Ostbayerische Technische Hochschule Regensburg

Fakultät für Informatik

Bachelorarbeit

im Studiengang Informatik

zur Erlangung des akademischen Grades
Bachelor of Science

Thema: Analyse, Konzeption und Implementierung eines Tools für das automatisierte Testen, Einreichen und Benoten von Programmieraufgaben

Autor: Andreas Huber <andreas.huber@st.oth-regensburg.de>
Matr.-Nr. 3180161

Version vom: 17. März 2022

Betreuer: Prof. Dr. Markus Heckner

Erklärung zur Bachelorarbeit

1. Mir ist bekannt, dass dieses Exemplar der Abschlussarbeit als Prüfungsleistung in das Eigentum der Ostbayerischen Technischen Hochschule Regensburg übergeht.

2. Ich erkläre hiermit, dass ich diese Abschlussarbeit selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Regensburg, den 17. März 2022

Andreas Huber

Inhaltsverzeichnis

IDE Integrated Development Environment - Integrierte Entwicklungsumgebung

Bash Bourne-again shell

LMS Leaning Management System

LDAP Lightweight Directory Access Protocol

SD Standardabweichung

1 Einleitung und Motivation

1.1 Herausforderung digitales Lernen

Deutschland gilt als sehr rückschrittlich im Thema Digitalisierung an Schulen und Universitäten. Nicht zuletzt hat die Corona-Pandemie den Rückstand Deutschlands in vielen Bereichen offengelegt. Neben Verwaltungen, Unternehmen, Schulen oder Gerichten hat die Pandemie vor allem viele Hochschulen dazu gezwungen, umzudenken und digitale Lerninhalte zu erstellen. Mit dieser Herausforderung kamen jedoch einige Einrichtungen und Lehrende[1] ohne jegliche Vorbereitungen schnell an ihre Grenzen. (Bundesministerium für Wirtschaft und Energie, n. d.)

Professor Doktor Markus Heckner hat sich dieser Problemstellung gestellt und hat gemeinsam mit Mitarbeitenden der Hochschule Regensburg an einer Lösung gearbeitet. Das Ergebnis ist ein optionales Zusatzstudium, um die sogenannten „Digital Skills“ der Studierenden zu verbessern und zu intensivieren. Langfristig gesehen sollen die Zusatzmodule die digitalen Fähigkeiten der Teilnehmer fördern und somit zur Einholung des digitalen Rückstands in Deutschland beitragen.

1.2 Kurs Digital Skills als Zusatzstudium

Digital Skills ist ein aus drei Semester bestehendes Zusatzstudium für alle Studierenden, mit Ausnahme von Studierenden der Fakultät Informatik und Mathematik, der Hochschule Regensburg. Wie der Name bereits sagt, findet das Zusatzstudium parallel zu dem Hauptstudium der Studierenden statt.

Das Zusatzstudium soll den Teilnehmern vor allem digitales Wissen näher bringen. In Semester 1 lernen die Studierenden unter dem Motto „Technologische Skills“ die Grundlagen der Programmierung, sowie das Verständnis von Internet of Things. Das zweite Semester befasst sich mit „Future Work Skills“ und bringt den Teilnehmern das Verständnis der Themen Data Science, Digitale Ethike, Agile Working, sowie Coaching Fähigkeiten bei. Im letzten Semester haben die Lernenden die Möglichkeit ein eigenes Digitalisierungsprojekt zu planen und umzusetzen.

[1] Aus Gründen der besseren Lesbarkeit wird in der gesamten Arbeit auf die gleichzeitige Verwendung der Sprachformen männlich, weiblich und divers (m/w/d) verzichtet. Sämtliche Personenbezeichnungen gelten gleichermaßen für alle Geschlechter. Ist eine spezifische Geschlechtergruppe gemeint, wird das entsprechende Adjektiv vorangestellt (z.B. „männliche Studenten“)

Die studienbegleitende Ausbildung wird den Studierenden nach Abschluss aller Veranstaltungen in Form eines Hochschulzertifikats, sowie anhand einer Erwähnung im Abschlusszeugnis, angerechnet.

Um im ersten Semester die Grundlagen der Programmierung automatisiert und trotzdem mit verständlichem Feedback zu lehren, wird eine zeit- und ortsunabhängige innovative Lernplattform benötigt. Die Teilnehmer sollen dabei trotzdem eine möglichst realitätsnahe Programmierumgebung kennenlernen. Um solch ein System nachhaltig bereitzustellen, benötigt man viel Planung und Verständnis der Studierenden. Genau mit dieser Aufgabe beschäftigt sich der Kern dieser Arbeit.

1.3 Struktur der Arbeit

Die Arbeit besteht aus mehreren wesentlichen Bestandteilen. Kapitel 2 behandelt alle nötigen Anforderungen, die für den Aufbau einer online Programmierplattform wichtig sind. Die Anforderungen werden hierbei unterteilt in funktionale und nichtfunktionale Anforderungen. Die funktionalen Anforderungen werden dabei aus zwei unterschiedlichen Sichten dargestellt: die Anforderungen aus der Sicht von Studierenden bzw. Teilnehmenden, sowie aus der Sicht von Lehrenden.

Kapitel 3 wiederum vergleicht und diskutiert zuerst vorhandene Lern-Systeme. Hierbei werden neben den Vorteilen auch die Probleme beim Einsatz an der Hochschule Regensburg besprochen. Zum Abschluss des Kapitels werden alle aufgeführten Online-Lernplattformen anhand der vorher aufgestellten funktionalen und nichtfunktionalen Anforderungen neutral mithilfe einer Tabelle bewertet und schließlich miteinander verglichen. Die finale Entscheidung, welches System am besten zu dem Zusatzstudium Digital Skills passt, fällt auf das System, welches bei dem Tabellenvergleich am meisten Punkte erreicht hat.

Die finale Entscheidung, sowie deren Konfiguration und Implementierung wird dann folgend in Kapitel 4 näher erläutert und beschrieben. Dabei geht es vor allem um technische Details, wie z.B. zusätzliche Software, die zur Unterstützung der Teilnehmer entwickelt werden muss.

Anschließend wird die Programmierlernplattform in Kapitel 5 in Form einer Feldstudie an nicht Informatik- oder Mathematik- studierenden Testprobanden getestet. Am Anfang des Kapitels wird das Testkonzept und der Aufbau der Studie erläutert. Danach folgt die Auswertung der Testergebnisse.

Schließlich fasst Kapitel 6 alles zusammen und gibt einen weiteren Ausblick auf die Zukunft des Zusatzstudiums Digital Skills, sowie auf den darin enthaltenen automatisierten Programmierkurs.

2 Anforderungsanalyse

Funktionale Anforderungen beschreiben, was das Projekt tun soll. Diese Informationen sind wichtig, um die richtigen Werkzeuge und Programme für die Umsetzung auszuwählen. Dabei geht es meist um sehr konkret formulierte Wünsche. Nichtfunktionale Bedingungen sind wiederum Bedingungen, wie zum Beispiel Zuverlässigkeit, Verfügbarkeit oder diverse Sicherheitsanforderungen. Sie lassen sich eher als Qualitätseigenschaften beschreiben.

2.1 Funktionale Anforderungen

2.1.1 Studierende

Studierende sollen eine Kursübersicht mit allen Aufgaben haben. Der persönliche Fortschritt der jeweiligen Aufgaben sollte dabei leicht ersichtlich sein. Mögliche Deadlines oder Abgabefristen sollen bei jeder Aufgabe deutlich erkennbar sein.

Des Weiteren müssen Studierende die Möglichkeit haben, ohne die Installation von zusätzlichen Programmen, die Aufgaben online bearbeiten, prüfen und abgeben zu können. Trotzdem sollen sie die Option haben, die Aufgaben in der Entwicklungsumgebung ihrer Wahl lösen zu können.

Eine weitere wichtige Anforderung bei der Prüfung der Aufgaben ist es, dass die Bearbeiter der Aufgaben sogenannte *human-readable* (für den Menschen lesbare) Fehlermeldungen erhalten müssen. Das bedeutet, dass die Fehlermeldungen bei der Überprüfung auch für nichttechnische Studierende leicht verständlich sein müssen. Fehlermeldungen bzw. konstruktives Feedback muss dabei automatisiert und jederzeit generiert werden können.

Die Möglichkeit Aufgabenversuche abzugeben, muss ebenfalls mit wenig Aufwand behaftet sein. Falls ein Versuch fehlschlägt, oder nicht die volle Punktzahl erhält, sollte der Teilnehmende jederzeit die Möglichkeit haben, einen neuen Versuch hochladen zu können.

2.1.2 Lehrende

Lehrende müssen neue Aufgaben anlegen und konfigurieren können, dazu gehört unter anderem die Festlegung einer Deadline bzw. eines Abgabedatums.

Des Weiteren müssen Dozenten eine Übersicht an Aufgaben des jeweiligen Kurses haben.

Einzelne Aufgaben sollten durch Lehrende temporär versteckt oder deaktiviert werden können.

Überdies hinaus muss es möglich sein, mehrere Administratoren zu den Kursen hinzuzufügen. Dadurch ist es möglich, dass verschiedene Personen die Aufgaben erstellen und die abgegebenen Lösungen herunterladen können. Dies ist gleichzeitig die nächste Anforderung: Administratoren müssen mit wenig Aufwand alle Abgaben der Teilnehmenden herunterladen können.

Ferner müssen die Aufgaben und die jeweiligen Deadlines auch nach der Erstellung editierbar sein.

Lehrende müssen den Aufgaben-Fortschritt der Teilnehmer je nach Kurs übersichtlich einsehen können.

2.2 Nichtfunktionale Anforderungen

Das eingesetzte System muss neben den funktionalen Anforderungen auch diverse nichtfunktionale Anforderungen erfüllen, um von der OTH als sinnvolle Lernplattform eingesetzt werden zu können.

Der erste wichtige Punkt ist, dass die Lernplattform möglichst zuverlässig ist. Zur Zuverlässigkeit gehört neben einer hohen Verfügbarkeit auch eine skalierende Performance, wenn viele Studierende gleichzeitig die Plattform nutzen wollen.

Ein weiterer Punkt ist die Wartbarkeit. Die Plattform sollte mit möglichst wenig Wartung sicher bestehen bleiben können. Außerdem sollte nur auf externe Systeme gesetzt werden, von denen ausgegangen werden kann, dass diese noch einige Jahre gepflegt werden. Hier empfiehlt sich ein Aufteilen der Plattform auf mehrere externe Tools, um bei einem Ausfall oder Außerbetriebnahme eines einzelnen Tools noch einen Notbetrieb gewährleisten zu können. Der Austausch gegen eine andere neue Softwarekomponente gestaltet sich dadurch leichter.

Die Ressourcenlast und damit verbundenen Kosten spielen eine weitere wichtige Rolle bei der Entscheidungsfindung. Wenn Teile der Lernplattform auf OTH-Servern gehostet werden müssen, sollten diese möglichst ressourcenschonend sein. Serverressourcen sind teuer und können das Projekt im Zweifelsfall unrentabel machen, wenn das System bei paralleler Nutzung durch mehrere Studierende eine inadäquate Serverlast voraussetzen würde. Selbiges gilt für

Lizenzgebühren möglicher Tools und Werkzeuge.

Neben den Kosten ist es auch wichtig, dass die Plattform zusammen mit dem Learning Management System (LMS) bzw. der Lernplattform der Hochschule arbeiten kann. Der Vorteil einer LMS-Integration wird später im Kapitel 3.2.2 näher erläutert.

Zu guter Letzt ist es wünschenswert, dass der Programmierkurs mit der Auswahl der Programmiersprache flexibel ist. So sollte es beispielsweise möglich sein, dass die erste Aufgabe mit der Programmiersprache Java gelöst wird, während die zweite mit Python gelöst werden muss.

3 Softwarearchitektur

Dieses Kapitel befasst sich mit der für das Projekt benötigten Toolchain und damit einhergehenden Softwarearchitektur.

Eine Toolchain ist eine Sammlung verschiedener Anwendungen, die gemeinsam eine Lösung bzw. ein Produkt erzeugen. Durch den Vergleich mit verschiedenen etablierten digitalen Lernplattformen ist es möglich, optimale bereits bestehende und gut getestete Software-Werkzeuge für die Hochschule Regensburg zu finden und schließlich einzusetzen.

Die Softwarearchitektur erläutert in diesem Zusammenhang die einzelnen Softwarekomponenten der Toolchain und beschreibt deren Zusammenspiel innerhalb eines Systems.

3.1 Versionskontrolle mit Git

3.1.1 Allgemein

Die Versionskontrolle oder Quellcodekontrolle ermöglicht es Änderungen an Softwarecode zu verfolgen. Die Verfolgung aller Änderungen erlaubt bei Bedarf die Wiederherstellung eines früheren Datenstands.

Versionskontrollsysteme, wie Git, speichern jede Änderung am Code in speziellen git-Datenbank-Dateien. Programmierer können dadurch in ihren Teams genau analysieren, wer welche Zeile wann geändert oder neu eingesetzt hat.

Durch Git können mehrere Entwickler an demselben Projekt bzw. Repository arbeiten und sogar dieselben Dateien gleichzeitig ändern. Git bietet hierfür Prozesse um solche Konflikte sauber lösen zu können. Beim Zusammenführen der beiden Zustände kann man entscheiden welche Zeile man aus welcher Änderung übernimmt.

Git bietet durch die Speicherung jeder Änderung nicht nur den Vorteil vom gemeinsamen Arbeiten, sondern auch eine Art „Backup“. Durch den genauen Verlauf des Quellcodes, können Fehlerursachen schneller analysiert, gefunden und verifiziert werden. Um die einzelnen Snapshots bzw. Schnappschüsse des aktuellen Standes zu markieren, gibt es in Git die sogenannten Commits.

Die folgenden Begriffsklärungen und Anleitungen rund um das Versionskontrollsystem Git sind wichtig, um den Rest der Arbeit besser verstehen zu können.

3.1.2 Repositories und Commits

Das Git-Repository ist der Kern des Projekts und umfasst alle Dateien, die von Git getracked werden sollen. Es ist sozusagen das Projekt an dem die Entwickler arbeiten. Um ein Repository anzulegen braucht es nur einen Ordner in dem man in der Konsole folgenden Befehl ausführt:

```
$ git init
```

Daraufhin wird in diesem Verzeichnis ein neuer `.git`-Ordner erstellt, welcher alle Informationen über das gerade erstellte Repository enthält. Sobald man nun beispielsweise eine neue Textdatei anlegt, wird diese Handlung im `.git`-Ordner getracked.

Git arbeitet mit Snapshots, die man manuell anlegen muss. Wenn der Entwickler nun täglich einen Absatz in die Textdatei schreibt, gibt es hierfür keine Historie. Git weiß nur, dass im ersten Snapshot keine Datei vorhanden war und nun eine Datei mit gefülltem Text vorliegt.

Um einen Verlauf der Erstellung zu speichern, muss der Entwickler sogenannte Commits mit aktuellen Zeitstempeln erstellen. Wann die Commits jeweils erstellt werden, ist dem Entwickler selbst überlassen. In der Regel werden Commits immer nach dem Erreichen eines Meilensteins erstellt. In diesem Fall wäre ein täglicher Meilenstein das Abschließen eines neuen Absatzes in der Textdatei. Zuerst können alle geänderten Dateien in der Konsole mit folgendem Befehl abgerufen werden:

```
$ git status
```

Die Ausgabe sieht in erläuterten Beispiel ungefähr so aus:

```
On branch main

No commits yet

Untracked files:
(use "git add <file>..." to include in what will be committed)
    hallo-welt.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Wie die Ausgabe bereits verrät, wurde bisher noch kein Commit angelegt. Außerdem verrät sie, dass die Datei `hallo-welt.txt` angelegt wurde, bisher jedoch nicht getracked wird. Nun kann der Entwickler die Änderung tracken, indem er folgenden Befehl ausführt:

```
$ git add hallo-welt.txt
```

Durch den Befehl `git add .` kann er auch mit einem Kommando alle *untracked* (nicht verfolgte) Änderungen in den neuen Commit hinzufügen. Jetzt weiß Git welche Änderungen commited werden sollen. Mit dem Befehl

```
$ git commit -m "Neue Hallo Welt Datei angelegt"
```

wird ein Commit mit der Nachricht „Neue Hallo Welt Datei angelegt“ angelegt. Dies ist nun ein neuer Snapshot, welcher in der Commit-Historie gespeichert wird. Sollte der Programmierer die Datei nun verändern, kann er die neuen Änderungen erneut tracken und wieder committen. An diesem Punkt könnte er jedoch jederzeit zu dem alten bereits committeten Zustand zurückkehren.

Damit Repositories im Internet verfügbar sind, benötigt man einen Server mit einer Kopie des lokalen Repositories. Diese Repositories nennt man auch Remote-Repositories. Der bekannteste Dienstleister für die Speicherung von Remote-Repositories ist GitHub. Der Vorteil von einer Online-Version des eigenen Repositories ist, dass andere Entwickler das Projekt klonen und anschließend mitarbeiten können. Nach dem Klonvorgang haben diese Personen ebenfalls eine lokale Kopie des Repositories auf ihrem Endgerät. Um nun einen lokalen Commit online verfügbar zu machen, muss dieser *gepushed* werden. Zuerst muss über folgenden Befehl geprüft werden, ob im lokalen Repository das richtige Remote-Repository referenziert wird:

```
$ git remote -v
origin https://github.com/user/repository-name (fetch)
origin https://github.com/user/repository-name (push)
```

Sollten die Adressen zum richtigen Online-Repository verweisen, kann mit folgendem Befehl ein Commit hochgeladen werden:

```
$ git push
```

Um mögliche Änderungen durch andere Mitarbeitende in das eigene lokale Repository zu laden, benötigt man einen sogenannten *Pull*:

```
$ git pull
```

Der Git-Workflow umfasst noch viele weitere Befehle und kann für Unerfahrene schnell sehr kompliziert werden. Aus diesem Grund enthalten die meisten gängigen Entwicklungsumgebungen grafische Oberflächen für die Verwendung von Git. So benötigt der Entwickler weder Konsole, noch tieferes Fachwissen über die genaue Verwendung bzw. Syntax der Befehle.

3.1.3 Branches

Git bietet neben der Kontrolle und Verfolgung von Commits und Änderungen auch die Möglichkeit echt parallel zu arbeiten. In einem großen Softwareprojekt sind zur Laufzeit auftretende Fehler in der Regel unumgänglich. Um keine Kunden zu verlieren, müssen einige diese Fehler gegebenenfalls schnellstmöglich behoben werden.

In diesem beispielhaften Fall, dass in einer Software ein gravierender Fehler auftritt, sollte möglichst schnell ein Update mit einer Lösung ausgearbeitet werden. Es kann jedoch sein, dass ein Teil des Entwicklungsteams gerade an einem sehr großen neuen Feature arbeitet, welches nicht halbfertig veröffentlicht werden darf. Es muss also eine Lösung geben, dass eine neue Version mit dem behobenen Fehler veröffentlicht werden kann, ohne, dass die Arbeit an dem neuen Feature gestört oder rückgängig gemacht werden muss. Git stellt mit dem Konzept von Branches eine Lösung für dieses Problem dar.

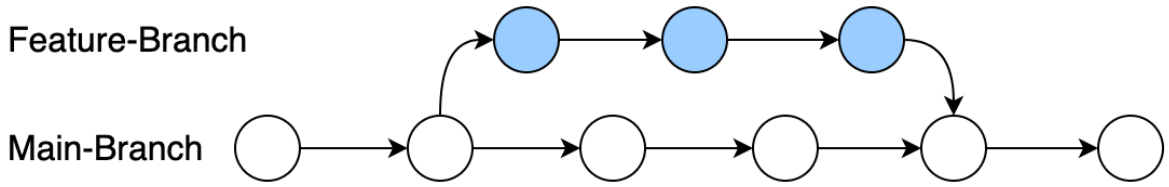

Abbildung 1: Git: Branches
Quelle: Eigene Darstellung

Branches (dt. Äste) teilen den linearen Entwicklungsablauf in mehrere parallele Zustände. Standardmäßig arbeitet man bei Git auf dem `Main`-Branch. Der Hauptzweig sozusagen (siehe Abbildung 1, der weiße Strang). Die weißen Kugeln stellen in Abbildung 1 jeweils einzelne Commits dar. Sobald ein neues Feature entwickelt wird, möchte der zuständige Programmierer

nicht, dass sich während der Entwicklung der Funktion etwas am restlichen Code ändert. Aus diesem Grund eröffnet der Feature-Programmierer, wie in Abbildung 1 ersichtlich, einen neuen Branch auf den aktuellen Commit des Hauptzweigs (blauer Zweig). Wenn sich der Hauptzweig jetzt durch Änderungen von anderen Kollegen ändert, merkt der Feature-Programmierer nichts davon, weil er sich auf einem anderen Ast (Branch) befindet und dabei seine ganz eigene Kopie des Projekts bearbeitet. Sobald er mit dem Feature fertig ist, kann er seinen Branch mit dem `Main`-Branch wieder zusammenführen. Falls sich in der Zwischenzeit die Dateien und Zeilen, die auch er bearbeitet hat, geändert haben, müssen die Konflikte in aller Regel manuell gelöst werden. Sollten nicht dieselben Zeilen bearbeitet worden sein, löst Git die Konflikte selbst und führt die zwei Zustände automatisch zusammen.

Eine bekannte Konvention unter Programmierern ist es, dass der `Main`-Branch immer eine lauffähige Version der Software hält. Es darf nie ein nicht-kompilierbarer oder unfertiger Stand auf dem Hauptzweig landen. So stellt man sicher, dass neue abzweigende Branches immer auf einer lauffähigen Basis aufbauen.

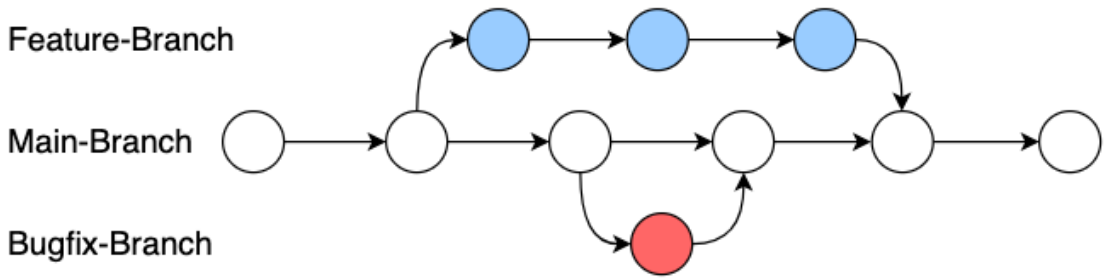

Abbildung 2: Git: Branches (Bugfix-Branch)
Quelle: Eigene Darstellung

Wie in Abbildung 2 zu erkennen, werden auch Fehlerbehebungen (Bugfixes) meist in einem anderen Branch behandelt. Nun lässt sich auch die anfangs erläuterte Problemstellung leicht erklären. Wenn ein gravierender Softwarefehler auftritt, kann das Team ausgehend vom lauffähigen `Main`-Branch einen neuen Zweig zur Fehlerbehebung eröffnen (siehe Abbildung 2, roter Kreis). Auf diesem Zweig können sie unabhängig von der Entwicklung eines neuen Features den Fehler beheben, mit dem Hauptzweig zusammenführen und das Update veröffentlichen. Wenn das Feature fertig ist, wird es mit dem Hauptzweig zusammengeführt und der Fehler auch darin automatisch behoben. Sollte der Fehler so gravierend sein, dass er die Entwicklung des `Feature`-Branches hindert, kann man jederzeit den nun fehlerfreien `Main`-Branch in den

`Feature`-Branch mergen und den `Feature`-Branch somit wieder auf den aktuellen Stand bringen. Mergen heißt hier nichts anderes als das Zusammenführen zweier Codebasen. Dies ist auch sinnvoll, wenn die Entwicklung eines Features einen längeren Zeitraum beansprucht. Ansonsten geht man die Gefahr ein, dass sich der Hauptzweig bei der Zusammenführung so sehr verändert hat, dass er dem `Feature`-Zweig zu sehr differenziert und ein Merge sehr viel manuelle Arbeit erfordert.

3.2 Vergleich vorhandener Systeme

3.2.1 CS50 der Harvard University

3.2.1.1 Allgemeines

CS50 ist die ursprüngliche Bezeichnung eines Lernkurses über Informatik, welcher von der Harvard University ins Leben gerufen wurde und weiterhin betreut wird. Der Kurs wurde aufgrund seines Erfolgs digitalisiert und wird nun als CS50x auf der Online-Lernplattform edX angeboten. Folgende Recherchen und Aussagen über CS50 sind jeweils immer auf die Online-Version CS50x bezogen.

Der Kurs CS50 lehrt Schülerinnen und Schüler die Grundlagen der Informatik. Dabei werden diverse Programmierübungen abgefragt. Aufgrund der hohen Anzahl an Teilnehmenden besitzt der Kurs ein automatisiertes Abgabe- und Benotungssystem.

Das System hinter CS50 wird mittlerweile vielfältig eingesetzt und wurde zu einem universalen Online-Lernsystem erweitert. Jeder kann sich durch eine Authentifizierung über die Plattform GitHub im Abgabesystem von CS50 einloggen und eigene Kurse erstellen. (Harvard University, n. d. b)

3.2.1.2 Ablauf für Studierende

Den Teilnehmerinnen und Teilnehmern wird jede Woche ein neues Kapitel präsentiert. Dabei können sie sich sowohl durch ein Vorlesungsvideo, als auch durch geschriebene Materialien über das Thema der Woche informieren. Mit Beginn der Woche bekommen die Teilnehmenden neben den Materialien auch Programmieraufgaben, welche sie mit dem vorher genannten System bearbeiten können. (Harvard University, n. d. c)

Die Programmieraufgaben können wahlweise über die, auf AWS Cloud9 basierenden, Online-Entwicklungsumgebung *CS50-IDE* von Harvard oder in jeder anderen beliebigen Entwicklungsumgebung der Wahl bearbeitet werden. Dies wird durch die flexible Architektur des Systems ermöglicht. Jede Funktionalität des vollautomatisierten Kurses geschieht durch öffentlich bereitgestellte Kommandozeilen-Tools. Dieses System hat den Vorteil, dass es unabhängig von der eingesetzten IDE (Entwicklungsumgebung) funktioniert, es wird lediglich ein Terminal mit den jeweiligen Tools benötigt. (Harvard University, n. d. d)

Um einen Lösungsversuch abzugeben wird das sogenannte Werkzeug *submit50* verwendet. Um den Code vor Abgabe auf Fehler zu überprüfen, stellt Harvard das Tool *check50* bereit. Auch die Sauberkeit und Qualität des Codes kann mithilfe eines Werkzeugs überprüft werden. Hierfür heißt die Softwarelösung *style50*. (Harvard University, n. d. e)

3.2.1.3 Ablauf für Lehrende

Um eigene Programmierkurse auf Basis der CS50-Technik zu erstellen, stellt Harvard die *me50-Plattform* kostenfrei zur Verfügung. Hierzu müssen Lehrende genau folgenden Link in ihrem Browser eingeben: https://submit.cs50.io/courses/new. Der Link führt zu einer Seite, um einen neuen Kurs anzulegen. Auf die Erstellungsseite gelangt man nur, über den Direktlink. Einen Knopf gibt es dafür nicht. Nach der Eingabe eines neuen Kursnamens öffnet sich die Einstellungsseite des neu angelegten Kurses. Dort können neben dem gerade festgelegten Namen auch die Beschreibung des Kurses, die beinhaltenden Aufgaben sowie die zuständigen Admin-Accounts der Dozenten festgelegt und geändert werden.

Neue Aufgaben bzw. Probleme können mithilfe der folgenden Anleitung erstellt werden: https://cs50.readthedocs.io/projects/check50/en/latest/check_writer. Nach der Erstellung eines Problems kann der sogenannte Slug (Name des Problems) in den Kurseinstellungen referenziert werden. Dadurch wird die Aufgabe automatisch im Kurs angezeigt.

In der Kursübersicht der me50-Plattform können Lehrende die Abgaben und Versuche der Studierenden sehen. Bei jedem Eintrag ist jeweils die erreichte Punktzahl von check50 und style50 sichtbar. Eine Übersicht der Abgaben können Dozenten jederzeit in Form einer .csv- oder als .json-Datei pro Aufgabe heruntergeladen werden.

3.2.1.4 Architektur

Die Harvard University hält den Aufbau von CS50 weitestgehend transparent. Viele der eingesetzten Werkzeuge sind öffentlich als Open-Source-Projekte unter der GitHub-Organisation „CS50“ zu finden (Harvard University, n. d. a). Darunter befinden sich unter anderem folgende Projekte:

- submit50: Abgabe von Code
- check50: Funktionalitätstests des Codes
- render50: Erzeugung von .PDF-Dateien aus Code

- ide50: Online-Entwicklungsumgebung
- style50: Überprüfung der Code-Qualität
- compare50: Plagiatserkennung von abgegebenen Projekten
- server50: Webserver

In Abbildung 3 ist der Aufbau der Lernplattform CS50 vereinfacht grafisch dargestellt. Die Rechtecke und Menschen sind die beteiligten Komponenten. Die Pfeile zwischen den Komponenten beschreiben die verbindende Relation. Von der `CS50-Programm`-Komponente ausgehende gestrichelte Pfeile zeigen, je nach ausgeführtem Befehl, mögliche Relationen.

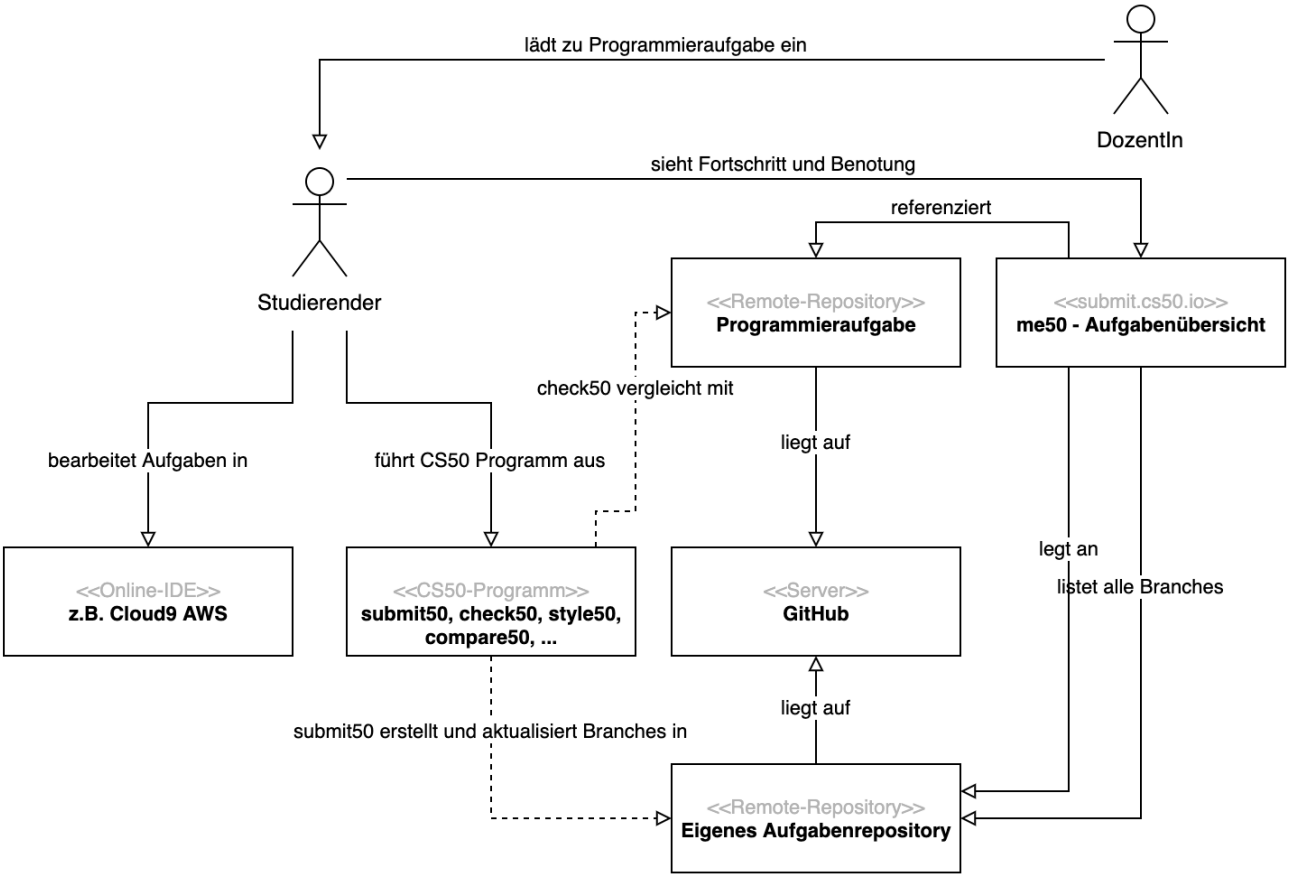

Abbildung 3: CS50 Architektur
Quelle: Eigene Darstellung

Teilnehmer dieses Online-Lehrangebots erhalten von den Dozenten einen Einladungslink zum Kurs. Dieser Link führt sie zur me50-Plattform (https://submit.cs50.io/), welche gleichzeitig als Aufgabenübersicht dient. Auf dieser Plattform können Studierende sowohl ihre Abgaben, als auch ihre Kurse übersichtlich verwalten. Neben der Weiterleitung zur me50-Plattform wird

gleichzeitig in der me50-GitHub-Organisation ein leeres Repository mit dem GitHub-Username des Teilnehmers angelegt.

Nach Annahme des Einladungslinks können Studierende in jeder beliebigen Entwicklungsumgebung die Aufgaben auf der CS50-Plattform (https://cs50.harvard.edu/) lösen. Das Korrigieren und Abgeben der Aufgabe benötigt mindestens Zugriff auf die Kommandozeilen-Tools check50 und submit50. Diese sind über den Python-Paketmanager frei verfügbar. Harvard stellt außerdem eine kostenlose Online-IDE bereit, welche die genannten Python-Programme bereits vorinstalliert zur Verfügung stellt.

Sobald der Studierende eine Aufgabe erfolgreich gelöst hat, kann er mithilfe des Namens der Aufgabe die Lösung überprüfen und schließlich abgeben. Eine Abgabe über die Konsole kann dann beispielsweise so aussehen:

```
~/hello/ $ submit50 cs50/problems/2021/x/hello
Connecting......
Authenticating...
Verifying.........
Preparing..................
Files that will be submitted:
./hello.c
Keeping in mind the course's policy on academic honesty, are you sure you want
    to submit these files (yes/no)? yes
Uploading....
.......
Go to https://submit.cs50.io/users/ndhbr/cs50/problems/2021/x/hello to see your
    results.
```

Der Parameter `cs50/problems/20212/x/hello` im Aufruf von submit50 ist der im vorherigen Abschnitt erläuterte Slug. Das Programm check50 wird ebenfalls mit dem Slug als Parameter aufgerufen. Mit dieser Information wissen die Programme, mit welcher Aufgabe sie die Lösung des Teilnehmenden vergleichen sollen.

Die letzte Zeile verlinkt erneut auf die me50-Plattform und zeigt dem Kursteilnehmer seine Abgaben dieser Aufgabe und seine damit erreichte Punktezahl. Gleichzeitig verlinkt die me50-Plattform auf das vorher erstellte GitHub-Repository des Teilnehmers. Die Plattform legt bei jeder neu abgegebenen Aufgabe einen neuen git-Branch für diese an. Nach der Abgabe des vorherigen Beispiels hätte die Plattform folgenden Branch in dem Repository angelegt `cs50/problems/2021/x/hello`. Durch die Commit-History in diesem Branch kann man später

den Verlauf der Versuche nachverfolgen.

3.2.1.5 Probleme beim Einsatz für die OTH

Die Toolchain von CS50 wäre grundsätzlich adäquat für den Einsatz an der OTH-Regensburg. Jedoch ist eines der Projekte aktuell noch nicht Open-Source: die Website zur Erstellung von neuen Kursen, Abgaben und Mitgliederverwaltung. Das heißt, dass der Quellcode dieses Projekts nicht öffentlich zugänglich ist.

Dieses Projekt ist gleichzeitig der größte Bestandteil des Netzwerks und ist deshalb essenziell für die Verwendung der Werkzeuge an der Regensburger Hochschule. Nach Rücksprache mit Herrn Carter Zenke der Harvard University ist ein Neuaufbau dieser Website mit einhergehender Veröffentlichung als Open-Source-Projekt gerade in Planung. Einen genauen öffentlichen Zeitplan hierfür gibt es aktuell nicht. Infolgedessen ist ein Einsatz des CS50-Systems an der OTH zum heutigen Datum nicht möglich.

3.2.2 Code FREAK der Fachhochschule Kiel

3.2.2.1 Allgemeines

Code FREAK ist eine All-in-one-Lösung für Online-Programmieraufgaben mit automatisiertem Feedback. Die von der Hochschule Kiel entwickelte Open-Source-Software soll den Einstieg in eine digitale Lernumgebung einfach machen. Die Zielgruppe von Code FREAK bezieht sich hierbei explizit auf Universitäten und Einrichtungen einer höheren Bildung. (Kiel University of Applied Sciences, 2019-2020)

Des Weiteren wirbt Code FREAK mit einer *LMS-Integration*. LMS ist die englische Abkürzung für Learning Management System (Lernplattform). Viele Schulen und Universitäten verwenden bekannte LMS-Systeme, wie Moodle, um Kurse, Fächer und Studierende zu verwalten (moodle.de, n. d.). Dies hat den Vorteil, dass Studierende kein extra Nutzerkonto für Code FREAK anlegen müssen. Sie können sich direkt mit ihrem gewohnten Hochschulzugang anmelden.

3.2.2.2 Ablauf für Studierende

Studierende können nach Anmeldung am System, an für sie sichtbaren Kursen (Assignments) teilnehmen. Dies geschieht meist durch einen vom Dozenten generierten Einladungslink. Jedes Assignment kann mehrere Aufgaben (Tasks) enthalten. Sobald der Einladungslink zum Assignment akzeptiert wurde, kann die teilnehmende Person alle Tasks aus dem Assignment sehen und bearbeiten. Die einzelnen Aufgaben können dann entweder über die integrierte Online-Entwicklungsumgebung, durch Hochladen der Lösung, oder durch die Angabe eines Git-Remote-Repository-Links bearbeitet werden.

, Mit dem Klick auf den „Start Evaluation“ Knopf wird die hochgeladene Lösung der Aufgabe überprüft. Die Ergebnisse der einzelnen Test-Schritte können dann in einem weiteren Tab eingesehen werden. Zu den Test-Schritten können neben üblichen Unit-Tests auch Stylechecks gehören. Ein Stylecheck kann zum Beispiel überprüfen, ob der geschriebene Code nach den für die Programmiersprache üblichen Konventionen formatiert ist. Die Evaluation einer Aufgabe kann beliebig oft wiederholt werden. Wenn alle Tests der Aufgabe bestanden wurden, wird der Task mit einem grünen Haken versehen. So sieht der Schüler in der Assignment-Ansicht auf einen Blick, welche Aufgaben er bereits gelöst hat.

3.2.2.3 Ablauf für Lehrende

Lehrende haben in Code FREAK die Möglichkeit Aufgaben im Task Pool zu erstellen. Im Task Pool können beliebig viele Tasks angelegt werden. Bei der Aufgabenkonfiguration ist es neben dem Erstellen der Anleitung auch möglich die einzelnen Test-Schritte festgelegzulegen. Außerdem können mehrere nacheinander ablaufende Testabläufe erstellt werden. So kann beispielsweise neben dem Ergebnis auch die Code-Qualität getestet werden.

Anschließend können Lehrende Assignments erstellen, welche aus den vorher angelegten Tasks im Task Pool bestehen. Der Task Pool hat den Vorteil, dass mehrere Kurse dieselben Aufgaben anbieten können, ohne diese redundant anlegen zu müssen. Dieses Vorgehen erhöht die Wartbarkeit der Aufgaben ungemein. In der Einzelansicht eines Kurses können Dozenten eine optionale Deadline, sowie eine maximale Bearbeitungsdauer angeben. Außerdem können Ergebnisse und Bewertungen der Studierenden als .CSV-Datei exportiert werden. Die Lösungsversuche der Studierenden können wiederum als .ZIP- oder als .TAR-Datei heruntergeladen werden.

3.2.2.4 Architektur

Code FREAK wird als fertiger Docker-Container ausgeliefert. Docker ist eine Software, um mithilfe von Containervirtualisierung einzelne Anwendungen voneinander zu isolieren. Durch die Containervirtualisierung werden unter anderem viele Abhängigkeits-, Sicherheits-, Netzwerk- und Einrichtungsprobleme beseitigt. Der Container enthält alle Bibliotheken und Abhängigkeiten, die die Software für die Laufzeit benötigt (Mouat, 2015). Docker entstand auf Basis von Linux-Containern, welche ein oder mehrere Prozesse vom restlichen System isolieren können (Red Hat, 2018).

Durch diese Handhabung kann Code FREAK mit nur einem Befehl in der Kommandozeile installiert und gestartet werden. Die Software benötigt grundsätzlich nur einen Container. Benutzt ein Studierender jedoch die integrierte Online-Entwicklungsumgebung, muss für jede Instanz ein zusätzlicher Container mit der Laufzeitumgebung der IDE gestartet werden. Jeder weitere Container benötigt weiteren Arbeitsspeicher und erhöht dabei die Prozessorlast.

3.2.2.5 Probleme beim Einsatz für die OTH

Beim Einsatz von Code FREAK an der Hochschule Regensburg gibt es einige Probleme. Das erste Problem bezieht sich auf den vorher erwähnten Arbeitsspeicher. Die Praxis zeigt, dass eine Instanz der Online-IDE schon nach wenigen Quellcodedateien rund drei Gigabytes an Arbeitsspeicher verwendet. Um eine reibungslose und parallele Nutzung für alle Studierenden des Kurses gewährleisten zu können, werden dementsprechend sehr hohe Serverkosten fällig. (Kiel University of Applied Sciences, n. d. b)

Eine weitere Hürde ist die Stabilität der Software. Code FREAK befindet sich, Stand heute, mitten in der Entwicklung, weshalb einige Funktionen und Features noch nicht ordnungsgemäß funktionieren. Darunter die vorher angeworbene LMS-Integration (Kiel University of Applied Sciences, n. d. a). Zum heutigen Zeitpunkt ist LDAP die einzige Möglichkeit sich mit dem System zu authentifizieren. Ein eigenes Anmeldeverfahren gibt es bisher noch nicht.

LDAP steht für Lightweight Directory Access Protocol und ist ein Netzwerkprotokoll zur Durchführung von Abfragen und Änderungen in einem verteilten Verzeichnisdienst. LDAP ist der De-facto Industriestandard für Authentifizierung und Autorisierung. (LDAP, n. d.)

Der Kurs Digital Skills soll den Teilnehmern einen Überblick über den Arbeitsalltag eines Informatikers geben. Dazu gehört unter anderem die Kommandozeile. Eine Anforderung der Lernplattform ist deshalb, dass man (optional) neue Aufgaben herunterladen und Lösungsversuche über Befehle in der Kommandozeile testen und abgeben kann. In Code FREAK kann man Aufgaben lediglich über die Oberfläche testen und abgeben.

3.2.3 GitHub Classroom

3.2.3.1 Allgemeines

GitHub Classroom ist ein weiterer Kandidat für den Einsatz an der Hochschule Regensburg. Die Plattform ermöglicht die automatisierte Erstellung von Repositories auf GitHub. Außerdem hilft Classroom dabei, Aufgabenvorlagen und dazugehörige Abgaben einfach zu verwalten und automatisch zu benoten. Dabei enthalten die von GitHub Classroom erstellten Aufgaben-Repositories bereits vorkonfigurierte Zugriffskontrollen. (GitHub Inc., n. d. c)

3.2.3.2 Ablauf für Studierende

Studierende bekommen pro Programmieraufgabe einen Einladungslink. Nach Annahme der Einladung wird pro Studierenden automatisch ein Repository für die jeweilige Aufgabe angelegt. Dieses Repository enthält die Vorlage, welche zum Bearbeiten der Aufgabe benötigt wird.

Sobald der Studierende eine Lösung zur Korrektur abgeben möchte, kann er per Push die Änderungen in das Remote-Repository hochladen. Je nach Konfiguration der Aufgabe starten daraufhin serverseitig ein oder mehrere Tests. Wenn alle Tests bestanden sind, hat der Studierende die Aufgabe erfolgreich abgeschlossen.

3.2.3.3 Ablauf für Lehrende

Um einen Aufgabe in GitHub Classroom zur Verfügung zu stellen, bedarf es zuerst einer GitHub Organisation, sowie einem Kurs in GitHub Classroom. Sobald beides erstellt ist, können Lehrende neue Assignments (Aufgaben) mithilfe von bestehenden Vorlage-Repositories erstellen. In der Vorlage befindet sich in der Regel ein Ordner mit Tests, welche das Ergebnis des Programms prüfen sollen. In GitHub Classroom kann demnach eine Reihe an Kommandozeilenbefehle festgelegt werden, die dann diese Tests ausführen und je nach Ergebnis bepunkten.

Eine Übersicht über die Abgaben und erreichten Punktzahlen der Studierenden, kann sich der Dozent jederzeit beim Klick auf eine Aufgabe anzeigen lassen. Gleichzeitig ist es in dieser Ansicht möglich alle Noten als .CSV-Datei und alle Repositories als .ZIP-Datei herunterzuladen.

3.2.3.4 Architektur

GitHub Classroom ist ein Bildungsservice der Firma GitHub Inc. und ist Stand heute kostenfrei (Arelia Jones, 2020). Die Software basiert auf der Automatisierung von Repositories. Jeder bei GitHub registrierte Nutzer, kann einen sogenannten *Classroom* erstellen und darin Aufgaben auf Basis vorhandener öffentlichen GitHub-Repositories erstellen.

3.2.3.5 Probleme beim Einsatz für die OTH

Es gibt keine Möglichkeit das System von GitHub Classroom auf einen lokalen Git-Server zu replizieren. Aufgrund dessen schafft man sich durch die Verwendung von Classroom eine externe Abhängigkeit an GitHub. Dies kann unter Umständen zu erheblichen Problemen führen, wenn der Dienst beispielsweise nicht erreichbar oder aufgelöst wird. Beides ist durch die Größe und Infrastruktur des Unternehmens nicht (häufig) zu erwarten.

Eine weitere Hürde ist der Bedarf an weiterer Software. GitHub Classroom alleine ist nicht ausreichend, um als vollständige Lernplattform für den Kurs Digital Skills geeignet zu sein. Hier bietet es sich an, eine eigene statische Website mit Anleitungen und Erklärungen zu bauen, welche dann jeweils auf Classroom Einladungslinks verweist. Als Online-Entwicklungsumgebung kann der Dienst Replit verwendet werden. In Replit ist es möglich, ein vorhandenes Git-Repository als Template (Vorlage) für eine neue Umgebung zu verwenden. Dieses Template könnte dann Hilfsprogramme für den Git-Workflow enthalten. Der Vorteil daran: Studierende bekommen erste Erfahrungen mit der Kommandozeile, müssen jedoch keine komplexen Git-Kommandos absetzen.

3.2.4 Bewertung und Entscheidung der vorhandenen Systeme

Die folgende Abbildung 4 visualisiert die erläuterten Systeme und bewertet diese anhand der vorher definierten Anforderungen (siehe Kapitel 2). Wie man der Grafik entnehmen kann, repräsentieren alle blau hinterlegten Zeilen die nichtfunktionalen Anforderungen. Alle rosa hinterlegten Zeilen wiederum repräsentieren die funktionalen Anforderungen aus der Sicht des Teilnehmers. Der letzte Teil, welcher gelb hinterlegt ist, spiegelt schließlich die funktionalen Anforderungen aus der Sicht der Lehrenden wider.

Die erste Spalte enthält die Namen der Anforderungen, während die zweite Spalte eine jeweilige Gewichtung der Anforderung enthält. Der valide Bereich dieser Spalte befindet sich

zwischen, einschließlich 0,00 und einschließlich 1,00. Eine 1,00 bedeutet, dass die Anforderung für den Einsatz an der Hochschule sehr wichtig ist. Je niedriger der Wert, desto unwichtiger die Anforderung.

Die Spalten drei, vier und fünf enthalten die Benotungen der Systeme zu den Anforderungen. Hierbei gilt ähnlich wie bei der Gewichtung: je höher der Wert, desto besser. Der valide Bereich befindet sich hier jedoch zwischen 0,0 und 3,0. Der Wert 0 bedeutet, dass das Feature nicht vorhanden ist, oder nicht zutrifft. Die Spalten enthalten hierbei das Ergebnis der Multiplikation aus der Bepunktung mit der in der zweiten Spalte enthaltenen Gewichtung.

Anforderungen:
Blau: Nichtfunktionale Anforderungen
Rosa: Funktionale - Studierende
Gelb: Funktionale - Lehrende

Punkte:
0: Nicht vorhanden
1: Mangelhaft
2: Befriedigend
3: Gut

Gewichtung:
0,00: Unwichtig
1,00: Sehr wichtig

Anforderung	**Gewichtung**	**submit50**	**GitHub Classroom**	**Code FREAK**
Open Source	0,50	0,5	0	1,5
Stabilität	1,00	3	3	1
Lokales Hosting möglich	0,70	0,7	0	1,4
Kosten bzw. benötigte Ressourcen	1,00	3	3	3
Moodle-/LMS-Integration	0,80	0	2,4	0,8
Anzahl möglicher Programmiersprachen	0,80	0,8	2,4	2,4
Intuitiv für Nicht-Informatiker nutzbar	0,80	0,8	1,6	2,4
Fortschritts-Übersicht	0,80	1,6	1,6	2,4
In-Browser-IDE	1,00	3	3	1
Unabhängigkeit IDE	1,00	3	3	2
Konstruktives Feedback	1,00	3	3	3
Code Style Check	0,80	2,4	1,6	2,4
Einfache Abgabe	0,80	1,6	1,6	2,4
Abgabemöglichkeit über Commandline	1,00	3	3	0
Versuchsverlauf	0,80	2,4	2,4	1,6
Aufwand neue Aufgaben zu erstellen	1,00	1	2	2
Deadline-Funktion	1,00	0	3	3
Aufgaben vorläufig unsichtbar machen	0,30	0	0	0,9
Mehrere Administratoren	0,50	1,5	1,5	0
Download aller Abgaben eines Kurses	1,00	2	3	3
Aufgaben editierbar	1,00	3	3	3
Liste der Aufgaben sortiert nach Kurs	0,50	0	0	0
Fortschrittsübersicht der Studierenden	0,80	1,6	1,6	2,4
Summe		**37,9**	**45,7**	**41,6**

Abbildung 4: Vergleich und Benotung der Systeme
Quelle: Eigene Darstellung

Zusammenfassend befinden sich in der letzten Zeile die Summen der Benotungen. In diesem Fall hat das System mit GitHub Classroom die meisten Punkte erreicht und wird somit als

geeigneter Kandidat für die Programmierplattform des Zusatzstudiums Digital Skills weiter evaluiert.

3.3 Finale Architektur Online-Learning Platform

3.3.1 Tutors als Aufgabensammlung

Das freie Open-Source-Projekt Tutors ist eine Sammlung von Softwarepaketen, die entwickelt wurden, um Online-Kurse mit Vorlesungen, Übungen, Videos und Kursmaterialien zu erstellen und durchzuführen. (Tutors Team, n. d.)

Durch die Funktionalität Kursmaterialien und Übungen zu erstellen, dient Tutors ideal als Aufgabensammlung des Programmierkurses. Auf der Plattform kann neben dem Programmierkurs auch der gesamte Inhalt des Zusatzstudiums eingeteilt in Semester und Module hochgeladen werden. Im Programmierkursmodul verlinkt jede Aufgabe jeweils auf die GitHub-Classroom-Aufgabe.

Neue Anleitungsseiten können mithilfe der in Informatik üblichen Sprache Markdown angelegt und präsentiert werden.

3.3.2 Replit als Online-Entwicklungsumgebung

Der Programmierkurs soll für jeden Teilnehmer ohne komplizierte Installationen durchführbar sein. Aus diesem Grund wurde die Entscheidung gefällt, eine Online Entwicklungsumgebung einzusetzen.

Der Vorteil an Replit ist, dass man dem Studierenden durch ein Vorlagenrepository alle nötigen Konfigurationen und Programme im Vorhinein bereitstellen kann. Sobald ein sogenanntes *Repl* (Bezeichnung für ein Projekt in Replit) mit der erwähnten Vorlage erstellt wurde, kann der Studierende ohne Installation geräteübergreifend online programmieren (Replit Docs, n. d. b). Der Teilnehmer muss daraufhin lediglich auf den „Run"-Knopf drücken, welcher die später näher erläuterte *OTH-Console* startet und schließlich alle benötigten Abhängigkeiten bereitstellt.

3.3.3 GitHub Classroom als Abgabe- und Bewertungssystem

GitHub Classroom eignet sich, für den Einsatz an der Hochschule, vor allem durch seine Flexibilität und Einfachheit. Durch diese Flexibilität ist es möglich, GitHub Classroom nur als eine austauschbare Komponente des Systems zu sehen. Classroom übernimmt im System die Rolle

des Aufgabenservers. Hier werden alle Aufgabenvorlagen, sowie alle Versuche der Studierenden gespeichert und bewertet.

Sollte diese Komponente des Systems ausfallen, besteht weiterhin Replit als Online-IDE, sowie Tutors mit den jeweiligen Anleitungen. Dadurch muss lediglich ein adäquater Ersatz für GitHub Classroom gefunden und installiert werden.

4 Konfiguration und Implementierung

4.1 Tutors als Aufgabensammlung

Da der Programmierkurs nur ein Teil eines ganzen Modulkatalogs des Kurses Digital Skills ist, übernimmt die Konfiguration von Tutors die hierfür zuständigen wissenschaftlichen Mitarbeiter des Zusatzstudiums.

Für die Einpflegung der Aufgaben werden lediglich Anleitungen sowie jeweils dazugehörige Thumbnails (Vorschaubilder) benötigt. Die Anleitungen werden im standardisierten und weit verbreiteten Format Markdown verfasst.

Anleitungen können mit verhältnismäßig wenig Aufwand verfasst werden und schließlich als `README.md`-Datei im Aufgaben-Repository abgelegt werden. Dies hat den Vorteil, dass die Anleitung auch in der Versionskontrolle der Aufgabe enthalten sind. Außerdem können GitHub und Replit beim Klick auf die Aufgabe die Anleitung neben Tutors auch zusätzlich formatiert anzeigen.

4.2 GitHub Classroom

4.2.1 Konfiguration

Für die Einrichtung wird eine GitHub Organisation erstellt. GitHub Organisationen können von jedem Nutzer GitHub-Nutzer erstellt werden und benötigen lediglich einen Namen und eine Kontakt-E-Mail-Adresse (GitHub Inc., n. d. b). Die Organisation dieses Kurses beherbergt alle Aufgabenvorlagen, die später näher erläuterte Vorlage für Replit, sowie alle Aufgabenrepositories der Studierenden.

4.2.2 Erste Aufgaben

Nach der Erstellung der OTH-Organisation können die Aufgabenvorlagen erstellt werden. Aufgabenvorlagen sind in GitHub Classroom normale Repositories, welche in GitHub als Template markiert wurden. Sie beinhalten meist zusätzlich Unit-Tests, um den Code darin zu prüfen.

Sobald die Organisation mit Aufgaben gefüllt ist, kann der Classroom für den Kurs ange-

legt werden. Anfangs ist ein Classroom, wie die Organisation auch, leer. Über die Oberfläche können neue Assignments erstellt werden. Assignments sind Aufgaben, welche dem Studierenden zur Verfügung stehen. Für jedes vorher angelegte Aufgabenrepository wird ein Assignment erstellt. Bei der Erstellung gibt man verschiedene Konfigurationsparameter an. Dazu gehört die Auswahl, ob eine Aufgabe von Einzelpersonen, oder einer Gruppe bearbeitet werden kann, oder ob die jeweiligen Versuche für alle Studierenden oder nur für die Lehrer sichtbar sind. Ferner gibt es die Möglichkeit eine Deadline, sowie den Starter Code anzugeben. Für den Starter Code wird in diesem Fall jeweils das Aufgabenrepository ausgewählt. (GitHub Inc., n. d. a)

4.2.3 Tests und Benotung

Im nächsten Schritt legt man die Benotung und das Feedback fest. Das Autograding (die Benotung) geschieht über Kommandos in der Konsole. In unserem Fall beinhaltet jedes Aufgabenrepository einen Ordner mit pytest-Tests. Pytest ist eine Code-Test-Bibliothek für Python. Die Assignments wurden so konfiguriert, dass GitHub nach jedem Push zum Repository des Studierenden die pytest-Tests gestartet werden. Wenn alle Tests erfolgreich sind, erhält der Studierende eine vorkonfigurierte Punktzahl. Durch Classroom ist es außerdem möglich, jedem Test eine individuelle Punktzahl zuzuweisen. So kann man dem Schüler neben der erfolgreichen Ausführung beispielsweise noch Bonuspunkte für das Fangen von nicht geplanten Eingaben vergeben. (GitHub Inc., n. d. a)

4.3 Erstellung eines Replit-Starter-Templates

4.3.1 Template-Repository

Projekte in Replit heißen Repls. Ein Projekt ist in Replit ein vollumfänglicher Arbeitsbereich, um neue oder bestehende Software zu entwickeln. Neben einer Dateiübersicht, einem Texteditor und einer Konsole enthält der Arbeitsbereich viele weitere Funktionen. Dazu zählt unter anderem eine eingebaute Oberfläche für die Versionskontrolle git, ein Debugger, eine lokale Key-Value-Datenbank, private Umgebungsvariablen, sowie ein spezieller Bereich für Unit-Tests.

Repls können auf Basis von bestehenden GitHub Repositories erzeugt werden. Durch den Import eines Repositories, werden alle im Repository vorhandenen Dateien in den neuen Arbeitsbereich kopiert. Diese Funktionalität ermöglicht das Bereitstellen von Hilfsprogrammen und Dateien, die die Bearbeitung der Kursaufgaben erleichtern.

Aus diesem Grund wurde ein öffentliches Repository in der vorher erstellten GitHub Organisation angelegt. Dies ist das Template, welches später von den Studierenden als Starter-Vorlage verwendet wird. Replit versteckt Dateien, welche sich in einem Ordner namens `/node_modules` befinden. Normalerweise wird der Ordner automatisch im Kontext mit externen Modulen der JavaScript-Bibliothek Node.js verwendet (npm Inc., n. d.). Das ist auch der Grund weshalb Replit diesen Ordner automatisch versteckt. Dieses Verhalten nutzen wir, um Hilfsprogramme und Konfigurationen zu verstecken.

Studierende werden den ganzen Kurs in einem Repl absolvieren. Jede Programmieraufgabe wird als Ordner im Projekt-Repl abgespeichert werden. Um die Aufgaben herunterladen, prüfen und schließlich abgeben zu können, benötigt man git-Kenntnisse, sowie Erfahrungen mit Test-Frameworks, wie zum Beispiel pytest. Im `/node_modules`-Ordner des Starter-Templates befinden sich diverse Wrapper-Tools, welche dem Studierenden die Arbeit abnehmen und sie dabei unterstützen.

4.3.2 Wrapper-Tools (get, check, submit)

Die Wrapper-Tools `get`, `check` und `submit` sind Bash-Skripte. Über den Konsolenbefehl `get <PROJEKT-REPOSITORY>` kann der Studierende die Aufgabe in seinen Arbeitsbereich laden. Das Skript durchsucht die im Code definierte Organisation nach einem Repository mit dem Namen und lädt es schließlich über den Befehl `git clone` in den Arbeitsbereich.

Der Befehl `check <PROJEKT-REPOSITORY>` erlaubt es die Aufgabe auf Fehler zu überprüfen. In jedem Aufgabenrepository befindet sich eine `.language`-Datei, welche die für die Aufgabe verwendete Programmiersprache enthält. Das `check`-Skript liest die `.language`-Datei aus und entscheidet daraufhin, welche Befehle zum Ausführen der Tests nötig sind. Python ist die vom Zusatzstudium Digital Skills verwendete Programmiersprache. In diesem Fall installiert das Skript zuerst die nötigen Abhängigkeiten mit `pip3 install pytest -quiet`. Der Parameter `-quiet` verhindert für den Teilnehmer unübersichtliche Konsolenausgaben. Sobald das Testframework für Python installiert ist, führt das Skript den Befehl `pytest` aus und startet somit die Ausführung der Korrekturtests.

Schließlich pusht der Befehl `submit <PROJEKT-REPOSITORY>` den Lösungsversuch in das GitHub Classroom Aufgabenrepository des Kursteilnehmers. Dazu überprüft das Skript zuerst, ob es in der Zwischenzeit Änderungen am Repository gab und lädt diese herunter. Anschließend erstellt das Skript einen automatisierten Commit und lädt diesen in das Remote-Repository in die GitHub-Organisation des Kurses hoch.

Damit die Skripte Ordnerunabhängig ausgeführt werden können, werden Konsolenaliase benötigt. Konsolenaliase können in einer Bash-Konsole beispielsweise über das Anhängen folgender Zeile an die `.bashrc`-Datei erstellt werden: `alias befehl=ëcho Hallo"`. Bash ist eine Art „Standard-Shell“ unter Linux und wird auch von Replit als Konsole eingesetzt (ubuntu Deutschland e.V., n. d.). Bei jeder neuen Konsolensitzung wird dann der Alias aus der Datei eingelesen und angewendet. Die genannte Datei befindet sich in der Regel im Benutzerverzeichnis, welches außerhalb des Arbeitsbereiches in Replit liegt. Alle Änderungen außerhalb des Arbeitsbereiches werden jedoch von Replit nach jeder Sitzung zurückgesetzt. Um dieses Problem zu beheben, wurde die nun folgende OTH-Console eingeführt.

4.3.3 OTH-Console

Sobald der Student in seinem Repl auf den Run-Knopf drückt, startet die sogenannte OTH-Console in der Konsole. Dies ist eine neue modifizierte Konsoleninstanz, welche alle für die Arbeit benötigten Konfigurationen enthält.

Sobald der Run-Knopf gedrückt wird, startet das Einrichtungsskript `setup.py`, welches die benötigten Dateien in das Benutzerverzeichnis schreibt.

Zuerst wird eine Konfigurationsdatei für Bash angelegt. In diese werden alle benötigten Aliase (`get`, `submit`, `check` und `github`) geschrieben. Außerdem wird GitHub, falls noch nicht vorhanden, zu den sogenannten „Known Hosts“ im SSH-Ordner hinzugefügt. Der Vorteil daran ist, dass der Student bei der ersten Verbindung mit GitHub (bspw. durch den `get`-Befehl) keine Authentizitätsprüfung bestätigen muss (ssh.com, n. d.). Als letztes wird in der Konfigurationsdatei noch das Aussehen des Bash Promptes festgelegt.

Im nächsten Schritt wird die passwortlose Authentifizierung mit GitHub eingerichtet. Hierzu benötigt man ein SSH-Schlüssle-Paar, welches automatisch, falls nicht vorhanden, durch das Einrichtungsskript erzeugt wird. Nach der Erzeugung wird es neben dem SSH-Ordner auch in die Repl-Nutzer-Datenbank geschrieben. Die Datebank ist ein simpler Key-Value-Speicher, welcher jeweils pro Replit-Projekt existiert (Replit Docs, n. d. a). Sobald der Studierende Replit neustartet und das Benutzerverzeichnis gelöscht wurde, holt sich das Einrichtungsskript die SSH-Keys aus der Datenbank und schreibt sie wieder zurück in die jeweiligen Dateien. Dasselbe Verfahren wird für die Konfiguration von Git angewandt. Git benötigt, um Änderungen zu pushen, einen Namen mit zugehöriger E-Mail Adresse (Software Freedom Conservancy, n. d.). Diese Daten werden zusammen mit den SSH-Keys in der Replit-Datenbank gespeichert.

Nach der Ausführung des Einrichtungsskripts, wird eine neue Bash-Konsolen-Instanz, mit

der gerade angelegten Konfigurationsdatei als Parameter, gestartet.

4.3.4 SSH-Keys

Um das vorher generierte SSH-Schlüsselpaar für die Authentifizierung gegen GitHub zu verwenden, muss der öffentliche Schlüssel noch zu dem GitHub Profil des Studierenden hinzugefügt werden. Hierfür enthält das Starter-Template ein weiteres Programm, welches mit dem Befehl `github` in der OTH-Console ausgeführt werden kann. Dieses weitere Pythonprogramm lädt den im SSH-Ordner gespeicherten öffentlichen Schlüssel und gibt ihn zusammen mit einem Link zum Hinzufügen von SSH-Keys in GitHub aus.

Des Weiteren überprüft das Programm, ob Git bereits konfiguriert ist. Um Commits zu erstellen, benötigt Git einen Namen sowie eine E-Mail-Adresse. Sind die benötigten Werte nicht in der Replit-Datenbank vorhanden, fragt das Programm den Nutzer nach dem Namen und der studentischen E-Mail-Adresse. Nach gültiger Eingabe der Daten werden diese Werte in der nutzerspezifischen Replit-Datenbank des Arbeitsbereiches hinterlegt. Damit Git die Werte übernehmen kann, werden sie, wie beim Einrichtungsskript auch, in eine dafür vorgesehene Git-Konfigurationsdatei geschrieben.

5 Studie: Test an fachfremden Studierenden

5.1 Allgemein

In einer Studie sollen Fehler und schwierig gestaltete Stellen der Programmierplattform gefunden und analysiert werden. Damit die Studie möglichst realitätsnah ist, fällt die Wahl hierbei auf eine Feldstudie mit fachfremden Studierenden.

Nach Vollendung der Tests sollen Schwachstellen und Lücken der Programmieraufgaben (Anleitungen, Fehlerbeschreibungen, Aufbau, Schwierigkeit, ...) gefunden, behoben und für zukünftige Aufgaben berücksichtigt werden. (Huber, 2022)

5.2 Methode

Um für jeden Durchlauf gleiche Testbedingungen sicherzustellen, wird ein Testkonzept ausgearbeitet. Bei jeder Durchführung wird sich auf die Regeln und den Ablauf des Testkonzepts bezogen.

Das Testkonzept legt im Grunde die Meta-Daten der Studie fest. Neben der Zielsetzung, der Zielgruppe und der Dauer des Tests wird in dem Konzept auch die Methodik festgelegt.

5.2.1 Teilnehmende

Die Zielgruppe der Studie lässt sich durch wenige Parameter definieren. Sie ist äquivalent zur Zielgruppe des Zusatzstudiums Digital Skills. Gesucht sind folglich Studierende, welche keine Studiengänge der Fakultät Informatik und Mathematik belegen. Durch diese Einschränkung qualifiziert man sich für die Teilnahme am Zusatzstudium Digital Skills und dadurch implizit auch für die Anteilnahme an der folgenden Studie.

Um eine gleichmäßig verteilte Stichprobenmenge zu erhalten, werden mindestens fünf Testpersonen aus fünf unterschiedlichen Studiengängen für die Durchführung benötigt. Wie in Tabelle 1 absolut dargestellt, ist das Geschlechterverhältnis mit zwei Drittel männlichen Teilnehmern sehr ausgeglichen. In folgender Tabelle ist neben dem Geschlecht auch das jeweilige Alter der Testpersonen, sowie der Studiengang und das Semester visualisiert.

Geschlecht	Alter	Studiengang	Semester
Weiblich	22	Betriebswirtschaftslehre (B.A.)	5
Weiblich	21	Bauingenieurwesen (B.Eng.)	5
Weiblich	22	Psychologie (M.Sc.)	3
Männlich	23	Brauwesen und Getränketechnologie (M.Sc.)	1
Männlich	23	Regenerative Energietechnik (B. Eng.)	6

Tabelle 1: Studienteilnehmer:innen

5.2.2 Ablauf des Tests

Die Studienbefragungen finden zwischen dem 11. Februar 2022 und dem 6. März 2022 statt. Die Tests werden als Interview online über eine Software für Videokonferenzen durchgeführt.

Am Anfang wird vom Beobachter der Durchführung der Studie, Andreas Huber, das Zusatzstudium Digital Skills anhand einer einseitigen Infografik vorgestellt.

Danach wird dem Teilnehmer die folgende Agenda vorgetragen und anschließend werden der Person vier Fragen gestellt. Die Fragen müssen jeweils mit Schulnoten von 1 bis 6 beantwortet werden. Sechs bedeutet hier immer eine gänzliche negative Neigung, während eins einer völligen Zusage entspricht. Die restlichen Noten bilden wie gewohnt Zwischenwerte.

Nachdem die Fragen von der teilnehmenden Person beantwortet wurden, muss diese die Bildschirmübertragung starten. Währen dessen notiert der Beobachter die Ergebnisse der Fragen in eine vorher angelegte Tabelle.

Der Teilnehmer wird anschließend darauf hingewiesen, dass die Studie der Think-Aloud-Methodik folgt. Die Think-Aloud-Methode ist eine Forschungsmethode, bei der der Testteilnehmer darum gebeten wird, seine Gedanken laut auszusprechen. Mit dieser Methode können mögliche Aufhänger und Probleme in den Anleitungen der Aufgaben leichter gefunden werden. Der Beobachter des Tests schreibt alle Anomalien kategorisiert nach Fortschritt und Teilnehmer des Tests auf.

Aufkommende Fragen werden können vom Beobachter beantwortet werden. Sollte eine Frage bzw. Aufgabe unlösbar erscheinen, gibt es am Ende jeder Aufgabe ein Lösungsvorschlag. Dieser sollte jedoch nur aufgeklappt werden, wenn keine realistische Chance der selbstständigen Lösung des Problems besteht.

Als Nächstes werden die im folgenden Kapitel 5.2.3 erläuterten Aufgaben durch den Testteilnehmer bearbeitet. Während der Bearbeitung wird die dafür benötigte Zeit durch den Beobachter mit einer Stoppuhr gestoppt.

Abschließend kann der Studierende die Bildschirmübertragung wieder beenden und sich auf das Nachgespräch konzentrieren. Hierbei werden dem Teilnehmer noch fünf weitere Fragen gestellt. Die Fragen müssen wie vorher in Schulnoten von 1 bis 6 beantwortet werden.

Der Beobachter notiert, wie gewohnt, die Ergebnisse der Fragen in der vorher erwähnten Datentabelle.

Die Dauer des Tests wird zwar als groben Leitfaden auf 30 bis 60 Minuten festgelegt, soll jedoch keine harten Limits festlegen. Der Test soll möglichst unabhängig ablaufen und kann bei möglichen Unverständnissen auch deutlich länger dauern. Gemäß einer Feldstudie darf das Ergebnis der Studie nicht durch eine mögliche Manipulation, wie beispielsweise durch eine zeitliche Barriere, verfälscht werden.

Auf eine Entschädigung der Teilnehmenden wird aufgrund mangelnder Budgets verzichtet. Alle Teilnehmenden absolvieren die Studie freiwillig.

5.2.3 Materialien

5.2.3.1 Fragen

Folgende Fragen werden den Teilnehmenden vor Beginn der Bearbeitung gestellt:

- Hast du schon einmal erwägt eine Programmierausbildung anzustreben? (1: will ich definitiv noch machen; 6: noch nie) [**PRE1**]
- Hast du schon einmal programmiert? (1: ständig; 6: noch nie) [**PRE2**]
- Wie fit fühlst du dich am PC? (1: sehr fit; 6: gar nicht) [**PRE3**]
- Würde für dich das Zusatzstudium Digital Skills in Frage kommen? (1: unbedingt; 6: auf keinen Fall) [**PRE4**]

Nach Bearbeitung der Aufgaben müssen die Studierenden noch folgende fünf Fragen beantworten:

- Wie schwer kam dir die Einrichtung, bis zu dem Punkt, an dem du die erste Aufgabe heruntergeladen hast, vor? (1: sehr leicht; 6: sehr schwer) [**PAST1**]
- Hattest du Schwierigkeiten mit Aufgabe 1? (1: nein, keine; 6: zu komplex) [**PAST2**]
- Hattest du Schwierigkeiten mit Aufgabe 2? (1: nein, keine; 6: zu komplex) [**PAST3**]
- Hast du verstanden was du in den Aufgaben genau gemacht hast? (1: ja vollkommen; 6: nein, gar nichts) [**PAST4**]
- Würdest du nach Abschluss des Tests deine Meinung zur Frage am Interesse eines Zusatzstudiums für Digital Skills ändern? (1: unbedingt; 6: auf keinen Fall) [**PAST5**]

Alle aufgezeigten Fragen wurden durch den Beobachter und Initiator der Studie, Andreas Huber, selbst erfunden. Die in eckigen Klammern stehenden Bezeichnungen dienen zur Identifikation der jeweiligen Fragen für die Auswertung.

5.2.3.2 Aufgaben

Der Test besteht aus vier für die Studie vereinfachten Aufgaben. Die erste Aufgabe beschäftigt sich lediglich mit der Einrichtung des Arbeitsplatzes in Replit und GitHub-Classroom. Die Studierenden sind dazu aufgefordert, das in Kapitel 4.3 besprochene Replit-Template zu klonen und in Replit einzurichten. Dazu gehört das Hinzufügen des SSH-Keys zu GitHub, sowie das Herunterladen der ersten Aufgabe.

Die zweite Aufgabe ist eine Programmieraufgabe mit der Sprache Python. Es handelt sich hierbei um eine vereinfachte Version der später in produktiv eingesetzten Aufgabe „Lab 4: Hello“. Die Vorlage der Aufgabe frägt den Nutzer nach seinem Namen. Nach der Eingabe schließt sich das Programm wieder. Die Aufgabe des Teilnehmers ist nun den Namen in folgendem Format wieder auszugeben: `Hallo, mein Name ist <NAME>`. Diese Aufgabe kann der Teilnehmer mit einer einzigen Codezeile lösen. Die Ausgabe der Lösung sieht dann so aus:

```
Wie ist dein Name? Andreas
Hallo, mein Name ist Andreas!
```

Der Name ist wie vorher beschrieben flexibel und abhängig von der Eingabe des Nutzers.

Die dritte Aufgabe ist ebenfalls eine Programmieraufgabe mit der Sprache Python. In dieser Aufgabe geht es um die erste Verwendung einer `for`-Schleife. Am Anfang der Anleitung

wird sehr ausführlich erklärt was eine `for`-Schleife ist, weshalb man sie braucht und wie man sie anwendet. Danach wird die Aufgabenstellung erklärt, welche ähnlich zur zweiten Aufgabe ist. Dieses mal wird der Nutzer nach Start des Programms nicht nur nach seinem Namen gefragt, sondern auch nach der Anzahl, wie oft der Name ausgegeben werden soll. Wie auch in der vorherigen Aufgabe schließt sich standardmäßig das Python-Skript gleich wieder. Der Testteilnehmer muss nun eine Schleife programmieren, dass der Name mit der aktuellen Zählervariable n-mal ausgegeben wird. Der Name und die Anzahl an Ausgaben wird wieder per Funktion als Parameter übergeben. Die Ausgabe in der Konsole soll wie folgt aussehen:

```
Wie ist dein Name? Andreas
Wie oft soll der Name ausgegeben werden? 5
Andi 0
Andi 1
Andi 2
Andi 3
Andi 4
```

Falls die Aufgaben reibungslos funktioniert haben, kann der Studierende die Herausforderung einer optionalen Bonusaufgabe annehmen. Um diese Bonusaufgabe zu lösen, muss der Teilnehmer die vorherige Aufgabe modifizieren, dass die Ausgabe in der Konsole wie folgt aussieht:

```
Wie ist dein Name? Andreas
Wie oft soll der Name ausgegeben werden? 5
Andi 1
Andi 2
Andi 3
Andi 4
Andi 5
```

Der Unterschied hier ist die Nummerierung. Im Standardfall zählt das Programm von 0 bis n - 1. Um die Bonusaufgabe zu bewältigen, muss das Programm von 1 bis n zählen.

Die vierte und letzte Aufgabe beschäftigt sich mit der Generierung von Text-Pyramiden, welche der ursprünglichen Version des Videospiels Super Mario Brothers entsprechen sollen. Der Nutzer wird nach Start des Programms nach der gewünschten Größe bzw. Höhe der Pyramide. Nach Eingabe eines gültigen Werts liefert das Skript, ausgegeben mit Raute-Zeichen, die linke Seite einer Super Mario Pyramide. Die Ausgabe der gelösten Aufgabe sieht beispielhaft so aus:

```
Height: -1
Height: 0
Height: 6
     #
    ##
   ###
  ####
 #####
######
```

Das Beispiel zeigt neben der Pyramide auch, dass das Skript ungültige Größenangaben erkennen und ignorieren soll. Aufgrund der erhöhten Schwierigkeit dieser Aufgabe wurde entschieden, die letzte Aufgabe für die Studie zu verwerfen.

5.2.3.3 Test-Classroom

Für die Vorbereitung der Studie wird eine neue GitHub-Organisation, sowie ein neuer GitHub-Classroom angelegt. Sowohl das Replit-Template als auch die im vorherigen Kapitel beschriebenen Aufgaben werden als Template-Repositories in der Test-Organisation abgelegt.

Für die Aufgaben werden mithilfe des Test-Frameworks pytest Benotungstests programmiert und in GitHub-Classroom konfiguriert. Jede gelöste Aufgabe belohnt den Studienteilnehmer pauschal mit 10 Punkten.

Neben den Programmieraufgaben werden auch ausführliche Anleitungen mit Bilder geschrieben. Nach der Erstellung von Vorschaubildern können diese in der, auch später produktiv genutzten, Plattform Tutors hochgeladen werden. Die Studienteilnehmer erhalten zu Beginn des Probelaufs einen Link zur Übersicht der benötigten Anleitungen in Tutors.

5.3 Deskriptive Ergebnisse

Nach Verarbeitung der Ergebnisse der Fragen, können nun die einzelnen Resultate beobachtet und analysiert werden. Die folgende Tabelle 2 enthält in Spalte 1 Identifikationsbezeichnung der Frage. Spalte 2 und 3 beinhaltet den Durchschnitt und die dazugehörige Standardabweichung (SD). Die 4., 5. und 6. Spalte enthalten den Medianwert, das Minimum und schließlich das Maximum der Antworten.

Ergebnisse / **Fragen**	**Durchschnitt**	**SD**	**Median**	**Min.**	**Max.**
Alter	22.200	0.837	22.000	21	23
PRE1	3.600	2.300	3.000	1	6
PRE2	2.600	1.670	3.000	1	5
PRE3	2.600	0.894	2.000	2	4
PRE4	2.600	1.820	2.000	1	5
PAST1	2.600	1.520	2.000	1	5
PAST2	3.200	1.300	4.000	1	4
PAST3	3.800	1.790	5.000	1	5
PAST4	2.200	1.100	3.000	1	3
PAST5	3.800	2.170	3.000	1	6

Tabelle 2: Auswertung der Studienergebnisse

Wie man der Alterszeile entnehmen kann, sind alle Teilnehmenden im Schnitt 22 Jahre alt. Dieses Ergebnis lässt sich vor allem aus der Anforderung schließen, dass alle Teilnehmenden immatrikulierte Studierende sein müssen.

Die Ergebnisse der Fragen PRE1 und PRE2 deuten darauf hin, dass die Interessen der Teilnehmer sehr gemischt sind. Der Bereich zwischen Minimum und Maximum ist in beiden Fragen sehr hoch. Dies verdeutlicht, dass es sowohl Teilnehmende gibt, die bereits ständig mit Programmieren zu tun haben, als auch welche, die noch absolut keine Berührungen mit der Materie hatten.

Das Wohlbefinden am Computer wurde mit der Frage PRE3 abgefragt. Ein Interpretationsversuch der Minima und Maxima ist, dass die Teilnehmenden aus Angst später Fehler

zu machen niemals die Note 1 vergeben haben. Damit ist diese Frage auch die einzige mit Schulnoten zu bewertende Frage, welche in der Umfrage keine Note 1 als Minimum enthält.

Die Frage PRE4 bezieht sich auf das Interesse an einem Zusatzstudium für Digital Skills. Der Durchschnitt der Antworten auf diese Frage liegt bei 2,600, der Median noch besser bei 2,000. Das heißt, dass die die Mehrheit der Testpersonen, vor der Bearbeitung der Aufgaben, einer Teilnahme an einem solchen Zusatzstudium nicht abgeneigt wären. Nach der Absolvierung der Programmieraufgaben werden die Teilnehmer erneut nach ihrer Meinung bzw. einer Änderung dieser Einschätzung gefragt (Frage: PAST5). Hier wird anhand des Durchschnitts, sowie des Medians eine deutliche negative Tendenz ersichtlich. Durch die hohe Standardabweichung bei der Frage PAST5, kann man nicht auf eine vollständige Abneigung der Studierenden schließen. Trotzdem kann die negative Tendenz des Interesses auf zu schwere oder dürftig erklärte Aufgaben hindeuten. Ein weiterer Interpretationsversuch ist, dass die Studierenden nach der Präsentation des Zusatzstudiums andere Vorstellungen zu den Modulen hatten.

Die Fragen PAST1, PAST2 und PAST3 beziehen sich jeweils auf das Schwierigkeitsempfinden der Studierenden gegenüber den drei Programmieraufgaben. Die erste Programmieraufgabe ist in diesem Fall die Einrichtungsaufgabe des Arbeitsbereiches. Mit dem Design der Aufgaben wird versucht die Schwierigkeit mit jeder Aufgabe zu steigern. Die Durchschnitte der Umfrage bestätigen den Erfolg eines linearen Schwierigkeitsanstiegs. Während PAST1 einen Durchschnitt von 2,600 aufweist, scheint die Schwierigkeit bei der ersten Python-Programmieraufgabe (PAST2) mit einem Durchschnittswert von 3,200 deutlich höher zu liegen. Erneut gesteigert hat sich das Schwierigkeitsempfinden mit der dritten Aufgabe (PAST3) auf einen Durchschnittswert von 3,800. Die Differenz der Durchschnitte beträgt bei beiden Anstiegen genau 0,800 Notenpunkte.

Das allgemeine Verständnis der Aufgaben wird mit der Frage PAST4 abgefragt. Für diese Frage wurde die 3 als Maximalnote aufgezeichnet. Der Durchschnitt liegt wiederum bei 2,200 Notenpunkte und ist somit im positiven Bereich.

6 Zusammenfassung und Ausblick

Zusammenfassend kann man sagen, dass man zusammenfassend sagen kann, dass man zusammenfassend sagen kann, dass man zusammenfassend sagen kann, dass man zusammenfassend sagen kann, dass ich absolut keinen Bock mehr hab.

Abbildungsverzeichnis

Literaturverzeichnis

Arelia Jones. (2020). *Set up your digital classroom with GitHub Classroom*. Verfügbar 28. Januar 2022 unter https://classroom.github.com

Bundesministerium für Wirtschaft und Energie. (n. d.). *Digitalisierung in Deutschland – Lehren aus der Corona-Krise*. Verfügbar 10. März 2022 unter https://www.bmwi.de/Redaktion/DE/Publikationen/Ministerium/Veroeffentlichung-Wissenschaftlicher-Beirat/gutachten-digitalisierung-in-deutschland.pdf?__blob=publicationFile

GitHub Inc. (n. d. a). *Create an individual assignment*. Verfügbar 7. Februar 2022 unter https://docs.github.com/en/education/manage-coursework-with-github-classroom/teach-with-github-classroom/create-an-individual-assignment

GitHub Inc. (n. d. b). *Creating a new organization from scratch*. Verfügbar 7. Februar 2022 unter https://docs.github.com/en/organizations/collaborating-with-groups-in-organizations/creating-a-new-organization-from-scratch

GitHub Inc. (n. d. c). *GitHub Classroom*. Verfügbar 28. Januar 2022 unter https://classroom.github.com

Harvard University. (n. d. a). *CS50 GitHub*. Verfügbar 28. Januar 2022 unter https://github.com/cs50

Harvard University. (n. d. b). *CS50: Introduction to Computer Science*. Verfügbar 28. Januar 2022 unter https://pll.harvard.edu/course/cs50-introduction-computer-science

Harvard University. (n. d. c). *CS50's Introduction to Computer Science*. Verfügbar 28. Januar 2022 unter https://www.edx.org/course/introduction-computer-science-harvardx-cs50x

Harvard University. (n. d. d). *Online - CS50 Docs*. Verfügbar 28. Januar 2022 unter https://cs50.readthedocs.io/ide/online/

Harvard University. (n. d. e). *submit50 - CS50 Docs*. Verfügbar 28. Januar 2022 unter https://cs50.readthedocs.io/submit50

Huber, A. (2022). *Testkonzept - Probedurchlauf des Pilotprojekts DSCC* [Unveröffentlichtes Testkonzept].

Kiel University of Applied Sciences. (n. d. a). *Code FREAK Documentation :: Code FREAK Docs*. Verfügbar 28. Januar 2022 unter https://docs.codefreak.org/codefreak/index.html

Kiel University of Applied Sciences. (n. d. b). *Installation Guide :: Code FREAK Docs*. Verfügbar 28. Januar 2022 unter https://docs.codefreak.org/codefreak/for-admins/installation.html#_dedicated_docker_host

Kiel University of Applied Sciences. (2019-2020). *Code FREAK | Code Feedback, Review & Evaluation Kit*. Verfügbar 28. Januar 2022 unter https://codefreak.org/#about

LDAP. (n. d.). *LDAP.com - Lightweight Directory Access Protocol*. Verfügbar 28. Januar 2022 unter https://ldap.com

moodle.de. (n. d.). *Lernerfolg mit Moodle*. Verfügbar 28. Januar 2022 unter https://moodle.de

Mouat, A. (2015). *Using Docker: Developing and Deploying Software with Containers*. O'Reilly Media, Inc.

npm Inc. (n. d.). *folders - Folder Structures Used by npm*. Verfügbar 7. Februar 2022 unter https://docs.npmjs.com/cli/v7/configuring-npm/folders

Red Hat. (2018). *Was ist ein Linux-Container?* Verfügbar 31. Januar 2022 unter https://www.redhat.com/de/topics/containers/whats-a-linux-container

Replit Docs. (n. d. a). *Database FAQ*. Verfügbar 7. Februar 2022 unter https://docs.replit.com/hosting/database-faq

Replit Docs. (n. d. b). *Replit and GitHub: Using and contributing to open-source projects*. Verfügbar 7. Februar 2022 unter https://docs.replit.com/tutorials/06-github-and-run-button

Software Freedom Conservancy. (n. d.). *1.6 Getting Started - First-Time Git Setup*. Verfügbar 7. Februar 2022 unter https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

ssh.com. (n. d.). *SSH Host Key - What, Why, How*. Verfügbar 7. Februar 2022 unter https://www.ssh.com/academy/ssh/host-key

Tutors Team. (n. d.). *Tutors Open Source Project*. Verfügbar 5. März 2022 unter https://tutors.dev/

ubuntu Deutschland e.V. (n. d.). *Bash*. Verfügbar 7. Februar 2022 unter https://wiki.ubuntuusers.de/Bash/