 6.22) können durch sogenannte Migrationen in die Datenbank eingepflegt werden . Für die Erstellung des Schemas wurde Prisma, ein Open-Source ORM⁶ Framework verwendet ([80]). ORM-Frameworks erleichtern die Kommunikation zwischen einer Anwendungssoftware und einer relationalen Datenbank, indem sie Objekte und deren Beziehungen zur Datenbank abbilden und die Abfrage von Daten vereinfachen. Prisma ist darauf ausgerichtet, die Entwicklung von Datenbankanwendungen in Node.js-Umgebungen zu erleichtern ([81]).

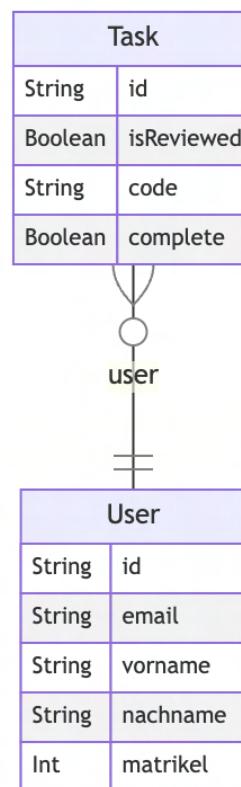


Abb. 6.23.: Datenbank Relationen (eigene Darstellung, erstellt mit ([82]))

⁶Object Relational Mapping

6.4.2. Ausführungs-Server

Sobald die Studierenden im Code-Editor ihren Code ausführen (siehe 6.3.1), werden sowohl der geschriebenen Code, als auch das Labyrinth an den Ausführungs-Server übertragen. Es wird im Port “5236“ gewartet bis die Website die Daten mittels “POST“ an den Server übermittelt. Zuerst wird ein Ordner mit zufälligem Namen erstellt. In diesen ZO⁷ wird eine “main.cpp“ Datei mit dem Code des Studierenden, eine “labyrinth.json“ Datei für das übergebene Labyrinth, sowie die für das Turtle-Programm benötigten Dateien “turtle.cpp“ und “turtle.h“ hineinkopiert. Es wird zudem ein Dockerfile angelegt, das zur Erstellung eines Docker-Images verwendet wird. Ein Dockerfile ist eine Textdatei, die eine Reihe von Anweisungen enthält, die Docker verwendet um ein Docker-Image zu erstellen. Das erstellte Image kann dann verwendet werden, um Docker-Container zu starten. Das Basisimage für dieses Docker-Image ist die neueste Version des GCC-Images. Das bedeutet, dass das Image auf einem Linux-System mit dem GCC⁸ als Basis, aufgebaut wird. In dem Dockerfile wird auch das Shell-Skript “run.sh“ innerhalb des Containers ausführbar gemacht. Das Skript erhält dadurch die Berechtigung, ausgeführt zu werden. Dieses Dockerfile wird auch in den ZO kopiert. Diese Erklärung kann den Code in Abbildung 6.24 entnommen werden.

```
const Dockerfile = `# Choose a base image that supports C++
FROM gcc:latest

# Copy the C++ executable to the container
ADD . /

# Set the working directory
WORKDIR /

RUN chmod +x run.sh

# Set the entry point to run the executable
CMD ["./run.sh"]`;
```

Abb. 6.24.: Dockerfile (eigene Darstellung)

Die Bash-Shell-Skriptdatei “run.sh“ mit dem Inhalt der in “RunScript“ definiert wird, hat die Aufgabe, den C++ Compiler “g++“ zu verwenden, um die Dateien “main.cpp“ und “turtle.cpp“ zu kompilieren und das ausführbare Programm als “Turtleprogramm“ zu erstellen. Dabei wird die Ausgabe in “compile.txt“ gespeichert. Die “2>&1“-Syntax bewirkt, dass sowohl die Standardausgabe, als auch Fehlermeldungen in die “compile.txt“-

⁷Abkürzung für “Ordner mit zufälligem Namen“

⁸GNU Compiler Collection

6. Implementation

Datei geschrieben werden. Das kompilierte Programm “Turtleprogramm“ wird anschließend ausgeführt und die Ausgabe wird in die Datei “turtleprogramm.txt“ geschrieben. Der Inhalt der Datei “kroeten_path.json“ wird in die Variable “path“ geladen. Schließlich werden die Ergebnisse in der Reihenfolge, in der sie generiert wurden, auf der Konsole ausgegeben. Zuerst der Kompilierungsoutput, dann der Ausgabeoutput des Programms und zum Schluss der Inhalt von “path“. Dieses Skript wird in den ZO unter den Namen “run.sh“ kopiert. Diese Erklärung kann den Code in Abbildung 6.25 entnommen werden.

```
| const RunScript = `#!/bin/bash
#2>&1 nimmt auch Fehlermeldungen mit
g++ main.cpp turtle.cpp -o Turtleprogramm -std=c++17 > compile.txt 2>&1
compile=$(cat compile.txt)

./Turtleprogramm > turtleprogramm.txt 2>&1
output=$(cat turtleprogramm.txt)

path=$(cat kroeten_path.json)

echo $compile
echo -----
echo $output
echo -----
echo $path
`;
```

Abb. 6.25.: RunScript (eigene Darstellung)

Sobald alle wichtigen Dateien in ZO kopiert wurden, wird ein Docker-Image aus dem Dockerfile im aktuellen Verzeichnis erstellt. Mit Hilfe der “-t“-Option erhält das Image den Namen des ZOs. Als nächstes wird ein Docker-Container mit dem zuvor erstellten Image gestartet. Dieser Container führt das in dem Dockerfile definierte Shell-Skript “run.sh“ aus.

```
try {
    // Kompiliere den Code im Docker
    execSync(`cd ${folder} && docker build -t ${folder} .`);
    // Führe den Code aus
    programm_output = execSync(
        `cd ${folder} && docker run ${folder}`
    ).toString();
} catch (err) {
    console.log(err);
}
```

Abb. 6.26.: Ausschnitt 1 des Ausführungs-Servers (eigene Darstellung)

6. Implementation

Die Ausgabe des ausgeführten Skripts wird in die Variable “programm_output“ geschrieben, wo es zuvor in einen String umgewandelt wurde. Wenn während des Versuchs, den Code im Docker auszuführen oder zu kompilieren, ein Fehler auftritt, wird dieser im catch-Block behandelt und auf der Konsole mit “console.log“ ausgegeben. Den Code zu dieser Erklärung kann in Abbildung 6.26 eingesehen werden.

Als nächstes wird “programm_output“ verarbeitet. Der Inhalt der Variable soll in drei separate Teile aufgeteilt werden. Es handelt sich dabei um den Kompilierungsoutput (compile) die Ausgabe (output) und den Schildkröten-Pfad (stringpath). Die Methode “split“ teilt “programm_output“ in ein Array mit drei Elementen. Mit Hilfe von “trim“ können führende/ abschließende Leerzeichen und Zeilenumbrüche entfernt werden. Es wird anschließend geprüft, ob “stringpath“ einen Inhalt hat und falls er nicht leer ist, wird “JSON.parse()“ verwendet, um den String in ein JavaScript-Objekt umzuwandeln, das dann in der Konstante “path“ abgespeichert wird. Wenn es ein Labyrinth mit Anweisungen für die Schildkröte gibt, wird der Weg, den die Schildkröte zurückgelegt hat, in “stringpath“ erfasst. Diese Erklärung kann dem Code in Abbildung 6.27 entnommen werden.

```
const [compile, output, stringpath] = programm_output
  .split("-----")
  .map(x => x.trim());
const path = stringpath ? JSON.parse(stringpath) : [];

// Delete the folder
execSync(`rm -rf ${folder}`);
```

Abb. 6.27.: Ausschnitt 2 des Ausführungs-Servers (eigene Darstellung)

Danach wird das Verzeichnis gelöscht, um Ressourcen freizugeben. Zum Schluss wird mit Hilfe der “res“-Variable, “compile“, “output“ und “path“ als HTTP-Antwort an die Webseite gesendet.

6.5. Ausführung und Benutzung

Die momentane Ausführung des Clients und der Website funktioniert über einen Server mit einer IP⁹, die von der HTWK gestellt wurde. Damit die Studierenden dorthin zugreifen können, müssen sie entweder im WLAN ([83]) oder über eine VPN Verbindung ([84]) mit dem Hochschulnetz der HTWK Leipzig verbunden sein. Anschließend müssen sie auf die richtige Turtle-Website ([85]) zugreifen und dort die Kurse absolvieren.

6.5.1. HTWK-Server aufsetzen

Es wird im Folgenden kurz gezeigt, wie der Server der HTWK aufgesetzt wurde. Eine ausführlichere Dokumentation befindet sich im Anhang (siehe A.3).

1. Verbindung mit VPN/WLAN der Hochschule ([84], [83])
2. Screen ([86]) installieren
3. Docker installieren ([87])
4. Datenbank einrichten ([88])
5. Node ([89]) installieren
6. Git-Repository ([67]) clonen
7. In den richtigen Ordner wechseln und dann die Website und den Server starten

⁹Internet Protokoll

7. Ergebnisse

In diesem Abschnitt werden abschließend alle Ergebnisse der Bachelorarbeit zusammengefasst und erläutert, die in den vorherigen Kapiteln erarbeitet wurden. Es wurde eine Lernplattform erstellt, auf der Studierende der HTWK Leipzig, die das Modul GdP besuchen, die Möglichkeit haben, verschiedene Kurse zu absolvieren.

Die in Kapitel 3.2.3 dargestellten Anforderungen wurden erfolgreich umgesetzt. Ein C++ Einführungskurs wurde auf Basis des GdP-Skripts und einer Turtle-Umgebung erstellt, der den Studierenden mithilfe verschiedener Übungen einen klaren Einblick in die Programmiersprache C++ vermittelt (siehe 5.1).

Es wurde ein weiterer Kurs entwickelt, um den Studierenden das Erstellen von Algorithmen näher zu bringen. Dieser Kurs wurde in verschiedene Abschnitte unterteilt, um den Lernprozess zu strukturieren und zu vereinfachen (siehe 5.2).

Nach Abschluss beider Kurse sind die Studierenden in der Lage, komplexe Probleme zu analysieren und Lösungsansätze zu entwickeln. Sie haben nicht nur ein allgemeines Verständnis der Programmiersprache C++, sondern sind auch in der Lage, sich mit Algorithmen auseinanderzusetzen, was eine wichtige Fähigkeit für ihr weiteres Studium darstellt. Bei der Umsetzung wurde auf die Einbindung von E-Learning Aspekten und die Integration multimedialer Inhalte geachtet (siehe 4). Dies ermöglicht den Studierenden ein interaktives und ansprechendes Lernerlebnis und hilft die Lehrinhalte effektiver und anschaulicher zu vermitteln.

8. Fazit und Ausblick

E-Learning wird auch zukünftig, vor allem in der Hochschullehre, ein präsentes Thema sein ([90]). Daher ist es von großer Bedeutung, das volle Potenzial des E-Learnings auszuschöpfen und kontinuierlich E-Learning Angebote zu fördern. Dieses Vorhaben wird von verschiedenen Initiativen, wie auch der sächsischen E-Learning Landesinitiative unterstützt ([91]). Im Rahmen dieser Bachelorarbeit wurde ein bedeutender Beitrag zur Weiterentwicklung des E-Learning Angebots der HTWK Leipzig geleistet.

Es ist geplant, dass der erstellte Kurs erstmals im Wintersemester 2023/24 eingesetzt wird. Dafür müssen noch weitere Diskussionen mit dem Modulbeauftragten des Moduls GdP abgehalten werden, um gegebenenfalls kleine Anpassungen vorzunehmen. Die Verwendung der erstellten Website soll im jetzigen Lehrplan des GdP-Moduls aufgenommen werden und ermöglicht es den Studierenden, eigene Algorithmen zu schreiben und zudem ihre im Skript erlernten Fähigkeiten auf praktische Weise zu testen und anzuwenden. E-Learning Angebote sollen nicht als Ersatz für herkömmliche Hochschullehre, sondern vielmehr als wertvolle Ergänzung betrachtet werden ([90]). Die entwickelten Kurse sind also eine sinnvolle Erweiterung des herkömmlichen Lernmaterials von GdP.

Die nächsten Schritte in diesem Projekt umfassen zunächst weitere Anpassungen und das Hinzufügen von neuen Aufgaben, Erklärungen und Übungen, um den Kursinhalt kontinuierlich zu verbessern und zu erweitern. Darüber hinaus besteht weiterer Entwicklungsbedarf im Bereich des Anmeldesystems und der Implementierung einer Peer-Review-Funktion. Die Einführung einer Peer-Review bietet den Studierenden vielfältige Lernvorteile und ermöglicht es ihnen, wertvolles Feedback von ihren Mitstudierenden zu erhalten ([92]). Das Grundgerüst des Projekts steht bereits, und es gibt zahlreiche Möglichkeiten, es in Zukunft zu erweitern und viele neue Ideen umzusetzen.

Abschließend ist zu betonen, dass die im Rahmen dieser Bachelorarbeit erstellte Website und die entwickelten Kurse auch zukünftig durch ihre agile Umsetzung und Gestaltung weiterentwickelt und genutzt werden können. Die kontinuierliche Anpassung und Erweiterung tragen dazu bei, den Lernprozess der Studierenden zu verbessern und die Qualität des E-Learning Angebots an der HTWK Leipzig zu steigern.

Abbildungsverzeichnis

1.1.	Projektabstimmung und Austausch (eigene Darstellung)	3
1.2.	Essentielle und Optionale Anforderungen (eigene Darstellung)	3
2.1.	Idealtypische Architektur eines Learning Management Systems ([20]) . . .	7
2.2.	Lerntheorien und Aufgaben digitaler Medien ([22])	8
2.3.	HEAD Wheel, Gaisch und Aichinger ([38])	11
3.1.	Mit Turtle erstellte Grafik ([41])	13
3.2.	Konzeptidee für den C++ Einführungskurs (eigene Darstellung)	15
3.3.	Konzeptidee 1 für die Willkommens-Seite (eigene Darstellung)	16
3.4.	Konzeptidee 2 für die Willkommens-Seite (eigene Darstellung)	16
3.5.	Konzeptidee für die Erstellung von Pseudocode (eigene Darstellung) . . .	17
3.6.	Konzeptidee 1 für die Gamification (eigene Darstellung)	18
3.7.	Konzeptidee 2 für die Gamification (eigene Darstellung)	18
3.8.	Konzeptidee für die Erstellung eines Algorithmus (eigene Darstellung) .	19
4.1.	Screendesign ([51])	21
4.2.	Standardlayout: typische Bereiche einer Website ([51])	22
4.3.	Footer der Turtle-Website (eigene Darstellung)	23
4.4.	Verwendete Farbpalette ([55])	24
4.5.	Das Spiel Pebble-Simulator mit Erklärungen (eigene Darstellung) . . .	26
4.6.	Fortschrittsanzeige bevor und nach Lösung der Aufgabe (eigene Darstellung)	27
4.7.	Zertifikat nach dem Kursabschluss (eigene Darstellung)	28
4.8.	Interaktive Umfrage (eigene Darstellung)	29
4.9.	Vorgehensweise für den Kurs (eigene Darstellung)	30
6.1.	Anschauung von benutzten Technologien (eigene Darstellung)	35
6.2.	Umsetzung Willkommenseite Teil 1 (eigene Darstellung)	37
6.3.	Umsetzung Willkommenseite Teil 2 (eigene Darstellung)	37
6.4.	Umsetzung Willkommenseite Teil 3 (eigene Darstellung)	38
6.5.	Umsetzung C++ Einführungskurs Teil 1 (eigene Darstellung)	38
6.6.	Umsetzung C++ Einführungskurs Teil 2 (eigene Darstellung)	39
6.7.	Umsetzung C++ Einführungskurs Teil 3 (eigene Darstellung)	39
6.8.	Umsetzung Algorithmus Teil 1, Bild 1 (eigene Darstellung)	40
6.9.	Umsetzung Algorithmus Teil 1, Bild 2 (eigene Darstellung)	40
6.10.	Umsetzung Algorithmus Teil 2, Bild 1 (eigene Darstellung)	41

Abbildungsverzeichnis

6.11. Umsetzung Algorithmus Teil 2, Bild 2 (eigene Darstellung)	41
6.12. Umsetzung Algorithmus Teil 3, Bild 1 (eigene Darstellung)	42
6.13. Umsetzung Algorithmus Teil 3, Bild 2 (eigene Darstellung)	42
6.14. Code-Editor ohne TurtleViewer (eigene Darstellung)	43
6.15. Code-Editor Properties (eigene Darstellung)	44
6.16. Code-Editor method POST (eigene Darstellung)	44
6.17. TurtleViewer-Komponente innerhalb Code-Editor (eigene Darstellung) . .	45
6.18. TurtleViewer-Komponente (eigene Darstellung)	45
6.19. Akkordeon-Komponente (eigene Darstellung)	46
6.20. Offenes und geschlossenes Akkordeon (eigene Darstellung)	46
6.21. Server-Client Modell ([74])	47
6.22. Datenbank Schema (eigene Darstellung)	48
6.23. Datenbank Relationen (eigene Darstellung, erstellt mit ([82]))	49
6.24. Dockerfile (eigene Darstellung)	50
6.25. RunScript (eigene Darstellung)	51
6.26. Ausschnitt 1 des Ausführungs-Servers (eigene Darstellung)	51
6.27. Ausschnitt 2 des Ausführungs-Servers (eigene Darstellung)	52

A. Anhang

A.1. Modulbeschreibung des Moduls GdP

Hierbei handelt es sich um den offiziellen Modulplan von GdP ([39]).

C963 – Grundlagen der Programmierung



Modul	Grundlagen der Programmierung Introduction to Programming
Modulnummer	C963 Version: 1
Fakultät	FIM-INF: Informatikstudiengänge - Fakultät Informatik und Medien
Niveau	Bachelor
Dauer	2 Semester
Turnus	Wintersemester
Modulverantwortliche	Prof. Dr. rer. nat. Mario Hlawitschka mario.hlawitschka@htwk-leipzig.de
Dozierende	Prof. Dr. rer. nat. Mario Hlawitschka mario.hlawitschka@htwk-leipzig.de
Sprache(n)	Deutsch
ECTS-Leistungspunkte	7 ECTS-Punkte
Workload	210 Stunden
Lehrveranstaltungen	4.50 SWS (2.50 SWS Vorlesung 2 SWS Praktikum)
Selbststudienzeit	147 Stunden 40 Stunden Vorbereitung Lehrveranstaltung 30 Stunden E-Learning 30 Stunden Selbststudium 40 Stunden Vorbereitung Prüfung 7 Stunden Sonstiges
Prüfungsvorleistung(en)	Keine
Prüfungsleistung(en)	Prüfung mündliches Fachgespräch Prüfungsduer: 20 Minuten Wichtung: 100%
Lehr- und Lernformen	- Programmierübungen in den Seminaren - Einsatz von e-Learning mit automatisierten Tests zur Eigenkontrolle
Medienform	keine Angabe
Lehrinhalte/Gliederung	- Einführung in die Programmierung und Hardware - Begriff des Algorithmus - Programmablaufpläne und Struktogramme - Imperative Programmierung - Kontrollstrukturen - Unterprogramme - Objektorientiertes Programmieren - Verwendung von objektorientierten Datenstrukturen - Vererbung sowie Schnittstellen und Klassen als deren Implementierungen - Ausnahmebehandlung - Vererbung - Grundlagen des Umgangs mit Dateien und Speicher

A. Anhang

Qualifikationsziele	Die Studenten kennen und verstehen Syntax und Semantik der Programmiersprachen C++ und Java. Sie sind in der Lage, formale und textuelle Beschreibungen von einfachen Algorithmen in kleine Programme gemäß des imperativen und objektorientierten Programmierparadigmas umzusetzen sowie einfache Probleme eigenständig zu lösen. Sie kennen Grundlagen der Objektorientiertheit, können Objekte identifizieren und als Klassen implementieren.
Zulassungsvoraussetzung	Keine
Empfohlene Voraussetzungen	Keine
Literaturhinweise	- U. Breymann: „Der C++ Programmierer“, Hanser, 2015. - B. Stroustrup: „Die C++ Programmiersprache“, Hanser, 2015.
Aktuelle Lehrressourcen	keine
Hinweise	Keine Angabe
Verwendbarkeit	Informatik Bachelor (20INB) Pflichtmodul Medieninformatik Bachelor (20MIB) Pflichtmodul Medieninformatik Bachelor Studienrichtung Bibliotheksinformatik (20MIB-BI) Pflichtmodul
Link zu Kurs/Lernressourcen im OPAL/Moodle/etc.	

A. Anhang

A.2. Inhaltliche Organisation GdP

Hierbei handelt es sich um die inhaltliche Organisation des Moduls GdP ([93]).

1. Woche:

- Organisatorisches
- Definitionen
- Was ist ein Algorithmus?
- Was ist ein Programm?

Seminar: Einführung in die Rechnerpools, unsere Arbeitsumgebung.

2. Woche:

- Programmablaufpläne
- Struktogramme

Seminar: Entwickeln von Programmen mit Hilfe von PAP und Struktogrammen

3. Woche:

- Die C++ Programmiersprache
- Der Compiler
- C++ Variablendeklarationen, Zuweisungen

Seminar: Überführen der Programme vom PAP zu Struktogramm zu Programm in C++

4. Woche:

- Bezeichner und Datentypen
- Kontrollstrukturen (Schleifen, Fallunterscheidungen)

Seminar: Programme mit Zykeln

5. Woche:

- Funktionen und Funktionsaufrufe

Seminar: Fortsetzung von Woche 4, Ergänzung von Funktionsaufrufen

6. Woche:

- Speicheraufbau des Rechners, insbesondere der Stapspeicher
- Rekursion

Seminar: Fortsetzung Übungsaufgabe zu Programmen mit Zeichenketten als Felder

A. Anhang

7. Woche:

- Klassische Probleme und deren rekursive Lösung

8. Woche:

- Felder in C++
- Call by Value, Call by Reference

Seminar: Übungsaufgabe zu Programmen mit Zeichenketten als Felder

9. Woche:

- Mehrdimensionale Felder

10. Woche:

- Objekte und Klassendefinitionen
- Polymorphismus, Vererbung, Klassenhierarchien

11. Woche:

- Speicheraufbau des Rechners: Der Heap
- Zeiger - unbenannte Variablen, automatische Zeiger

12. Woche:

- Datenstrukturen: Verkettete Listen

13. Woche:

- Ausblick: Zykel neu gedacht, Anonyme Funktionen

14. Woche:

- Ausblick: Templates, Template-Metaprogramming

A.3. Bash Skript - Server

Hierbei handelt es sich um die Bash-Skripte für den Server.

```
Bash Skript – Server neu aufsetzen
Verbindung mit VPN/WLAN der Hochschule mit den Zugangsdaten
$ ssh name@ip
Installiere Screen
$ sudo apt update
$ sudo apt install screen
Docker installieren
$ sudo apt install apt-transport-https ca-certificates curl software-
properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-
key add -
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu focal stable"
$ apt-cache policy docker-ce
$ sudo apt install docker-ce
$ sudo systemctl status docker
Datenbank einrichten
$ sudo docker run --name tutledb -e POSTGRES_PASSWORD=***** -d postgres
Node installieren
$ sudo apt install nodejs
Git Repository clonen
$ git clone https://github.com/mirellalms/Turtle_Website.git
$ git clone https://github.com/mirellalms/Turtle_Server.git
Anpassungen in .env vornehmen
$ cd Turtle_Website
$ nano .env
DATABASE_URL="postgresql://postgres:*****@localhost:5432/tutledb?schema=publi
c"
CODE_SERVER="http://localhost:5236/run"
NEXTAUTH_URL=http://ip:3000/
NEXTAUTH_SECRET=*****
Anpassungen in CodeEditor.tsx vornehmen
$ nano src/components/CodeEditor.tsx
const server = "http://ip:5236/run";
Sessions umbenennen
$ screen
$ screen -S CodeServer
Server starten
$ screen -r Server
$ cd Turtle_Server
$ npm i
$ sudo node index.js (Shift+A+D)
Website starten
$ screen -r Website
$ cd Turtle_Website
$ npm i
$ npx prisma migrate deploy
$ npx prisma generate
$ sudo npm run dev (Shift+A+D)
```

Bash Skript – Anpassungen

Verbindung mit VPN/WLAN der Hochschule mit den Zugangsdaten

```
$ ssh name@ip
```

Git

```
$ cd Turtle_Server  
$ git pull  
$ cd ..  
$ cd Turtle_Website  
$ git pull  
$ cd ..
```

Screen Sessions löschen

```
$ killall screen
```

Anpassungen in .env vornehmen

```
$ cd Turtle_Website  
$ nano .env  
DATABASE_URL="postgresql://postgres:*****@localhost:5432/turtledb?schema=public"  
CODE_SERVER="http://localhost:5236/run"  
NEXTAUTH_URL=http://ip:3000/  
NEXTAUTH_SECRET= *****
```

Anpassungen in CodeEditor.tsx vornehmen

```
$ nano src/components/CodeEditor.tsx  
const server = "http://ip:5236/run";
```

Sessions umbenennen

```
$ screen  
$ screen -S CodeServer  
$ screen -S Website  
$ screen -ls
```

Server starten

```
$ screen -r CodeServer  
$ cd Turtle_Server  
$ npm i  
$ sudo node index.js (Shift+A+D)
```

Website starten

```
$ screen -r Website  
$ cd Turtle_Website  
$ npm i  
$ npx prisma migrate deploy  
$ npx prisma generate  
$ sudo npm run dev (Shift+A+D)
```

Prisma

```
$ screen -r Prisma  
$ cd Turtle_Website  
$ npx prisma studio (Shift+A+D)
```

Literaturverzeichnis

- [1] Jesse Walker. C-Turtle. <https://github.com/walkerje/C-Turtle>. Accessed: 2023-08-17.
- [2] Patricia Arnold, Lars Kilian, Anne Thilloesen, and Gerhard Zimmer. *Handbuch E-Learning: Lehren und Lernen mit digitalen Medien*. utb, 2013. Accessed: 2023-08-08.
- [3] Duden. Personal Computer. <https://www.duden.de/node/110049/revision/1455145>. Accessed: 2023-08-08.
- [4] Statistisches Bundesamt, editor. *Daten aus den Laufenden Wirtschaftsrechnungen (LWR) zur Ausstattung privater Haushalte mit Informationstechnik*. Statistisches Bundesamt, 2022. Accessed: 2023-08-08.
- [5] Hochschule für Technik Wirtschaft und Kultur Leipzig. Technik ausleihen und zurückgeben. <https://bibliothek.htwk-leipzig.de/ausleihe-und-nutzung/technik-ausleihen-und-zurueckgeben>. Accessed: 2023-08-08.
- [6] Hochschule für Technik Wirtschaft und Kultur Leipzig. Computerarbeitsräume (1. OG). <https://bibliothek.htwk-leipzig.de/en/ausleihe-und-nutzung/vor-ort-arbeiten/computerarbeitsraeume>. Accessed: 2023-08-08.
- [7] Bundesministerium für wirtschaftliche Zusammenarbeit und Entwicklung. Organisation für wirtschaftliche Zusammenarbeit und Entwicklung. [https://www.bmz.de/de/ministerium/internationale-organisationen/oecd#:~:text=Die%20Organisation%20f%C3%BCr%20wirtschaftliche%20Zusammenarbeit,Development%20\)%20gibt%20es%20seit%201961](https://www.bmz.de/de/ministerium/internationale-organisationen/oecd#:~:text=Die%20Organisation%20f%C3%BCr%20wirtschaftliche%20Zusammenarbeit,Development%20)%20gibt%20es%20seit%201961). Accessed: 2023-08-08.
- [8] Bundesministerium für Wirtschaft und Energie. *Digitalisierung in Deutschland – Lehren aus der Corona-Krise Gutachten des Wissenschaftlichen Beirats beim Bundesministerium für Wirtschaft und Energie (BMWi)*. BMWi, 2021. Accessed: 2023-08-08.
- [9] Turtlegrafik. https://python-online.ch/turtle.php?inhalt_links=turtle/navigation.inc.php&inhalt_mitte=turtle/turtle.inc.php. Accessed: 2023-08-17.
- [10] Dr. Jürgen Fleig. *Was Motivation ist und wie Motivation erklärt wird*. business-wissen, 2021. Accessed: 2023-08-08.

Literaturverzeichnis

- [11] Prof. Dr. Lisa Marie Warner. *Selbstwirksamkeitserwartung*. Dorsch Lexikon der Psychologie, 2022. Accessed: 2023-08-08.
- [12] Eva Schuepbach, Urs Guggenbühl, Cornelia Krehl, Heinz Siegenthaler, and Ruth Kaufmann-Hayoz. *Didaktischer Leitfaden für E-Learning*. hep Verlag Bern, 2020. Accessed: 2023-08-09.
- [13] Sebastian Becker Jana Brehmer. *E-Learning ... ein neues Qualitätsmerkmal der Lehre?* Georg-August-Universität Göttingen, 2017. Accessed: 2023-08-09.
- [14] Steve Wheeler. *e-Learning and Digital Learning*. Springer US, Boston, MA, 2012. Accessed: 2023-08-11.
- [15] Daniel Otto and Sara Becker. *E-Learning and Sustainable Development*. 2019. Accessed: 2023-08-11.
- [16] Michael Kerres. *Mediendidaktik, Konzeption und Entwicklung mediengestützter Lernangebote*. Oldenbourg Wissenschaftsverlag, München, 2013. Accessed: 2023-08-11.
- [17] Kleinmann Bernd and Wannemacher Klaus. *E-Learning an deutschen Hochschulen*. HIS GmbH Hannover, 2004. https://his-he.de/fileadmin/user_upload/Publikationen/Projektberichte_alte_Website/Hochschulplanung/hp165.pdf .Accessed: 2023-08-15.
- [18] Sonja Klante. Was sind Lernmanagement-Systeme? [https://wb-web.de/material/medien/was-sind-lernmanagement-systeme-1.html#:~:text=Lernmanagement%2DSysteme%20\(LMS\)%20wurden,Kommunikation%20zwischen%20Lernenden%20und%20Lehrenden](https://wb-web.de/material/medien/was-sind-lernmanagement-systeme-1.html#:~:text=Lernmanagement%2DSysteme%20(LMS)%20wurden,Kommunikation%20zwischen%20Lernenden%20und%20Lehrenden). Accessed: 2023-09-07.
- [19] Mirella Willems. Turtle Website Github. https://github.com/mirellawlms/Turtle_Website. Accessed: 2023-09-04.
- [20] Rolf Schulmeister. *Lernplattformen für das virtuelle Lernen, Evaluation und Didaktik*. Oldenbourg Wissenschaftsverlag, Berlin, Boston, 2005. Accessed:2023-08-15.
- [21] Martin Ebner, Sandra Schön, and Walther Nagler. Das themenfeld „lernen und ehren mit technologien “. *Lehrbuch für Lernen und Lehren mit Technologien: 2. Auflage (2013)*, page 11, 2013. Accessed:2023-08-16.
- [22] Linda Grogorick and Susanne Robra-Bissantz. Digitales lernen und lehren: Führt corona zu einer zeitgemäßen bildung?digital learning and teaching: Does corona

Literaturverzeichnis

- lead to modern education? *HMD Praxis der Wirtschaftsinformatik*, 2021. Accessed: 2023-08-11.
- [23] e teaching.org. Lerntheorien. <https://www.e-teaching.org/didaktik/theorie/lerntheorie>, 2016. Accessed: 2023-08-10.
- [24] Grotlüschen, Anke and Pätzold, Henning . *Lerntheorien: in der Erwachsenen-und Weiterbildung*. wbv Publikation, Stuttgart, Deutschland, 1. auflage edition, 2020. Accessed: 2023-08-10.
- [25] Susanne Meir. Didaktischer Hintergrund Lerntheorien. https://lehrerfortbildung-bw.de/st_digital/moodle/01_praxis/einfuehrung/material/2_meir_9-19.pdf. Accessed: 2023-08-10.
- [26] Prof. Klaus Hering. E-Learning. <https://bildungspotrait.sachsen.de/opal/auth/RepositoryEntry/34750496768?5>. Accessed: 2023-08-10.
- [27] Sebastian Höhne. Konstruktivismus. <http://www.lernpsychologie.net/lerntheorien/konstruktivismus>. Accessed: 2023-09-03.
- [28] George Siemens. Connectivism:a learning theory for the digital age. http://www.itdl.org/Journal/Jan_05/article01.htm. Accessed: 2023-08-10.
- [29] Hochschule für Soziale Arbeit FHNW. Konnektivismus kurz erklärt. fhnw.ch/plattformen/oersozialarbeit/connectivism.php. Accessed: 2023-08-10.
- [30] Prof. Dr. Oliver Bendel. Gamification. <https://wirtschaftslexikon.gabler.de/definition/gamification-53874>. Accessed: 2023-08-29.
- [31] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. *From Game Design Elements to Gamefulness: Defining Gamification*. 2011. Accessed: 2023-09-25.
- [32] Daniel Tolks and Michael Sailer. *Gamification als didaktisches Mittel in der Hochschulbildung*. 2021. Accessed: 2023-09-25.
- [33] Athanasios Mazarakis and Paula Bräuer. Welche gamification motiviert? In *Workshop Gemeinschaften in Neuen Medien (GeNeMe) 2017*, pages 259–268. TUDpress, Dresden, 2017. Accessed: 2023-08-29.
- [34] Martin Lehner. *Didaktik*. utb, Stuttgart, Deutschland, 2020. Accessed: 2023-08-11.

Literaturverzeichnis

- [35] Uwe Fahr. Was ist Hochschuldidaktik? <https://uwe-fahr.de/2020/09/hochschuldidaktik/>, 2020. Accessed: 2023-08-10.
- [36] Linde Frank and Auferkorte-Michaelis Nicole. *Diversität in der Hochschullehre – Didaktik für den Lehralltag*. Verlag Barbara Budrich, Stuttgart, Deutschland, 2021. Accessed: 2023-08-13.
- [37] Lena Nölkenbockhoff. Diversitätsbewusst lehren. <https://lehrpfade.th-koeln.de/diversitaet-digitale-lehre/#was-ist-ein-hybrides-prasenzseminar>, 2022. Accessed: 2023-08-10.
- [38] Martina Gaisch, Silke Preymann, and Regina Aichinger. Diversitätsparadigmen neu gedacht: Schnittmengen zwischen hochschulischer Vielfalt und unternehmerischen Sinnwelten. https://www.researchgate.net/publication/316541707_Diversitatsparadigmen_neu_gedacht_Schnittmengen_zwischen_hochschulischer_Vielfalt_und_unternehmerischen_Sinnwelten/figures?lo=1, 04 2017. Accessed: 2023-10-10.
- [39] Hochschule für Technik Wirtschaft und Kultur Leipzig. C963 – Grundlagen der Programmierung. https://modulux.htwk-leipzig.de/uploads/pdfs/modulux/module/c963/modulux_modul_c963_2023-09-26_12-29-15.pdf. Accessed: 2023-08-16.
- [40] Nikita Silaparasetty. The Beginner’s Guide to Python Turtle. <https://realpython.com/beginners-guide-python-turtle/>. Accessed: 2023-08-17.
- [41] Turtle-Modul von Python. <https://www.python-lernen.de/python-turtle.htm>. Accessed: 2023-08-17.
- [42] Dr. Markus Siepermann. Pseudocode. <https://wirtschaftslexikon.gabler.de/definition/pseudocode-46714>. Accessed: 2023-08-18.
- [43] Data Science Team. Pseudocode. <https://datascience.eu/de/programmierung/pseudocode/>. Accessed: 2023-08-18.
- [44] Was ist ein Pseudocode? https://www.rfdz-informatik.at/wp-content/uploads/2020/10/RE_I_Pseudocode.pdf. Accessed: 2023-08-18.
- [45] Heinrich Müller and Frank Weichert. *Algorithmenentwurf*. Springer Fachmedien Wiesbaden, Wiesbaden, 2017. Accessed: 2023-08-29.

Literaturverzeichnis

- [46] Ulrich Brenner. Skript zum Vorkurs Programmierung in C++ - Wintersemester 2018/2019. <https://www.or.uni-bonn.de/lectures/ws18/vorkurs/skript.pdf>. Accessed: 2023-08-29.
- [47] Vorkurs C++ Programmierung. Accessed: 2023-08-29.
- [48] W3Schools. <https://www.w3schools.com/>. Accessed: 2023-08-29.
- [49] Open HPI. <https://open.hpi.de/courses>. Accessed: 2023-08-29.
- [50] Dr. Nils Haldenwang Prof. Dr. Michael Brinkmeier. Studienvorkurs Informatik. <https://www.informatik.uni-osnabrueck.de/fileadmin/documents/Arbeitsgruppen/Didaktik/Vorkurs/Skript.pdf>, 2018/19. Accessed: 2023-08-29.
- [51] Böhringer Joachim, Bühler Peter, Schlaich Patrick, and Sinne Dominik. Kompendium der Mediengestaltung. <https://link.springer.com/book/10.1007/978-3-642-54583-2>, 2014. Accessed: 2023-08-27.
- [52] Footer einer Website: Design und SEO-Tipps. <https://www.ionos.de/digitalguide/websites/webseiten-erstellen/was-ist-ein-footer/>, 2021. Accessed: 2023-08-29.
- [53] Alessandra Maino. Farbpsychologie im Webdesign. <https://usersnap.com/de/blog/farbpsychologie-im-webdesign/>. Accessed: 2023-08-25.
- [54] Lisa-Marie Volpert. Farbpsychologie im Webdesign. <https://www.kreativkarussell.de/magazin/farbpsychologie-im-webdesign/#:~:text=Die%20Farbpsychologie%20nimmt%20im%20Bereich,einfachste%20Weg%20Eindruck%20zu%20hinterlassen.>, 2020. Accessed: 2023-08-25.
- [55] coolors. <https://coolors.co/008000-d7f5dd-1a7d36-f7f7f8-1e1e1e>. Accessed: 2023-08-29.
- [56] undraw. <https://undraw.co/>. Accessed: 2023-08-28.
- [57] Svgrepo. <https://www.svgrepo.com/>. Accessed: 2023-08-28.
- [58] fontawesome. <https://fontawesome.com/start>. Accessed: 2023-08-28.
- [59] Hochschule für Technik Wirtschaft und Kultur Leipzig. Vorlagen im Corporate Design. <https://www.htwk-leipzig.de/intern/kommunikation-marketing/logo-und-vorlagen-im-corporate-design#c95526>. Accessed: 2023-08-28.

- [60] KFEDUCATION. Digitales Storytelling in der Bildung. <https://kf-education.com/digitales-storytelling-in-der-bildung/>, 2020. Accessed: 2023-08-22.
- [61] Dana Tretter. *Erzähl mir nix?! Geschichten, Narration und Storytelling in der Bildung*. KFEDUCATION, 2022. Accessed: 2023-08-22.
- [62] Helge Fischer, Matthias Heinz, Lars Schlenker, Sander Münster, Fabiane Follert, and Thomas Koehler. *Die Gamifizierung der Hochschullehre – Potenziale und Herausforderungen*, pages 113–125. 2017. Accessed: 2023-08-29.
- [63] C64 Wiki. Hallo Welt (Programm). [https://www.c64-wiki.de/wiki/Hallo_Welt_\(Programm\)](https://www.c64-wiki.de/wiki/Hallo_Welt_(Programm)), 2022. Accessed: 2023-09-04.
- [64] Suzan Kamel-ElSayed and Stephen Loftus. *Using and Combining Learning Theories in Medical Education*. Springer, 2018. Accessed: 2023-09-12.
- [65] David Tischlinger. CMS-Revolution: Vom Backend zum Frontend. <https://blog.hubspot.de/website/frontend-backend#:~:text=Was%20ist%20Frontend%20und%20Backend, die%20Nutzenden%20nicht%20zug%C3%A4nglich%20ist>. Accessed: 2023-08-20.
- [66] Mirella Willems. Turtle Server. https://github.com/mirellawlms/Turtle_Server. Accessed: 2023-09-04.
- [67] GitHub. Github-Dokumentation. <https://docs.github.com/de/get-started/quickstart/hello-world>. Accessed: 2023-09-07.
- [68] Visual Studio Code. Visual Studio Code. <https://code.visualstudio.com/>. Accessed: 2023-09-12.
- [69] nextjs. The React Framework for the Web. <https://nextjs.org/>. Accessed: 2023-09-12.
- [70] TypeScript. TypeScript. <https://www.typescriptlang.org/>. Accessed: 2023-09-12.
- [71] React. Komponenten und Props. [https://de.legacy.reactjs.org/docs/components-and-props.html#:~:text=React%20%C3%BCbergibt%2C%20wenn%20es%20ein,%E2%80%9Cprops%E2%80%9D%20\(Eigenschaften\)](https://de.legacy.reactjs.org/docs/components-and-props.html#:~:text=React%20%C3%BCbergibt%2C%20wenn%20es%20ein,%E2%80%9Cprops%E2%80%9D%20(Eigenschaften)). Accessed: 2023-09-07.
- [72] Monaco Editor. <https://www.npmjs.com/package/@monaco-editor/react>. Accessed: 2023-09-07.

- [73] Object Types. <https://www.typescriptlang.org/docs/handbook/2/objects.html>. Accessed: 2023-09-07.
- [74] Digital Guide IONOS. Was ist ein Server? <https://www.ionos.de/digitalguide/server/knowhow/was-ist-ein-server-ein-begriff-zwei-definitionen/>, 2023. Accessed: 2023-09-07.
- [75] redhat. *Was ist eine API?* redhat, 2023. <https://www.redhat.com/de/topics/api/what-are-application-programming-interfaces>. Accessed: 2023-09-07.
- [76] Leo Lemke. Rest apis einfach erklärt – definition, funktionsweise + anwendungsbeispiel. <https://www.friendventure.de/wordpress/rest-api-erklaerung/>, 2020. Accessed: 2023-09-07.
- [77] Roy Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000. Accessed: 2023-09-07.
- [78] kinsta. *Verständnis von API-Endpunkten.* 2023. <https://kinsta.com/de/wissensdatenbank/api-endpunkt/#:~:text=Ein%20API%2DEndpunkt%20ist%20ein,auf%20dem%20API%2DServer%20angibt.>, Accessed: 2023-09-07.
- [79] PostgreSql. <https://www.postgresql.org/>. Accessed: 2023-09-07.
- [80] Prisma. Is Prisma an ORM? <https://www.prisma.io/docs/concepts/overview/prisma-in-your-stack/is-prisma-an-orm>. Accessed: 2023-09-12.
- [81] Prisma. Prisma. <https://www.prisma.io/>. Accessed: 2023-09-12.
- [82] Simon Knott. Prisma ER Diagram. <https://prisma-erd.simonknott.de/>. Accessed: 2023-09-12.
- [83] Hochschule für Technik Wirtschaft und Kultur Leipzig. Eduroam WLAN an der Hochschule. <https://itsz.htwk-leipzig.de/serviceangebote-dienste/wlan>. Accessed: 2023-09-07.
- [84] Hochschule für Technik Wirtschaft und Kultur Leipzig. VPN Virtual Private Network. <https://itsz.htwk-leipzig.de/serviceangebote-dienste/vpn-zugriff-auf-das-hochschulnetz>. Accessed: 2023-09-07.
- [85] Mirella Willem. HTWK VPN Zugriff auf Turtle-Website. <http://141.57.9.103:3000>. Accessed: 2023-09-07.

Literaturverzeichnis

- [86] Linuxzise. *How to use Linux Screen.* 2021. <https://linuxize.com/post/how-to-use-linux-screen/>. Accessed: 2023-09-15.
- [87] How to install and use Docker on Ubuntu 20.04. <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-20-04>. Accessed: 2023-09-07.
- [88] Docker PostgreSQL. https://hub.docker.com/_/postgres. Accessed: 2023-09-07.
- [89] Node. <https://nodejs.org/de/about>. Accessed: 2023-09-07.
- [90] Die Zukunft von Lehren und Lernen. <https://bildungsportal.sachsen.de/impulse/zukunft/>. Accessed: 2023-09-07.
- [91] Das E-Learning-Informationsportal für Sächsische Hochschulen. <https://bildungsportal.sachsen.de/portal/>. Accessed: 2023-08-10.
- [92] Raoul Mulder, Jon Pearce, and Chi Baik. Peer review in higher education: Student perceptions before and after participation. *Active Learning in Higher Education*, 15, 2014. Accessed: 2023-09-15.
- [93] Hochschule für Technik Wirtschaft und Kultur Leipzig. Inhaltliche Organisation. <https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/25768460288/CourseNode/1631759207016055012>. Accessed: 2023-09-04.