Entwicklung und Implementierung eines Lernsystems für den Service GitHub als Progressive Webanwendung für Studierende der Informatik

Projektbericht für das Modul
Praxisprojekt MI Sommersemester 2024
im Studiengang Medieninformatik
an der Fakultät für Informatik und Ingenieurwissenschaften
der Technischen Hochschule Köln

vorgelegt von: Anouk Martinez
Matrikel-Nr.: 11154860
Adresse: Am Sandberg 28
51643 Gummersbach
anouk_olivia_elsa.martinez_wieczorek@smail.th-koeln.de

eingereicht bei: Prof. Dr. Hoai Viet Nguyen

Gummersbach, 06.09.2024

Kurzfassung/*Abstract*

0.1 Deutsche Kurzfassung

Im Rahmen des Praxisprojektes wurde ein Problemfeld hinsichtlich mangelnder Kompetenzen Studierender zum Thema Git und Github analysiert. Hierzu wurde unter Anderem eine Umfrage unter Studierenden der Informatik an der Technischen Hochschule Köln in der Fakultät 10 durchgeführt, um Daten zu möglichen Ursachen unter Studierenden zu finden.

Um einen möglichen Lösungsansatz umzusetzen und die Lernbarriere zu senken, wurde ein Entwurf für ein System entwickelt, mithilfe dessen Befehle und Arbeitsabläufe beigebracht werden sollen. Das System nutzt dafür eine spielerische Analogie, bei der der Nutzdende mithilfe von Befehlseingaben in einer Konsole einem Koch dabei helfen soll, Gerichte zuzubereiten und abzuschicken. Es wurden verschiedene Webtechnologien verwendet, um einen spielbaren Prototyp als progressive Webanwendung zu entwickeln.

0.2 English Abstract

Over the course of the practice project, a problem field regarding the lack of competencies of students in regard to Git and GitHub was analyzed. Among other methods, a survey was deployed among students in IT at the Technische Hochschule Köln in faculty 10, to collect data on possible causes among students.

To work out a possible solution and lower the barrier of entry, a draft was developed for a system that teaches commands and workflows. The system uses a playful analogy for this, in which the user helps a cook to prepare and send out meals. Different web technologies were then used to develop a playable prototype as a progressive web app.

Inhaltsverzeichnis

Abbildungsverzeichnis

Glossar

API Application Programming Interface (API). Eine Schnittstelle, die einen Datenaustausch zwischen verschiedenen Softwareanwendungen ermöglicht.. 10

Branches Abzweigungen einer Version in Versionskontrollsystemen. Auf einem Branch kann der Code parallel zum existierendem Hauptcode bearbeitet werden.. 9

Commit Vorgang in Versionskontrollsystemen wie Git, bei dem Änderungen an Dateien dauerhaft gespeichert werden. Ein Commit enthält eine Nachricht, die die Änderungen beschreibt, und ermöglicht das Verfolgen des Fortschritts und das Wiederherstellen früherer Versionen.. 2

Git Ein kostenloses Versionsverwaltungssystem, entwickelt von Linus Torvald.. 2

GitHub Eine Entwickler Plattform zum Speicher, Erstellen und Verwalten von Code.. 2

HTML Hyper Text Markup Language (HTML). Die standardisierte Auszeichnungssprache zur Strukturierung und Darstellung von Inhalten im Web.. 12

IDE Integrierte Entwicklungsumgebung. Bietet meist einen Text Editor, sowie fortgeschrittenere Features zur Syntax Überprüfung beim Programmcode schreiben.. 2

Local Verweis auf ein lokales Repository in Versionskontrollsystemen wie Git, das meist auf dem eigenem Rechner liegt.. 2

Merge Konflikte Konflikte, die beim Zusammenführen von verschiedenen Versionen des Codes passieren können, wenn parallel an der selben aktuellen Version gearbeitet wird.. 9

Remote Verweis auf ein externes Repository in Versionskontrollsystemen wie Git, das meist auf einem Server liegt.. 2

Repository Speicherort in Versionskontrollsystemen wie Git, der alle Dateien, Ordner und den gesamten Versionsverlauf eines Projekts enthält.. 2

SSH Key Kryptografischer Schlüssel, der für die sichere, passwortlose Anmeldung zwischen Computern über das Secure Shell Protokoll (SSH-Protokoll) genutzt wird.. 2

Story Modus Ein Modus des Systems, der eine Geschichte erzählt.. 20

1 Einleitung

Die Frage, wie sich Lernende freiwillig und effektiv mit Bildungsinhalten auseinandersetzen können, ist zentral für die Gestaltung moderner Bildungssysteme. Besonders in der Informatik besteht die Herausforderung darin, Studierende erfolgreich an neue Technologien heranzuführen.

Im Rahmen dieses Praxisprojekts im Fachbereich Medieninformatik wird untersucht, wie ein spielerisches Lernsystem Informatikstudierende dabei unterstützen kann, relevante Technologien effizient zu erlernen. Sowohl die zugrunde liegende Problemstellung als auch der entwickelte Lösungsansatz werden im Detail beschrieben

Zusätzlich werden die Ergebnisse einer projektbegleitenden Umfrage präsentiert, die Einblicke in die Bedürfnisse und Herausforderungen der Zielgruppe bietet.

Hinweis zur geschlechtergerechten Sprache

Alle Bezeichnungen von Personen werden durch eine Aktion formuliert, die die Person auszeichnet. Beispielsweise wird statt Nutzer das Wort Nutzende verwendet, oder statt Student das Wort Studierender. Diese Formulierungen soll stets alle Geschlechter umfassen.

Gummersbach, September 2024

2 Problemstellung

Zu Beginn des Studiums fällt es vielen Informatikstudierenden schwer, sich mit den Konzepten und Arbeitsweisen von Versionsverwaltungssystems vertraut zu machen (siehe Abschnitt 2.2). Mögliche Ursachen hierbei sind beispielsweise fehlende Motivation und Zeit, sich außerhalb der Vorlesungen mit dem Thema auseinanderzusetzen. Hinzu kommt oft ein Mangel an langfristigem Lerneffekt sowie technische Hürden, wie Probleme bei der Authentifizierung mit einem SSH Key.

2.1 Mögliche Ursachen des Problemfeldes

In der Vorarbeit zum Praxisprojekt wurden drei Hauptursachen für die genannten Schwierigkeiten identifiziert und analysiert. Anschließend wurde eine Umfrage unter Studierenden der Fakultät 10 der Technischen Hochschule (TH) Köln durchgeführt, um diese Hypothesen zu überprüfen.

2.1.1 Fehlende Motivation

Eine Ursache könnte in der geringen Eigenmotivation der Studierenden liegen, sich mit dem Thema auseinanderzusetzen. So erscheint das Erlernen von Werkzeugen wie Git für einige möglicherweise als aktuell irrelevant für ihren Studienstand. Viele Studierende verwenden zudem grafische Oberflächen in einer IDE wie Visual Studio Code, um Git und GitHub zu nutzen, ohne die zugrunde liegenden Funktionen wirklich zu verstehen. Dies könnte zu Bedienfehlern und Konflikten führen, beispielsweise wenn ein Commit Ablauf nicht korrekt durchgeführt wird.

2.1.2 Fehlende langfristiger Lerneffekt

Einige Studierende beschäftigen sich zwar selbstständig mit Git und GitHub, lesen jedoch lediglich Befehlsketten, ohne diese anzuwenden. Dadurch bleibt der langfristige Lerneffekt aus, und das grundlegende Verständnis der Konzepte von Remote und Local Repository wird oft vernachlässigt.

2.1.3 Technische Probleme

Eine weitere mögliche Ursache für den Wissensmangel sind technische Probleme. Manche Studierende haben keinen Zugriff auf einen Computer und können Git nur eingeschränkt nutzen. Andere verfügen zwar über die notwendige Hardware, haben jedoch Schwierigkeiten bei der Einrichtung von Git, insbesondere der Authentifizierung via SSH-Key, was den Lernprozess erheblich behindert.

Die Umfrage ergab, dass fast alle genannten Faktoren von zehn oder mehr Studierenden als Hindernisse empfunden wurden. Technische Probleme wurden jedoch nur siebenmal als Grund genannt, und fehlende Hardware war für keinen der Teilnehmenden ein Problem.

2.2 Prüfung der Thesen anhand einer Umfrage der Zielgruppe

Zur Überprüfung der oben genannten Thesen wurde eine Umfrage unter den Studierenden der Fakultät 10 der TH Köln durchgeführt. Der erste Teil der Umfrage sammelte demografische Daten der Teilnehmenden sowie Informationen zu ihren Erfahrungen mit den Versionsverwaltungssystemen Git und GitHub sowie gegebenenfalls weiteren Systemen. Abschließend wurden die Studierenden nach ihrer Meinung zu einem möglichen Lernsystem befragt.

Die vollständigen Ergebnisse der Umfrage sind im Anhang dieses Berichts zu finden. Im Projektbericht selbst werden nur die Abschnitte der Umfrage ausgewertet, die für die jeweilige Problemstellung relevant sind.

Die Umfrage wurde in deutscher und englischer Sprache durchgeführt. Da die Fragen in beiden Versionen identisch waren, wurden die Ergebnisse zusammengeführt und gemeinsam ausgewertet.

2.2.1 Git und GitHub Kenntnisse

In der Umfrage sollten primär die Gründe für mögliche Wissensmängel ausgearbeitet werden. Auf Grundlage dessen und der Wünsche der Studierenden sollte im weiteren Verlauf des Projektes dann ein System, zugeschnitten auf die Interessen der Zielgruppe, entwickelt werden.

Aus der Umfrage ließ sich schließen, dass alle Gründe, abgesehen von Hardwareproblemen, vergleichbar gleichmäßig ausgewählt wurden. Insgesamt konnte festgestellt

werden, dass nicht nur Schwierigkeiten bestehen, sich die Kenntnisse selbst beizubringen, sondern auch ein Mangel an Studierenden, die Git über die Konsole direkt verwenden. Relevant ist dies im Kontext, da über die grafischen Oberflächen möglicherweise die Zusammenhänge der Git-Kommandos nicht verstanden werden.

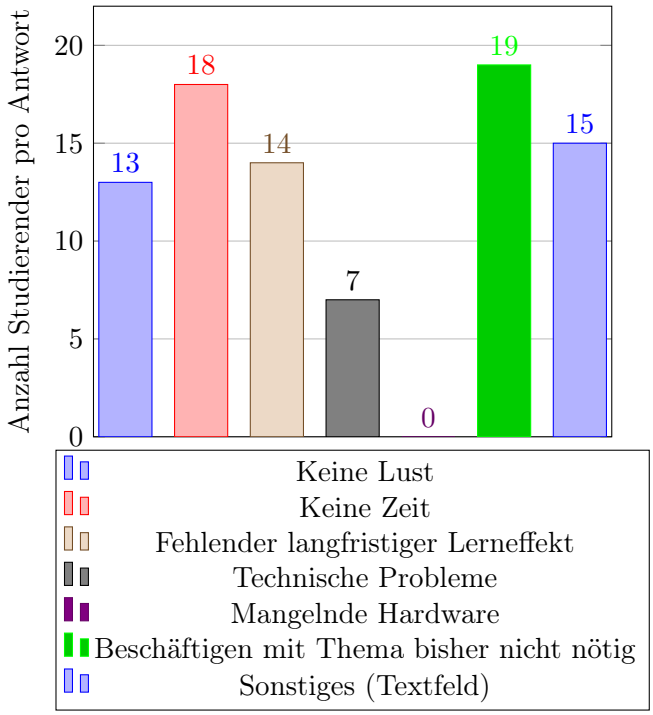

Abbildung 2.1: Grafische Darstellung der Ergebnisse zur Frage: Mögliche Gründe für Wissensmängel im Bereich

Generell bewerteten viele Studierende, trotz dessen, dass viele dieser Git bereits im Studium verwendet hatten, ihre Kenntnisse eher mittelmäßig bis niedrig.

Die wichtigste Frage, die mit der Umfrage analysiert werden sollte, ist jedoch die zum Interesse des Systems. Durch die Ergebnisse hat sich herausgestellt, dass die Mehrheit der Befragten generell ein Interesse an einem Lernsystem zum Thema Git und GitHub hätte.

Zudem ergab sich, dass tiefergehende Lerninhalte generell als wichtiger eingestuft wurden, als ein interessantes Setting mit ansprechendem Humor zu haben. Im Kontext des Projektes hatte dies zunächst wenig Einwirkung, da der Prototyp um das Konzept der Koch-Analogie strukturiert wurde.

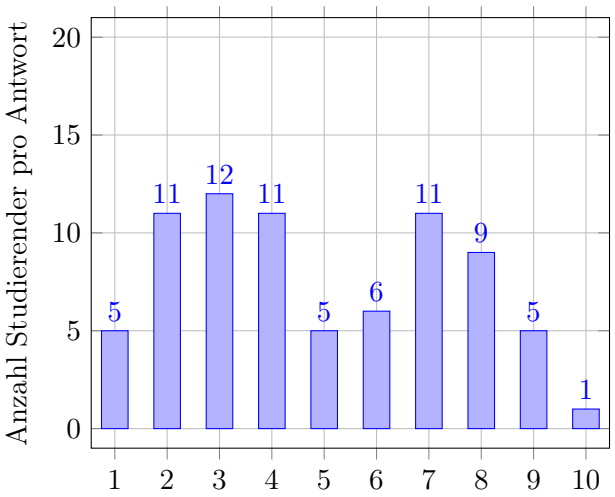

Abbildung 2.2: Grafische Darstellung der Ergebnisse zur Frage: Wissen von Git

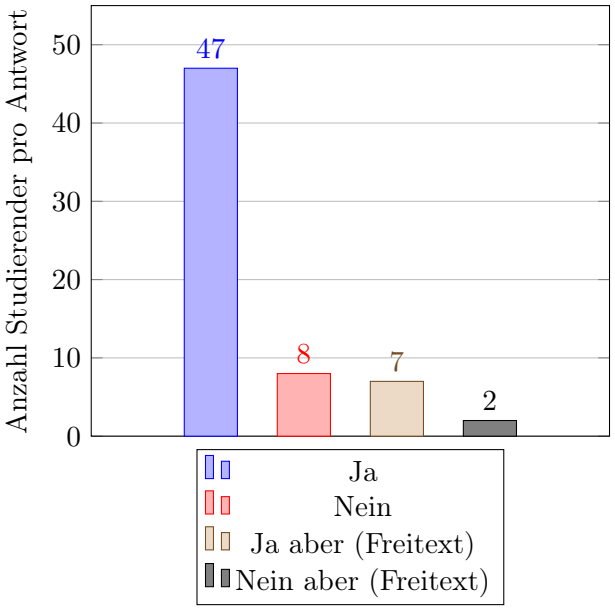

Abbildung 2.3: Grafische Darstellung der Ergebnisse zur Frage: Interessen an einem Lernsystem

Nähere Details zu den Antworten in den Freitextfeldern, sowie die restlichen Inhalte der Umfrageergebnisse befinden sich im Anhang des Projektberichtes.

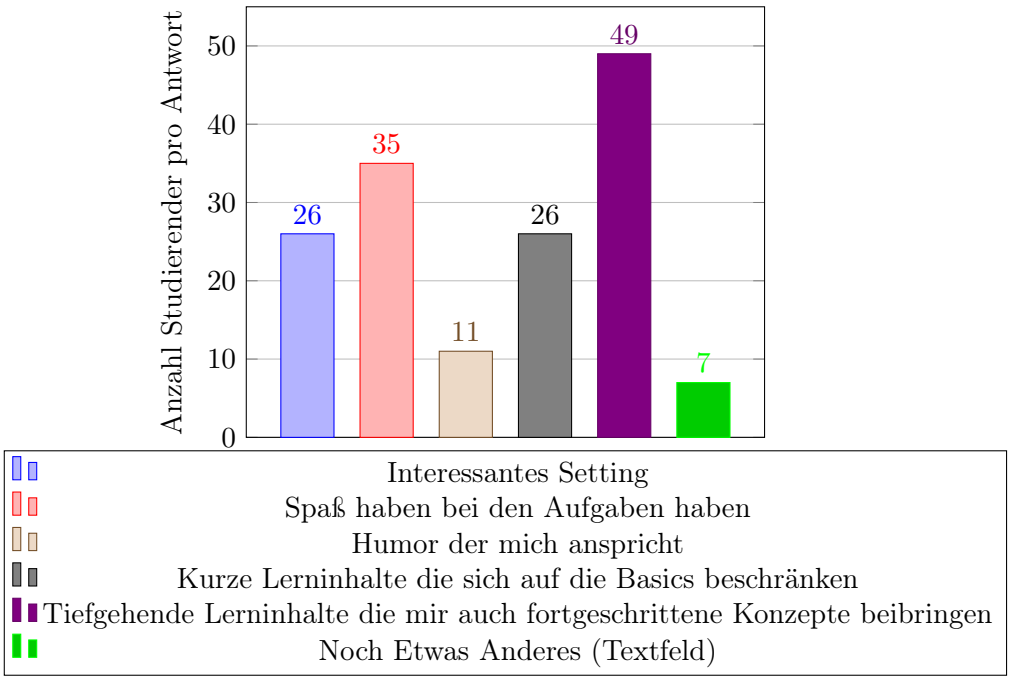

Abbildung 2.4: Grafische Darstellung der Ergebnisse zur Frage: Wünsche an einem Lernsystem

3 Lösungsansatz

Mithilfe der Informationen aus der Problemfeldanalyse wurde eine mögliche Lösung für das Problem entwickelt. Im Rahmen des Praxisprojektes sollte so ein Lernsystem für Git und GitHub entwickelt werden, in dem Studierende sich selbst Wissen aneignen können, und dieses auch direkt praktisch anwenden, um den Lerneffekt zu verbessern.

Die Frage, wie Gestaltungen eines Systems beim Lernen helfen kann, wurde bereits in den 1990er Jahren von verschiedensten Forschern untersucht. Unter anderem wurden hierbei von Thomas W. Malone Untersuchungen zum Zusammenhang von Computerspielen und Motivation untersucht. [1] Das oberste Ziel bei der Gestaltung eines Systems bleibt hierbei, dass Nutzende das System nicht aus Zwang nutzen, sondern aus eigener intrinsischer Motivation.

Dass Spiele beim Lernen von Inhalten ebenfalls effektiver sind, als herkömmliche Materialien, wurde ebenfalls bereits viel im Kontext des Feldes Gamification untersucht.

Mit dem Lernsystem für Git soll das Wissen daher ebenfalls mittels einer Art Spiel vermittelt werden, in dem die Konzepte um Git in eine leichter verständliche Analogie verpackt werden.

3.1 Hauptziele des Systems

Um herauszuarbeiten, was das System leisten muss, wurden zunächst die Phasen der Nutzung ausgearbeitet. Beispielsweise sollen diese sein, die Motivation, das System zu nutzen, das Verhalten beim Nutzen des Systems, und anschließend den Lerneffekt, der nachträglich gewonnen werden kann.

Potenzielle Konfliktpunkte des Systems bestehen darin, dass Studierende möglicherweise gar nicht erst zur Nutzung des Systems motiviert sind oder im Verlauf der Nutzung feststellen, dass sie Verständnisschwierigkeiten haben, was zu einem Abbruch der Nutzung aus Frustration führen könnte. Daher soll sichergestellt werden, dass Studierende sich mit den Inhalten auseinandersetzen möchten, andererseits allerdings auch, dass sie die Inhalte verstehen.

Motivation zur Nutzung : Studierende müssen sich grundsätzlich mit dem System auseinandersetzen wollen.

Nutzung allgemein : Das System muss einfach gestaltet werden, um Frustration bei der Nutzung zu vermeiden. Im schlimmsten Fall führen Konflikte zum Abbrechen der Nutzung des Systems.

Lerneffekt : Damit Studierende eine positive Meinung auf das System behalten und möglicherweise zu diesem zurückkehren oder es weiterempfehlen möchten, muss ein langfristiger Lerneffekt entstehen. Das System sollte zudem ein konsistentes Ende haben, das Nutzende gedanklich mit der Geschichte abschließen lassen.

Weiterhin wurde sich mit den nötigen Hauptfunktionalitäten des Systems auf funktioneller Ebene auseinandergesetzt. Hierbei wurden Fragen gestellt wie "Was braucht das System, um erfolgreich zu sein?", sowie "Wie kann die Zielgruppe effektiv angesprochen werden?".

Es wurden drei Schlüsselpunkte ausgearbeitet.

Motivation zum Selbstlernen : Damit das System Erfolg hat, müssen Studierende das System gerne benutzen. Dazu sollten Frustrationspunkte wie zu schwierige Aufgaben ohne Hilfestellungen oder verwirrende Bedienung vermieden werden.

Gesammeltes Wissen : Das System sollte im Idealfall nicht nur zum Lernen der Services dienen, sondern auch als eine Art Nachschlagewerk für alle Kommandos und Abläufe, die man zur Nutzung von Git benötigt. Hierbei werden sich auf die Befehle fokussiert, die direkt in der Kommandozeile ausgeführt werden.

Zielgruppe und Lerntypen ansprechen : Damit Studierende motiviert werden, das System zu benutzen, sollte dieses nicht nur einfach zu benutzen sein, sondern die Nutzenden auch auf persönlicher Ebene ansprechen. Hierzu wurde eine Analogie entworfen, die es den Studierenden einfacher machen soll, sich mit den Aufgaben zu identifizieren. Das System soll sich nicht so anfühlen, als wenn man Aufgaben erledigen muss, sondern eher wie ein Spiel, das neben einem langfristigen Lerneffekt den Nutzenden auch Spaß bereitet.

3.2 Konzept

Im Lösungsansatzteil wurde zunächst überlegt, wie man das System konzeptionell gestalten könnte. Hierbei wurden die Grundsätze zunächst technologiefrei entworfen. In späteren Überlegungen, insbesondere im Kontext der Proof of Concepts, wurde bereits grundsätzlich überlegt, welche groben Technologien für das System relevant sind.

3.2.1 Analogie

Die Analogie, die im Kontext der Praxisprojekts gewählt wurde, ist die eines Kochs, der mithilfe von Git Befehlen ein Gericht kocht und anschließend an einen Kunden schickt. Hierbei werden alle nötigen Befehle behandelt und in einzelnen Kapiteln aufgeteilt. Diese Kapitel umfassen in der derzeitigen Version des Systems:

1. Tutorial für die Steuerung des Spiels
2. Git-Commit durchspielen
3. Git Branches Erstellen und Verwalten
4. Merge Konflikte verstehen und lösen

Es wurden vor der Implementierung des Systems verschiedene theoretische Implementierungen dieser Analogie aufgestellt. Beispielweise wurde überlegt, wie man Branches im Kontext eines Restaurants darstellen könnte. Die Umsetzung der Analogie wurde zusammenfassend in einer Liste gegenübergestellt.

git clone Das kopieren eines Repositorys auf die lokale Maschine sollte im System noch relativ nah an der praktischen Anwendung sein, um Verwirrung zu vermeiden. In der visuellen Darstellung werden durch das erste Klonen alle benötigten Zutaten an den Arbeitsplatz geholt.

git add Das Hinzufügen einer Datei wurde mithilfe von Zutaten dargestellt. Hierbei sind drei Lebensmittel zur Verfügung gestellt, ein Blatt Salat, eine Tomate und Hackfleisch. Um einen Burger vollständig zusammenzustellen, werden dann alle Zutaten dem Commit oder visuell dem Burger hinzugefügt und dann verpackt.

git commit Das "Verpacken" eines Commits wird mithilfe des Burgers visualisiert. Sobald eine Bestellung vollständig hinzugefügt wurde, wird diese abgeschlossen.

git push Ein Push wird mit dem Abschicken der Bestellung gleichgesetzt.

Branches Verschiedene Branches sind im Kontext des Systems mit Arbeitsplatten gleichgesetzt. Hierbei können mehrere Köche zusammen an einem Arbeitsplatz arbeiten und sich gegenseitig zuvorkommen, oder an unterschiedlichen Arbeitsplatten arbeiten, und später zu der "Hauptplatte" zurückkehren.

Merge Konflikte Merge Konflikte können im System durch überschneidende Zutaten dargestellt werden. Beispielsweise bereitet ein Koch einen Burger ohne Zwiebeln zu, ohne das Wissen, dass in einem späteren Push die Bestellung doch noch aktualisiert wurde, und der Burger nun keine Zwiebeln mehr haben soll.

Im Verlauf der Implementierungsphase wurde die Analogie nicht mehr angepasst.

3.2.2 Bedienung

Das Spiel selbst soll von den Nutzenden über eine Art Mini-Konsole bedient werden, um einen starken Praxisbezug zu gewährleisten. Studierende sollen, wie sie es auch in einer realen Situation tun würden, über die Konsole die richtigen Befehle eingeben. Im System wird dieser Prozess durch eine Geschichte unterstützt, und es werden viele Hinweise und Erklärungen gegeben, dazu, welche Befehle eingegeben werden sollen. Mehr dazu, wie genau die Funktionsweise dieser Konsole sein soll, lässt sich in Abschnitt 3.3.1 finden.

In der Planung wurde zunächst überlegt, eine möglichst realitätsnahe Konsole auf dem Interface zu haben, die realistische Eingaben und Ausgaben entgegennimmt. Später wurde diese Darstellung allerdings vereinfacht. Es wurde ebenfalls überlegt, eine tatsächliche Anbindung an GitHub zu haben, über die mit der richtigen Eingabe eines Kommandos ein tatsächliches Repository erstellt wird. Diese Idee wurde allerdings wieder verworfen, da sie annimmt, dass jeder Nutzende des Systems bereits ein GitHub Profil besitzt, beziehungsweise dem System die nötigen Berechtigungen zur API Anbindung geben möchte. Außerdem würde dieser Ansatz voraussetzen, dass der Nutzende die Aufgaben in der richtigen Reihenfolge abarbeitet.

Letztendlich wurde sich für eine minimalistische Implementierung der Konsole entschieden, die je nach Aufgabe zugeschnittene Feedbacks bei richtigen und falschen Eingaben geben kann.

Wie die Konsole aussehen soll, wurde zunächst in Wireframes skizziert. Mehr zu dem Thema der Gestaltung lässt sich in Abschnitt 3.4 finden.

3.3 Proof of Concepts

Bevor mit der Implementierung des Systems begonnen wurde, wurde überlegt, welche die wichtigsten Funktionen sind, und welche Risiken diese bergen könnten. Diese Risiken waren unter anderem wichtig, da ohne die Implementierung möglicher Fallbacks ein Versagen des Systems eintreten könnte. Mehr zu der Implementierung der jeweiligen Proof of Concepts lässt sich in dem Kapitel zur Implementierung finden.

3.3.1 Mock Konsole

Beschreibung
Einer der wichtigsten Funktionen des Systems ist eine Mock Konsole, auf welcher der Nutzende Eingaben tätigen kann. Nach einer Eingabe soll die Konsole oder das Spiel

dem Nutzendem Feedback geben, und das Bild in der Mitte des User-Interfaces soll sich verändern.

Exit Kriterium

- Eine richtige Eingabe wurde getätigt, und das Bild in der Mitte hat sich geändert. Die Konsole gibt ein positives Feedback aus.
- Eine falsche Eingabe wurde getätigt, und das Bild in der Mitte hat sich nicht geändert. Die Konsole gibt ein negatives Feedback aus.

Fail Kriterium

- Die Konsole reagiert nicht auf Nutzendeneingaben.

Fallback
Der Nutzende kann in den Einstellungen zwischen manuellen Eingaben und Auswahl aus einem Menü von Befehlen wählen. Diese Befehle haben dieselbe Funktionalität, nur dass der Nutzende diese bereits vorher einsehen kann und die nicht komplett selber eingeben muss. Dieser Fallback könnte ebenfalls nützlich sein, wenn der Nutzende keine Texteingabe zur Verfügung hat oder die Aufgaben zu schwierig findet, und lieber aus einer Auswahl von Lösungen wählen würde.

3.3.2 Dialogeingabe

Beschreibung Durch Dialog soll der Nutzende in die Story des Systems gezogen werden und sowohl Erklärungen als auch Feedback zu den Eingaben und Aktionen erhalten. Der Nutzende soll auf die Dialogbox klicken können, um die nächste Dialoglinie angezeigt zu bekommen.

Exit Kriterium

- Drei Beispieldialoglinien wurden erfolgreich durchgespielt, und der Nutzende kann nun eine Eingabe auf der Mock Konsole durchführen.

Fail Kriterium

- Der Nutzende weiß nicht, wo dieser klicken muss, um den Dialog fortzuschreiten.
- Das System reagiert nicht auf Klicken des Nutzenden, und der Dialog wird nicht fortgesetzt.

Fallback
Die Dialogbox weist einen Knopf auf, mit dem der Dialog fortgeschritten werden kann. Zusätzlich kann der Nutzende ebenfalls durch Eingaben über die Tastatur den Dialog fortschreiten.

3.3.3 PWA Funktionalitäten

Beschreibung
Das System soll Progressive Webapp (PWA) Funktionen aufweisen, um für den Nutzenden möglichst zugänglich zu sein. Hierzu soll eine offline HTML Seite implementiert sein, die eine sehr simple Version der anderen Proof of Concepts aufweist, mit dem Unterschied, dass es einen Hinweis oben darauf gibt, dass die Seite momentan offline betrachtet wird.

Exit Kriterium

- Eine PWA kann erfolgreich installiert werden, und eine Offline Version der Seite kann betrachtet werden.

Fail Kriterium

- PWA Funktionen nicht verfügbar und Seite lädt trotzdem nicht ordentlich.

Fallback
Offline-Funktionen nicht verfügbar machen. Nutzende bekommt einen Hinweis darauf, dass dieser bitte eine Verbindung zum Internet herstellen soll.

3.4 Wireframes

Bevor mit der Implementierung des Systems begonnen wurde, wurden mehrere Skizzierungen zur Gestaltung des Systems erstellt. Einige ältere Skizzierungen des Levelscreen Designs wurden hierbei ausgelassen.

Die ersten Wireframes sollen die generelle Gestaltung der Menüs skizzieren. Hierbei sollen alle Level in einer Art Bibliothek gesammelt sein. Die Levels sollen in einer beliebigen Reihenfolge gespielt werden können. Außerdem soll es noch einen weiteren Tab geben, um nur auf die Informationsinhalte zuzugreifen. Im Level selber befinden sich diese unter dem Informations-Button. Durch eine separate Funktion hierbei soll sichergestellt werden, dass Studierende immer einfachen Zugriff auf alle Informationen haben und alle Kommandos mitsamt ihren Erklärungen an einem gesammelten Ort finden.

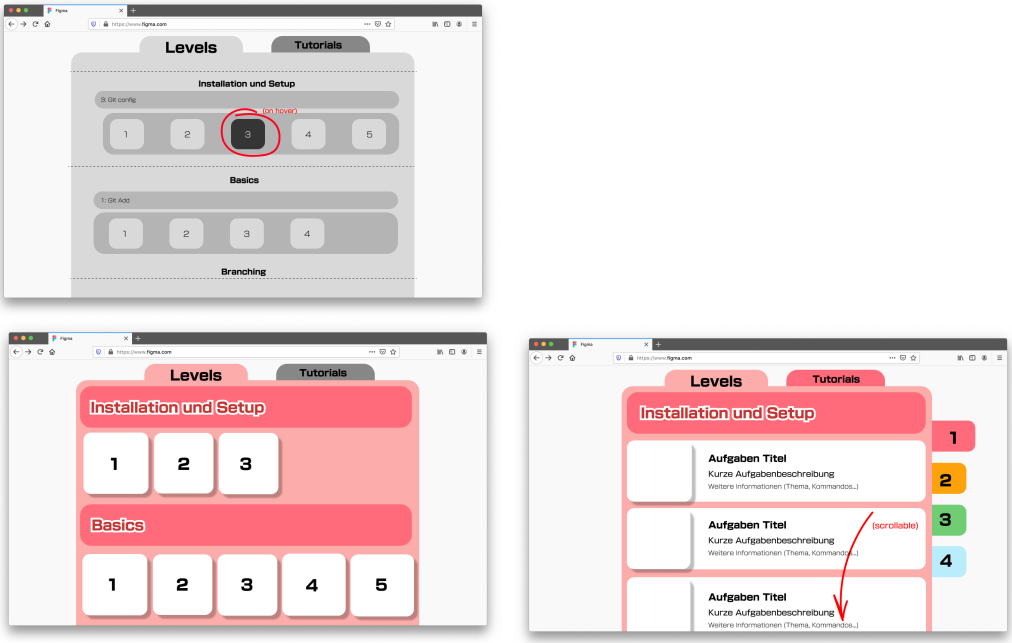

Abbildung 3.1: Wireframes zur Gestaltung der Levelauswahl

Oben dargestellte Grafik soll die Planung des Leveldesigns verdeutlichen. Hierbei wurde unter anderem darauf geachtet, dass das User-Interface minimal gehalten wird, um unnötige Verwirrung zu vermeiden. Über die Konsole sollen Eingaben getätigt werden, die in dem Bild mittig reflektiert werden. Zudem gibt es eine Menüleiste, in welcher der Nutzende gegebenenfalls ein Level überspringen kann, zurück ins Hauptmenü gelangt oder zusätzliche Erklärungen für das aktuelle Thema aufrufen kann.

Es wurden mehrere verschiedene Skizzen für das Interface erstellt. In der Umsetzung wurde letztendlich eine Kombination verwendet. Links sind die Menüpunkte zu finden, rechts an der Seite ist die Mock Konsole für Eingaben. Mittig in der finalen Version sind das Bild zum Szenario sowie die Dialogbox.

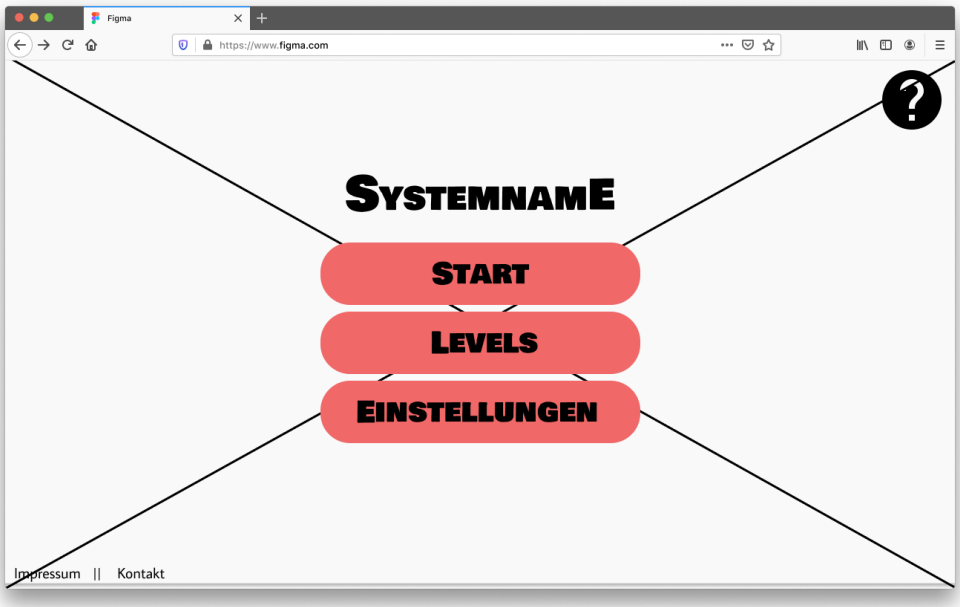

Abbildung 3.2: Wireframes zur Gestaltung des Hauptmenüs

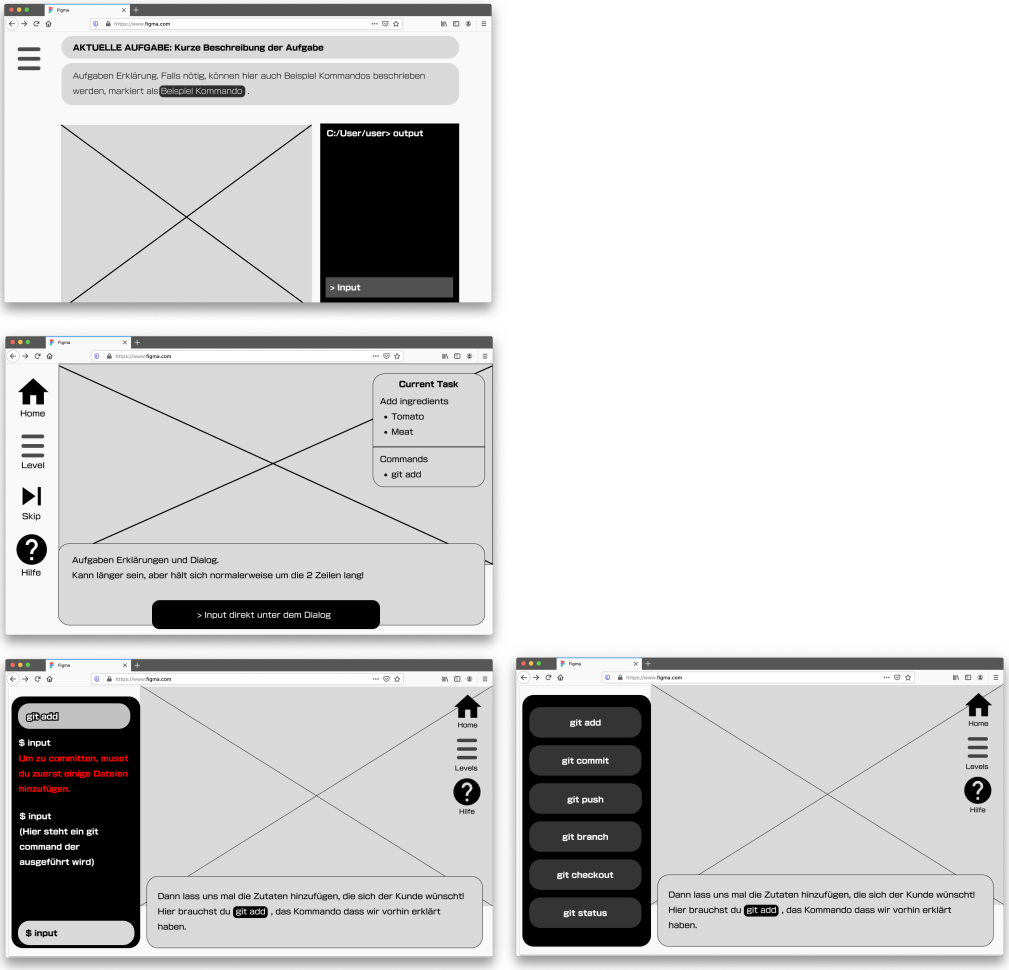

Abbildung 3.3: Wireframes zur Gestaltung des Spiele-Interface

4 Implementierung

Für das System wurde ausschließlich ein Frontend entwickelt, dass in sich geschlossen bereits alle Dialoge und Bilder beinhaltet.

Für die Implementierung wurden verschiedenste Web Technologien verwendet, die in dem Abschnitt Technologien näher beleuchtet werden.

4.1 Technologien

CSS und Tailwind Für das generelle Styling des Frontends wurde im Projekt Tailwind CSS eingesetzt. Dieses Framework erleichtert es, Änderungen schnell und effizient umzusetzen und somit eine möglichst kurze Feedback-Schleife sicherzustellen. In Kombination mit Vite wurden bei Änderungen im Styling der Server direkt neu geladen, was den Entwicklungsprozess um vieles vereinfachte.

Typescript Als Programmiersprache für logische Funktionen wurde TypeScript statt Javascript verwendet, um es einfacher zu machen, Klassen und Interfaces zu erstellen, sowie eine Typsicherheit zu gewährleisten.

Vite/Vue/HTML HTML wurde im Projekt in Verbindung mit Vue.js verwendet. Vite diente zum Erstellen des Grundgerüsts des Vue-Projekts. Anschließend wurde der integrierte Entwicklungsserver von Vite genutzt, um eine möglichst schnelle Feedback-Schleife für visuelle Anpassungen zu ermöglichen.

Deno Deno wurde in diesem Projekt als Alternative zu Node.js verwendet. Dies erfolgte teils aus persönlichem Interesse, um die neue Technologie zu erlernen. Deno wurde als Ersatz für Node.js, da es von den selben Entwicklern stammt, und mehr Features bietet, die mehr Entwicklungsfreiraum für die Zukunft bieten als herkömmliche Laufzeitumgebungen. Zusätzlich bietet es eine Rückwärtskompatibilität mit Node.js, die alle alten Bibliotheken unterstützt.

PWA Funktionalitäten Zu Beginn des Projektes wurde überlegt, wie man das System möglichst zugänglich machen kann. Zunächst wurde in Erwägung gezogen, das System als Desktop Applikation zu implementieren. Hierzu wäre das Framework Electron verwendet worden. Hierbei könnten alle bisher erwähnten Frameworks wie gewohnt verwendet werden. In mehreren Beratungsgesprächen

hatte sich allerdings herausgestellt, dass es sinnvoller wäre, eine Webapp zu entwickeln, die unabhängig vom System des Nutzenden funktioniert, und zudem Offline Funktionen aufweist. Es wurde sich daher entschieden, das System als Web-Applikation mit progressive Webapp (PWA) Funktionen zu implementieren. Grund hierfür war unter anderem, dass die Applikation möglichst unabhängig funktionieren soll und auch die Möglichkeit bieten soll, heruntergeladen zu werden. Studierende können hierbei eine lokale Version des Systems auf ihren Geräten haben.

4.2 Implementierung der Proof of Concepts

Im folgenden Abschnitt werden alle Proof of Concepts noch einmal reflektiert betrachtet. Hierbei wird sich auf die ausformulierten Proof of Concepts aus Kapitel 3 bezogen.

4.2.1 Mock Konsole

[1] Die Mock Konsole wurde anhand einer Vue Komponente implementiert, die an bestimmten Stellen im Dialog automatisch aktiviert und deaktiviert wird. Bei Eingaben wird ein Feedback innerhalb der Konsole ausgegeben, welches zu der Aufgabe passt.

Der Fallback für auswählbare Optionen wurde im Rahmen des Praxisprojektes nicht erstellt. Der zeitliche Rahmen war ungenügend. Außerdem wurden letztendlich andere Features als wichtiger eingestuft.

4.2.2 Dialog

Der Dialog wurde ebenfalls in einer eigenen Vue Komponente umgesetzt. [2] Wenn keine Eingabe benötigt wird, kann der Nutzende des Systems auf die Dialog-Box klicken, um die nächste Zeile zu sehen. Es wurde zudem entschieden, einige Wörter hervorzuheben, die besonders wichtig sind, oder einen Hinweis auf die kommende Eingabe geben sollen. Wenn keine Eingabe nötig ist, wird dem Nutzendem ebenfalls ein Hinweis gegeben.

[1] `https://github.com/AnoukMartinez/martinez-pp-ss-2024/blob/main/src/components/Console.vue`

[2] `https://github.com/AnoukMartinez/martinez-pp-ss-2024/blob/main/src/components/DialogueBox.vue`

4.2.3 PWA Funktionalitäten

Die PWA Funktionen wurde zu Beginn der Implementierung eingerichtet, und funktionieren online. Die Anwendung ist herunterladbar, benötigt allerdings trotzdem eine Verbindung zum Server. In der momentanen Version der Applikation gibt es daher keine Offline Funktionalitäten.

5 Fazit und Reflexion

Abschließend werden noch einmal alle Teile des Projekts reflektiert betrachtet. Zudem werden mögliche zukünftige Funktionen des Projekts betrachtet.

5.1 Schwierigkeiten bei der Implementierung

Eine der größten Herausforderungen bei der Umsetzung des Systems war die Organisation der Vue-Komponenten. Zwar war im Vorfeld geplant, dass beispielsweise bei Nutzendeneingaben ein entsprechendes Feedback generiert werden soll, jedoch wurde die Kommunikation zwischen den Komponenten nicht detailliert genug durchdacht. Stattdessen erfolgte die Implementierung situationsbedingt, abhängig von den jeweils benötigten Funktionalitäten.

Dies lag teilweise daran, dass sich die Funktionsweise von Vue generell erst während der Projektarbeit parallel beigebracht wurde. So war es intransparent, wie die Komponenten strukturiert werden sollten.

Dies führte zu einem anfänglich komplexen und unübersichtlichen Einsatz von Vue-Emits, der im weiteren Verlauf der Entwicklung schrittweise reduziert und vereinfacht werden musste.

5.1.1 Analogie

Was zudem größtenteils zu Schwierigkeiten führte, war vor allem der Einsatz einer Analogie zum Erklären der Sachverhalte. Besonders herausfordernd war, immer Äquivalente zu den Themen zu finden, die mit der Analogie erklärbar sind. Beispielsweise war es sehr schwierig, einen sinnvollen Vergleich für das Klonen und Pullen eines Repository zu finden.

Aus der Umfrage der Studierenden ergab sich, dass die Inhalte eine höhere Priorität einnehmen als das Setting und die Geschichte, die eventuell mit dem System kommt. Vor allem im Hinblick auf die Umfrageergebnisse lässt sich daher rückblickend sagen, dass die Verwendung einer Analogie im Kontext eines Lernsystems für Git eher

ungeeignet ist. Im Falle der weiteren Entwicklung des Systems sollte die Analogie durch simple, praxisbezogene Visualisierungen ersetzt werden.

Das größte Risiko bei der Verwendung einer Analogie ist hierbei Verwirrung unter den Nutzenden. Um nicht den gesamten Fortschritt rückgängig zu machen, der im Rahmen des Projektes geleistet wurde, könnte dem Nutzenden eine Wahl gegeben werden, zwischen einem Story Modus und einem Modus, der sich nur auf die Lerninhalte fokussiert.

5.2 Zukünftige Ergänzungen

Insgesamt lässt sich sagen, dass das Projekt momentan noch nicht in einem Zustand ist, in dem das System praktisch anwendbar wäre. Für die Fertigstellung eines minimal realisierbaren Systems fehlen noch folgende Aspekte:

Rückmeldungen Um das System benutzendenfreundlich und effizient zu gestalten, ist es notwendig, Rückmeldungen von Testnutzenden einzuholen. Auf Basis dieser Rückmeldungen muss das System detailliert angepasst werden, um eine frustrationsfreie Lernerfahrung zu gewährleisten.

Erklärungen Um einen besseren Lerneffekt zu gewährleisten, wurde ein Infoknopf in der Seitenleiste eingebaut, der für jedes Kapitel zusätzliche Erklärungen bereitstellen soll. Diese Erklärungen sind allerdings nur teilweise geschrieben worden, und erfordern noch einer inhaltlichen Erweiterung.

Weitere Kapitel Für die inhaltliche Vervollständigung des Systems fehlt ein Kapitel zum Thema Merge-Konflikte. Dieses wurde im Rahmen des Projekts nicht umgesetzt, da es den zeitlichen Umfang überschritten hätte. Zusätzlich wäre ein Mock-Editor zu integrieren, in dem Nutzende Merge-Konflikte simulieren können. Hierbei muss die Eingabe semantisch geprüft werden.

Verständlicheres Feedback Eine sinnvolle Ergänzung des Systems wären Ausgaben in der Konsole, die eine echte Ausgabe in der Git Bash simulieren.

Zudem könnten folgende optionale Funktionen noch zusätzlich implementiert werden. Diese sind allerdings nicht zwingender Weise für die Fertigstellung des Projektes benötigt.

Fallback für Konsoleneingaben Wie in den Proof of Concepts erwähnt, war initial eine Alternative für Texteingaben geplant, in welcher der Nutzende aus einer Auswahl an Kommandos durch Klicken wählen kann. Dann können Nutzende des Systems in den Einstellungen den Modus auswählen, den sie präferieren.

Offline-Funktionen Zuletzt sollten dem System Offline-Funktionen im Kontext der PWA-Funktion gegeben werden, um möglichst zugänglich für alle Studierenden zu sein.

Literatur

[1] Thomas W. Malone. „Toward a Theory of Intrinsically Motivating Instruction“. In: *Cognitive Science* 5.4 (1981), S. 333–369. DOI: 10.1207/s15516709cog0504_2.

Anhang

Im Rahmen der Projektarbeit wurde eine Umfrage unter Studierenden der Fakultät 10 durchgeführt. Teilnehmende sollten hierbei verschiedene Fragen zum Thema Git, GitHub und ihrer Demografik beantworten. Im folgendem Abschnitt sind die vollständigen Ergebnisse dieser Umfrage zu finden.

Bei einigen Fragen wurden Freitextfelder zur Verfügung gestellt. Die Ergebnisse dieser Felder sind gesondert in Listen direkt unter den grafischen Darstellungen zu finden. Zudem anzumerken ist, dass zwei Unfragen parallel durchgeführt wurden, einmal in deutscher Sprache, und einmal in englischer Sprache verfasst. Diese Maßnahme wurde getätigt, um die Umfrage möglichst zugänglich zu machen, und um möglichst viele diverse Antworten zu erhalten. Alle im Abschnitt dargestellten Ergebnisse sind eine Kombination beider Ergebnisse. Die Fragestellungen wurden hierbei identisch formuliert. Alle Antworten wurden am letzten Stand, dem 08.10.2024 um 15:35 abgerufen.

Demografik

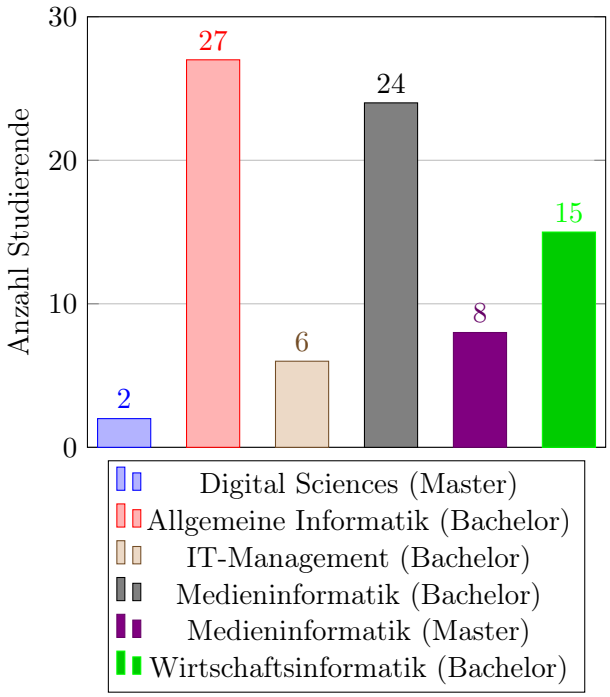

Abbildung 1: Grafische Darstellung der Ergebnisse zur Frage: Studiengänge der Teilnehmer

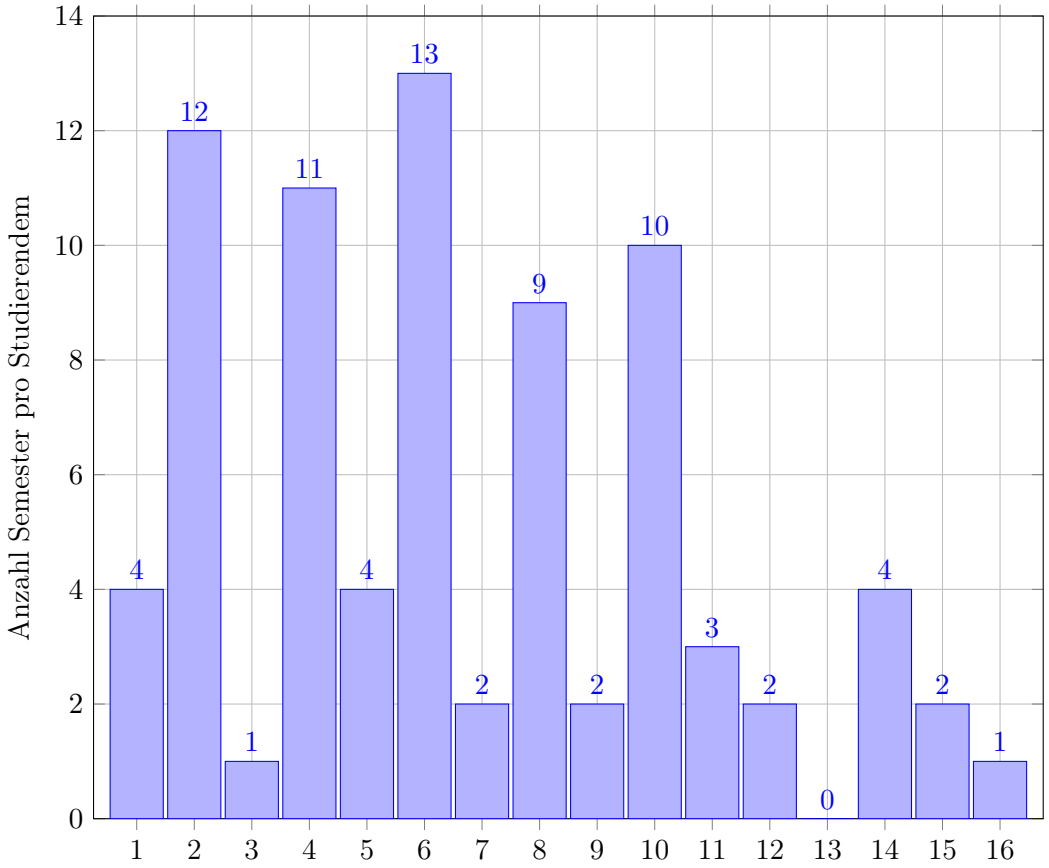

Abbildung 2: Grafische Darstellung der Ergebnisse zur Frage: Semester der Teilnehmer

Git und Github Erfahrungen

Im Abschnitt der Umfrage zu Erfahrungen sollte sichergestellt werden, dass die Befragten sich bereits mit Git auseinandersetzen mussten. Zudem sollte erforscht werden, in welchem Zusammenhang Git verwendet wurde, und welche Schnittstellen zu Nutzung verwendet wurden.

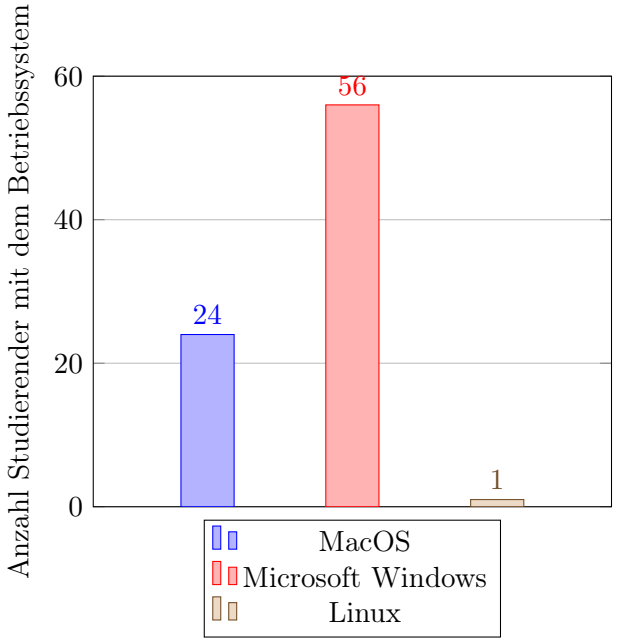

Abbildung 3: Grafische Darstellung der Ergebnisse zur Frage: Betriebssysteme der Teilnehmer

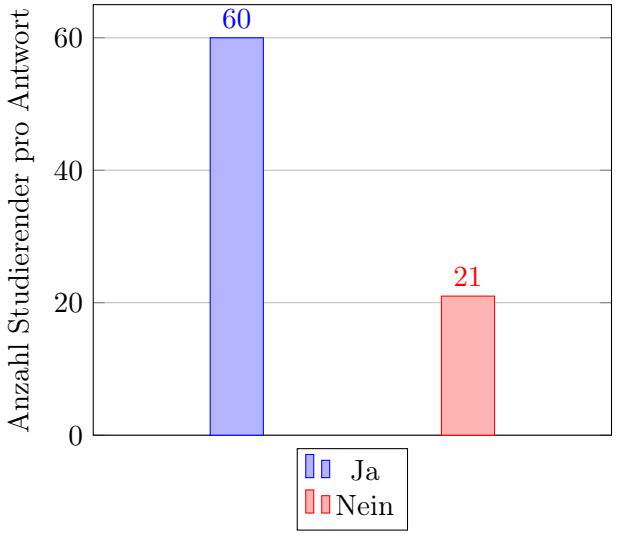

Abbildung 4: Grafische Darstellung der Ergebnisse zur Frage: “Ich kenne den Unterschied zwischen Git und Code Hosting Plattformen wie GitHub.”

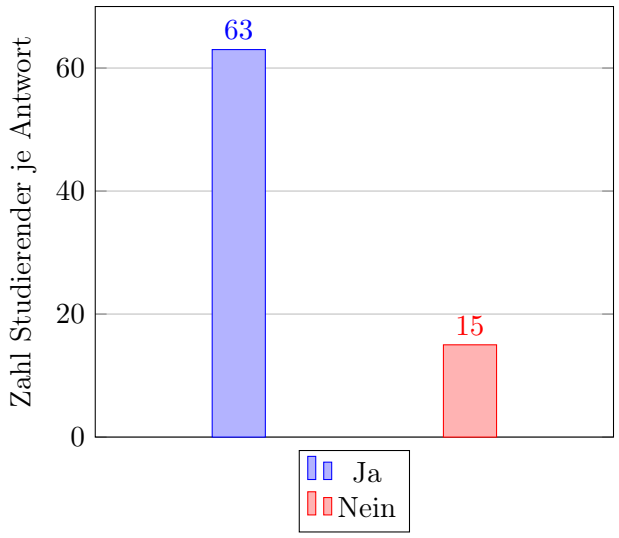

Abbildung 5: Grafische Darstellung der Ergebnisse zur Frage: “Ich habe bereits im Rahmen meines Studiums an ein oder mehreren Kursen teilgenommen, in denen die Nutzung von Git oder Services wie GitHub erforderlich war.”

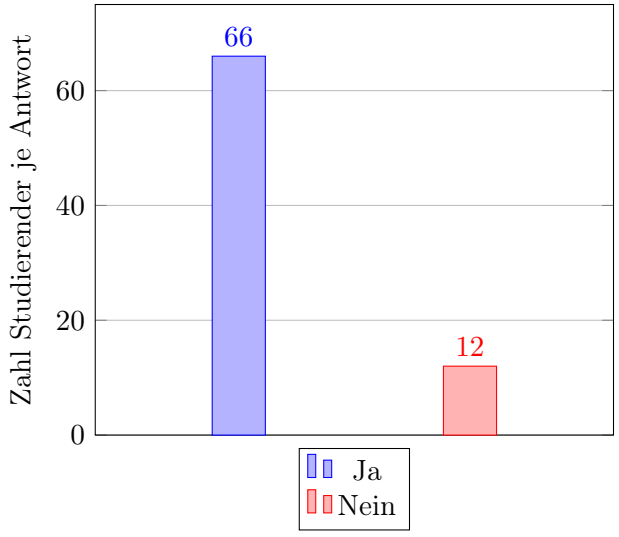

Abbildung 6: Grafische Darstellung der Ergebnisse zur Frage: “Ich habe bereits Git oder Services wie GitHub im Kontext meiner Projekte der Hochschule verwendet.”

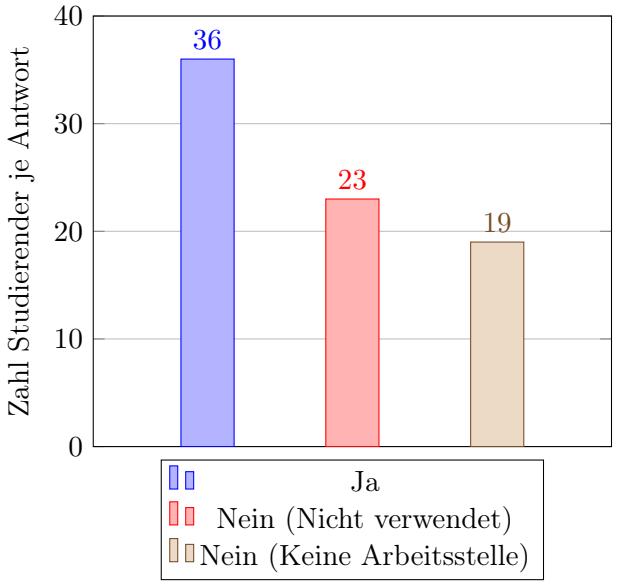

Abbildung 7: Grafische Darstellung der Ergebnisse zur Frage: "Ich habe bereits Git oder Services wie GitHub im Rahmen meiner Arbeit verwendet."

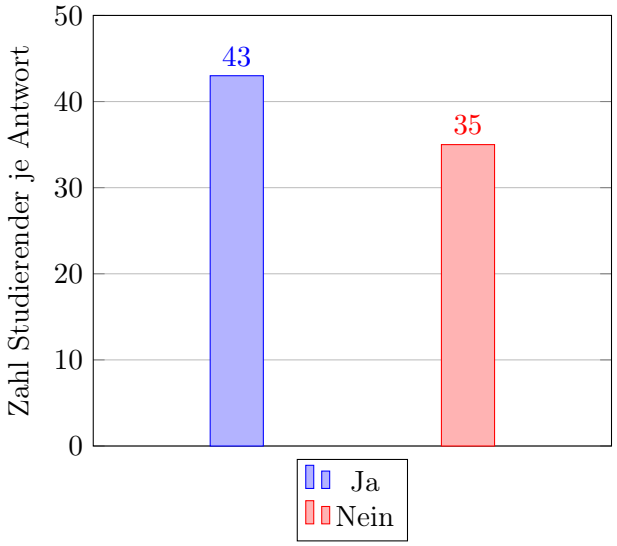

Abbildung 8: Grafische Darstellung der Ergebnisse zur Frage: "Ich habe bereits Git oder Services wie GitHub im Kontext meiner persönlichen privaten Projekte verwendet."

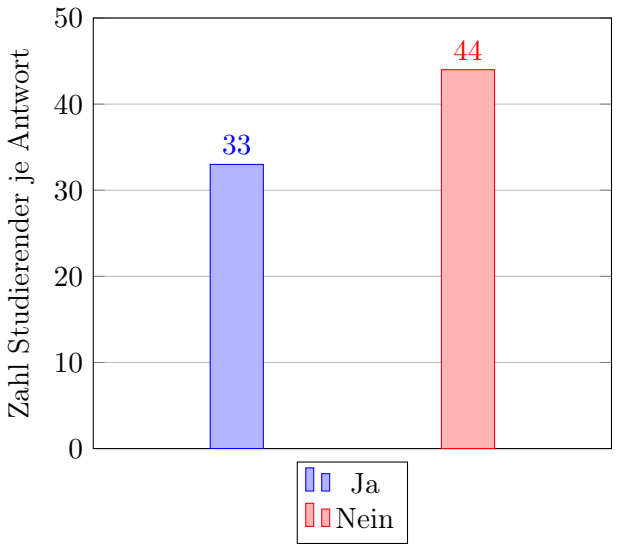

Abbildung 9: Grafische Darstellung der Ergebnisse zur Frage: “Ich habe schon einmal einen GitHub GUI Client verwendet.”

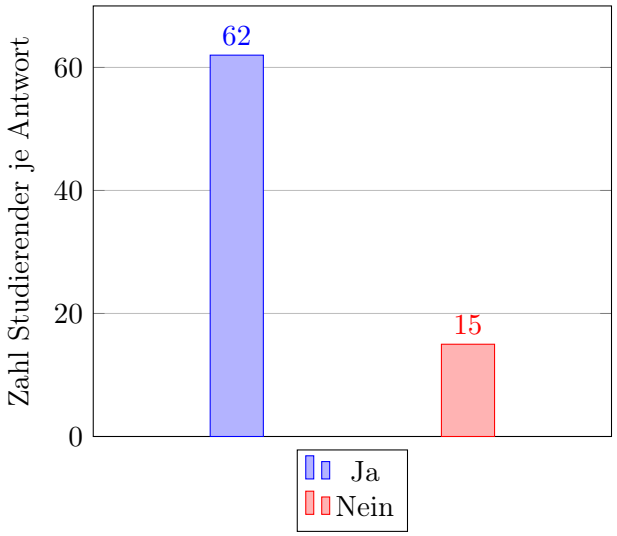

Abbildung 10: Grafische Darstellung der Ergebnisse zur Frage: “Ich habe schon einmal GitHub über das integrierte grafische Interface einer IDE verwendet.”

Selbsteinschätzung der Teilnehmenden

Im dritten Teil der Umfrage wurde Teilnehmenden eine Möglichkeit gegebene, ihre

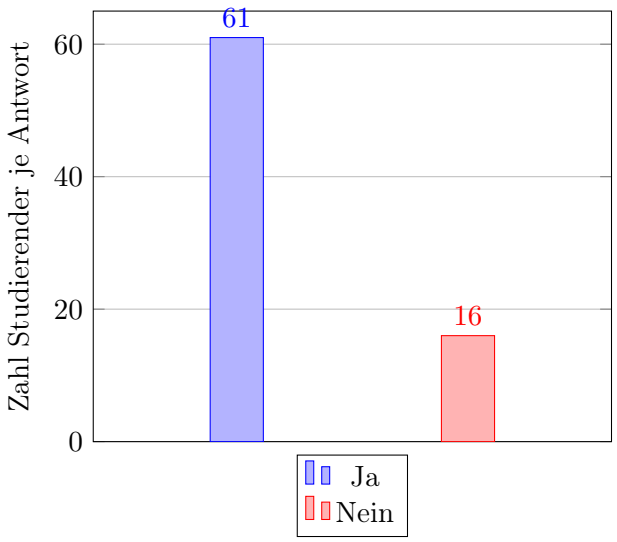

Abbildung 11: Grafische Darstellung der Ergebnisse zur Frage: “Ich habe bereits Git über die Kommandozeile verwendet.”

eigenen Kompetenzen einzuschätzen und zu bewerten. Hierbei wurden verschiedenste Aussagen getätigt, dessen Zutreffen im Anschluss auf einer Skala von Eins bis Zehn bewertet wurden.

Dabei wurde der folgende Info Text mit den ungefähren Vergleichswerten beigefügt. Hierbei soll eine 1 auf der Skala etwa vergleichbar sein mit der Aussage “Ich habe noch nie von dem Service/Tool gehört”. Eine 10 hingegen soll auf der Skala vergleichbar sein mit der Aussage “Ich habe den Service/das Tool vollkommen verstanden und kenne ein paar fortgeschrittene Konzepte”. Fortgeschrittene Konzepte sollen hier beispielsweise Branches für Git sein, oder generelle Code Review Funktionen für GitHub. Diese Vergleiche wurdem im Kontext der Umfrage in der Fragestellung spezifiziert.

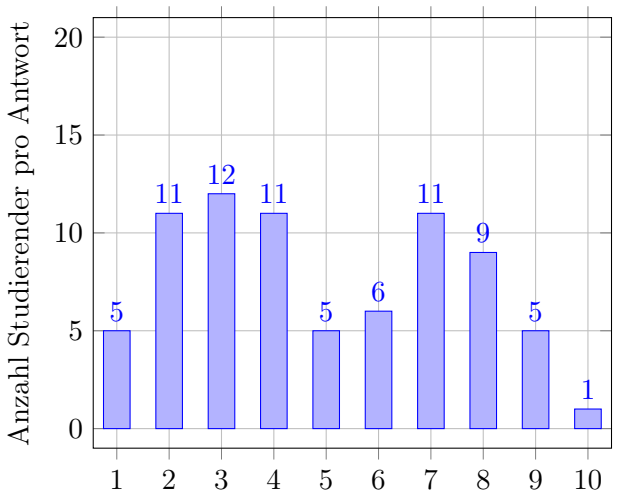

Abbildung 12: Grafische Darstellung der Ergebnisse zur Frage: Wissen von Git

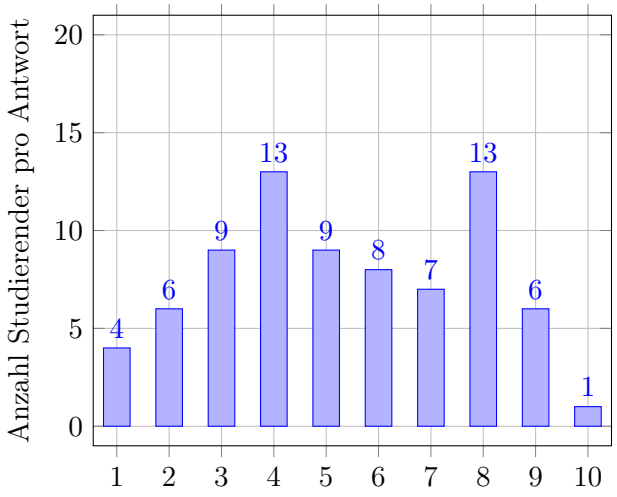

Abbildung 13: Grafische Darstellung der Ergebnisse zur Frage: Wissen von GitHub

Interesse an einem Lernsystem

Um anhand der Umfrage bestätigen zu können, ob ein Lernsystem wirklich von der Zielgruppe gewünscht wird, wurde die Frage speziell in einem eigenem Teil behandelt. Hierbei wurde unter anderem gefragt, ob Studierende generell Interesse an einem Lernsystem hätten, sowie was ihnen bei einem solchem System wichtig wäre.

Zunächst wurden die möglichen Wissensmängel in einer Selbsteinschätzung der Studie-

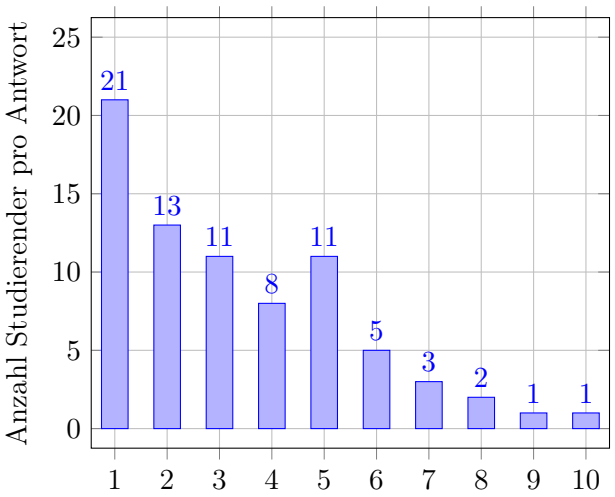

Abbildung 14: Grafische Darstellung der Ergebnisse zur Frage: Wissen von Anderen Services

renden betrachtet. Hierbei wurde ebenfalls die Möglichkeit gegeben, ein Freitextfeld auszuwählen. Die gegebenen Antworten sind ebenfalls in diesem Dokument beschrieben, jeweils in einer Auslistung unter der grafischen Darstellung der vorgegebenen Ergebnisse.

Freitext: Sonstige Gründe für Wissensmängel

- An der Th wurde es nicht richtig erklärt bzw. beigebracht bekommen
- Ich weiß wie Git funktioniert, weiß aber auch, dass mein Wissen nicht wirklich über die Basics hinausgeht.
- Es besteht keine Notwendigkeit
- Keine Erklärungen in den Fächern. Man wird einfach rein geworfen. Hätte mir eine kleine einführung gewünscht
- Gute Git/Hub Kenntnisse sind kein muss. Meistens reichen einfache Push/-Pull/Clone Aktionen.
- Ich musste mich bisher nicht mehr in das Thema einarbeiten. Damit ist gemeint, dass ca. 6 Befehle alle meine Use-Cases bis jetzt abgedeckt haben.
- Alles Wissen das ich besitze habe ich aufgrund von erfahrenen Freunden. Den richtigen Umgang oder irgendetwas im Allgemeinen dazu wurde an der TH nie (verständlich) beigebracht sondern immer einfach vorrausgesetzt

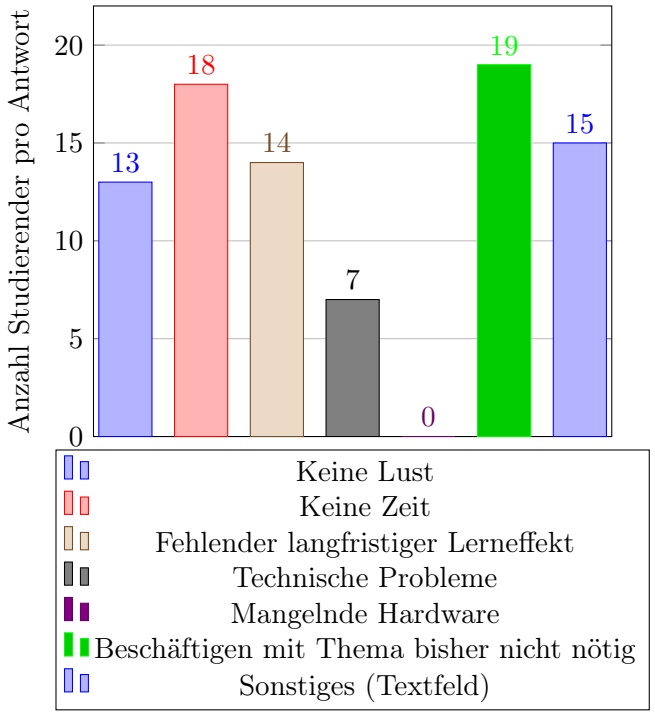

Abbildung 15: Grafische Darstellung der Ergebnisse zur Frage: Mögliche Gründe für Wissensmängel im Bereich

- Beim selber erlernen bringt man sich aus Zeitgründen nur das nötigste bei, um die Aufgaben erfüllen zu können. Wenn es danach nicht mehr gebraucht wird wird das Gelernte wieder vergessen.
- Man geht in der Hochschule quasi nie über das Klonen und Pushen hinaus.
- Wozu reines Git wenn es Github gibt
- Nur mit minimalen funktionen in bestimtmen kursen. Es wird nie regelmäßig verwendet, sodass ein regelmäßiger kontakt mit git entstanden ist
- Wird in den Vorlesung nur kurz vorgestellt und dann soll man alles darüber wissen I guess...
- nach einer Zeit wenn man es nicht mehr verwendet, ist das Wissen sehr darüber sehr schwammig. Und die Basis wieder nochmals aufzuholen, ist immer eine sehr hohe Hürde.
- Bisher war es nie sonderlich zwingend, daher war die Motivation nicht sonderlich hoch. Und nach dem Benutzen (was selten war), habe ich alles wieder vergessen

- nicht oft genug damit gearbeitet.

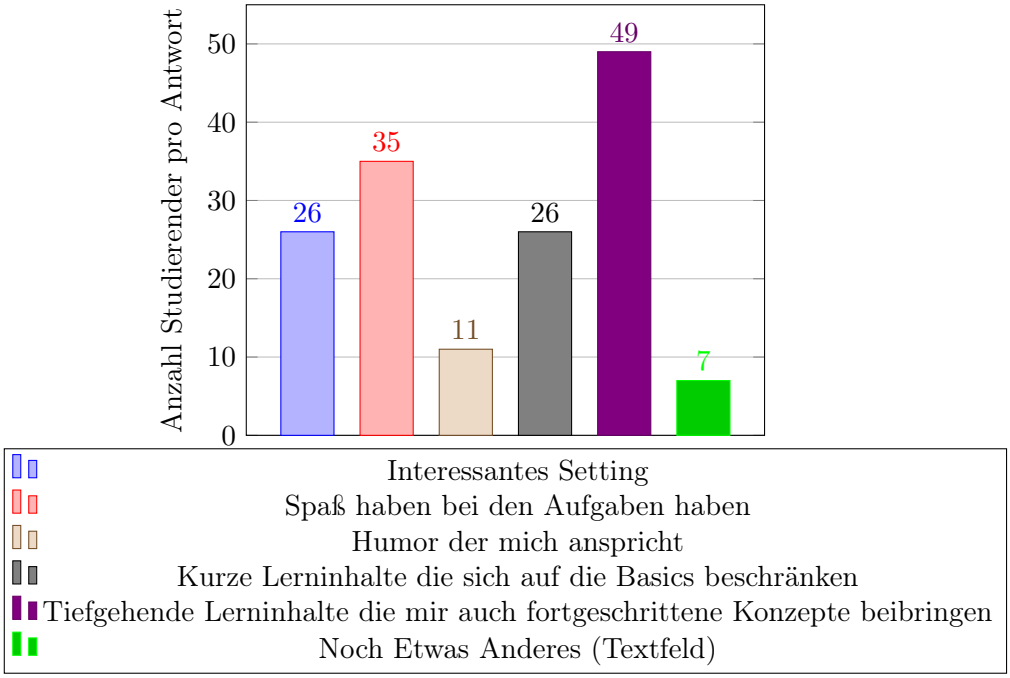

Abbildung 16: Grafische Darstellung der Ergebnisse zur Frage: Wünsche an einem Lernsystem

Freitext: Sonstige Wünsche für ein Lernsystem

- simplere gestaltung
- Visualisierungen!!
- Einen gut verständlichen Einstieg der nicht überfordert und sich auf die Basics beschränkt mit aufbauend schwerer werdenden Lerninhalten
- in kleiner abschließbare Module unterteilen, dass es nicht so ein großer Block ist
- Active Recall, Spaced Repetition, Interleaving nicht nur einmal lernen, sondern im Kopf behalten.
- das die Profs nicht von nicht vorhandenen Vorwissen ausgehen
- Möglichkeit, online von zuhause jederzeit zugreifen zu können. Support bei Anwendungsproblemen, wenn Beispiele praktisch umgesetzt werden

Freitext: Ja, aber...

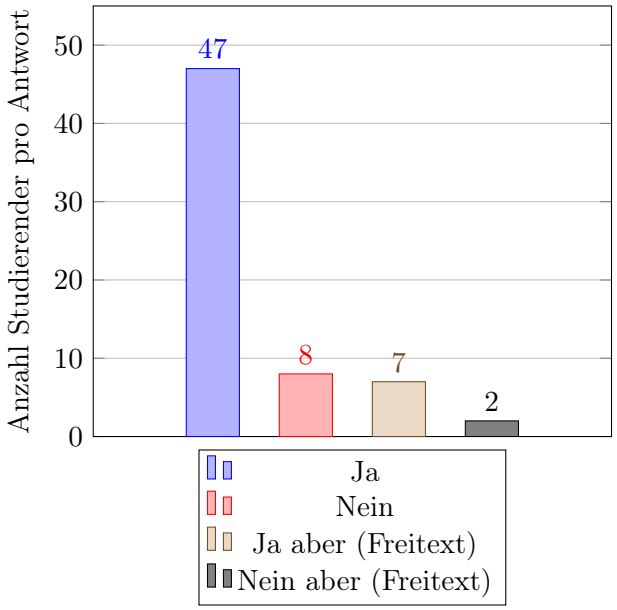

Abbildung 17: Grafische Darstellung der Ergebnisse zur Frage: Interessen an einem Lernsystem

- Nur falls ich es alleine nicht bereits checken sollte, sprich ich jetzt noch keine Erfahrung damit jemals gehabt hätte.
- Gamificationinhalte sollten dennoch eine gewissen Seriosität haben
- Es sollte generell standert in der lehre sein und ein einheitlciehs sytsem geben
- nur wenn auch tiefgehendere Lerninhalte vermittelt werden
- nur solange es nicht zu langatmig ist und die lernkurve steilgenug ist.
- Aktuell ist der Bedarf bei mir nicht mehr groß. Am Anfang des Studiums sehr gern
- es müsste Effizient wissen beibringen und meinen wissensstand berücksichtigen. sodass man nicht gelangweilt wird

Freitext: Nein, aber...

- würde das System auch tiefgehende Konzepte umfassen, hätte ich durchaus Interesse
- hätte es evtl genutzt vor ein paar Jahren, als ich es gelernt hatte, wenn es zügig den Stoff behandelt.

Zum Schluss der Umfrage, wurde Teilnehmenden noch einmal die Möglichkeit gegeben, generelle Anmerkungen zur Umfrage oder zum Thema zu geben. Hierzu wurde ein einzelnes Freitextfeld bereitgestellt.

- Gamification für eine bessere Zukunft.
- Vielleicht ist für Anfänger auch wichtig zu verstehen, was anders an Git im Vergleich z.B. zu SVN oder Perforce/Helix Core ist.
- Etwas verwirrt hat mich, dass bei der Frage "Git Kenntnisse" das Verwenden von Branches auf die selbe Erfahrungsstufe gesetzt wird wie Git "vollständig" verstanden zu haben, wobei ich Branches eher bei den Git Grundlagen verordnen würde. Von daher war ich mir nicht ganz sicher wo ich mich auf der Skala einordnen soll
- Sollte unbedingt im Rahmen eines Pflichtkurs der Informatik eingehender behandelt werden. Mehr als, wie man ein commit macht, wurde leider nie erklärt. Ganz zu schweigen was ein Branch ist und wie man die handelt mit bspw. mehreren Projektteilnehmern.
- Ich finde es toll, dass eine Umfrage zu diesem Thema gemacht wird und hoffe, das Lernsystem zum Lernen von GIT noch kennenlernen zu dürfen. :)
- While I don't have a particularly deep understanding of git, I feel comfortable enough with my basic knowledge. So even though I rated my knowledge of git and GitHub rather low, it's always been enough for me.

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift