HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

moodle

Moodle Plugin StudentQuiz

Erweiterungen

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil
Herbstsemester 2016

Autoren: Lukas Dürrenberger, Steven Ryser, Alexis Suter
Betreuer: Prof. Frank Koch, Dozent für Wirtschaftsinformatik

Erstelldatum	Mittwoch, 05. Oktober 2016
Änderungsdatum	Freitag, 23. Dezember 2016
Druckdatum	Freitag, 23. Dezember 2016
Autoren	Lukas Dürrenberger (lduerren@hsr.ch) Steven Ryser (s1ryser@hsr.ch) Alexis Suter (a2suter@hsr.ch)

Abstract

StudentQuiz erlaubt es Studenten Fragen für die Mitstudenten zu erfassen, sowie mit bereits erfassten Fragen Quiz zu erstellen und lösen. Moodle, als eine Lernplattform mit über 80 Millionen Benutzer, bietet standardmässig bereits Funktionalitäten an, um Quiz für Studenten zu erstellen, dies kann aber nur von Lehrpersonen oder Personen mit erweiterten Rechten durchgeführt werden. Genau an diesem Punkt setzt das StudentQuiz an und gibt den Studenten die Möglichkeit ihr Wissen während dem Semester zu stärken, aber es auch spezifisch als Tool für die Prüfungsvorbereitung zu verwenden. Die Grundsteine der zwei StudentQuiz Moodle Plugins wurden in einer vorhergehenden Arbeit umgesetzt.

Im Rahmen dieser Studienarbeit soll nun das vorhandene Projekt so überarbeitet werden, dass es den Validierungsprozess für das offizielle Plugin Directory von Moodle bestehen kann und zusätzlich mit der neusten Version von Moodle kompatibel ist. Zu den nötigen Anpassungen und Fehlerbehebungen sollen zudem noch einzelne neue Erweiterungen sofern möglich hinzugefügt werden.

Durch unsere Abhängigkeit von Drittpersonen konnten während der Dauer der Studienarbeit die beiden Plugins nicht ins Moodle Plugin Directory aufgenommen werden, aber es wurden etliche Anpassungen vorgenommen und gemeldete Probleme gelöst, so dass nun vor allem nur noch mehr Zeit für die Validierung benötigt wird. Um die Wartbarkeit der beiden Plugins zu verbessern und mit dem Wissen, dass weitere Arbeiten dafür ausgeschrieben werden, wurde die Dokumentation von den einzelnen Komponenten noch etwas verbessert und Hinweise zu möglichen Erweiterungen oder Änderungen angegeben.

Aufgabenstellung

Aufgabenstellung SA „Moodle Plugin Studentquiz“ definitiv

1. Betreuer

Prof. Frank Koch, Dozent für Wirtschaftsinformatik

2. Ausgangslage, Problembeschreibung

Quizzenger.hsr.ch ist ein an der HSR entwickeltes Fragen-Pool, das auf crowdsourced Content ausgelegt ist und mittels Badges und Games die Motivation seiner User schürt. Moodle ist das weltweit bedeutendste Learning Management System und erzielt an der HSR über 150'000 Zugriffe monatlich. Beide Systeme sind GPLv3 und basieren auf PHP.

Um die Ideen des Quizzengers für eine grössere Usergroup auch ausserhalb der HSR verfügbar zu machen, wurde die Idee des Quizzengers im FS16 als das neue Moodle Plugin «Studentquiz» umgesetzt. Studentquiz kann als Aktivität innerhalb eines Moodlekurses angelegt werden und die Kurs-Nutzer können darin Fragen erfassen, diskutieren und bewerten sowie in Form von Quizzes filtern und beantworten. Studentquiz berechnet die Schwierigkeit sowie die durchschnittliche Bewertung der einzelnen Fragen und vergibt Leistungspunkte für das Beantworten und Erstellen von Fragen an seine Nutzer.

Der Quizzenger hat sich im Pilotbetrieb einer HSR-Klasse im FS16 bewährt. Künftig soll aber mit dem Studentquiz innerhalb von Moodle gearbeitet werden.

3. Aufgabenstellung

Die folgenden Punkte sind in drei verschiedene Prioritätsstufen unterteilt:
A =hohe Priorität, B=mittlere Priorität, C =tiefe Priorität.

Priorität A:

1. Durchlaufen des Moodle Plugin Validation Prozesses: https://docs.moodle.org/dev/Plugin_validation und Bereitstellung des Plugins (inkl. Questionbehaviour) auf dem Moodle Plugin Directory https://docs.moodle.org/dev/Plugin_contribution.
2. Erstellen eines Student-Dashboards (das auch für Teacher zugänglich ist), vielleicht mit folgenden Punkten:
   a. Filtereinstellung (z.B. gelöst, ungelöst, falsch gelöst)
   b. Lernfortschritt (wie viele Fragen beantwortet), Ranking
3. Neue Spalten «Anzahl Durchführungen» und «Anzahl Kommentare»
4. Option pro Quiz, mit der neue Fragen zunächst auf Hidden gestellt und vom Teacher freigegeben werden müssen. Alternativ auch Markierung mittels «Approved-Tag», dass dann auch im Filter wählbar sein sollte
5. Verbesserung Punktesystem
   a. Eigene Fragen sollen nicht bewertet werden können
   b. Für das Beantworten eigener Fragen werden keine Punkte vergeben
   c. Für das mehrfache Beantworten einer Frage werden nur beim ersten Mal Punkte erzielt
   d. Für eine halbrichtige Antwort werden keine Punkte vergeben
6. Testing / Upgrade Moodle 3.2
   a. PHPUnit-Tests erstellen und ausbauen
   b. Acceptence-Tests (Behat, manuell (vor allem UIs))

7. Reengineering / Refactoring: Vor allem beim Filtering
    a. Code ist schlecht erweiterbar
    b. Evtl auf Button «Run Selected Questions» verzichten
    c. Dynamisches Filtern mit JavaScript (Anpassung Frage Buttons)
       Siehe SA S 27
8. Bug-Fixes
    a. Link auf Quiz von Teacher funktioniert nicht auf Moodle 3.0 (funktioniert das in Moodle 3.1?)
    b. Bewerten einer Frage kann unter Moodle 3.1 umgangen werden
    c. Backup and Restore funktionieren nicht für Studentquiz
    d. Nach Abschluss des Quiz Review („Summary of your previous attempts") wird nur die letzte Teilnahme Quiz angezeigt
9. Notification an die Autoren, wenn Dozierende Fragen verändern oder löschen / Ein „Last edited by"-Entry
10. Teacher können Mindestanforderungen setzen (Activity Completion)
11. Lösung „Orphaned activities (section 999)" verbessern (OA soll nicht erscheinen, sondern die Fragen sollen einfach nur noch in der Questionbank sein)

Priorität B:

1. Erstellen eines Teacher-Dashboards, vielleicht mit folgenden Punkten:
    a. Übersicht aller Fragen
    b. Markierung neuer Fragen
    c. Kommentare
2. Menüpunkt «Quiz and Questions» verbessern? Rename Navigation "Quiz and Questions" to "Create Question or Quiz"
3. Import / Export von einzelnen Fragen (Siehe SA S 27)
4. Tags mit Auto-Completion
5. User-Interface-Sprache
    a. Deutschsprachiges User-Interface (Duzi)
    b. Französischsprachiges User-Interface
6. Navigation: Gewisse Seiten (z.B. Übersicht) erscheinen nicht mehr im Quiz unter dem entsprechenden Kursabschnitt

Priorität C:

1. Neben dem «Move» einzelner Fragen sollte auch ein «Copy» angeboten werden
2. Gamification (Lösungsmöglichkeiten abschätzen)
3. Nach Abgabe Überprüfung optional (Bericht)
4. Ausführlichere Dokumentation der Architektur / Files des Plugins, da man sich bisher nur über die Files einen Überblick schaffen kann
5. Studenten-Dashboard: Übersicht zu den Teilnahmen eines Quizzes, filterbar nach Student/Score hinzufügen
6. Session-Timeout bei Bearbeitung verhindern

4. Zur Durchführung

Es handelt sich um eine Software-Entwicklungsarbeit im Umfeld von Moodle, PHP und MySQL. Auf der fachlichen Seite geht es um e-Assessment als innovative Lern- und Prüfungsform. Solide PHP-Programmierkenntnisse und die Bereitschaft, sich rasch in Moodle und die SA bzw. das Plugin Studentquiz einzuarbeiten, werden vorausgesetzt. Gute Englisch-Kenntnisse sind erforderlich.

Aufgrund der Auslandsabwesenheit des Betreuers findet ein Grossteil der Besprechungen online statt. Die Besprechungen sind von den Studierenden mit einer Traktandenliste vorzubereiten und die Ergebnisse sind in einem Protokoll zu dokumentieren, das in einem kollaborativen Bereich (Dropbox) abgelegt wird.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben.

5. Dokumentation und Abgabe

Die Anwendung sollte weitestgehend selbsterklärend sein, so dass eine knappe Benutzeranleitung (strukturiert nach den Rollen Admin, Teacher, Student; ausgerichtet anhand vergleichbarer Moodle-Dokumentationen) ausreichend ist.

Wegen der beabsichtigten Weiterentwicklung wird auf Modularität und Erweiterbarkeit sowie Qualität der technischen Dokumentation erhöhter Wert gelegt. Auch die Entwicklungsumgebung ist zu beschreiben (Server-Konfiguration, eingesetzte Tools, Projektdateien, Makefiles, etc).

Die Dokumentation zur Projektplanung und -verfolgung ist gemäss den Richtlinien der Abteilung Informatik anzufertigen. Die Detailanforderungen an die Dokumentation der Recherche- und Entwicklungsergebnisse werden entsprechend dem konkreten Arbeitsplan festgelegt.

Die gesamte Arbeit (Quellcode, Buildskripte, Testcode, Testdaten, Benutzeranleitung, Technischer Bericht, Aufgabenstellung, Projektplanung, Protokolle) sind schlussendlich als .zip abzugeben.

6. Termine

Siehe Termine Bachelor- und Studienarbeiten

Studienarbeiten	
bis Ende April 2016	Einbringen eigener Themen gemäss E-Mail Aufforderung.
23.05. - 8.7.16	Die Betreuer erfassen ihre Arbeiten im AVT-Tool.
2.6.16	Präsentation der SA-Themen
11.7.16	Die Arbeiten werden unter https://avt.hsr.ch veröffentlicht.
bis 19.8.16	Die Studierenden, die aufgrund ihrer Vorleistungen zum vorzeitigen Vergabeprozess im AVT zugelassen sind, können sich um 1 Arbeit mit Priorität 1, um 3 Arbeiten mit Priorität 2 und um 3 Arbeiten mit Priorität 3 bewerben.
von 22.8. bis 16.9.16	Zuteilung durch den Studiengangleiter
bis 19.9.16	Einrichten der Arbeitsplätze in den Labors
19.9.16	Beginn der Studienarbeit, Ausgabe der Aufgabenstellung durch die Betreuer.
20.12.16	Die Studierenden geben den Abstract zur Kontrolle an ihren Betreuer/Examinator frei. Die Studierenden erhalten vorgängig vom Studiengangsekretariat die Aufforderung mit den Zugangsdaten zur Online-Erfassung des Abstracts.
23.12.16	Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract zur Weiterverarbeitung an das Studiengangsekretariat frei.
23.12.16	Abgabe des Berichtes an den Betreuer bis 17.00 Uhr.

7. Beurteilung

Die Bewertung erfolgt gemäss dem abgegebenen Formular <Beurteilung_SA_empty.doc>:

Im Übrigen gelten die Abläufe und Regelungen Studien- und Bachelorarbeiten sowie die Allgemeinen Infos Bachelor- und Studienarbeiten des Studiengangs Informatik.

8. Quellen

- Video Tutorial Studentquiz https://tube.switch.ch/videos/637cda8a
- Studentquiz
  - Abgabe Studienarbeit Schuler / Moshfegh FS16
  - http://sinv-56036.edu.hsr.ch/moodle/ (Server mit Moodle 3.0 und PHP5)
  - https://github.com/frankkoch/moodle-mod_studentquiz
  - https://github.com/frankkoch/moodle-qbehaviour_studentquiz
- Quizzenger
  - https://quizzenger.hsr.ch
  - https://github.com/frankkoch/Quizzenger-2
- https://docs.moodle.org/dev/Plugin_contribution_checklist
- http://docs.moodle.org/en/Development:Guidelines_for_contributed_code
- http://docs.moodle.org/en/Developer_documentation#How_you_can_contribute

Management Summary

Ausgangslage:

Moodle, eine online Lernplattform mit über 80 Millionen Benutzern, bietet für die Überprüfung des Lernfortschritts Funktionalitäten an, so dass Lehrer Quiz für ihre Studenten erstellen können. Diese Quiz können aber nur durch die Lehrkraft erstellt, mit Fragen gefüllt und zur Beantwortung freigegeben werden. Das ganze System ist auf den aktiven Aufwand der Lehrkraft und das passive Beantworten auf Seite der Studenten aufgebaut. Genau diesen Punkt möchten wir mit unserem StudentQuiz ändern. StudentQuiz soll nicht nur eine Möglichkeit für Lehrer sein, ihren Studenten Fragen für die Wissenskontrolle anzubieten, sondern gibt auch den Studenten die Möglichkeit ihr Wissen während dem Semester zu verfestigen und das StudentQuiz auch als Tool für die Prüfungsvorbereitung zu verwenden. Die Grundsteine der zwei StudentQuiz Moodle Plugins wurden in einer vorhergehenden Arbeit umgesetzt.

Vorgehen/Technologien:

Im Rahmen unserer Studienarbeit bauen wir auf den zwei Moodle Plugins der Vorgänger auf und überarbeiten das vorhandene Projekt, so dass es den Validierungsprozess für das offizielle Plugin Directory von Moodle bestehen kann und zusätzlich mit der neuesten Version von Moodle kompatibel ist. Als erstes müssen vorhandene Fehler behoben und eine stabile Version für den Validierungsprozess erstellt und eingereicht werden. Zu den nötigen Anpassungen sollen zudem noch neue Erweiterungen und Übersichten hinzugefügt werden. Alle Arbeiten werden hauptsächlich in PHP und jQuery gemacht sowie die Datenbank mit MySQL verwaltet.

Ergebnisse:

Das StudentQuiz wurde so gestaltet, dass die Studenten zur aktiven Teilnahme am Fragepool und der Vorbereitung und Vertiefung eines Kurses bewegt werden. StudentQuiz erlaubt es Studenten Fragen für Mitstudenten zu erfassen sowie mit bereits erfassten Fragen Quiz zu erstellen und zu beantworten. Zusätzlich können die Fragen bewertet und kommentiert werden, um für bessere Fragen und ein besseres Verständnis zu sorgen. Der Lehrer hat die Möglichkeit Fragen visuell zu bestätigen, somit wird die Qualität der Fragen sichergestellt. Zwei verschiedene Übersichtsseiten pro Quiz, eine für die Rolle Student und eine für die Rolle Teacher, ermöglichen eine schnelle und einfache Übersicht über interessante Punkte des Quiz. Um die Wartbarkeit der beiden Plugins zu verbessern und mit dem Wissen, dass weitere Arbeiten dafür ausgeschrieben werden, wurde die Dokumentation von den einzelnen Komponenten verbessert und Hinweise zu möglichen Erweiterungen oder Änderungen angegeben.

Durch unsere Abhängigkeit von Drittpersonen konnten während der Dauer der Studienarbeit die beiden Plugins nicht ins Moodle Plugin Directory aufgenommen werden, aber es wurden etliche Anpassungen vorgenommen und alle gemeldeten Probleme gelöst. Somit ist das erfolgreiche Bestehen der Validierung nur noch eine Frage der Zeit.

Inhaltsverzeichnis

1. Einführung

1.1. Vision

Das Moodle Plugin StudentQuiz soll Studenten die Möglichkeit bieten in Moodle eigene Fragen zu erstellen und die der Mitstudenten zu beantworten. Hierfür sollen die Eigenschaften von Moodle voll ausgeschöpft werden und den Studenten ein gutes Werkzeug in die Hand legen.

1.2. Ziele

Ziel der Arbeit ist ein StudentQuiz Moodle Plugin zu entwickeln, das die Studenten zur aktiven Teilnahme am Fragepool, der Vorbereitung und Vertiefung eines Kurses bewegt. Zur einfacheren und offiziellen Benutzung des StudentQuiz soll der Validierungsprozess von Moodle durchlaufen und das Plugin anschliessend in das Moodle Plugin Directory aufgenommen werden. Mithilfe von StudentQuiz sollen Studenten eigene Fragen für Mitstudenten erfassen können, sowie mit bereits erfassten Fragen Quiz erstellen und lösen. Zusätzlich können die Fragen bewertet und kommentiert werden, um für bessere Fragen und ein besseres Verständnis zu sorgen. Der Lehrer hat die Möglichkeit Fragen visuell zu bestätigen, wodurch die Qualität der Fragen sichergestellt wird. Darüber hinaus haben wir einige Ziele in der Aufgabenstellung unter der Priorität A und Teile von B umgesetzt.

1.3. Rahmenbedingungen

Es wird das Vorprojekt des Moodle Plugins StudentQuiz fortgeführt. Dieses wurde von einer Studentengruppe als Semesterarbeit erstellt.

Da das Plugin auf Moodle basiert, gibt es einige Abhängigkeiten davon. Hierbei muss beachtet werden, dass keine Änderungen und Anpassungen an Moodle selber gemacht werden sollen. Die grösste Abhängigkeit steht zum Quiz-Plugin und Question-Plugin von Moodle, die Standardbestandteil dessen sind.

1.4. Vorgehen

Das Vorgehen an die Probleme wird in verschiedene Phasen unterteilt, die sich auch im Kapitel 5 detailliert behandelt. Durch die Aufgabenstellung wurde die Ausgangslage analysiert, um ein klares Bild der Struktur und des Problems zu erhalten. Hier wurden oft auch unterschiedliche Lösungswege betrachtet. Nach der Analyse wurde jeweils die beste Methode für die Lösung gewählt. Ergaben oder zeigten sich daraus weitere Komplikationen, die nicht im Rahmen der Semesterarbeit gelöst werden konnten, wurden diese jeweils in den Reflektionen dokumentiert.

2. Vorprojekt

Als Vorlage für dieses Projekt dient das Vorprojekt von den Studenten Dena Moshfegh und Manuel Schuler [1], welche das Moodle Plugin StudentQuiz ursprünglich erstellten. Es wurde auf Basis von unterschiedlichen Moodle-Komponenten entwickelt. Hauptsächlich wurde das Quiz-Plugin zur Hilfe genommen.

2.1. Architektur

Die Architektur des StudentQuiz-Plugins basiert hauptsächlich auf dem Standardaufbau für Moodle Plugins. Eine Ausführliche Beschreibung der Filestruktur findet man auf dem offiziellen Moodle Wiki für Entwickler [2]. Eine genaue Beschreibung der Architektur findet man in der Arbeit der Vorgänger. Insbesondere soll auf die Domain-Analyse verwiesen werden. Weitere wichtige Kapitel sind Design und Implementation.

Auch in dieser Arbeit wird im Bezug zur Architektur eingegangen. Dabei werden Erweiterungen und Anpassungen aufgezeigt.

2.2. Erweiterungen und Anpassungen

In dieser Arbeit wurden mehrere Erweiterungen gemacht. Dazu gehören einige Anpassungen der StudentQuiz-Hauptseite. Weiter wurde das Dashboard und Punktesystem stark überarbeitet. Zudem wurde die Navigation vereinfacht und einzelne Punkte entfernt.

Grundsätzlich wurde aber am Konzept und an der Architektur nicht viel verändert.

2.3. Anpassungen für Validation Process

Um die beiden Plugins ins Moodle Plugin Directory einreichen zu können, musste insbesondere noch Namensgebungen sowie Zugriffsüberprüfungen angepasst werden. Genauere Details zur Umsetzung ergehen aus dem Kapitel 5.1.

3. Projektaufbau

3.1. Moodle

Moodle ist eine Open-Source Lernplattform basierend auf PHP und kann gratis mit geringem Aufwand auf einem beliebigen Webserver installiert werden. Die Software wurde Modular aufgebaut, daher auch das M in Moodle, und bietet somit eine optimale Grundlage für Erweiterungen wie StudentQuiz.

Anfangs Dezember wurde die neue Version von Moodle veröffentlicht [3]. Nebst Erneuerungen wie Private Nachrichten und eine neue Diagramm API, ist die markanteste Erneuerung wohl das neue Boost Theme. Es kommt im modernen Bootstrap [4] Design daher und soll benutzerfreundlicher als vorherige Themes sein.

3.2. Use Cases

3.2.1. Use Case Diagramm

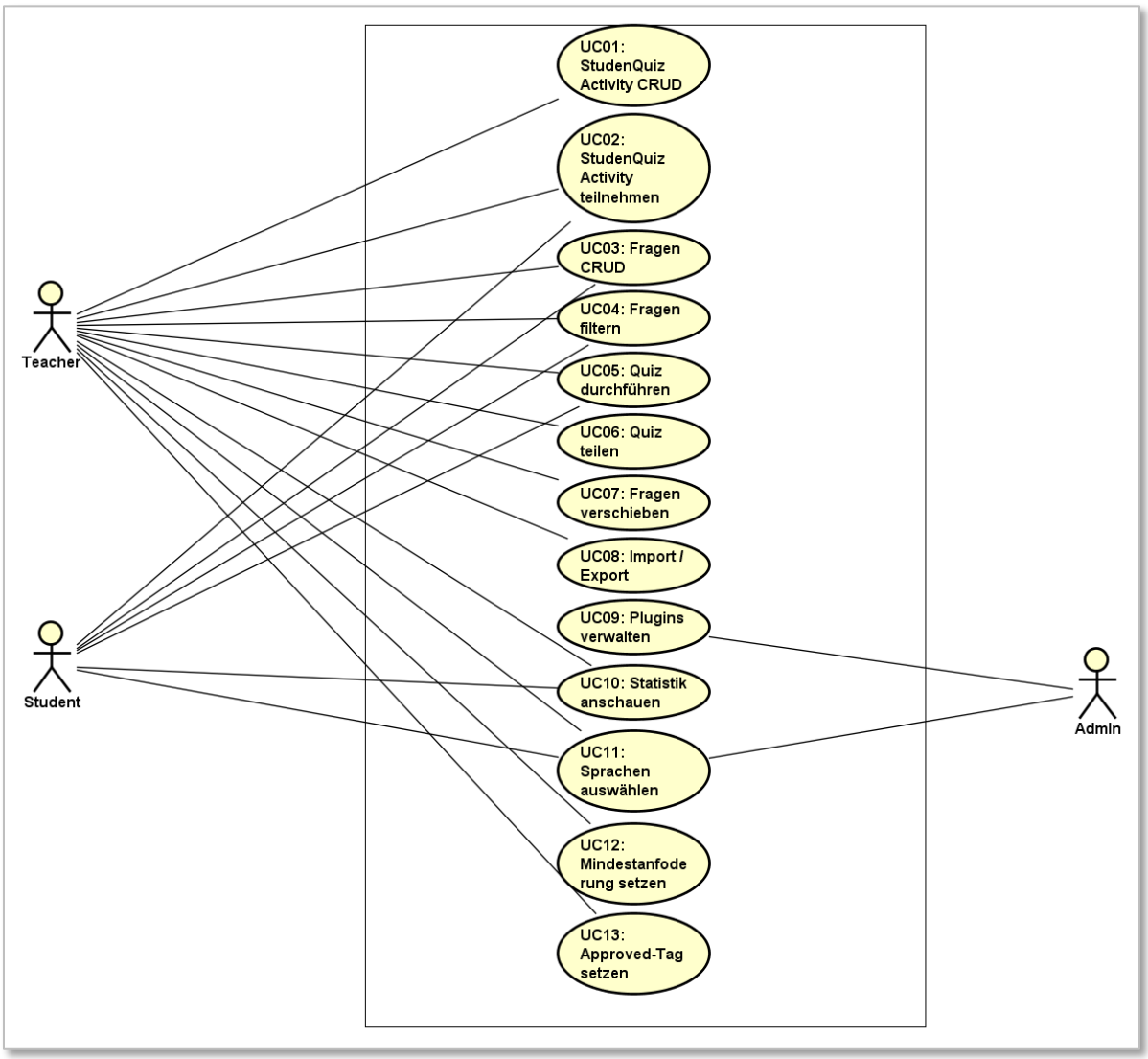


Abbildung 1: Use Case Diagramm

3.2.2. Aktoren und Stakeholders

Aktoren	Beschreibung
Teacher	Hat die Rolle Teacher inne. Ist ein Kursleiter und verantwortlich für die Inhalte in dem jeweiligen Kurs. Kann das StudentQuiz erstellen und löschen, sowie alles Dazugehörige verwalten.
Student	Hat die Rolle Student inne. Ist in einem Kurs angemeldet und kann das StudentQuiz des Lehrers benutzen. Kann dabei eigene Fragen erfassen, kontrollieren lassen, andere Fragen mitbeantworten, Kommentieren und Bewerten.
Kursteilnehmer	Alle Teilnehmer eines Kurses, das beinhaltet die Rollen Teacher und Student.
Administrator	Hat die Rolle Admin inne. Hat die Rechte für alle Use Cases sowie für die Verwaltung der Plugins.

Tabelle 1: Aktoren und Stakeholders

3.2.3. Use Case Beschreibung (Brief)

Die Use Cases UC01 bis UC09 wurden vom vorhergehenden Projekt übernommen. Einige wurden dabei angepasst und zusammengeführt. Die übrigen Use Cases (UC10 bis UC13) sind Neue, welche wir in dieser Arbeit erstellt und definiert haben.

UC01 StudentQuiz Activity CRUD

Teacher können eine oder mehrere StudentQuiz Activities in einem Kurs erstellen, verwalten und löschen.

UC02 StudentQuiz Activity teilnehmen

Alle Kursteilnehmen (sofern zugelassen) können am StudentQuiz teilnehmen. Dabei können einige weitere Use Cases durchgeführt werden, wie zum Beispiel neue Fragen erstellen, vorhandene Fragen beantworten oder eigene Fragen löschen.

UC03: Fragen CRUD

Alle Kursteilnehmer können neue Fragen erfassen. Dabei sollen folgende Punkte möglich sein:

- Neue Frage erstellen
- Aus verschiedenen Fragetypen auswählen
- Frage mit Tags klassifizieren
- Frage verwalten
    - Beachtet werden muss dabei, dass Teacher alle Fragen verwalten können. Die Studenten dürfen aber nur ihre eigenen Fragen verwalten können.
- Frage löschen

UC04: Fragen filtern

Kursteilnehmer können die Fragen im StudentQuiz nach verschiedenen Kriterien filtern. Die Kriterien umfassen folgende Punkte:

- Bewertung
- Schwierigkeit
- Tag

- Durchführungen
- Kommentare
- Fragetitel
- Frageinhalt
- Bestätigte Fragen
- Datum
- Abhängig von der Rolle
    - Teacher: Vorname, Nachname
    - Student: Meine Fragen

UC05: Quiz durchführen
Kursteilnehmer können Fragen aussuchen und mit diesen ein Quiz starten. Dabei können die Fragen beantwortet, bewertet und falls gewünscht kommentiert werden.

UC06: Quiz teilen
Teacher können erstellte Quizinstanzen mithilfe eines Links den Studenten zur Verfügung stellen.

UC07: Fragen verschieben
Teacher können ausgewählte Fragen in eine andere Kategorie verschieben.

UC08: Import / Export
Teacher können Fragen in den Fragepool importieren und ausgewählte Fragen aus dem Fragepool exportieren.

UC09: Plugins verwalten
Der Administrator kann die Plugins aktualisieren und verwalten.

UC10: Statistik anschauen
Studenten können ihre eigene Statistik in jedem StudentQuiz anschauen. Teacher können die gesamte Statistik des jeweiligen StudentQuiz anschauen.

UC11: Sprachen auswählen
Kursteilnehmer können ihre eigene Sprache auswählen (sofern der Administrator die Sprache im Moodle installiert hat). Drei Sprachen sollen verfügbar sein: Englisch, Deutsch und Französisch.

UC12: Mindestanforderungen setzen
Teacher können für jedes StudentQuiz eine „Activity completion“ setzen, die von den Studenten erfüllt sein muss, bevor sie für den Lehrer als erledigt gekennzeichnet wird.

UC13: Approved-Tag setzen
Um die Qualität der Fragen sicherzustellen, können Teacher Fragen mittels eines bewährten Häkchens bestätigen. Genauso gut können sie die Bestätigung der Fragen wieder rückgängig machen.

3.2.4. UC13 Approved-Tag Fully Dressed

Primary Actor	**Teacher**
Overview	Um die Qualität der Fragen sicherzustellen, können Teacher Fragen mittels eines bewährten Häkchens bestätigen. Zusätzlich können sie die Bestätigung der Fragen wieder rückgängig machen.
Aktoren und Stakeholders	Teacher: • Möchte die Qualität der Fragen sicherstellen, deshalb überprüft er sie. Wenn die Frage korrekt ist, wird sie bestätigt, ansonsten wird die Frage angepasst. Student: • Möchte die Gewissheit, dass seine Frage richtig ist • Möchte mit bestätigten Fragen üben
Pre-Condition	UC03: Frage wurde erfolgreich erstellt
Post-Condition	Der Bestätigt-Wert der ausgewählten Frage wurde verändert
Main success scenario	1. Der Teacher wählt eine nicht bestätigte Frage auf der Hauptseite des StudentQuiz aus 2. Der Teacher überprüft die Frage und deren Antworten auf ihre Korrektheit 3. Der Teacher bestätigt die ausgewählte und korrekte Frage
Alternative	1a. Frage ist schon bestätigt: 1a. Der Teacher überprüft die Frage nochmals 1a. Der Teacher ändert den Bestätigungswert der Frage auf nicht bestätigt 1b. Der Teacher lässt die Frage unberührt 3a. Die Frage ist nicht korrekt: 3a. Der Teacher korrigiert die Frage und bestätigt sie 3b. Der Teacher lässt die Frage unbestätigt
Frequency of Occurrence	Mehrmals täglich, nach Belieben durch den Teacher

Tabelle 2: Fully Dressed Use Case: Approved-Tag

3.3. Struktur

3.3.1. Tabellenstruktur

Die Datenstruktur des StudentQuiz ist zum Teil stark abhängig von den umliegenden Daten, die zu den Quiz und den Questions gehören. Für die Durchführung eines Quiz wurden die wichtigsten Klassen zum Thema zusammengefasst. Dies kann zukünftigen Entwicklern zu einem besseren Verständnis der Funktionsweise des StudentQuiz verhelfen.

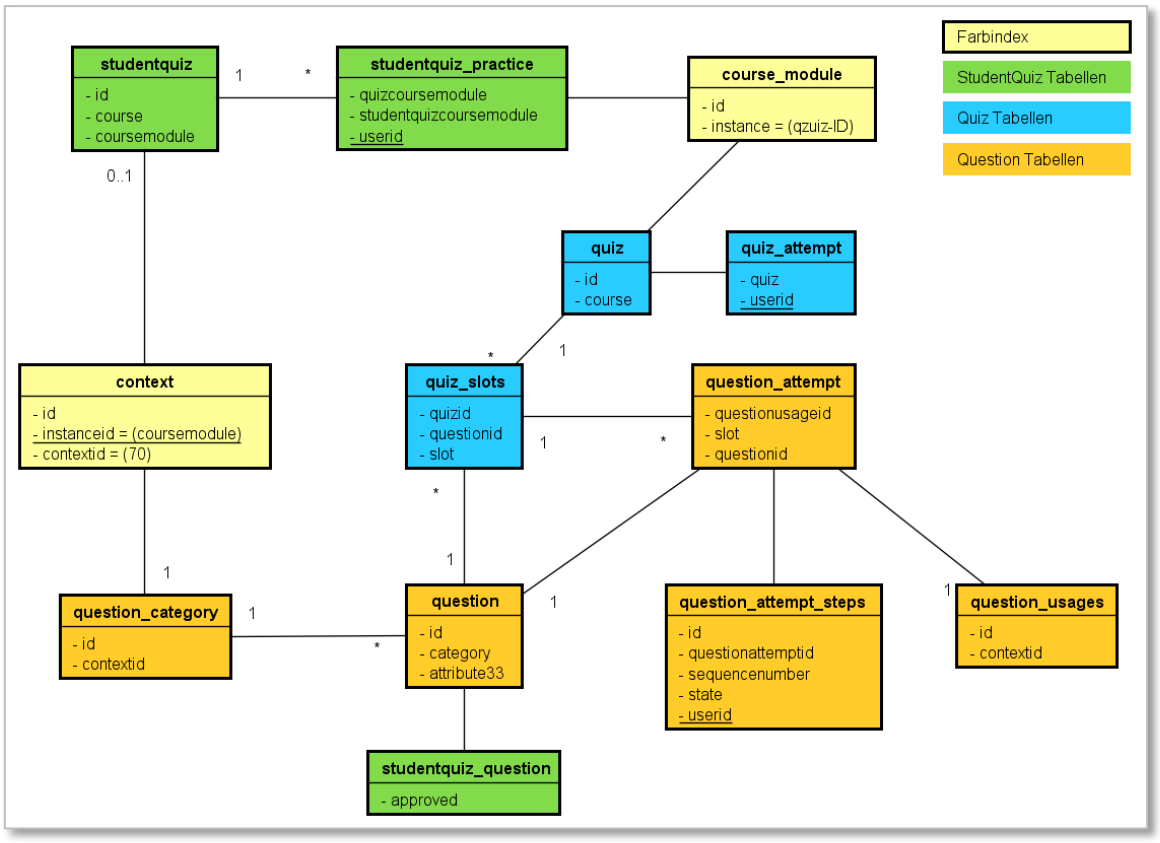


Abbildung 2: Datenbanktabellenstruktur

Im Folgenden werden einzelne wichtig Tabellen ausführlicher beschrieben. Nicht hier beschriebene Tabellen des StudentQuiz werden in der Dokumentation der Vorgänger genauer beschrieben.

studentquiz

Dies ist die Haupttabelle von StudentQuiz. Grundlegende Daten sind hier gespeichert. Wichtig sind hier die Felder course und coursemodule. Course ist der aktuelle Kurs und coursemodule ist die ID der Activity im Kurs.

question_category

Jedem StudentQuiz ist eine Kategorie von Fragen zugewiesen. Dies entspricht der Questionbank, die die Fragen beinhaltet.

studentquiz_practice

Speichert die Teilnahme des Users am StudentQuiz. Gespeichert wird die ID über das course_module, welches die Activity beschreibt. Eine einzelne Durchführung wird in quiz_attempt gespeichert und die Instanz wird im Quiz gespeichert.

question_attempt
Speichert die einzelnen Versuche auf einer Frage durch einen User. Der User wird aber auf dem question_attempt_steps gespeichert. Doch gibt es mehrere Wege um an Daten zu kommen.

studentquiz_question
Diese Tabelle speichert die Fragen zum StudentQuiz. Es geht momentan noch vor allem darum um zu prüfen ob die Fragen approved wurden.

3.3.2. Dateistruktur

Dieses Kapitel beschreibt kurz, welches File für was zuständig ist. Es werden nicht alle Files des Studentquiz durchgegangen, nur die die sehr wichtig sind.

lib.php
Die Schnittstelle von Moodle an das Plugin, für einzelne Einstellungen und Events. Z.B. wenn ein neues StudentQuiz erstellt wird.

locallib.php
Bietet diverseste Methoden für den internen Gebrauch im Plugin.

mod_form.php
Methode für die Settings eines neuerstellten StudentQuiz.

renderer.php
Klasse, welche Methoden zum Rendering der Views anbietet. (Wird benötigt für Haupt-View und die Reports (Dashboard)

reportlib.php
Library für die Berechnungen und Sammlungen der Report-Daten. Wird hauptsächlich vom Dashboard benutzt. Aber auch zur Berechnung der Grades.

reportquiz.php
Anzeige-File für das Dashboard. Die URL für das Dashboard wird auch auf dieses File referenziert.

reportrank.php
Anzeige-File für das Ranking.

settings.php
Einstellungen für das Plugin werden hier definiert. Diese sind im Moodle unter Siteadministration verfügbar.

version.php
Definition der Version und etwaigen Abhängigkeiten.

view.php
Hauptseite des StudentQuiz.

Viewlib.php
Bietet die wichtigsten Methoden für das Erstellen und Starten von StudentQuiz.

4. Projektmanagement

4.1. Rollen und Verantwortlichkeiten

4.1.1. Prof. Frank Koch

Prof. Frank Koch Dozent an der HSR agiert als Betreuer der Studienarbeit und ist dafür verantwortlich, dass die Arbeit problemlos und zielführend umgesetzt wird.

4.1.2. Lukas Dürrenberger

Lukas Dürrenberger ist Informatikstudent an der HSR im Vollzeitstudium. Für die Studienarbeit übernimmt er die Rolle als Projektmanager und ist zusätzlich noch Teil des Entwicklerteams.

4.1.3. Steven Ryser

Steven Ryser ist Informatikstudent an der HSR im Vollzeitstudium. Er übernimmt bei der Studienarbeit insbesondere die Verwaltung von Dokumenten und das Protokollieren von unseren Sitzungen. Zusätzlich ist er Teil des Entwicklerteams.

4.1.4. Alexis Suter

Alexis Suter ist Informatikstudent an der HSR im Vollzeitstudium. Er ist Teil des Entwicklerteams und übernimmt die Verantwortung für komplexere Aufgabenstellungen.

4.2. Projektplan

4.2.1. Aufwandschätzung

Die Zeiteinteilung wird auf die einzelnen Iterationen abgestimmt sein. Des Weiteren wird auf die Priorisierung der einzelnen Aufgaben Rücksicht genommen. Die gewünschten Aufgaben welche mit der Priorität A gekennzeichnet sind, sollten alle erledigt werden. Aufgaben mit der Priorisierung B sind als „Nice to have“ Aufgaben nicht so wichtig. Wir wollen diese auch erreichen, falls dafür genügend Zeit zur Verfügung steht. Der Aufwand liegt also ganz klar bei den A priorisierten Aufgaben. Die Aufwandschätzung wird also wie folgt festgelegt:

In der Construction1-Phase werden alle Bugs in der alten Version behoben und Optionen verbessert. Das Ziel ist eine fehlerfreie Version, welche so den Moodle Plugin Validierung Prozess beginnen kann.

In der Construction2-Phase werden alle Tasks und Features welche eine hohe Priorisierung haben, und somit zur geplanten Arbeit gehören, realisiert. Das Ziel ist, dass alle Priorität A Aufgaben verwirklicht werden. Dabei wird laufend Fehlerbehandlung betrieben.

In der Construction3-Phase, der letzten Construction-Phase, liegt das Hauptaugenmerk auf einer stabilen und fehlerfreien Version und das Anpassen des Plugins auf die Moodle Version 3.2. Hier werden zusätzlich, soweit genügend Zeit vorhanden ist, möglichst viele Priorität B Aufgaben entwickelt.

4.2.2. Iterationen

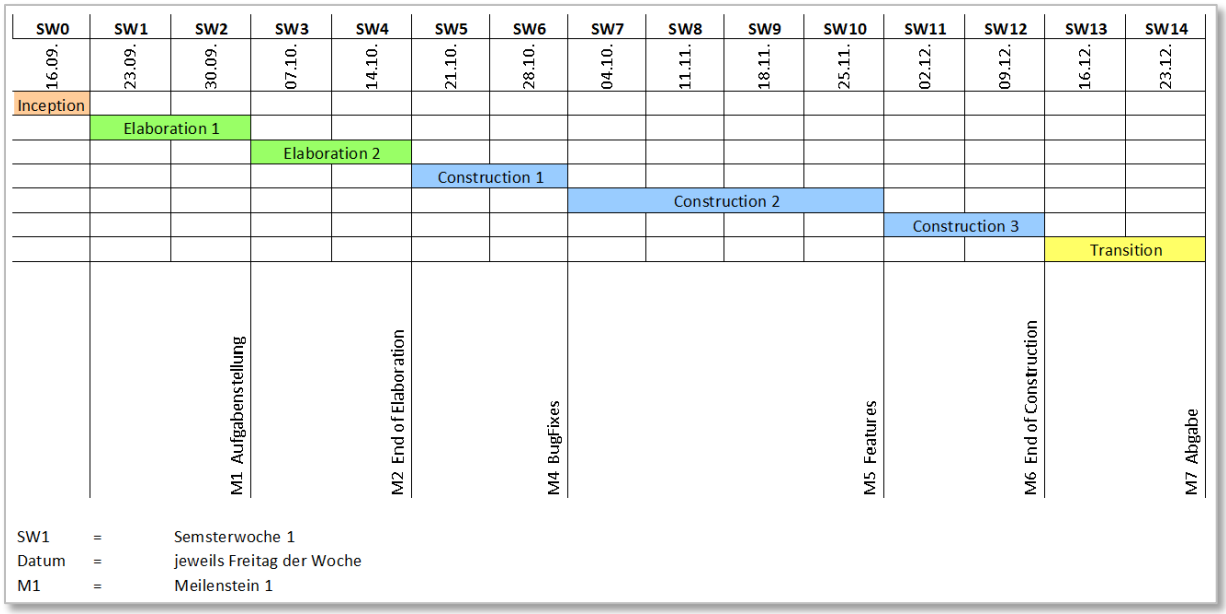


Abbildung 3: Iterationen

4.2.2.1. Inception

Während der Inception Phase sollen die Vorschläge zur Aufgabenstellung genauer analysiert werden und das Team kann sich finden und einrichten.

4.2.2.2. Elaboration

In dieser Phase wird die Aufgabenstellung klar definiert und die Ziele sowie die gemeinsame Vision ausgearbeitet. Zusätzlich steht Zeit zur Verfügung um sich in den bestehenden Quellcode einzulesen und Moodle an sich besser kennen zu lernen.

4.2.2.3. Construction

Die Construction Phase ist insbesondere der Weiterentwicklung des StudentQuiz gewidmet. Zuerst sollen die bestehenden Bugs gefixt werden, danach die neuen Erweiterungen mit Priorität A umgesetzt werden und schlussendlich steht noch Zeit zur Verfügung um allfällige Fehler zu korrigieren, sowie die Möglichkeit einige Priorität B Features zu implementieren.

4.2.2.4. Transition

Während der Transition Phase wird der Code nur noch minimalst verändert um allfällige Bugs noch zu fixen. In dieser Phase gilt es insbesondere die Dokumentation fertig zu stellen und alle anderen Dokumente aufzufrischen.

4.2.3. Zeitplan

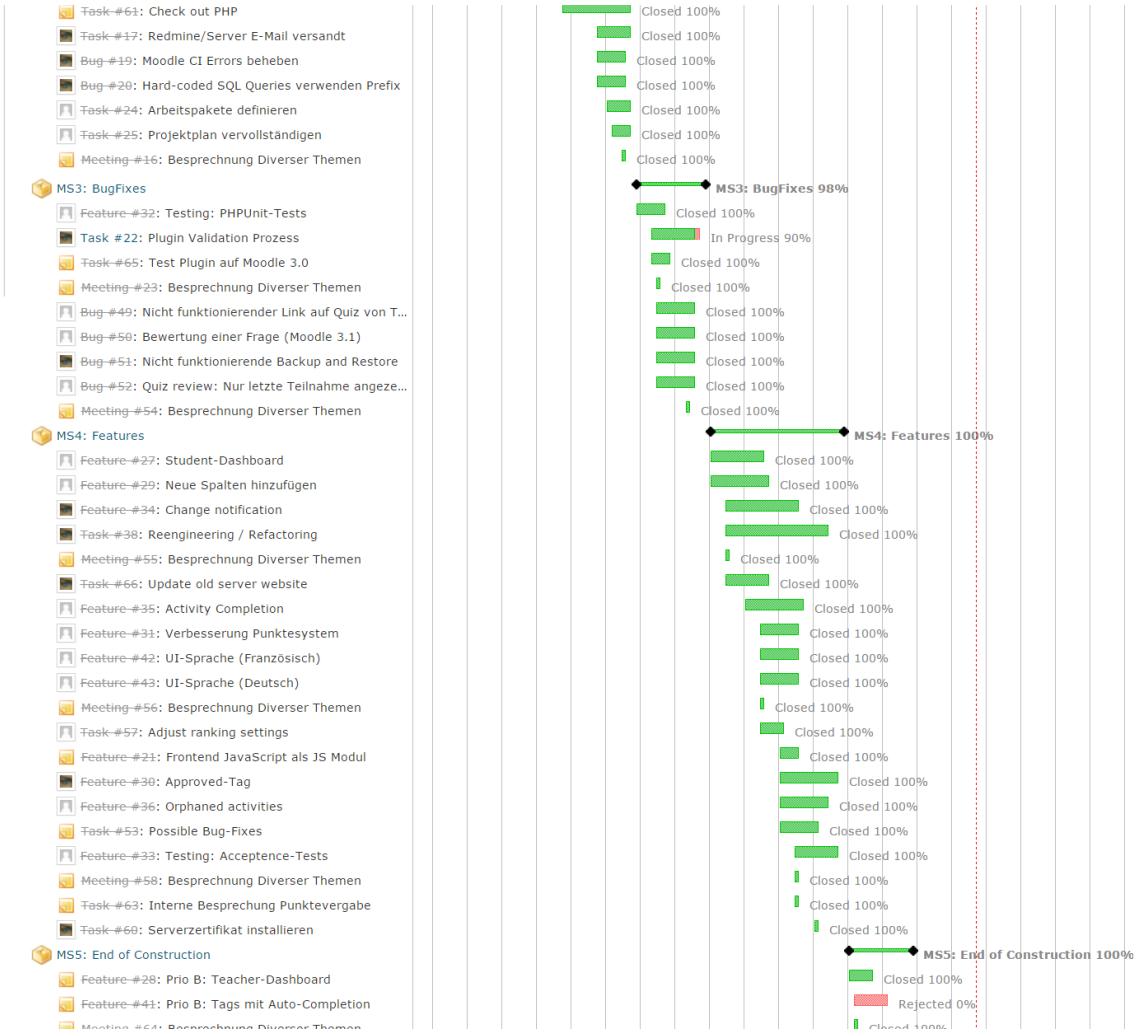


Abbildung 4: Zeitplan als Ganttdiagramm (Ausschnitt)

Dies ist ein Ausschnitt des Ganttdiagramms. Das komplette Ganttdiagramm ist in der digitalen Abgabe zu finden.

Alle Arbeitspakete wurden in vier Kategorien unterteilt:

- Bug
- Feature
- Task
- Meeting

Die Kategorie Bug enthält alle Aufgaben die entweder die Korrektur bekannter Bugs beinhalten oder neue Bugs die wir während der Arbeit erkannt haben.
In der Kategorie Feature sind alle Aufgaben, welche eine neue oder zusätzliche Erweiterung beschreiben.
Die Kategorie Task besteht aus allen Aufgaben die keine direkten Codeänderungen zur Folge haben, sondern eher längere Aufgaben die der Entwicklungsumgebung, zum Bsp. Administration oder Infrastruktur, dienen.
Die letzte Kategorie ist selbsterklärend. Hier wurden alle Meetings erfasst.

Im Ganntdiagramm ist gut erkennbar, dass die meisten Aufgaben erfolgreich abgeschlossen wurden.

Da bei der Priorität A Aufgabenentwicklung Qualität im Vordergrund stand, gibt es ein paar Prioritäten B Aufgaben die wir nicht mehr realisieren konnten.

Auch sieht man dass es noch ein offenes Arbeitspaket gibt. Das "Test validation process". Der ist von unserer Seite abgeschlossen, aber von Moodle noch nicht bestätigt, sondern befindet sich noch in deren Queue. Und natürlich die Projektabgabe, welche erst abgeschlossen sein wird, wenn wir alle Dokumente eingereicht haben.

4.3. Infrastruktur

4.3.1. Redmine

Wir hatte Redmine [5] als eine freie, webbasierte Projektmanagementsoftware bereits in der Software Engineerings Vorlesung kennen gelernt und teils für das Engineering Projekt an der HSR verwendet. Für die Arbeit am StudentQuiz half uns die Software insbesondere dadurch, dass sie es erlaubt eine grobe Planung der einzelnen Iterationen durchzuführen. Somit erhielten wir schon sehr früh einen Überblick zu den einzelnen Iterationen. Des Weiteren wurden alle grösseren Bugs, Features und Tasks erfasst, womit wir im Detail sehen konnten, was noch ansteht und wer für was zuständig ist. Und schlussendlich gibt das Ganttdiagramm eine komplette Projektübersicht, welche man sich von manch anderen Tools nur wünschen könnte.

4.3.2. Trello

Obwohl Redmine im Gesamtbild sehr gut ist, so ist die Software doch auch schon etwas älter, was man insbesondere bei der Benutzbarkeit zu spüren bekommt. Wenn man zum Beispiel einen neuen Task erstellen möchte, so muss man zuerst zum spezifischen Projekt navigieren, dann ein neues Issue erstellen und dort zwischen all den verschiedenen Optionen alles richtig einstellen. Für kleine Aufgaben bei welchen man nicht einmal weiss, ob sie wirklich einen Eintrag ins Redmine benötigen, hatten wir uns auf Trello [6] festgelegt. Trello arbeitet mit Spalten und Karten so ziemlich nach dem Kanban Prinzip [7]. Es kann frei verwendet werden und durch die einfache Bedienung können neue Aufgaben in Sekunden schnelle hinzugefügt werden.

Trello fand für uns vor allem zu Beginn einen grösseren Nutzen, bis wir alles im Redmine richtig eingeordnet hatten. Da es so einfach ist ein neues Trello Board zu erstellen, ist Trello das perfekte Tool um einfach Mal anzufangen.

4.3.3. Toggl

Aus einer Erfahrung mussten wir feststellen, dass es oft sehr schwierig ist, sich die verwendete Zeit für eine bestimmte Aufgabe zu merken oder des Öfteren vergisst man es einfach aufzuschreiben. Toggl [8] kann frei verwendet werden und bietet eine einfache Zeiterfassung an. Durch Browser Integrationen konnten wir schlussendlich sogar Timers direkt auf Redmine oder Trello starten. Ohne Toggl hätten wir höchst wahrscheinlich keine solch detaillierte Zeiterfassung, der einzige Nachteil war, dass wir die Zeiten am Schluss manuell ins Redmine eintragen mussten.

4.3.4. GitHub

GitHub [9] ist wohl mittlerweile der bekannteste Anbieter zum Hosten von Git Repositories. Da der bestehende Source Code sowieso bereits auf GitHub [10] [11] lag, war es sehr einfach einen Fork davon zu erstellen, welcher dann unser aktiver Arbeitsbereich wurde.

4.3.5. VM Server

Von der HSR wurde uns eine VM als Server zur Verfügung gestellt, womit wir stetig eine aktuelle

Version hatten, auf welche man von überall her zugreifen konnte. Die Moodle Installationen auf dem Server dienten insbesondere um den Vorschritt mit Prof. Koch zu präsentieren und diskutieren. Der Server wurde des Weiteren verwendet um Redmine darauf zu hosten.

4.3.6. Lokaler Webserver

Für die eigentliche Entwicklung am StudentQuiz hatte sich schlussendlich jeder einen lokalen Webserver mit Hilfe von XAMPP [12] eingerichtet, so konnte jeder unabhängig voneinander arbeiten. Die einzelnen Änderungen wurden dann in separate Branches über GitHub synchronisiert.

4.3.7. PHPStorm

JetBrains [13] bietet für Studenten gratis Lizenzen zu all ihren IDEs unter anderem auch PHPStorm [14]. Insgesamt bot uns PHPStorm eine solide Umgebung an und kam auch problemlos mit der doch relativ grossen Codebase von Moodle zurecht.

4.4. Risikomanagement

Risk ID	Risiko	Massnahme	Eintretens-wahr-scheinlich-keit	Priorität
R01	Personalausfall	Gute Planung und Kommunikation	5%	**gering**
R02	Unrealistische Zeitplanung	Zeitplan festlegen (Meilensteine, usw.), Reserve sicherstellen	50%	**mittel**
R03	Hardwareausfall (Server)	Server wird von der HSR gewartet	1%	**gering**
R04	Hardwareausfall (Clients)	Redundante Systeme bereitstellen	5%	**gering**
R05	Unzureichende Softwaretechnologie	Bewährte Tools einsetzen	20%	**gering**
R06	Vergolden des Systems	Definition of Done definieren	75%	**hoch**
R07	Unerwartete Bugs	TDD, CodeReviews durchführen	50%	**mittel**
R08	Verlust Projektdaten	Backups erstellen und Wiederherstellung ermöglichen	5%	**gering**

Tabelle 3: Risikoanalyse

4.5. Qualitätsmassnahmen

4.5.1. Travis CI

Schon zu Beginn der Studienarbeit war uns klar, dass wir für die Weiterentwicklung des StudentQuiz eine Continuous Integration System benötigen, damit wir stetig unsere Unit-Tests, aber auch weitere statischen Code Analysen ausführen können. Dadurch, dass wir sehr früh auf das Moodle Plugin CI Softwarepaket [15] gestossen sind und dieses insbesondere für die Verwendung mit Travis CI [16] entwickelt wurde, fiel uns die Entscheidung leicht und setzten innert wenigen Minuten eine Travis Konfigurationsdatei auf und schon erhielten wir die ersten Reports. Für Open-Source Projekte kann Travis CI frei verwendet werden und integriert sich direkt mit GitHub.

4.5.2. Moodle Plugin CI

Das Softwarepaket ist freiverfügbar und den Tipps für das Einrichten von Travis CI, kann man die ersten Code Analysen schneller starten, als mit vielem anderen. Moodle Plugin CI bietet die folgende Code Checkers an:

- PHPUnit - Unit Testing Framework
- Behat - Acceptance Testing Framework
- Moodle Code Checker - Moodle Code Style [17] Überprüfung
- PHP Linting - Allgemeiner PHP Code Checker
- PHP Copy/Paste Detector - Erkennen von dupliziertem Code
- PHP Mess Detector - Ein weitere PHP Code Checker
- CSS Lint - Allgemeiner CSS Syntax Checker
- JSHint - Allgemeiner JavaScript Code Checker
- Shifter - Überprüfung von YUI Code

Durch die zur Verfügung gestellten Logs konnten wir viele unschöne Codestellen ausbessern und somit die Codequalität von StudentQuiz erhöhen.

4.5.3. Versionierung

Um möglichen Datenverlust durch Hardware- oder Softwarefehler zu minimieren, haben wir für das StudentQuiz Git eingesetzt. Die dezentralisierte Art von Git gab uns die Sicherheit, dass immer jemand noch eine Kopie des geschriebenen Codes besitzt. Bei einem plötzlichen Ausfall würde somit nur eine minimale Menge an Daten verloren gehen, nämlich nur das, was noch nicht gepusht wurde. Glücklicherweise kam es jedoch nie zu Problemen oder Verlusten von Source Code.

Damit es zu möglich wenigen Konflikten auf dem Git Repository kam, arbeitete jede Person auf ihrem eigenen Branch. Sobald die Änderungen abgeschlossen waren und Travis CI für den Commit grünes Licht gab, übernahm Lukas Dürrenberger als Merge Dictator [18] und führte die Branches in den Entwicklungsbranch ein. Falls ein Branch etwas älter war, wurde zuerst ein Rebase auf den Entwicklungsbranch durchgeführt und nach einem erneuten Durchlauf von Travis CI zusammengeführt.

4.5.4. Sitzungsprotokolle

Durch Sitzungsprotokollen zu jeder Besprechung mit Prof. Koch werden Traktanden, Beschlüsse und offene Punkte erfasst, wodurch wir eine stetige Übersicht der aktuellen Situation haben und bei Ungewissheit nachschauen können, was in der Vergangenheit beschlossen wurde.

4.5.5. Manuelle Massnahmen

Die automatisierten Tests decken leider noch nicht alle Aspekte des StudentQuiz ab und so wurden während der Entwicklung die einzelnen Änderungen auch immer noch manuell verifiziert, ob diese auch tatsächlich funktionieren.

5. Vorgehen und Methoden

5.1. Moodle Plugin Validation Process

5.1.1. Aufgabenstellung

Durchlaufen des Moodle Plugin Validation Prozesses und Bereitstellung des Plugins (inkl. Question Behaviour) auf dem Moodle Plugin Directory.

5.1.2. Ausgangslage

Moodle ist modular aufgebaut und bietet in ihrem Plugin Directory [19] über 1200 verschiedene Plugins an. Um die Plugins im Directory auf einer gewissen Qualität zu halten, muss jedes Plugin durch einen Validierungsprozess [20] gehen. Dabei geht es insbesondere darum die Syntax nach ihren Regeln zu überprüfen und sicher zu stellen, dass die Umsetzung im Code sinnvoll ist und keine Probleme verursacht. Als Grundlage dazu gibt es offizielle Richtlinien, wobei diese nur einige allgemeine Aspekte anspricht.

5.1.3. Lösung / Umsetzung

5.1.3.1. Anpassungen nach den Richtlinien

Die Richtlinien zum Plugin Validierungsprozess beschreiben insbesondere allgemeine Dinge, wie zum Beispiel was für Komponenten vorhanden sein müssen und welche Variablen pflichtgemäss gesetzt werden müssen. Diese Vorgaben wurden zum grössten Teil bereits in der Vorarbeit umgesetzt und wir konnten die meisten Punkte direkt abhaken. Zur Anpassung des Codes gibt es bis auf den Hinweis zur Frankenstyle Namensgebung keine Anweisungen, aber es werden weiter Dokumentationen verlinkt. Nachdem einige Bugs gefixt wurden und der Code nach bestem Wissen angepasst wurde, reichten wir das Activity, wie auch das Question Behaviour Plugin ein.

5.1.3.2. Automatisierte Überprüfungen

Ähnlich unserer Lösung über Travis CI, wurden nach einer gewissen Zeit automatische Überprüfungen [21] [22] durchgeführt und es zeigte sich, dass wir doch noch ein paar Stellen übersehen hatten. Bislang wurde jQuery direkt mit dem Plugin geliefert, doch da jQuery komprimiert ist und einige Quirks hat, gefiel dies den Überprüfungen natürlich nicht so stark. Mit etwas Nachforschen konnten wir dann herausfinden, dass der von Moodle empfohlene Weg über ein JavaScript Module, auch bekannt unter der Abkürzung AMD [23]. Zusätzlich bemerkte der Überprüfungsreport, dass wir in gewissen Plugins keine Überprüfung hatten, ob es sich um eine interne Datei handelt. Mit den erwähnten und weiteren kleinen Anpassungen wurde eine neue Version erstellt und erneut hochgeladen.

5.1.3.3. Feedback

Durch die hohe Auslastung des Moodle Teams wegen dem Release von Moodle 3.2, bekamen wir für längere Zeit keine Rückmeldung zu unserem Plugin. Durch verschiedene Unterhalten in öffentlichen Chaträumen, erhielten wir einige Tipps und konnten schlussendlich auch ein grobes Feedback zu den beiden Plugins erhalten. Es stellte sich heraus, dass wir die Frankenstyle Namensgebung nicht ganz korrekt eingesetzt hatten. Des Weiteren fehlten noch Überprüfungen, ob ein Nutzer überhaupt eingeloggt ist und beim Question Behaviour wurden noch problematische Direktaufrufe zu den Parametervariablen $_POST und $_GET verwendet. Und diese problematischen Codestellen wurden überarbeitet und in einem neuen Release veröffentlicht.

5.1.3.4. Resultat

Da der Validierungsprozess von Drittpersonen abhängig ist und sich durch die verzögerten Feedbacks das Ganze in die längt gezogen hat, konnte wir leider der Aufgabenstellung nicht völlig gerecht werden, aber mit der geleisteten Arbeit und den Anpassungen sollte dies sicherlich der nächsten Gruppe gelingen.

5.1.4. Reflektion

Eine der grössten Herausforderungen bei der Umsetzung war das Finden von den nötigen Informationen. Das Moodle Wiki enthält sehr viele nützliche Aspekte und im Vergleich zu anderen grossen Projekten, ist es denen weitvoraus, aber trotzdem fehlen oftmals Details, wie zum Beispiel wann denn genau Funktionen nach Frankenstyle benannt werden müssen und wann es unter keinen Umständen so sein darf. Öfters steht auf den Seiten auch eine Moodle Versionsnummer, aber ob die Seite jetzt nur für die angegebene Version gilt oder für alle darauffolgende Versionen ebenfalls, ist nie klar definiert.

Für den Validierungsprozess wünschten wir uns insbesondere eine Checkliste mit den wichtigsten Punkten, welche sicherlich angeschaut werden. Auch würde es wohl den meisten Pluginentwickler helfen, wenn der gesamte Prozess etwas transparenter wäre.

5.2. Student-Dashboard

5.2.1. Aufgabenstellung

Erstellen eines Student-Dashboards (das auch für Teacher zugänglich ist), vielleicht mit folgenden Punkten:

a. Filtereinstellung (z.B. gelöst, ungelöst, falsch gelöst)
b. Lernfortschritt (wie viele Fragen beantwortet), Ranking

5.2.2. Ausgangslage

Zur Übersicht der Studenten und Dozenten wird ein Dashboard verlangt, welches zusätzliche (statistische) Informationen beinhaltet. Das Dashboard soll zum einen über den eigenen Fortschritt, sowie eine Art Rangliste über die Studenten bereitstellen. Dozenten sollen zudem die Möglichkeit haben den Fortschritt der einzelnen Studenten einzusehen.

5.2.3. Lösung / Umsetzung

Das Dashboard soll in der Übersicht des StudentQuiz einblendbar sein, wie die Filter über Auf- und Zuklappen. Alternativ kann das Dashboard als Menüpunkt in der Navigation unter dem StudentQuiz erscheinen, was jedoch das Laden einer weiteren Seite voraussetzt.

Der Aufbau des Dashboards soll folgendermassen sein: Zuoberst kommen die Persönlichen Status-Informationen. Darunter erscheint direkt die Ranking-Tabelle aller Studenten, sortiert nach maximal erreichter Punktzahl in diesem Quiz. Der User selber soll darin fett markiert sein. Der Dozent sieht zuerst nur die Ranking-Tabelle und kann durch Anklicken eines Studenten dessen persönlicher Status im StudentQuiz einsieht.

Die Status-Informationen sollen bestehen aus:

1 Menü Eintrag Dashboard, Wie viele Fragen er hat, (Auch die die noch nicht approved sind) Fragen, die er noch nicht beantwortet hat, bzw. wie viele er beantwortet hat.

Eine klare Definition wurde aber während der Umsetzung mit Herrn Koch erarbeitet. Dabei wurden einige Anpassungen für die Reihenfolge und Bezeichnungen durchgeführt.

Das Grade (Benotung) wurde eingeführt und in das Dashboard genommen. Das Grade ist eine Summe der maximal erreichten Punktzahl pro Frage im Studentquiz. Dieser Wert kann weiterverwendet werden an anderen Orten und wurde für den Beginn in das Dashboard übernommen.

Die Berechnungen gewisser Werte durch die Vorgänger stimmten noch nicht ganz. Diese Fehler wurden Grossteils verbessert.

5.2.4. Berechnung einzelner Punkte anhand eines Beispiels

Hier wird anhand eines kurzen Szenarios die Berechnung einzelner Punkte beschrieben.

Szenario
StudentQuiz mit 5 Fragen a je 1 Punkt.
Anzahl Durchführungen mit allen Fragen: 2

You have answered correctly

Hier können maximal 5 Punkte erhalten werden. D.h. pro Frage im StudentQuiz wird nur einmal inkrementiert. Normalerweise gilt nur die letzte Antwort für das Inkrement.

Your grade total

Das Grade ist eine Art Benotung anhand der korrekt beantworteten Fragen. Da im Szenario pro Frage nur 1 Punkt vergeben wird, können auch hier maximal 5 Punkte erreicht werden.

Total of your answers

In den zwei Durchführungen, können maximal 10 Antworten gegeben werden.

Total of correct answers

Die Anzahl der korrekten Antworten aus den zwei Durchführungen. Also maximal 10.

5.2.5. Reflektion

Das Dashboard ist ein gutes Mittel um den Teilnehmern einen Überblick über dessen Leistungen zu bieten.

Mögliche Anpassung, wäre eine klarere und verständlichere Übersicht.

5.3. Teacher-Dashboard

Das Teacher-Dashboard ist dem Student-Dashboard zwar sehr ähnlich, stellt aber auch Daten zu den Studenten dar.

5.3.1. Aufgabenstellung

Den Lehrern soll – ähnlich dem Student-Dashboard – eine Übersicht über Zahlen im Quiz gegeben werden. Zusätzlich sollen auch die Daten der Studenten des Modules angezeigt werden.

5.3.2. Ausgangslage

Es wird vom Student-Dashboard ausgegangen nur, dass dann die Daten aller Studenten im Modul dazu genommen werden.

5.3.3. Lösung / Umsetzung

Für die Tabelle über alle Schüler, werden für jeden Schüler die Daten dessen Dashboards gesammelt. Diese werden dann im renderer.php aufgebaut.

5.3.4. Reflektion

Das Teacher-Dashboard bildet eine gute Erweiterung des Student-Dashboards, indem die eigenen Ergebnise sowie die der Schüler angezeigt werden.

5.4. Neue Spalten

5.4.1. Aufgabenstellung

Neue Spalten «Anzahl Durchführungen» und «Anzahl Kommentare»

5.4.2. Ausgangslage

Sobald sich der Fragenkatalog etwas füllt, kann es passieren, dass man sehr schnell den Überblick verliert, welche Fragen denn nun schon oft beantwortet wurden und welche eher selten. Ausserdem wäre es oft praktisch zu wissen, ob jemand etwas Neues zu einer eigenen Frage geschrieben hat. Durch das Einfügen von zwei neuen Spalten in die Fragenübersicht, soll beiden Problemen entgegengewirkt werden.

5.4.3. Lösung / Umsetzung

Ähnlich der Anleitung im Developer Manuel wurden zwei neue Column-Klassen angelegt und die benötigten SQL Ergänzungen herausgesucht. Zusätzlich wurden noch die Filteroptionen erweitert, so dass es nun auch möglich ist, zum Beispiel nach Fragen ohne Kommentare zu suchen.

Die erste Implementierung zu den Anzahl Durchführungen war jedoch fehlerhaft, da die Durchführungen von den Quiz gezählt wurden, anstatt die Durchführungen der einzelnen Fragen. Nach einer Anpassung des SQL Statements funktioniert nachher jedoch alles problemlos.

5.4.4. Reflektion

Das Hinzufügen von neuen Spalten und insbesondere das Filtern mit handgestrickten SQL Statements, gefällt dem Software Engineer in uns gar nicht so gut, aber wenn man es mit Legacy Code zu tun hat, hat man nicht immer die Wahl schön abstrakten Code zu schreiben.

5.5. Approved-Tag

5.5.1. Aufgabenstellung

Option pro Quiz, mit der neue Fragen zunächst auf Hidden gestellt und vom Teacher freigegeben werden müssen. Alternativ auch Markierung mittels «Approved-Tag», das dann auch im Filter wählbar sein sollte

5.5.2. Ausgangslage

Beim Einsatz des Quizzengers musste Prof. Koch feststellen, dass es meist sehr schwierig war die Übersicht zu behalten, welche Fragen nun bereits von ihm überprüft wurden und welche nicht. Dies unter anderem deshalb, weil es schlussendlich über hundert Fragen gab und jede Woche neue

Fragen erfasst wurden. Deshalb dann die Idee eines «Approved-Tag», welches für überprüfte Fragen gesetzt werden kann.

5.5.3. Lösung / Umsetzung

5.5.3.1. Taggingsystem

Da das StudentQuiz in vielen Fällen nur Operationen auf der Quizebene kennt, gab es vorab keinen direkten Weg ein Approved-Tag für jede Frage zu speichern. Eine Möglichkeit die zuerst überprüft wurde, wäre gewesen, dass man das Fragentaggingsystem verwendet und dort einfach einen speziellen Tag einführt, welcher dann eine Frage als bestätigt markiert. Doch da Studenten eigene Tags erstellen könnten und das Taggingsystem fix mit dem Moodle Quiz verankert ist, entschieden wir uns gegen eine solche Lösung.

5.5.3.2. Schlussendliche Lösung

Um nun als einer Frage einen Status verliehen werden konnte, musste eine neue Datenbanktabelle erstellt werden. Diese wird stetig mit den neu erstellten Fragen ergänzt, welche standardmässig auf unbestätigt gesetzt werden. In der Fragenübersicht des StudentQuiz kann nun eine Lehrperson oder jemand mit den vollen Bearbeitungsrechten einzelne Fragen mit einem Klick auf das Kreuz bestätigen oder mit dem Klick auf den Haken bestätigt Fragen wieder auf unbestätigt zurücksetzten. Alternativ kann auch die gesamte Auswahl von Fragen bestätigt werden, indem der Knopf unter- oder oberhalb der Fragenübersicht verwendet wird.

5.5.3.3. Frage verstecken

Nach einem Gespräch mit Prof. Mehta wurde festgelegt, dass das Verstecken von nicht bestätigen Fragen im Prinzip keinen Sinn macht. Die Studenten sollen sofort Zugriff auf die erstellten Fragen haben und nicht warten müssen, bis eine Lehrperson diese Frage bestätigt.

5.5.4. Reflektion

Zu Beginn war uns gar nicht klar, wie Schwierig die Umsetzung dieser Aufgabe sein könnte. Wir am Anfang wollten vor allem verhindern, eine neue Datenbanktabelle einzuführen, doch schlussendlich war dies die beste Lösung und sie scheint sehr gut zu funktionieren. Und wer weiss, ob die Tabelle in naher Zukunft nicht auch für weitere Assoziationen mit einzelnen Fragen verwendet werden kann.

5.6. Punktesystem

5.6.1. Aufgabenstellung

Verbesserung Punktesystem

a. Eigene Fragen sollen nicht bewertet werden können
b. Für das Beantworten eigener Fragen werden keine Punkte vergeben
c. Für das mehrfache Beantworten einer Frage werden nur beim ersten Mal Punkte erzielt
d. Für eine halbrichtige Antwort werden keine Punkte vergeben

5.6.2. Ausgangslage

Das Punktesystem war bis anhin noch etwas dürftig und einfach gestaltet. Erweiterungen waren schwierig umzusetzen. Es gab zudem noch einzelne Fehle. Insbesondere wenn ein Versuch noch nicht beendet wurde. Ein anderer erwähnenswerter Fehler war, dass bei speziellen Fragetypen zu viele Punkte vergeben wurden. (Multiple-Choice-Fragen mit Sub-Questions).

5.6.3. Lösung / Umsetzung

Die Analyse des Problems war schwieriger als die Lösung selber. Insbesondere aus dem Grund dass die Abfrage der Ranking-Daten relativ kompliziert ist. Weiter wurden zusätzliche Berechnungsfehler relativ spät entdeckt. Am wichtigsten ist die Funktion get_user_ranking(), in der das SQL-Statement für die Ranking-Daten liegt. Dieses Statement musste einigen Anpassungen unterzogen werden.

5.6.4. Reflektion

Das Ranking ist so wie es ist recht gut gelöst. Möglicherweise könnte es besser beschriftet werden oder mit neuen Werten erweitert werden.

5.7. Testing / Upgrade Moodle 3.2

5.7.1. Aufgabenstellung

Testing / Upgrade Moodle 3.2

a) PHPUnit-Tests erstellen und ausbauen
b) Acceptance-Tests

5.7.2. Ausgangslage

Wir wussten vor dem Start der Studienarbeit, dass irgendwann während der Dauer der Arbeit die neue Version 3.2 von Moodle veröffentlich werden wird. Eine ähnliche Situation hatte die Gruppe vor uns mit dem Update von Moodle 3.0 auf 3.1. Aus der Erfahrung heraus, dass ein Update in gewissen Aspekten des Plugins Probleme hervorrufen könnte, entschieden wir uns die Testsuiten des Plugins zu erweitern.

5.7.3. Lösung / Umsetzung

5.7.3.1. Update

Schon sehr früh versuchten wir die damals noch in Entwicklung stehende 3.2 Version und waren beim ersten Versuch sehr positiv überrascht. StudentQuiz funktionierte auf den ersten Blick einwandfrei und auch unsere Tests zeigten keinerlei Abweichungen.

5.7.3.2. Visuelles

Die einzigen Anpassungen welche für das Update nötig waren, sind visueller Natur. Da das StudentQuiz leider gewissen HTML Tags fest im Code verankert hat, gab es beim Update ein paar visuelle Unstimmigkeiten, welche aber ohne weiter schnell angepasst werden konnten.

5.7.3.3. Unit Tests

Für wichtige Funktionen wurden die vorhandenen Unit-Tests erweitert. Wichtig war dabei, dass durch viele Abhängigkeiten vieles nicht getestet werden kann. Der Aufwand wäre hier unverhältnismässig. Auch Mocking (Simulation von Komponenten) ist nur schwer möglich. Wegen der Moodle Plugin-Validierung ist es nicht möglich das Test-Framework zu erweitern. Somit wurde das Testing auf die wichtigsten Funktionen beschränkt.

5.7.3.4. Acceptance Tests

Da Unit Tests nur gewisse Teile der Pluginfunktionen überprüfen können, definierten wir Acceptance Tests. Diese Integrationstests prüfen nicht die einzelnen Features auf ihre Funktionalität hin, sondern das System mit allen Funktionalitäten als Ganzes. Am effizientesten geht das mithilfe unserer Use Cases. Dabei wurden die Informationen der jeweiligen Use Cases zu Test Cases umgemünzt. Es wurde je eine Testanleitung mit wichtigen Prüfpunkten für die Rolle des Teacher und die Rolle des

Studenten erarbeitet und definiert.

5.7.3.5. Boost Theme

Wie in 3.1 erwähnt ist eine der markantesten Erneuerungen das neue Boost Theme. Obwohl wir frühe Versionen von Moodle 3.2 getestet hatten, war uns nicht bewusst, dass es für Moodle 3.2 eine Änderung im Standardtheme geben würde, sodass wir dann erst gegen Ende unserer Arbeit feststellen mussten, dass das Boost Theme mit einem anderen Gedanken an die Navigation herangeht. Für die effektive Nutzung von StudentQuiz mit dem Boost Theme empfehlen wir den Navigationsblock einzublenden, damit man trotzdem noch Zugriff auf die Statistiken und das Ranking bekommt.

5.7.4. Reflektion

Die Umstellung auf das Boost Theme war für uns eine Überraschung, denn wir hatten ja frühe Versionen von der neuen Version getestet, aber da hatte es noch keinen Umstieg gegeben.

Die Codestruktur von Moodle ist oft nicht sehr durchsichtig mit vielen freien Funktionen in den Dateien der Library. Auch StudentQuiz wurde nach dieser Art geschrieben. Durch die fehlenden Abstraktionen war es somit sehr schwierig Unit Tests zu schreiben.

5.8. Filtering Reengineering

5.8.1. Aufgabenstellung

Reengineering / Refactoring: Vor allem beim Filtering

a) Code ist schlecht erweiterbar
b) Evtl. auf Button «Run Selected Questions» verzichten
c) Dynamisches Filtern mit JavaScript (Anpassung Frage Buttons) Siehe SA S 27

5.8.2. Ausgangslage

Die Filterfunktion der Fragenübersicht gehört zu einem der zentralen Features vom StudentQuiz. Mit der Annahme, dass eine Activity es mit einem grossen Fragenkatalog zu tun haben wird, bieten die Filter verschiedenste Optionen um spezifische Fragen heraus zu suchen. In der nahen Zukunft, kann es durchaus sein, dass weiter Filteroptionen benötigt werden, dies jedoch in den vorhandenen Code zu integrieren, wird wegen der Komplexität doch recht zeitaufwändig. Ein Refactoring oder gar Reengineering soll hier Abhilfe schaffen.

5.8.3. Lösung / Umsetzung

Nach längerem Versuch den Filtering Code irgendwie umzustellen oder in einem grösseren Ausmass zu verbessern, musste festgestellt werden, dass sich die Implementierung stark an die offiziellen Filterfunktionen lehnen. Der Code scheint unnötig komplex für jemand der neu an das Projekt sitzt und von Moodle relative wenig versteht, aber beim genauen Studieren der Klassen und Funktionen in Verbindung mit der Moodle Dokumentation wird vieles verständlicher. Da sich die Funktionen stark an Moodle selbst orientiert hatten und gewisse Strukturen für das Filtern benötigt werden, wurde davon angesehen ein Reengineering durchzuführen und an dessen Stelle wurden kleinere Refactorings getätigt. Um es zukünftigen Entwicklern jedoch einfacher zu machen, wurde ein neues Manual erstellt mit detaillierteren Informationen zum Filtern.

Eine visuelle Änderung, welche auch Auswirkungen auf die Bedienung hat, war das Entfernen der beiden Knöpfe "Run selected questions" und "Run filtered questions". Stattdessen wurde ein "Start

Quiz" Knopf ober und unterhalb des Fragenkatalogs angebracht. Es werden nun immer nur die ausgewählten Fragen ausgeführt, aber da standardmässig alle Fragen auf der aktuellen Seite ausgewählt werden, entspricht die Standardaktion im Prinzip dem Ausführen von gefilterten Fragen. Falls eine Frage nicht ins Quiz aufgenommen werden sollte, kann diese einfach abgewählt werden.

5.8.4. Reflektion

Eine gewisse Enttäuscht ist schon zu spüren, dass wir den Filtering Code nicht in einem grossen Ausmass verändern konnten. Denn obwohl man mit etwas investierter Zeit sich relative schnell im Code zurechtfindet, so muss man für eine Erweiterung doch immer verschiedene Codestellen ändern und es kann schnell passieren, dass ein Teil vergessen geht und es zu Problemen kommt.

5.9. Bugfixes

5.9.1. Aufgabenstellung

a) Link auf Quiz von Teacher funktioniert nicht auf Moodle 3.0 (funktioniert das in Moodle 3.1?)
b) Bewerten einer Frage kann unter Moodle 3.1 umgangen werden
c) Backup and Restore funktionieren nicht für Studentquiz
d) Nach Abschluss des Quiz Review („Summary of your previous attempts“) wird nur die letzte Teilnahme Quiz angezeigt

5.9.2. Ausgangslage

Wie jede Software enthielt auch StudentQuiz nach der Abgabe der vorangegangenen Arbeit noch gewisse Bugs. Auch während der Einarbeitungszeit fielen uns hier und dort noch ein paar Kleinigkeiten auf. Diese Bugs galt es somit nun zu Beginn zu beheben.

5.9.3. Lösung / Umsetzung

5.9.3.1. Teacher Quiz Link

Dieser Bug schien zuerst etwas unerklärlich, da gewisse Links zu Quiz funktionierten, während andere eine Fehlermeldung verursachten. Doch beim genaueren Analysieren wurde festgestellt, dass dem Link eine falsche ID gegeben wurde und ob nun ein Link funktionierte oder nicht, hing davon ab, ob in der Datenbanktabelle an einem ganz anderen Ort die ID existierte oder nicht. Anstatt der Quiz ID musste somit einfach die Course Module ID verwendet werden.

5.9.3.2. Umgehen der Fragenbewertung

Die Aufforderung, dass eine Frage beantwortet werden muss, wird alleinig auf der Clientseite mittels JavaScript umgesetzt. Da sich nun ein paar Kleinigkeiten zwischen den Moodle Versionen im HTML Code geändert hatten, gab der JavaScript Code plötzlich immer an, dass die Bewertung bereits getätigt wurde. Durch das Anpassen des JavaScript Code konnte das Problem gelöst werden.

5.9.3.3. Backup und Restore von StudentQuiz

Wenn zum Beispiel von einem ganzen Kurs, welcher eine StudentQuiz Activity enthält, gemacht wurde, kam es beim Restoren des Backups zu einem Fehler, was dann der Import in einem fehlerhaften Zustand liess. Der Grund war, dass der einen Datenbanktabelle eine neue Spalte hinzugefügt wurde, welche aber im Backup nicht berücksichtigt wurde und beim Restoren kam es dann zu Konflikten mit der Datenbank. Da es sich bei diesen Werten jedoch um Course Module IDs handeln, welche sich je nach Kurs ändern, musste eine Lösung implementiert werden, welche beim Import diese Course Module IDs auf die richtige Zahl setzt.

5.9.3.4. Nur die letzte Quiz Durchführung

Während der Einarbeitungszeit wurde festgestellt, dass am Ende eines Quiz es jeweils immer nur eine Durchführung angezeigt wurde. Dies hatte uns etwas verwirrt und stellten nach etwas nachforschen fest, dass für jede Auswahl von Fragen ein komplett neues Quiz in der Datenbank angelegt wurde. Das heisst also auch, dass wenn man dieselben Fragen zum Beispiel zehn Mal durchläuft, man am Ende zehn unabhängige Quiz erstellt hat.

Lösung

Um diesem Verhalten entgegenzuwirken wurde eine Funktion eingebaut, welche anhand von einer Liste von Fragen IDs feststellen kann, ob ein Quiz mit den gleichen Fragen bereits existiert und falls dies so ist, wird dieses Quiz verwendet, ansonsten wird ein neues erstellt. Somit verkleinert sich die totale Anzahl der erstellten Quiz und es werden mehr Durchführungen nach dem Quiz angezeigt.

5.9.3.5. Hardcoded Datenbankpräfixe

Einer der ersten Bugs welche wir entdeckten war, dass das Plugin nicht zu funktionieren schien, wenn man die Datenbanktabellen ohne mdl_ Präfix erstellt hatte. Grund dafür war, dass in einem Codeteil festgeschriebene Präfixe verwendet wurden, anstatt die dafür vorgesehenen geschweiften Klammern. Nach einer kurzen Anpassung lief dann auch alles problemlos.

5.9.4. Reflektion

Die Lösungen zu den einzelnen Bugs waren in den meisten Fällen ziemlich trivial, stattdessen wurde viel mehr Zeit benötigt um das grundlegende Problem erst einmal aufzuspüren. Moodle war da in vielen Fällen sehr hilfreich, in dem man genaue Fehlermeldungen und Stacktraces erhielt. Nur manchmal fehlten noch die aktuellen Zustände der Variablen, welche dann aber mit einfachen echo Statements oder einem Dateilogging umgangen werden konnte.

5.10. Notification

5.10.1. Aufgabenstellung

Notification an die Autoren, wenn Dozierende Fragen verändern oder löschen / Ein „Last edited by“-Entry

5.10.2. Ausgangslage

Beim Einsatz von Quizzenger wurde festgestellt, dass es des Öfteren Fragen gibt, welche Schreibfehler enthalten oder falsche Informationen weitergeben. Eine Lehrperson hat nun die Möglichkeit fehlerhafte Fragen zu bearbeiten, leider wird dabei dann aber dem Studenten die Änderungen nicht mitgeteilt. Somit kann es durchaus passieren, dass sich ein Student falsche Informationen merkt, da dieser nie erfahren hat, dass die Frage korrigiert wurde.

5.10.3. Lösung / Umsetzung

5.10.3.1. Änderungsbenachrichtigung

Es soll nun eine Notification E-Mail versendet werden, falls eine Lehrperson Änderungen an der Frage eines Studenten vornimmt. Da das StudentQuiz auf dem Moodle Quiz aufbaut und die Fragen nicht selbst verwaltet, können wir Änderungen an Fragen nicht direkt feststellen, sondern können nur bemerken, falls eine Aktion zum Bearbeiten ausgeführt wurde. Eine Änderungsbenachrichtigung zu einer Frage wird somit nur dann verschickt werden, wenn die Bearbeitung über den Fragenkatalog des StudentQuiz ausgeführt wird und wenn die folgenden Bedingungen erfüllt sind:

- Die Person welche die Frage soeben bearbeitet hat, hat die nötigen Rechte um alle Fragen zu bearbeiten, d.h. sie ist eine Lehrperson.
- Die bearbeitete Frage gehört nicht der Person welche die Frage soeben bearbeitet hat.
- Die Änderungen sind nicht älter als fünf Sekunden.

Der letzte Punkt soll insbesondere verhindern, dass keine unbeabsichtigten E-Mails versendet werden.

5.10.3.2. Bestätigungsbenachrichtigung

Nachdem die Änderungsbenachrichtigung eingebaut war und diese problemlos funktionierte, kam noch der Wunsch auf, ebenfalls einen Notification zu verschicken, wenn eine Lehrperson eine Frage bestätigt. Da sich die beiden Aktionen an verschiedenen Stellen im Code implementiert werden, musste die Benachrichtigungsfunktion in die allgemeinere lib.php Datei verschoben werden, was aber ein kleiner Aufwand war. Wenn eine Lehrperson nun also eine Frage bestätigt oder die Frage auf unbestätigt setzt, dann wird der Student eine E-Mail bekommen, welche ihn auf dieses Ereignis hinweist.

5.10.3.3. "Last edited by"-Entry

Der Fragenkatalog ist mit vierzehn Spalten bereits sehr gefüllt, weshalb wir uns entschieden haben keine weitere Spalte mit einem "Last edited by" Eintrag zu erstellen. Damit der Student jedoch weiss, wer dann jetzt die Frage bearbeitet hatte, wird im E-Mail der Name der bearbeitenden Person angegeben.

5.10.4. Reflektion

Eine Problematik mit Notification E-Mails ist, dass es gut vorkommen kann, dass zu viele E-Mails versendet werden, so dass es nur noch lästig wird. In den Profileinstellungen können die einzelnen Benachrichtigungen deaktiviert werden, oder der Administrator kann diese sogar auf der gesamten Moodle-Plattform unterdrücken.

5.11. Activity Completion

5.11.1. Aufgabenstellung

Das StudentQuiz implementiert die Activity Completion noch nicht. Dies wird von vielen Lehrern angewendet.

5.11.2. Ausgangslage

Der Wert des Grades (Eine Art Benotung) muss hierfür gesetzt werden, was relativ kompliziert ist. Da es nach jeder Durchführung des Quiz diese Note neu berechnet werden muss. Dabei ist das Problem noch, dass das Quiz im Quiz-Plugin läuft. Hierfür bietet sich aber an, die Berechnung am Ende des Quiz im QuestionBehaviour-Plugin des StudentQuiz zu machen.

Bei der Analyse wurde noch entdeckt, dass die von StudentQuiz erstellte Instanz (ein Quiz) die Grade-Einstellung dabeihat, was das Fluten des Grade-Books besser erklärt. Für das Quiz-Plugin lässt sich der Grade-Eintrag nicht abschalten.

Ein Abgleich des Grades bei jedem Aufruf des StudentQuiz würde erheblich viel Leistung brauchen und ist somit nicht ideal. Ausserdem würde das Grade erst dann berechnet werden und wäre somit an anderen Orten nicht aktuell. Weiter wäre eine Zwischentabelle möglich, was den Rahmen dieser Arbeit sprengen würde und einiges an Komplexität einbringen würde.

5.11.3. Lösung / Umsetzung

Das Grade wird nun beim Aufruf des Grade bei Review speichern.

5.11.4. Reflektion

Idee für Reengineering: Für StudentQuiz die Quiz-Durchführung selbst anhand der Question-Engine implementieren. Das StudentQuiz würde somit möglich einiges an Flexibilität gewinnen. Die Implementation würde dann im Endeffekt dem Quiz-Moduls sehr ähnlich sein. Ein Reengineering kommt deshalb in Frage, da eine bessere Flexibilität vom Quiz-Modul nicht erwartet werden kann. https://docs.moodle.org/dev/Using_the_question_engine_from_module

Problem: Gradeberechnung

Es gibt dabei noch ein Problem, das auftritt falls das Modul Questionbehaviour nicht installiert ist, oder die Bewertungen abgeschaltet sind. Dann wird die Berechnung nicht ausgelöst. Es wäre möglich dies zu umgehen, indem das StudentQuiz-Questionbehaviour für alle Quiz-Instanzen ausgeführt wird und dort (im Behaviour) abgefragt wird ob auf dem StudentQuiz entsprechende Option zum Aktivieren der Bewertung/Bemerkung eingestellt ist. All dies aus dem Grund, dass das Questionbehaviour in die Quiz-Instanz bei der Durchführung übernommen wird.

Versteckte Grade-Kategorie

Es gäbe die Möglichkeit eine Kategorie für die Grades der Quiz-Instanzen zu erstellen. Diese würde so unsichtbar wie möglich eingestellt. Fragt sich ob die Kategorie auf die Quiz-Instanz gespeichert werden kann.

5.12. Orphaned Activities

5.12.1. Aufgabenstellung

Section 999 (Verwaiste Aktivitäten (Abschnitt 1000))

StudentQuiz-Durchführungen erscheinen in der Modul-Übersicht in einer verwaisten Sektion. Diese sind für normale Studenten nicht sichtbar. Aber für Dozenten, die das Modul gerade bearbeiten sind diese sichtbar.

5.12.2. Ausgangslage

Das grundlegende Problem liegt daran, dass für alle Durchführungen des StudentQuiz eine eigene Sektion im Kurs erstellt wird, die generell nicht sichtbar ist. Wird diese nicht erstellt, kann das Quiz nicht angezeigt werden aus nachfolgenden Gründen.

Beim Aufruf auf das Quiz werden die Rechte des Studenten geprüft. Dabei muss das Quiz mit entsprechender Course-Module ID in einer Sektion dieses Modules erscheinen. Dass pro StudentQuiz nur ein (Moodle-)Quiz erstellt wird (Statt für jede Art der Durchführung), wäre sehr wünschenswert, doch technisch momentan nicht machbar.

Diese Einträge entstehen somit durch das Quiz-Plugin.

5.12.3. Lösungsfindung

Es wurde keine Programmatische Lösung hierfür gefunden, da dies zu viel Zeit gekostet hätte. Es wurden mehrere Ansätze überprüft.

Verstecken durch CSS: Nicht möglich, da kein direkter Zugriff auf die Seite möglich ist.

Kein Eintrag in die Sektion: Dies hat wie erwähnt zur Folge, dass die Rechte des Users durch das Moodle-Quiz nicht überprüft werden können.

5.12.4. Reflektion

Das Problem ist momentan nicht lösbar. Es gäbe mehrere unwahrscheinliche Möglichkeiten, die kurz erläutert werden:

1. Die Durchführung der StudentQuiz müsste durch das Plugin selber durchgeführt werden. Also so wie das Quiz-Plugin selber.
2. Das Quiz-Plugin erhält mehr Möglichkeiten der Übersteuerung.
3. Eine Sektion liesse sich definitiv unsichtbar machen.

5.13. Navigation

5.13.1. Aufgabenstellung

Das Menü des StudentQuiz ist etwas überladen und sollte angepasst werden. Einzelne Punkte sollen umbenannt werden.

Navigations-Problem: Bei Beendung einer StudentQuiz-Durchführung, wird nicht mehr auf die StudentQuiz-Seite zurückgeführt.

5.13.2. Ausgangslage

Das Menü ist relativ einfach anpassbar, die Anpassungen müssen aber manuell im Code geschehen. Hierzu steht die Methode studentquiz_extend_navigation in der Datei lib.php des StudentQuiz-Plugins zur Verfügung.

Da die Durchführung des StudentQuiz durch das Quiz-Modul von Moodle durchgeführt wird, gelangt die Navigation nach dem Quiz nicht zurück zum StudentQuiz, wo das Quiz gestartet wurde.

5.13.3. Lösung / Umsetzung

Das Menü wurde wie folgt angepasst:

- Eintrag „Quiz and Questions“ wurde entfernt, da er nicht nötig ist. Gleicher Link, wie beim Quiz-Name.
- Statistics wurde eingeführt
- Diverse Umbenennungen

Das Navigationsproblem konnte nicht gelöst werden, da die Quiz-Durchführung durch das Quiz-Plugin gehandhabt wird. Dieses Plugin ist auch nicht so flexibel, dass die Navigation angepasst werden kann.

5.13.4. Reflektion

Um das Navigationsproblem zu lösen erfordert es grössere Anpassungen. Auch hier würde sich eine StudentQuiz-Interne Durchführung des Quiz empfehlen.

5.14. Übersetzungen / Sprache

5.14.1. Aufgabenstellung

Das StudentQuiz soll auch auf Deutsch und Französisch verfügbar sein.

5.14.2. Ausgangslage

Bisher wurde das StudentQuiz nur auf Englisch angeboten. Die Strings werden aber alle in einer Sprachdatei gehalten (lang/en/studentquiz.php).

5.14.3. Lösung / Umsetzung

Für Deutsch und Französisch wurde jeweils ein Sprachfile erstellt. Dort wurden alle Strings übersetzt. Neue Strings wurden ebenfalls übersetzt.

5.14.4. Reflektion

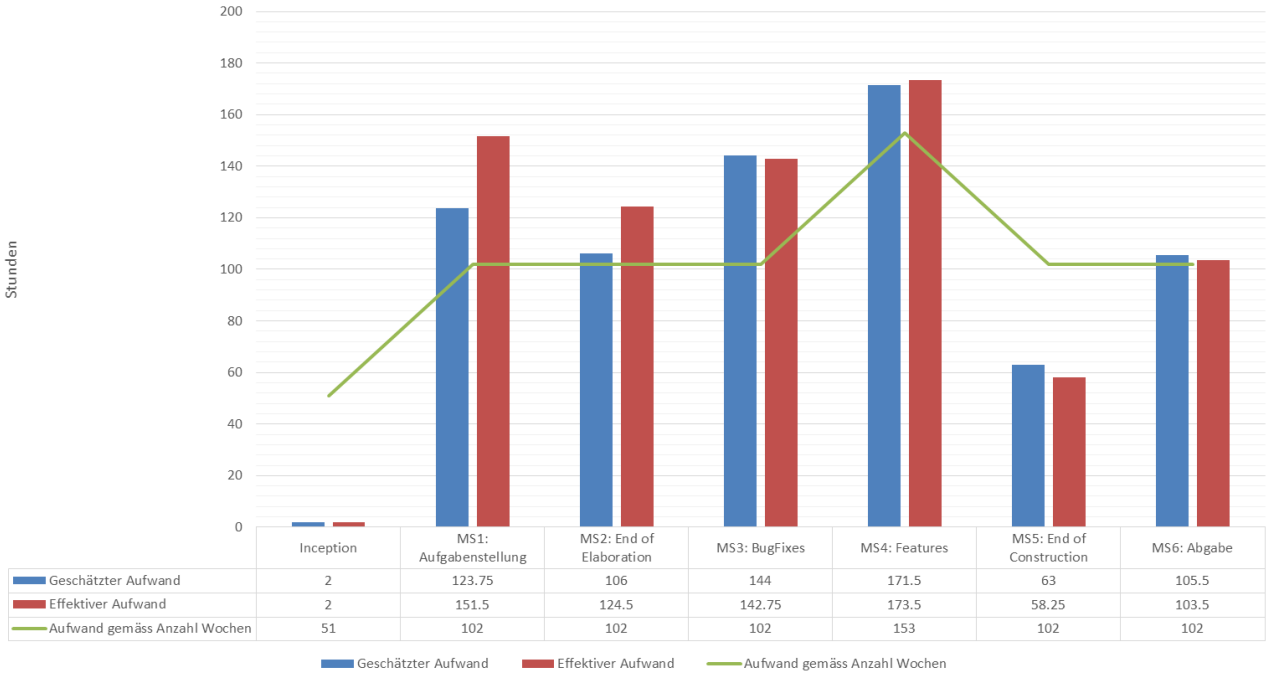
Die Übersetzung des Plugins ging recht gut, da die allermeisten Strings in der Richtigen Datei abgelegt wurden.

Es wäre zusätzlich noch nötig, nicht mehr benutzte Strings zu entfernen, da diese die Dateien überladen könnten. Ansonsten ist darauf zu achten, dass bei neuen Strings immer jeweils die Übersetzung dazu genommen wird.

6. Projektauswertung

6.1. Zeitauswertung

6.1.1. Aufwandschätzung zu Aufwand-Ist



	Inception	MS1: Aufgabenstellung	MS2: End of Elaboration	MS3: BugFixes	MS4: Features	MS5: End of Construction	MS6: Abgabe
Geschätzter Aufwand	2	123.75	106	144	171.5	63	105.5
Effektiver Aufwand	2	151.5	124.5	142.75	173.5	58.25	103.5
Aufwand gemäss Anzahl Wochen	51	102	102	102	153	102	102

Abbildung 5: Aufwandschätzung

In diesem Diagramm sieht man sehr gut wie der geschätzte Aufwand (blau) mit dem effektiven Aufwand (rot) einhergeht. Der Aufwand war schlussendlich meistens ein bisschen höher als geschätzt. Man kann gut erkennen, dass die Schätzung am Beginn des Projekts einiges zu tief ist, sprich das Einrichten der Arbeitsumgebungen und vor allem das Einarbeiten in Moodle und die vorhergegangene Arbeit aufwändiger war als gedacht. Im Verlaufe des Projekts gleichen sich die zwei Zeitbalken jedoch immer weiter an, bis schlussendlich die Schätzung ein klein wenig über dem effektiven Aufwand war. Daraus ist ersichtlich, dass wir nicht nur besser im Entwickeln wurden, sondern auch im Schätzen des Aufwands.

Der kleine Aufwand in der Inception-Phase ist einerseits der Unwichtigkeit dieser Phase in unserer spezifischen Arbeit zuzuschreiben, als auch der Unsicherheit einzelner Entwickler, ob die Studienarbeit angetreten werden kann. Zum Glück fingen wir doch trotz Unsicherheit mit dem Einarbeiten an und konnten diese Arbeit zum Erfolg führen.

Die grüne Linie zeigt sehr schön den theoretischen Aufwand pro Meilenstein an. Dieser wird anhand der Anzahl Wochen in einem Meilenstein errechnet und zeigt die Höhe des Aufwands an, wenn man jede Woche gleich viel in die Arbeit investieren könnte.

6.1.2. Aufwand im Verlauf der SA

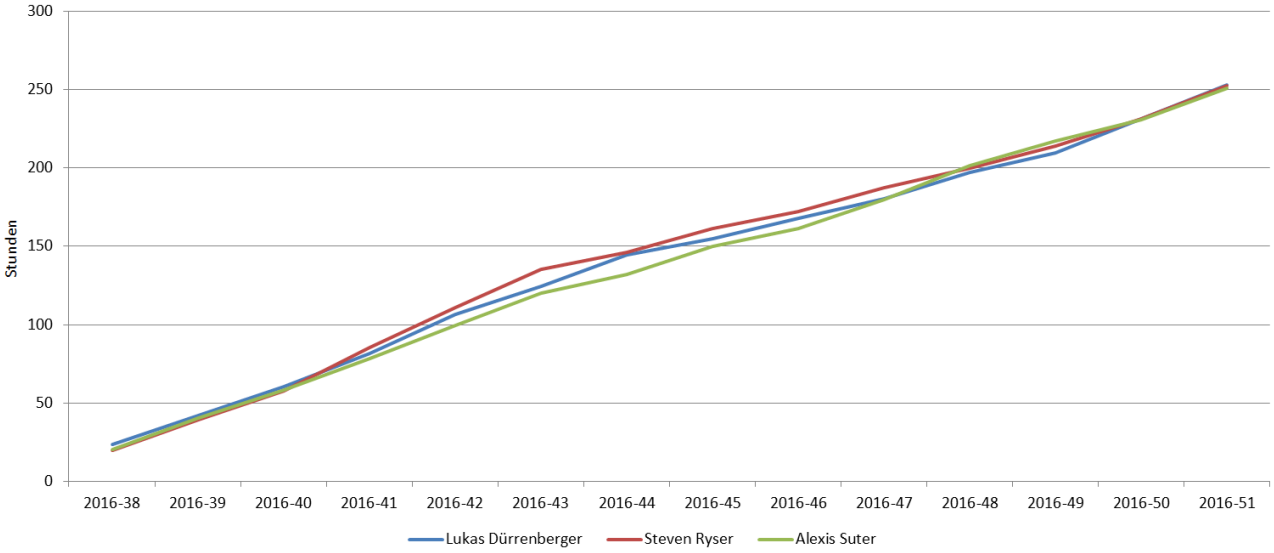


Abbildung 6: Aufwand

In diesem Diagramm erkennt man gut wie gross der Aufwand der einzelnen Entwickler im Verlauf der Studienarbeit war. Es zeigt sich, dass manchmal der Eine etwas mehr zu tun hatte und ein andermal jemand anderes. Schlussendlich zeigt sich, dass alle Entwickler etwa gleich viel zum Erfolg des Projekts beigetragen haben. Im Allgemeinen können wir stolz auf diese ziemlich gleichmässige Verteilung sein.

6.1.3. Ist-Aufwand jedes Entwicklers

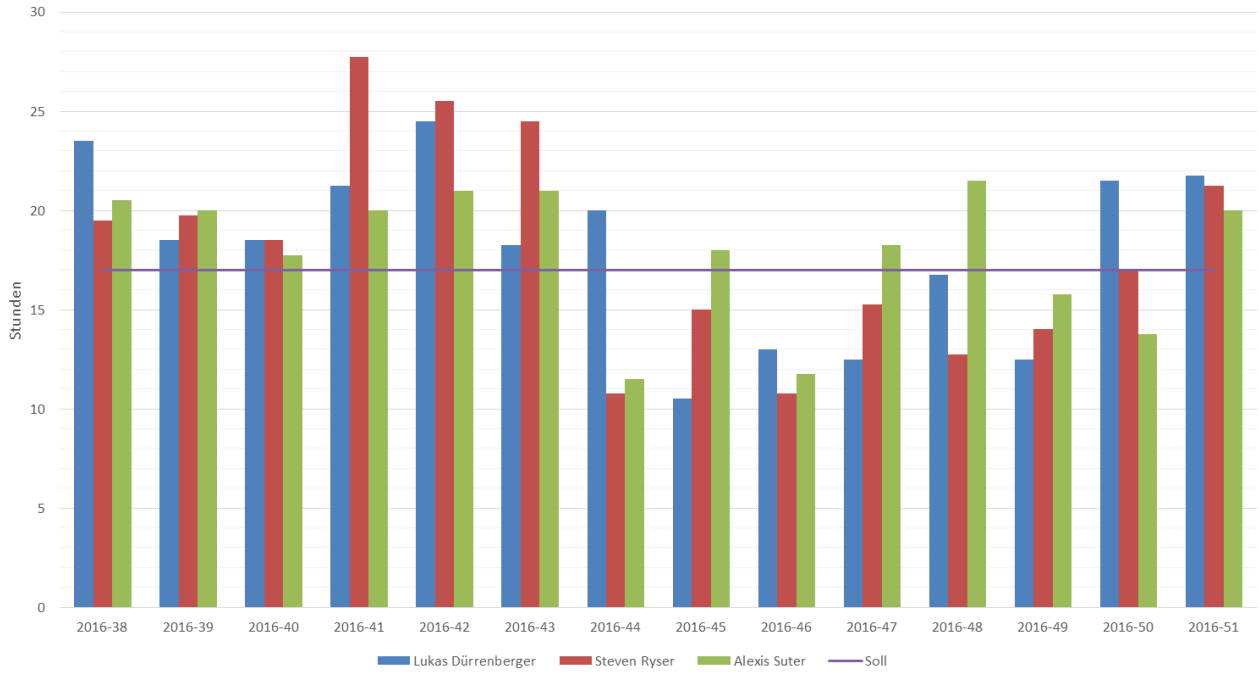


Abbildung 7: Ist-Aufwand

Dieses Diagramm zeigt anschaulich die Verteilung des individuellen Aufwands über die 14 Wochen des Projekts. Man erkennt an der Fluktuation der jeweiligen Wochen, wie unterschiedlich diese Wochen waren und wann wir mehr oder weniger am Projekt arbeiten konnten.

Die violette Linie zeigt sehr schön den theoretischen Aufwand pro Woche an. Dieser wird berechnet, indem der theoretische Gesamtaufwand durch die Anzahl Wochen des Projekts geteilt wird und zeigt an, wieviel Aufwand man pro Woche investieren müsste, wenn man jede Woche gleich viel in die Arbeit investieren wollte.

6.1.4. Aufwandschätzung jedes Meilensteins

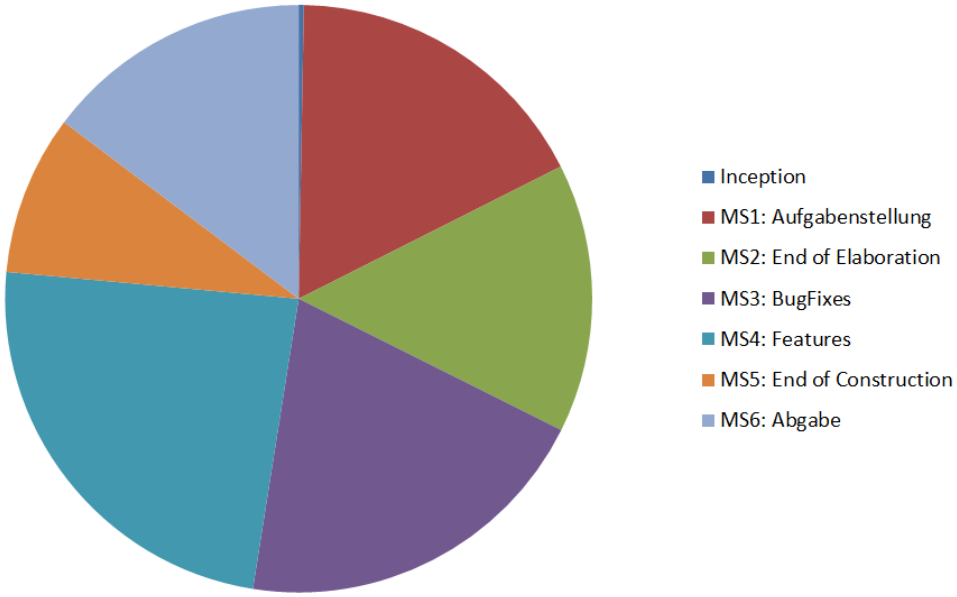


Abbildung 8: Aufwandschätzung nach Meilensteine

Hier sieht man die Verteilung des geschätzten Aufwands pro Meilenstein am Gesamtaufwand. Zum Beispiel wurde für die Construction-Phase (MS3, MS4 und MS5) etwas mehr als die Hälfte der Zeit eingeplant.

6.1.5. Effektiver Aufwand pro Meilenstein

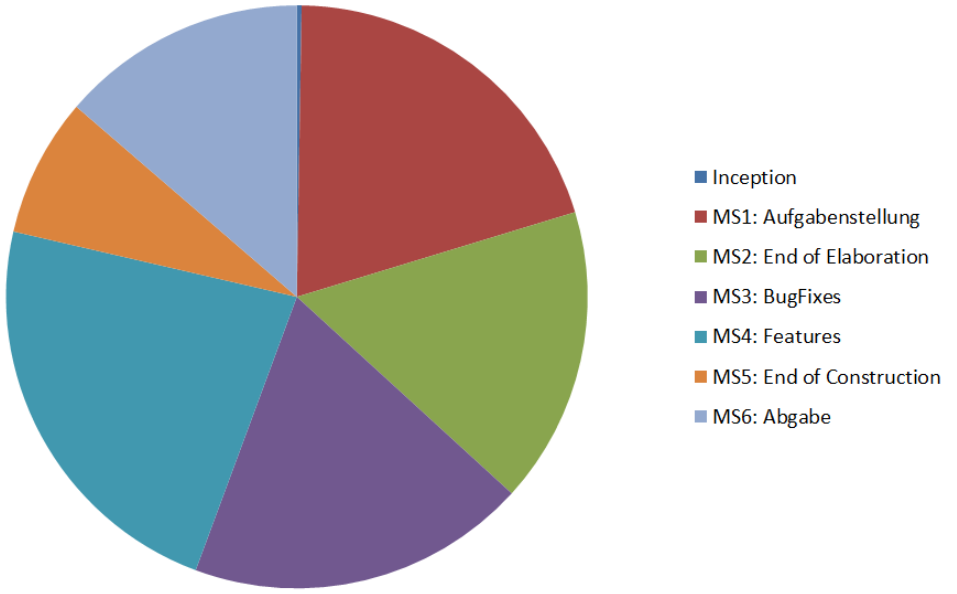


Abbildung 9: Effektiver Aufwand nach Meilensteine

Hier sieht man die Verteilung des effektiven Aufwands pro Meilenstein am Gesamtaufwand. Es sind ein paar Grössenveränderungen erkennbar. Aber im Vergleich zur Schätzung ist es nicht schlecht.

6.2. Codestatistik

Die Metriken in diesem Kapitel wurden alle mithilfe des MetricsReloaded Plugins für IntelliJ erfasst.

6.2.1. Moodle Code Style Check

Um die Moodle Plugin Validierung zu bestehen, muss der Code den Moodle Style Guidelines genügen. Dementsprechend haben wir unseren Code gemäss diesen Guidelines ausgerichtet. Getestet wurden primär die PHPDocs und der Quellcode.

6.2.2. Lines of Code Metrik

Vorhergehendes Projekt:

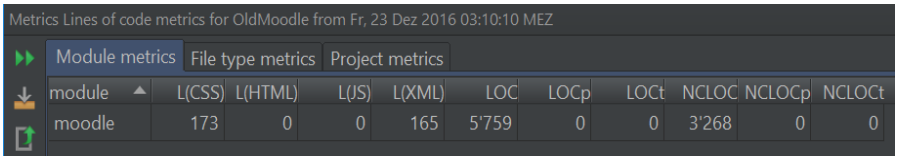


Abbildung 10: Lines of Code (LOC) des vorhergehenden Projekts

Nach unserer Studienarbeit:

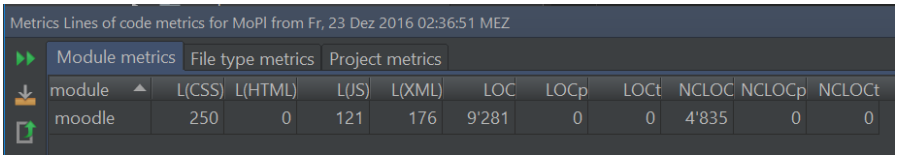
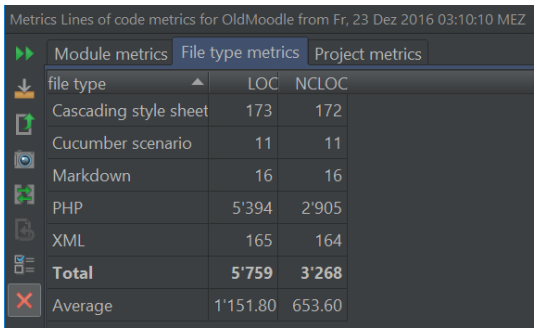


Abbildung 11: Lines of Code nach unserer Studienarbeit

Anhand dieser Codestatistik ist gut ersichtlich, dass einige neue Features und zahlreiche Verbesserungen zum StudentQuiz hinzugekommen sind. Wir haben über 3500 neue Zeilen Code hinzugefügt. Die totale Grösse (Total Lines of Code) umfasst 9281 Codezeilen.

6.2.3. File type Metrik

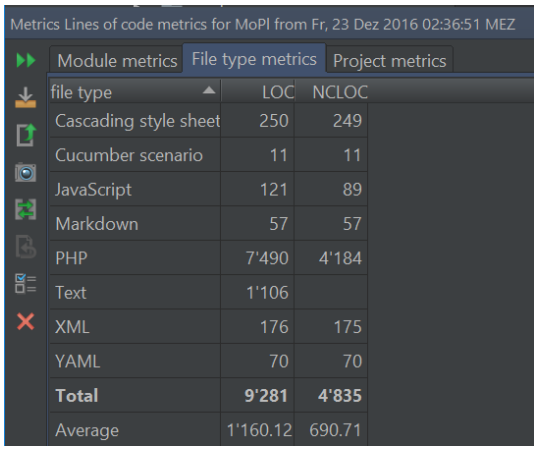
Vorhergehendes Projekt:



file type	LOC	NCLOC
Cascading style sheet	173	172
Cucumber scenario	11	11
Markdown	16	16
PHP	5'394	2'905
XML	165	164
Total	**5'759**	**3'268**
Average	1'151.80	653.60

Abbildung 12: Lines of Code des vorhergehenden Projekts nach Dateitypen

Nach unserer Studienarbeit:



file type	LOC	NCLOC
Cascading style sheet	250	249
Cucumber scenario	11	11
JavaScript	121	89
Markdown	57	57
PHP	7'490	4'184
Text	1'106	
XML	176	175
YAML	70	70
Total	**9'281**	**4'835**
Average	1'160.12	690.71

Abbildung 13: Lines of Code nach unserer Studienarbeit

Die durchschnittliche Codezeilenanzahl der „File type“ liegt bei 1160. Der weitaus grösste Teil macht hier der PHP Teil aus, was auch zu erwarten war. Mit 7490 Codezeilen aus dem Gesamten ergibt einen PHP Anteil von 80%.

7. Resultat und Ausblick

7.1. Zielerreichung

Wir haben uns am Anfang der Studienarbeit zusammen verschiedene Ziele gesetzt. Nun am Ende dieser Arbeit ist es schön zu sehen, dass die alle, als wichtig definierten, Verbesserungen und Erweiterungen realisiert und vervollständigt werden konnten. Das StudentQuiz ist mit allen neuen Features funktionstüchtig und bestand alle unsere Tests bisher. Es können Fragen erstellt, beantwortet, bewertet und kommentiert werden. Die Quiz zu beantworten macht Spass. Es wird in Zukunft ein gutes Tool sein, um den Studenten zu ermöglichen, sich aktiv im Kurs einzubringen.

Leider geht die offizielle Moodle Validierung der Plugins länger als angenommen und konnte so nicht mehr in im Zeitrahmen unserer Studienarbeit ins Moodle Plugin Directory aufgenommen werden. Der Prozess ist aber im Gange und wir haben alle gewünschten Anpassungen der Validierungsmitglieder bereits umgesetzt. Bis zur bestandenen Durchführung dauert es also nur noch ein bisschen. Somit kann auch dieser Punkt von unserer Seite her als Erfüllt betrachtet werden.

Aus Sicht der Priorisierung unserer Aufgaben haben wir alle Aufgaben mit Prioritätsstufe A realisiert und sogar einige Punkte der Prioritätsstufe B entwickelt und integriert. Zusätzlich haben wir das StudentQuiz Inhaltlich und Grammatikalisch auf Vordermann gebracht und der Spracheinstellung zwei zusätzliche Sprachen hinzugefügt, um den Wirkungsbereich des Plugins zu erhöhen und eine noch simplere Benutzung des StudentQuiz zu ermöglichen.

7.2. Ausblick / Erweiterungsmöglichkeiten

Während dieser Arbeit wurden einige Punkte entdeckt, die in Zukunft in Betracht gezogen werden können. Diese werden im Folgenden genauer erläutert.

7.2.1. Dashboard

Mögliche Anpassung, wäre eine klarere und verständlichere Übersicht. (Usability) Auch eine Erklärung der Werte wäre für User sehr nützlich.

7.2.2. Punktesystem

Das Ranking könnte ebenfalls eine Erklärung für die Werte beinhalten. (Usability) Eventuell könnte es mit zusätzlichen Spalten bereichert werden.

7.2.3. Eigene Quiz-Durchführung

Die Idee wäre eine eigene Quiz-Durchführung im StudentQuiz zu implementieren. Dies ist ein etwas kritischer Punkt, denn der Aufwand kann recht gross werden. Dabei kann aber einiges vom Question-Plugin aus Moodle benutzt werden. Der Vorteil daran ist, dass das Handling der Resultate besser implementiert werden kann und einige Fehler ausgeräumt werden können. Beispielsweise kann dadurch die Flutung der Orphanned Activities und der Grades verhindert werden.

7.2.4. Notifications

Momentan wird für jedes Un-/Approval oder nach jeder Bearbeitung einer Frage direkt ein E-Mail versendet, was teilweise zu einer kurzen Verzögerung beim Laden der Seite führt. Moodle sammelt normalerweise E-Mails welche versendet werden sollen und ein Cronjob schickt dann alle paar Minuten die E-Mails heraus. Eventuell wäre es von Vorteilen, den Plugin Code so umzuschreiben, dass die E-Mails in die Cronjob Wartschlage gespeichert werden, anstatt eine E-Mail direkt zu

versenden.

7.2.5. Architektur Refactoring

Moodle ist mittlerweile schon etwas in die Jahre gekommen. Dies merkt man als Benutzer nicht direkt, aber beim Betrachten des Quellcodes stellt man fest, dass der grundliegende Architektur insbesondere der objektorientierte Ansatz fehlt. Bei der Entwicklung von StudentQuiz wurden die Architekturzüge von Moodle übernommen, vor allem wohl weil es so von gewissen Teilen durch Moodle verlangt wird. Für zukünftige Gruppen würden wir empfehlen einen Versuch zu starten um eine neue Architektur für StudentQuiz zu erstellen, die auf konkreten Interface Abstraktionen aufbaut. Nicht nur würde dies den Code viel verständlicher machen, sondern es würde um vieles einfacher werden neue Features zu implementieren und zusätzlich wäre eine Kleinigkeit flächendeckende Unit Tests umzusetzen. Ein Refactoring sollte insbesondere den Zugriff auf die Datenbankelemente über Model Klassen oder sogenannte Repositories lösen, damit nicht in jeder zweiten Funktion ein handgestricktes SQL Statement gefunden werden kann.

8. Literaturverzeichnisse

8.1. Quellenverzeichnis

[1] D. Moshfegh und M. Schuler, «Moodle Integration Student-Quiz,» Hochschule für Technik Rapperswil, 2016.

[2] Moodle, «Moodle Plugin files,» [Online]. Available: https://docs.moodle.org/dev/Plugin_files. [Zugriff am 23 Dezember 2016].

[3] Moodle, «Moodle 3.2 release notes,» [Online]. Available: https://docs.moodle.org/dev/Moodle_3.2_release_notes. [Zugriff am 23 Dezember 2016].

[4] Bootstrap, «Bootstrap - Most popular HTML, CSS, and JS framework,» [Online]. Available: https://getbootstrap.com/. [Zugriff am 23 Dezember 2016].

[5] Redmine, «Redmine,» [Online]. Available: https://redmine.org/. [Zugriff am 23 Dezember 2016].

[6] Trello, «Trello,» [Online]. Available: https://trello.com/. [Zugriff am 23 Dezember 2016].

[7] D. Radigan, «Kanban,» Atlassian, [Online]. Available: https://www.atlassian.com/agile/kanban. [Zugriff am 23 Dezember 2016].

[8] Toggl, «Toggl,» [Online]. Available: https://toggl.com/. [Zugriff am 23 Dezember 2016].

[9] GitHub, «GitHub,» [Online]. Available: https://github.com/. [Zugriff am 23 Dezember 2016].

[10] F. Koch, «StudentQuiz Activity Plugin,» [Online]. Available: https://github.com/frankkoch/moodle-mod_studentquiz. [Zugriff am 23 Dezember 2016].

[11] F. Koch, «StudentQuiz Question Behaviour Plugin,» [Online]. Available: https://github.com/frankkoch/moodle-qbehaviour_studentquiz. [Zugriff am 23 Dezember 2016].

[12] Apache Friends, «XAMPP,» [Online]. Available: https://www.apachefriends.org/. [Zugriff am 23 Dezember 2016].

[13] JetBrains, «JetBrains,» [Online]. Available: https://www.jetbrains.com/. [Zugriff am 23 Dezember 2016].

[14] JetBrains, «PHPStorm,» [Online]. Available: https://www.jetbrains.com/phpstorm/. [Zugriff am 23 Dezember 2016].

[15] Moodle HQ, «Moodle Plugin CI,» [Online]. Available: https://github.com/moodlerooms/moodle-plugin-ci. [Zugriff am 23 Dezember 2016].

[16] Travis CI, «Travis CI,» [Online]. Available: https://travis-ci.org/. [Zugriff am 23 Dezember 2016].

[17] Moodle, «Moodel Coding style,» [Online]. Available: https://docs.moodle.org/dev/Coding_style. [Zugriff am 23 Dezember 2016].

[18] S. Chacon und B. Straub, Pro Git, Apress, 2014.

[19] Moodle, «Moodle plugins directory,» [Online]. Available: https://moodle.org/plugins/. [Zugriff am 21 Dezember 2016].

[20] Moodle, «Plugin Validation,» [Online]. Available: https://docs.moodle.org/dev/Plugin_validation. [Zugriff am 23 Dezember 2016].

[21] Moodle, «Prechecker StudentQuiz Activity,» [Online]. Available: https://integration.moodle.org/job/Precheck%20remote%20branch/29456/artifact/work/smurf.html. [Zugriff am 23 Dezember 2016].

[22] Moodle, «Prechecker StudentQuiz Question Behaviour,» [Online]. Available: https://integration.moodle.org/job/Precheck%20remote%20branch/29457/artifact/work/smurf.html. [Zugriff am 23 Dezember 2016].

[23] Moodle, «Javascript Modules,» [Online]. Available: https://docs.moodle.org/dev/Javascript_Modules. [Zugriff am 23 Dezember 2016].

8.2. Tabellenverzeichnis

8.3. Abbildungsverzeichnis

8.4. Glossar

Activity	Eine Aktivität in einem Kurs die einer Sektion/Woche zugewiesen ist. Kann auch ein StudentQuiz sein.
StudentQuiz	Die Moodle Activity StudentQuiz, die von Teachers erstellt werden kann.
Instanz	Eine Instanz ist eine Quizgenerierung welche unterschiedliche Fragen eines StudentQuiz beinhaltet.
Durchführung	(Quiz-)Durchführung ist die Durchführung eines StudentQuiz durch einen Studenten. Durchgeführt wird immer eine Instanz eines StudentQuiz.
Moodle-Quiz	Das Standardquiz von Moodle.

9. Anhang

9.1. A: Inhalt der digitalen Abgabe

1. Studienarbeit
2. Aufgabenstellung
3. Poster
4. Persönliche Stellungnahme – Lukas Dürrenberger
5. Persönliche Stellungnahme – Steven Ryser
6. Persönliche Stellungnahme – Alexis Suter
7. Acceptance Tests Student
8. Acceptance Tests Teacher
9. Benutzeranleitung – Student Manual
10. Benutzeranleitung – Teacher Manual
11. Benutzeranleitung – Administrator Manual
12. Benutzeranleitung – Developer Manual
13. Sitzungsprotokolle
14. Eigenständigkeitserklärung
15. Quellcode – StudentQuiz Activity
16. Quellcode – StudentQuiz Question Behaviour
17. Gantt Diagramm
18. Daten Diagramm

9.2. D. Sitzungsprotokolle

Die Sitzungsprotokolle sind zusammen gefasst als Archive in der digitalen Abgabe zu finden.

9.3. E: Manuals

Die vier Manuals sind in der digitalen Abgabe zu finden.