HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Moodle Plugin StudentQuiz

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2017

Autoren	Denis Manente, Simon Schaefer
Betreuer	Prof. Frank Koch
Experte	Stephan Meier
Gegenleser	Prof. Dr. Olaf Zimmermann

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

ALARP Ist ein englisches Akronym und bedeutet «as low as reasonably practicable»(so niedrig, wie vernünftigerweise praktikabel). , 96

Capabilities Capabilities umschreiben ein Recht eine Aktion auszuführen, welches einer Role in einem Context zugefügt werden kann. 17

CD Bezeichnet eine Sammlung von Techniken, Prozessen und Werkzeugen, die den Softwareauslieferungsprozess (englisch Continuous Delivery) verbessern.

CI Kontinuierliche Integration (englisch Continuous Integration) ist ein Begriff, der den Prozess des fortlaufenden Zusammenfügens von Komponenten zu einer Anwendung beschreibt. Das Ziel der kontinuierlichen Integration ist die Steigerung der Softwarequalität.

CLI Command-line interface.

Coding Style Unter Coding Style oder Coding Style Guide verstehen Softwareentwickler eine Sammlung von Konventionen, die eingehalten werden sollen, um Programmcode einheitlich und lesbar zu schreiben. 21

Confluence Applikation von Atlassian, welches sich mit JIRA kombinieren lässt und für die Ablage von Projektdokumentationen eignen..

Construction Phase des Rational Unified Process. In der Construction liegt der Fokus auf der Implementierung der einzelnen Features. Sie beginnt nach dem Ende der Elaboration und Endet mit dem Beginn der Transition.

Context Ein Context ist eine konzeptionelle und hierarchisch verschachtelbare Gruppierung von Zugriffsrechten und Objekten.

Continuous Integration Bezeichnet einen Entwicklungsprozess, bei welchem Software auch nach kleinen Anpassungen vollständig installiert und automatisiert getestet wird. 21

Controller Im Zusammenhang des Model-View-Controller Patterns koordiniert der Controller die Ausführung von Befehlen zwischen View und Model. 15

Docker Software zur Virtualisierung von Softwareumgebungen. 62

Elaboration Phase des Rational Unified Process. Nach der Inception liegt der Fokus auf der Erarbeitung der Anforderungen, Risikoanalysen, und Entscheidungskriterien der Architektur..

Fragetypen Moodle liefert in der Standardausführung die Implementierung von verschiedenen Fragetypen. Der einfachste ist vermutlich eine simple "Ja/Nein" Frage, oder Wertberechnungen. Die Kompliziertesten sind vermutlich die Embbeded Types, welche aus mehreren in einander verschachtelte Fragen bestehen. 10, 54

GUI Graphical User Interface: Allgemeine Bezeichnung für eine Grafische Benutzeroberfläche wie zum Beispiel eine im Browser dargestellte Website. 11, 24, 25

HSR Hochschule für Technik Rapperswil.

Inception Phase des Rational Unified Process. In der Inception werden die grobe Aufgabenstellung und die Rahmenbedingungen des Projekts definiert..

Integration Test Mit einem oder mehreren Integration Tests kann geprüft werden, ob sich dass das Gesamtsystem erwartungsgemäss verhält. 21

magische Konstante Mit einer magischen Konstanten wird eine meist willkürlich gewählter und selten konfigurierbarer Wert innerhalb des Codes bezeichnet. 19

Major Release Ein Major Release beschreibt die inkrementelle Veröffentlichung einer neuen Softwareversion, welche zusätzliche Funktionalitäten und Änderung der bereits verfügbaren Schnittstellen umfasst. 38, 94, *siehe auch* Semantic Versioning

Minor Release Ein Minor Release beschreibt die inkrementelle Veröffentlichung einer neuen Softwareversion, welche zusätzliche Funktionalitäten, aber keine Änderung der bereits verfügbaren Schnittstellen umfasst. 22, 38, 94, *siehe auch* Semantic Versioning

Mockup Ein Mockup in der Softwareentwicklung bezeichnet einen rudimentären Wegwerfprototypen der Benutzerschnittstelle einer zu erstellenden Software.

Model Im Zusammenhang des Model-View-Controller Patterns repräsentiert das Model die Objekte und Methoden der Software, ohne die Darstellung oder die Interaktion mit dem Benutzer zu implementieren. 15

Moodle Moodle ist eine frei verfügbare Open Source Software unter der GNU Public Licence Version 3. 9

MVC Das Model-View-Controller (MVC) Pattern beschreibt eine Softwarestruktur, in welcher die Logik zur Darstellung von Daten von der eigentlichen Datenmodellierung und der Interaktionssteuerung getrennt sind. 15

Pagination Eine Pagination ist ein grafisches Navigationselement, welches potentiell lange Listen von Elementen oder Tabellen mit vielen Zeilen in überschaubare Portionen unterteilt, sodass der Benutzer sich jeweils nur einen Teil der Liste anschauen und bei den nächsten Teil auf Wunsch nachladen kann. Bei StudentQuiz wird eine Pagination verwendet um eine potentiell sehr lange Tabelle von Fragen in kleinere Seiten von 20 Fragen pro Seite zu unterteilen. 18

Question Bank Die Question Bank ist eine Kernfunktionalität von Moodle, welche die Verwaltung von Prüfungsfragen unabhängig von ihrer Verwendung ermöglicht. 10, 11, 17, 34

Question Behaviour Ein Question Behaviour ist ein abstraktes Konzept der Question Engine, welche die Interaktionen zwischen dem Studenten und einer Frage innerhalb einer Question Usage steuert. 13, 54

Question Engine Die Question Engine fasst die verschiedenen Interaktionsmöglichkeiten zwischen Fragen und Nutzern von Fragen zusammen. 17, 33

Question Usage Eine Question Usage eine von einem Benutzer ausgewählte Menge von Fragen dar. Sie umfasst neben den Fragen auch die dazugehörigen Beantwortungsversuche und Bewertungszustände der jeweiligen Frage durch den jeweiligen User. Mit "Start Quiz" wird jeweils eine neue Question Usage für den Nutzer angelegt. 29, 30, 54

Renderer Ein Renderer umfasst Funktionen zur Generierung von HTML und anderen Ausgabenformate von Datenobjekten. 15

Reverse Proxy Der Reverse Proxy ist ein Proxy, der Ressourcen für einen Client von einem oder mehreren Servern holt. Die Adressumsetzung wird in der entgegengesetzten Richtung vorgenommen, wodurch die wahre Adresse des Zielsystems dem Client verborgen bleibt..

Role Beschreibt eine Rolle, die ein Benutzer in einem bestimmten Context einnehmen kann. Ein Nutzer kann mehrere Rollen haben.

RUP Rational Unified Process. Iterativer Prozess zur Projektentwicklung..

Section Ein Moodle Kurs kann in Sections unterteilt werden. Diese Unterstützen den Dozenten bei der Gruppierung seiner Unterrichtsmaterialien in chronologische oder thematische Einheiten. 17, 19

Semantic Versioning Definiert ein einheitliches Bezeichnungssystem für Softwareversionierungsnummern.

Separation of Concerns Objektorientierter Code soll so strukturiert werden, dass funktionale Einheiten klar definierte, abgegrenzte Funktionen erfüllen. Dies verbessert einerseits die Lesbarkeit des Codes und ermöglicht eine bessere Stabilisierung durch Unit Tests. 18

Staging Ein Verfahren, eine neue Produktversion funktionell aufzubauen, zu testen und zu reviewen bevor diese für die produktive Umgebung freigegeben wird. , 95

State Machine Eine State Machine (deutsch: Zustandsautomat) modelliert ein Verhalten anhand von Zuständen, welche durch definierte Aktionen ineinander übergehen.. 33

Stored Procedures Eine auf einer Datenbank programmierte Abfolge von Befehlen. Diese kann verwendet werden, um regelmässig anfallende oder durch bestimmte Ereignisse ausgelöste Aufgaben zu erledigen. 31

Themes Ein Theme ist eine Sammlung von Darstellungsregeln, Schriftarten, Farben und Grössen, welche das optische Erscheinungsbild einer Moodle-Installation definiert. 10

Transition Phase des Rational Unified Process. In der Transition wird das Projekt finalisiert und dem Auftraggeber übergeben..

Unit Test Mit einem oder mehreren Unit Tests können die verschiedenen Berechnungsschritte einer Methode eingehend geprüft werden.

View Im Zusammenhang des Model-View-Controller Patterns repräsentiert die View die Darstellung der Benutzerschnittstelle. 15

Views Eine View im Kontext einer Datenbank wird häufig zur Berechnung von Aggregatswerten über mehrere Relationen verwendet. Die Datenbank beobachtet dabei die zur Berechnung notwendigen Relationen und aktualisiert die Ergebnisse, wenn Änderungen an diesen vorgenommen werden. 31

1 Aufgabenstellung

Die folgende Aufgabenstellung wurde zu Beginn dieser Bachelorarbeit formuliert.

Die darin erwähnten Anlagen Todo-Liste und Bug Report finden sich im Anhang A.2

Aufgabenstellung Bachelor Thesis „Moodle Plugin StudentQuiz“

1. Ausgangslage, Problembeschreibung

Studierende schätzen Trainings-Prüfungen zur Vorbereitung auf summative Assessments. Aus diesem Grunde entwickelte die Hochschule für Technik in Rapperswil das Moodle Plugin StudentQuiz. Mit StudentQuiz können Studierenden eigene Fragen erstellen und in einem Pool miteinander teilen. Auch wenn der Beitrag einzelner Studierender nur klein ist, entstehen bei grösseren Gruppen schnell beachtliche Fragen-Sammlungen.

Die gesammelten Fragen können in StudentQuiz nach einer Vielzahl von Kriterien zu einem Quiz gebündelt werden. Beim Durchspielen der Quizzes können die Studierenden die Fragen kommentieren und bewerten. Aus den Nutzungsdaten ermittelt StudentQuiz die Qualität der Fragen und honoriert Studierenden Punkte für Beiträge und richtige Antworten. Die erstellten Fragen können zudem in weiteren Moodle-Tests recycliert werden.

2. Aufgabenstellung

Es handelt sich um eine GPLv3 Entwicklung im Umfeld von PHP und JavaScript für Moodle ab Version 3.3. Auf der fachlichen Seite geht es um e-Assessment und Gamification als innovative Lern- und Prüfungsform. Das aktuelle StudentQuiz steht als Activity Plugin (Version 2.0.3) und als zusätzliches Question Behaviour Plugin (Version 2.0.2) als Ausgangsbasis auf Github zur Verfügung. Folgende Aufgaben sollen umgesetzt werden:

2.1 Funktionale Anforderungen

Die funktionalen Anforderungen bestehen im Wesentlichen aus den nachstehenden Punkten. Detailliert werden diese in der Anlage «Todo StudentQuiz3 Stand 21.09.17», Kapitel «New Features» beschrieben. Die funktionalen Anforderungen wurden auch auf Redmine übernommen; Zugang: _____, Login: _____, Passwort: _____

- Learning Assistance
  Studenten können ihren Lernfortschritt einsehen, unbeantwortete bzw falsch beantwortete Fragen filtern, sowie den eigenen Lernfortschritt mit dem der Gruppe vergleichen.
- Moodle Mobile App / Moodle Desktop
  Quizzes sollten innerhalb der Moodle Mobile App durchgeführt werden können
- Orphaned Section 999
  Im Moodle 3.3 wurden Orphaned Sections abgeschafft. Die Lösung durchgespielte Quizzes in der Orphaned Section 999 abzuspeichern funktioniert deshalb nur noch ungenügend. Da das Speichern durchgespielter Quizzes nur einen geringen Mehrwert hat, soll ganz darauf verzichtet werden und die Orphaned Section 999 aufgelöst werden.
- Restriction of Question Types
  Der Teacher soll konfigurieren können welche Fragetypen für eine SQ-Activity zugelassen sind.
- Konfiguration auf Activity-Level
  Momentan kann SQ nur site-wide konfiguriert werden.

2.2 Verbesserung bestehender Funktionen

Die angestrebten Verbesserungen bestehen im Wesentlichen aus den nachstehenden Punkten. Detailliert werden diese in der Anlage «Todo StudentQuiz3 Stand 21.09.17», Kapitel «Improvement Features» beschrieben. Die angestrebten Verbesserungen wurden auch auf Redmine übernommen; Zugang: _____, Login: _____, Passwort: _____

- Endgültige Lösung gelöschter Fragen
  Gelöschte Fragen sind weiterhin ausgegraut sichtbar und bearbeitbar. Diese sollten ganz verschwinden
- Deaktivierung Gradebook
  Im Gradebook sollte weder eine SQ-Activity noch ein SQ-Quiz noch eine SQ-Quiz-Instanz erscheinen
- Löschen Kommentare
  Kommentare können aktuell nur vom Ersteller und Admin gelöscht werden. Diese sollten aber auch vom Lehrer und Manager gelöscht werden können.
- Anonymisierung Kommentare
  Für die Anonymität/Pseudonyme von Kommentaren wird aktuell die swissEduPersonUniqueID vom SwitchAAI verwendet - ein Wert der im LDAP der HSR indirekt abgefragt werden kann (objectSid). Damit ist Anonymität nicht gewährt.
- Zugriff «non-editing teacher»ext
  “Non-editing teacher” sollte gleiche Rechte wie ein Student haben
- Performance-Optimmierung
  Bei sehr grossen Fragemengen führt ein Klick auf «StartQuiz» zu sehr langen Wartezeiten
- Verbesserungen im User-Interface
  Kleinere Arbeiten bzgl Labels, Tags, Sortieren, Filtering
- Verbesserung Reports
- Allenfalls Beseitigung Bugs

2.3 Bugs

- Leider wurden in der Vorgänger-Software () einige Bugs bekannt. Diese sollten möglichst zu Beginn der Arbeit bereinigt werden und dienen damit auch zur Einarbeitiung. Die Bugs werden in der Anlage «Bug Report Stand 21.09.17», beschrieben und wurden auch auf Redmine übernommen; Zugang: _____, Login: _____, Passwort: _____

2.4 Nicht funktionale Anforderungen

- Open Source
  Freie Software gemäss Lizenz GNU GPLv3+
- Technologien
  StudentQuiz sollte sowohl unter PHP 5.6 als auch 7.0 laufen
- Usability
  StudentQuiz sollte unter allen Core-Themes laufen, inklusive Boost. Auf mobiles Handling wird Wert gelegt.
- Scalability
  StudentQuiz erzeugt höchste Nutzungsraten. Im Pilotversuch WI1 im HS17 mit ca 80 Studierenden und 500 Fragen wurden knapp 15'000 Zugriffe erzeugt, was extrem hoch ist. StudentQuiz soll aber auch Gruppen mit bis zu 500 Studierenden und 5'000 Fragen bedienen können. Dann ist mit circa 500'000 Zugriffen innerhalb eines Semesters zu rechnen. Das System soll so getestet werden, dass es mindestens 50 Zugriffe pro Minute bedienen kann.

- Mehrsprachigkeit
  StudentQuiz ist in Englisch, Deutsch und Französisch verfügbar. Änderungen im UI sind entsprechend zu übersetzen.
- Dokumentation
  Da das Projekt nach der Bachelorarbeit weiterentwickelt wird, wird Wert auf eine gute Dokumentation für die verschiedenen Anwender und Entwickler gelegt. Die englischsprachigen Manuals für Administrator, Teacher und Student sind zu aktualisieren.

3. Zur Durchführung

Interesse an Design und Entwicklung einer Open Source Web-Anwendung im Bereich E-Learning wird vorausgesetzt.

Mit dem Betreuer finden Besprechungen gemäss Absprache statt. Die Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse sind in einem Protokoll zu dokumentieren, das dem Betreuer per E-Mail zugestellt wird. Die Projektsprache ist Englisch.

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.

4. Dokumentation und Abgabe

Wegen der beabsichtigten Weiterverwendung der Ergebnisse wird auf Vollständigkeit und Qualität der Dokumentation in englischer Sprache erhöhter Wert gelegt.

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 Dokumentation ist vollständig auf CD in drei Exemplaren abzugeben.

Neben der Dokumentation sind abzugeben:

- ein Poster zur Präsentation der Arbeit
- alle zum Nachvollziehen der Arbeit notwendigen Ergebnisse und Daten (Quellcode, Buildskripte, Testcode, Testdaten usw.)
- Material für eine Abschlusspräsentation (ca. 20’)

5. Termine

14.09.17	Beginn der Bachelorarbeit, Ausgabe der Aufgabenstellung durch die Betreuer.
19.12.17	Die Studierenden geben den Abstract für die Diplomarbeitsbroschüre zur Kontrolle an ihren Betreuer/Examinator frei. Die Studierenden erhalten vorgängig vom Studiengangsekretariat die Aufforderung mit den Zugangsdaten zur Online-Erfassung des Abstracts für die Broschüre.

	Die Studierenden senden per Email das A0-Poster zur Prüfung an ihren Examinator/Betreuer. Vorlagen sowie eine ausführliche Anleitung betreffend Dokumentation stehen unter den allgemeinen Infos Diplom-, Bachelor- und Studienarbeiten zur Verfügung.
22.12.17	Der Betreuer/Examinator gibt das Dokument mit dem korrekten und vollständigen Abstract der Broschüre zur Weiterverarbeitung an das Studiengangsekretariat frei.
22.12.17	Abgabe des Berichtes an den Betreuer bis 12.00 Uhr. Fertigstellung des A0-Posters bis 12.00 Uhr.
08.01.18 – 02.02.18	Mündliche BA-Prüfung.
02.03.18	Bachelorfeier und Ausstellung Bachelorarbeiten

6. Beurteilung

Eine erfolgreiche Bachelorarbeit erhält 12 ECTS-Punkte (1 ECTS Punkt entspricht einer Arbeitsleistung von ca. 25 bis 30 Stunden). Die Bewertung erfolgt gemäss nachstehender Kriterien:

Gesichtspunkt	**Gewicht**
1. Organisation, Durchführung (Projektplanung u. Nachführung Arbeit gemäss Projektplan, Selbstständigkeit, Einsatz, Zusammenarbeit mit Auftraggeber, Betreuer)	1/6
2. Bericht (Inhalt des Projektschlussberichts, Gliederung, Darstellung, Sprache der gesamten Dokumentation)	1/6
3. Inhalt 3.1 Problemanalyse (Vorstudie, Literaturstudium, Anforderungsspezifikation, Anforderungsanalyse, Domainanalyse) 3.2 Lösungsentwurf (Lösungsvarianten und deren Beurteilung, Variantenentscheid, Konzept, Entwurf) 3.3 Realisierung und Test	1/2 1/6 1/6 1/6
4. Präsentation und Mündliche Prüfung zur Bachelorarbeit	1/6

Im übrigen gelten die Abläufe und Regelungen Studien- und Bachelorarbeiten im Studiengang Informatik.

7. Betreuer

HSR, Prof. Frank Koch, Dozent für Wirtschaftsinformatik

8. Anlagen

- Dokument «Todo STudentQuiz3 Stand 21.09.17»
- Dokument «Bug Report Stand 21.09.17»

Rapperswil, den 21.09.17

2 Abstract

Moodle ist eine frei verfügbare, weltweit eingesetzte Open Source Software für Dozenten und Studenten auf allen Ebenen von der primären bis zur universitären Ausbildung. Eine engagierte und weitverzweigte Gemeinschaft entwickelt neben den Kernfunktionalitäten eine Vielzahl von zusätzlichen Plugins und Themes, welche wie Moodle selbst unter der GNU General Public Licence V3 lizenziert sind.

Ein solches Plugin ist StudentQuiz, welches im Rahmen von zwei vorangehenden Studentenarbeiten entwickelt wurde. Es ermöglicht den in einen Kurs eingeschriebenen Studenten eine Sammlung von Repetitionsfragen aufzubauen und diese gegenseitig zu bewerten, zu kommentieren und zu üben. Die dabei entstehenden Daten werden pro Frage ausgewertet, sodass sie bezüglich ihrer Schwierigkeit und Bewertung gruppiert und gezielt geübt werden können. Eine Punktewertung und Rangliste belohnt gute Fragen und fördert die aktive Beteiligung. Dozenten des Kurses können die Korrektheit der Fragen bestätigen und schlechte Fragen löschen.

Im Rahmen dieser Bachelorarbeit erweitern wir StudentQuiz um verschiedene Features zur Abbildung und Nutzung des persönlichen Lernfortschritts für Studenten und überarbeiten die Architektur des Plugins fundamental damit das Plugin auch bei grösseren Datenmengen skaliert. Mit unserem Beitrag stellen wir die zukünftige Nutzbarkeit, Wartbarkeit und Weiterentwicklung des Plugins sicher, welche durch frühere Architektur- und Designentscheidungen gefährdet war.

3 Management Summary

3.1 Ausgangslage

Wir übernahmen zu Beginn dieser Bachelorarbeit das Projekt StudentQuiz in der Version 2.0.3, das Ergebnis von zwei aufeinanderfolgenden Studienarbeiten, mit einer Mängelliste von bekannten Problemen, einer Liste von Anpassungswünschen und den Zielen zur Umsetzung neuer Features.

3.2 Vorgehen und Technologien

Wir entschieden uns für einen iterativen Prozess um die verschieden Aufgaben anzugehen. Aufbauend auf unseren Vorkenntnissen in der Softwareentwicklung in der Skriptsprache PHP, arbeiteten wir uns ausgehend von der bestehenden Codebasis von StudentQuiz in die Moodle Plugin Entwicklung ein.

Dazu konnten wir sowohl auf die Dokumentation der Vorarbeiten, wie auch diejenige der Moodle Community zurückgreifen. Wir richteten uns auf der von der HSR zur Verfügung gestellten Infrastruktur eine flexible Staging-Umgebung ein, auf der wir die verschiedenen Konstellationen von Moodle und StudentQuiz Versionen testen konnten.

Nach der Einarbeitung priorisierten wir auf der Mängelliste die Punkte, welche die Nutzung des Plugins für die im Mai 2017 veröffentlichte Moodle Version 3.3 problematisch bis unzumutbar machten. Parallel dazu erarbeiteten wir unsere Lösungsansätze für gewünschten neuen Features und Verbesserungen.

Während wir uns schon mit der Abarbeitung der Mängel und der Umsetzung verschiedener kleinerer Anpassungen befassten, wurde immer klarer, dass die längerfristige Entwicklung und die Skalierbarkeit des Plugins durch die zugrundeliegende Architektur gefährdet ist. Wir entschieden uns die Umbauten in der Architektur vorzunehmen und die damit verbundenen Risiken in Kauf zu nehmen.

3.3 Ergebnisse

Anfang November veröffentlichten wir einen Minor Release für StudentQuiz in der Version 2.1.0, welcher kritische Probleme der bisherigen Architektur oberflächlich löste, wodurch das Plugin auch wieder unter der Moodle Version 3.3 und der zeitgleich veröffentlichten Version 3.4 benutzbar wurde. Diese Version von StudentQuiz war aber weiterhin nicht skalierbar und bediente sich nach wie vor fragwürdiger Methoden um die Use Cases umzusetzen.

Anfang Dezember finalisierten und veröffentlichten wir die Version 3.0.0 als Major Release, welche den Abschluss unserer tiefgehenden Umbauten in der Architektur des Plugins, die Einführung des persönlichen Lernfortschritts für Studenten, sowie die Umsetzung der grösseren und kleineren Anpassungen und Mängelbehebungen markiert. Diese Version wurde erfolgreich auf der produktiven Moodle Instanz der HSR installiert und wird seitdem verwendet.

3.4 Ausblick

StudentQuiz kann und soll weiterentwickelt werden:

- Die **Integration für Moodle Mobile** wird von vielen Nutzern sehnlichst erwartet und die Attraktivität des Plugins deutlich steigern.
- Die Ergänzung und Weiterentwicklung unserer **automatisierten Acceptance und Unit Tests** wird es zukünftigen Entwicklern erleichtern, ihre Beiträge anhand komplexer Szenarien zu verifizieren.
- Aus Zeitgründen konnten wir im Rahmen dieser Bachelorarbeit zahlreiche grösseren und kleineren **Verbesserungen** nicht mehr umsetzen, welche die Testbarkeit und Wartungsfreundlichkeit des Codes weiter verbessern.
- Die **Personal Learning Assistance** von StudentQuiz kann nun weiter modernisiert und optimiert werden. Unsere konzeptionellen Vorschläge können dazu nächste Schritte aufzeigen.

4 Technischer Bericht

4.1 Ausgangslage

4.1.1 Moodle

Moodle ist eine frei verfügbare und unter der GNU Public Licence V3 lizenzierte Open Source Software, welche seit 1999 von Schulen, Universitäten und anderen Ausbildungsstätten genutzt wird. Sie ermöglicht es den Dozenten eines Kurses Unterrichtsmaterialien aller Art den eingeschriebenen Studenten zur Verfügung zu stellen.

Versionen Die Entwicklung der ersten Moodle Version geht auf den Australier Martin Dougiamas zurück, welcher noch heute als Unternehmer in der weltweiten Moodle Bewegung aktiv ist. Während dieser Bachelorarbeit wurde die Moodle Version 3.4 veröffentlicht. Dies geschah im Rahmen des seit 2013 eingeführten Versionierungsrhythmus, welcher jeweils im Mai und im November einen halbjährlichen Major-Release, also einen Sprung in der ersten Dezimalstelle der Versionsnummer vorsieht. Wie aus Abbildung 1 abgelesen werden kann, sind für diese Bachelorarbeit die Moodle Versionen 3.1 bis 3.4 relevant.

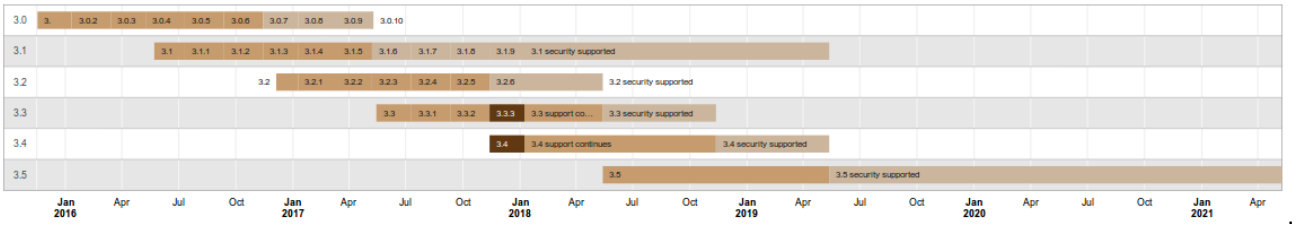


Abbildung 1: Veröffentlichungstermine und offizielle Supportlaufzeiten der verschiedenen Moodle Versionen

Verbreitung Gemäss eigener Statistik[9] werden aktuell knapp 100'000 Moodle Installationen in 234 Ländern betrieben. In der Abbildung 2 fällt auf, dass Moodle insbesondere in Spanien überproportional häufig eingesetzt wird.

4.1.2 Moodle Aktivitäten

Der Name Moodle selbst ist ein englisches Akronym, welches zu **m**odular **o**bject-**o**riented **d**ynamic **l**earning **e**nvironment ausgeschrieben wird. Ausgehend von diesem modularen Ansatz bietet die Moodle auch schon in der Standardausführung eine Vielzahl von modular einsetzbarer Aktivitäten (Activities) an, welche die Studenten bei der Auseinandersetzung mit dem vermittelten Stoff unterstützen und motivieren. So kann ein Dozent in einem Kurs eine Umfrage hinzufügen, über welche er von den Studenten ein Meinungsbild zu seiner Unterrichtseinheit einholen kann, oder eine Forum-Aktivität einbinden, über welche sich die Studenten zu kontroversen Themen oder Aufgabenstellungen im Rahmen des Kurses austauschen können. Von den Dozenten oft genutzt ist auch die Möglichkeit einem Kurs eine Quiz-Aktivität hinzuzufügen.

Vereinigte Staaten von Amerika (USA)	10,795
Spanien	8,162
Brasilien	5,382
Mexiko	5,293
Italien	3,801
Großbritannien	3,695
Indien	3,640
Kolumbien	3,149
Deutschland	2,909
Russland	2,561

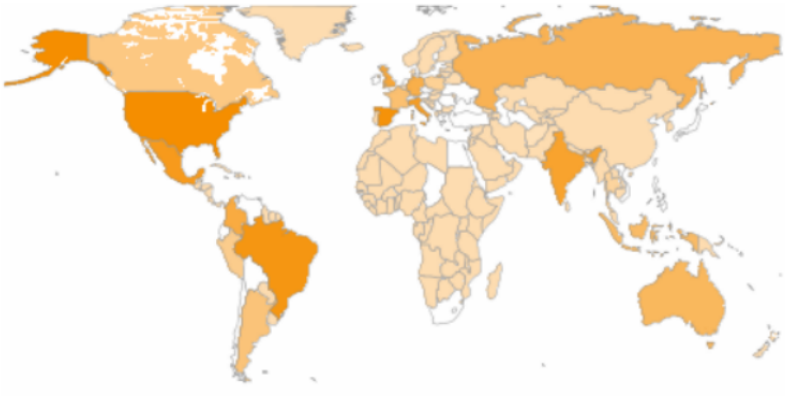


Abbildung 2: Top 10 der Moodle-Installationen pro Land und globale Verbreitung (Stand Dezember 2017)

4.1.3 Moodle Aktivität: Quiz

Die Quiz Aktivität standen schon in den ersten Moodle Versionen zur Verfügung. Die ursprüngliche Metapher war vermutlich ein ausgedrucktes Prüfungsblatt mit einer Auswahl von Fragen, welche die Studenten eines Kurses zu beantworten haben. Dieser doch sehr universelle Bestandteil eines Unterrichts verlangte nach einer Abbildung in der virtuellen Moodleumgebung. Die automatisierte Auswertung der gegebenen Antworten im Gegensatz zur zeitaufwendigen, manuellen Korrektur, wird wohl so manchen Dozenten schon überzeugt haben, die Quiz Aktivität für grössere und kleinere Zwischenprüfungen zu nutzen. Aufgrund der vielseitigen Anwendungsszenarien und didaktischen Konzepte wurde im Lauf der Entwicklung von Moodle mehrfach überarbeitet, erweitert und optimiert. In seiner heutigen Version bietet die Quiz Aktivität bereits in der Standardausführung 15 verschiedene Fragetypen und unterstützt die systematische Wiederverwendung von Fragen mit einer Question Bank.

4.1.4 Moodle Feature: Question Bank

Die Question Bank ist eine Kernfunktionalität, welche den Dozenten eines Kurses zur Verfügung steht. Sie dient der von jeglichen Aktivitäten unabhängigen Verwaltung von Prüfungsfragen. So können Fragen in hierarchischen Kategorien organisiert, aus Kursen exportiert und in andere importiert werden.

4.1.5 Moodle Plugins

Ein zentraler Bestandteil der Moodle Software ist ihre Erweiterbarkeit durch eine Vielzahl von Plugins. Die Software ist absichtlich so gebaut worden, dass die Integration von zusätzlichen Funktionalitäten, eigenen Themes, Schnittstellen zu anderen Systemen auf den verschiedensten Ebenen der Architektur möglich ist. Diese sogenannten Plugins werden von einer weitverzweigten und vielseitigen Gemeinschaft von Softwareentwicklern und Anwendern gepflegt, weiterentwickelt und geprägt. Das offizielle Moodle Plugin Directory verwaltet per Dezember 2017 über 1400 Plugins.

4.1.6 StudentQuiz Projekthistorie

Bei dieser erstaunlichen Vielfalt von verschiedenen Plugins und der langjährigen Entwicklungsgeschichte von Moodle, stellt sich die Frage, welche Use Cases noch nicht durch bestehende Lösungen abgedeckt werden, welche die Entwicklung eines weiteren Plugins legitimieren. StudentQuiz ermöglicht es den Studenten eigene Fragen in die Question Bank eines Kurses einzutragen und anschliessend beliebig oft zu beantworten. Damit soll die Leistung der Studenten nicht wie in einer gewöhnlichen Quiz Aktivität bewertet werden, sondern zum wiederholten Üben ermutigt werden. Während die Metapher des Quiz Moduls also ein Prüfungsblatt ist, stelle man sich bei StudentQuiz eine gemeinsame Lernkartei vor, welche durch alle Studenten eines Kurses ergänzt, bewertet, kommentiert und vor allem auch zum Üben benutzt werden kann. Mit diesem Ansatz bindet StudentQuiz die Gemeinschaft der Studenten in einen gemeinsamen Lernprozess ein. Die Qualität der Fragen wird neben dem Bewertungssystem durch die Nutzer auch dadurch gewährleistet, dass die Dozenten im Gegensatz zu den Studenten, Fragen bezüglich ihrer Korrektheit bestätigen können oder im Falle einer unpassenden oder schlichtweg falschen Frage diese korrigieren oder gleich löschen können.

StudentQuiz wurde im Frühlingssemester 2016 erfolgreich im offiziellen Moodle Plugin Directory aufgenommen. Das Plugin erfreut sich wachsender Bekanntheit und wurde Anfang September 2017 auf rund 120 Moodle Installationen betrieben.

Im Austausch mit den verschiedenen Anwendern und nicht zuletzt auch mit eigenen Praxiserfahrungen aus dem Betrieb an der HSR wurden Probleme und Mängel erkannt, sowie verschiedene Ideen angedacht, in welche Richtung StudentQuiz weiterentwickelt werden könnte.

4.1.7 Legitimation für eine Bachelorarbeit

Bevor mit einer Bachelorarbeit begonnen werden kann, muss ermittelt werden, ob die Problemstellung genug Bereiche für einen konzeptionellen, theoretischen und einen praktischen Teil hergibt.

Praktischer Teil Vordergründig ist diese Bachelorarbeit als sehr praxisorientiertes Programmierprojekt zu sehen. Unser Betreuer Prof. Frank Koch nimmt darin die Rolle des Auftraggebers ein, welcher uns mit der Weiterentwicklung seines bestehenden Softwareprojektes nach seinen Vorstellungen beauftragt. Konkrete Anpassungen am Graphical User Interface (GUI) der Applikation und die Ergänzung des bestehenden Funktionsumfangs um neue Funktionalitäten sind typische Aufgaben, welche für sich genommen noch keine Bachelorarbeit legitimieren, sondern auch einfach einem Freelance-Entwickler in Auftrag gegeben werden könnten.

Konzeptioneller Teil Neben der Erarbeitung und Abwägung der Ursachen und Lösungsvorschläge komplexerer Probleme oder Fehler besprechen wir im konzeptionellen Teil dieser Arbeit die Entwicklung der neuen Funktionalitäten aus verschiedenen Blickwinkeln. Nicht zuletzt soll unser Beitrag dazu sein die Weiterentwicklung des Projekts zu ermöglichen.

Theoretischer Teil Für viele der im praktischen oder konzeptionellen angewandten Methoden und Einsichten greifen wir auf theoretische Grundlagen zurück, welche uns auf unserem Ausbildungsweg vermittelt wurden.

4.2 Problembeschreibung

Ausgehend von der Aufgabenstellung gruppieren wir in diesem Abschnitt die wichtigsten Schwerpunkte, welche es in dieser Bachelorarbeit zu bearbeiten gilt. Die detaillierte und vollständige Liste der gewünschten Verbesserungen und Mängelliste sind im Anhang beigelegt.

- **Orphaned Section 999** Unter diesem Titel bearbeiten wir ein mit der Moodle Version 3.3 entstandene Problematik, welche die Durchführungen von einzelnen Quiz in einer unschönen Darstellung sowohl in der Datenbank wie auch gegenüber den Dozenten und nicht zuletzt auch in Form von massiven Latenzproblemen zu Tage tritt. Es gilt abzuwägen in welcher Form die Architektur und das Datenmodell des Plugins angepasst werden müssen.
- **Performance** Rückmeldungen aus der praktischen Nutzung des Plugins haben grosse Latenzzeiten bei Anwendungen auf Kursen mit dreistelligen Studenten und Fragezahlen bemängelt. Wir sollen die Performanceprobleme von StudentQuiz analysieren und soweit beheben, dass die diesbezüglich angegebenen nicht funktionalen Anforderungen erfüllt werden.
- **Personal Learning Assistance** Um das gewünschte neue Feature sauber umzusetzen, bedarf es in der Elaboration einer detaillierten Klärung der benötigten Anpassungen am Datenmodell.
- **Testing** Die Testbarkeit von bereits vorhandenem Code ist ein wichtiger Grundstein der Softwareentwicklung. In der Moodlewelt werden dabei verschiedene Technologien und Lösungen verwendet. Wir lesen uns in die für uns relevanten Testing Strukturen ein und ermittelten daraus, welche funktionalen und nicht funktionalen Tests zusätzlich formuliert und automatisiert werden müssen, um eine längerfristig solide Testbarkeit des Plugins sicherzustellen.
- **Kompatibilität Moodle Mobile App** Der übernommene Stand des StudentQuiz kann in der der Moodle Mobile App nicht verwendet werden. Wir ermitteln die dazu nötigen Anpassungen.

Neben diesen Schwerpunkte arbeiten wir uns in folgende weitere Themen ein, um die gewünschten Verbesserungen umzusetzen.

- **Badge System und Activity Completion API** Moodle bietet es Dozenten an für die erfolgreiche Bearbeitung von Activities oder auch der erfolgreichen Teilnahme an bestimmten Kursen den Studenten Badges zur Auszeichnung auszustellen. Neben dem Beheben der gefundenen Fehlern bei der Anbindung der dahinterstehenden Completion API möchten wir ausloten, ob damit zusätzliche kreative Möglichkeiten entstehen.
- **Backup/Restore/Import, Uninstall und Upgrade** Wir möchten verstehen, wie das Plugin diese verschiedenen Mechanismen implementiert um bei Zukünftigen Änderungen am Datenbankschema oder anderen Anpassungen, sowie für die Behebung der gemeldeten Bugs einen guten Überblick über die verschiedenen Konsequenzen zu haben.
- **Notifications** Bestimmte Aktionen, wie das Kommentieren einer Frage, sollen Benachrichtigungen an die betreffenden Studenten auslösen. Um dies implementieren können, müssen wir uns mit dem Event System von Moodle auseinandersetzen.

4.3 Analyse

In diesem Abschnitt möchten wir die wichtigsten Ergebnisse unserer Einarbeitung in die bestehende Codebasis von StudentQuiz 2.0.3 präsentieren. Diese bilden die Grundlage aufgrund derer wir unsere Lösungsansätze für die zu gesetzten Ziele erarbeiteten. Wir verzichten auf eine vollständige Dokumentation der bestehenden Architektur, sondern verweisen dafür auf die vorausgehende Studienarbeit.[31] Wer sich für die aus dieser Arbeit resultierende Architektur interessiert, findet sie in unserer Softwaredokumentation ab Abschnitt 5. An dieser Stelle möchten wir uns auch für die Unterstützung und Dokumentation unserer Vorgänger bedanken, deren Erfahrung wir bei mehreren Gelegenheiten während der Analyse einbeziehen durften.

4.3.1 Eine Erweiterung, zwei Plugins

StudentQuiz besteht aus zwei separaten Moodle Plugins.

Das Moodle Plugin `mod_studentquiz` implementiert ein sogenanntes Activity Modul. Dabei wird die Activity StudentQuiz eingeführt und somit in einem Kurs verwendbar. Auch die zusätzlichen Ansichten wie Ranking und Statistics werden durch dieses Modul definiert und berechnet.

Das Moodle Plugin `qbehaviour_studentquiz` implementiert ein sogenanntes Question Behaviour für Fragen in Moodle. Ein Question Behaviour erweitert die von Moodle vorgesehen Interaktionen zwischen dem Studenten und einer Frage um die Funktionalität, nach welcher Studierende nach der Beantwortung einer Frage diese bewerten müssen und optional kommentieren können.

Die gewählte Architektur bietet keine Möglichkeit, die Funktionalität vom Question Behaviour in StudentQuiz zu integrieren. Denn sobald ein Student ein Quiz durchführt, wird diese alleinständig von der Quiz Activity verwaltet. StudentQuiz hat jedoch die Konfiguration dieser Quiz Activity vornehmen können, sodass für die Beantwortung der Fragen das Question Behaviour von StudentQuiz verwendet werden soll.

Wer das StudentQuiz Modul installiert, muss nicht zwingend das Question Behaviour von StudentQuiz installieren, dann werden lediglich die gegebenen Antworten ausgewertet. Es ist aber nicht möglich das Question Behaviour von StudentQuiz ohne das StudentQuiz Modul sinnvoll zu verwenden, da die benötigte Datenstruktur im Verantwortungsbereich des Moduls liegt.

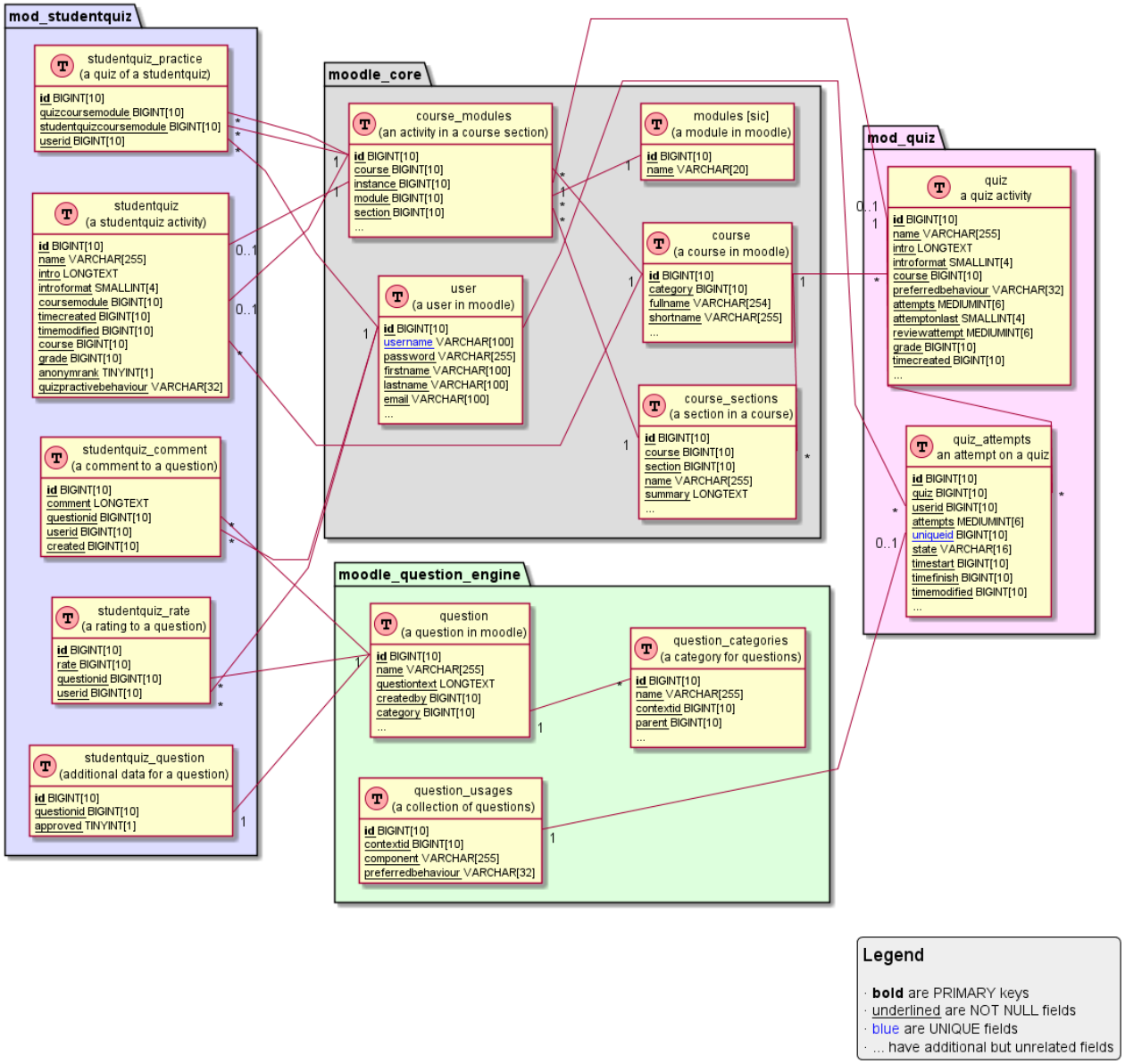


Abbildung 3: Datenmodell Stand StudentQuiz 2.0.3

Das Datenmodell in Abbildung 3 zeigt die effektive Umsetzung der Datenbank mit allen expliziten Beziehungen. Nicht direkt erkennbar ist, dass sich das Question Behaviour im Feld `preferredbehaviour` als String in der Quiz Tabelle versteckt, welches StudentQuiz entsprechend konfiguriert hat. Dafür zeigt es auf, dass die Tabelle `question_usages` über die Tabelle `quiz_attempts` mit dem Quiz verbunden ist. Denn die Question Usage ist der Einstiegsknoten, um an die Question Attempt Daten der Studenten zu gelangen. Die Question Usage wird vom Quiz konfiguriert und das Question Behaviour im Feld `preferredbehaviour` ist ebenfalls als String abgelegt.

4.3.2 Einstiegspunkte

Um das Verhalten eines Moodle Plugins zu analysieren empfehlen wir die verschiedenen Einstiegspunkte durchzugehen. Mit einem Einstiegspunkt markieren wir hier eine im Browser aufrufbare URL, welche die Ausführung von StudentQuiz Skriptes aktiviert. Wir unterscheiden in einem PHP-Projekt ausführbare Skripte mit prozeduralen Befehlen von deklarativen Klassen und Funktionsbeschreibungen.

- **view.php** Startseite mit Frageliste und Filterformular.
- **reportrank.php** Rangliste
- **reportquiz.php** Statistik
- **settings.php** Darstellung der systemweiten Plugineinstellungen.
- **index.php** Liste aller StudentQuiz Activities dieses Kurses
- **import.php** Importieren von Fragen
- **export.php** Exportieren von Fragen
- **save.php** Entgegennahme von Ratingwerten und Kommentaren
- **comment_list.php** Abfrage Kommentarliste
- **remove.php** Löschen von Kommentaren

Der von Moodle empfohlene Aufbau dieser Skripte umfasst die folgenden Schritte:

1. Aufruf des globalen Konfigurationsskriptes `config.php` welches die die wichtigsten globalen Objekte wie DB, USER, SESSION, PAGE und OUTPUT initialisiert und Moodle's Standardbibliotheken einbindet.
2. Einbindung der lokalen Bibliotheken und Klassen des Plugins (locallib.php und weitere)
3. Validierung und Entgegennahme der erforderlichen Aufrufparameter
4. Sicherstellung der Benutzerauthentifikation
5. Allfällige Umleitungen an andere Einstiegspunkte aufgrund von bestimmten Aufrufparametern.
6. Initialisierung und Bearbeitung des lokalen Models
7. Initialisierung eines lokalen Renderers.
8. Aktualisierung der globalen PAGE Variable mit Werten aus dem Model oder dem Renderer.
9. Ausgabe der Kopfzeile des globalen Renderers
10. Ausgabe des lokalen Renderers mit `echo`
11. Ausgabe der Fusszeile des globalen Renderers.

Diese bewährte Struktur entspricht einer Anwendung des sogenannten Model-View-Controller (MVC) Patterns. Der Einstiegspunkt übernimmt dabei die Rolle des Controllers, der Renderer diejenigen der View und das Model wird durch die in dieser Ansicht instanziierte Klassen der Businesslogik (StudentQuiz, Versuch, Question Bank) repräsentiert.

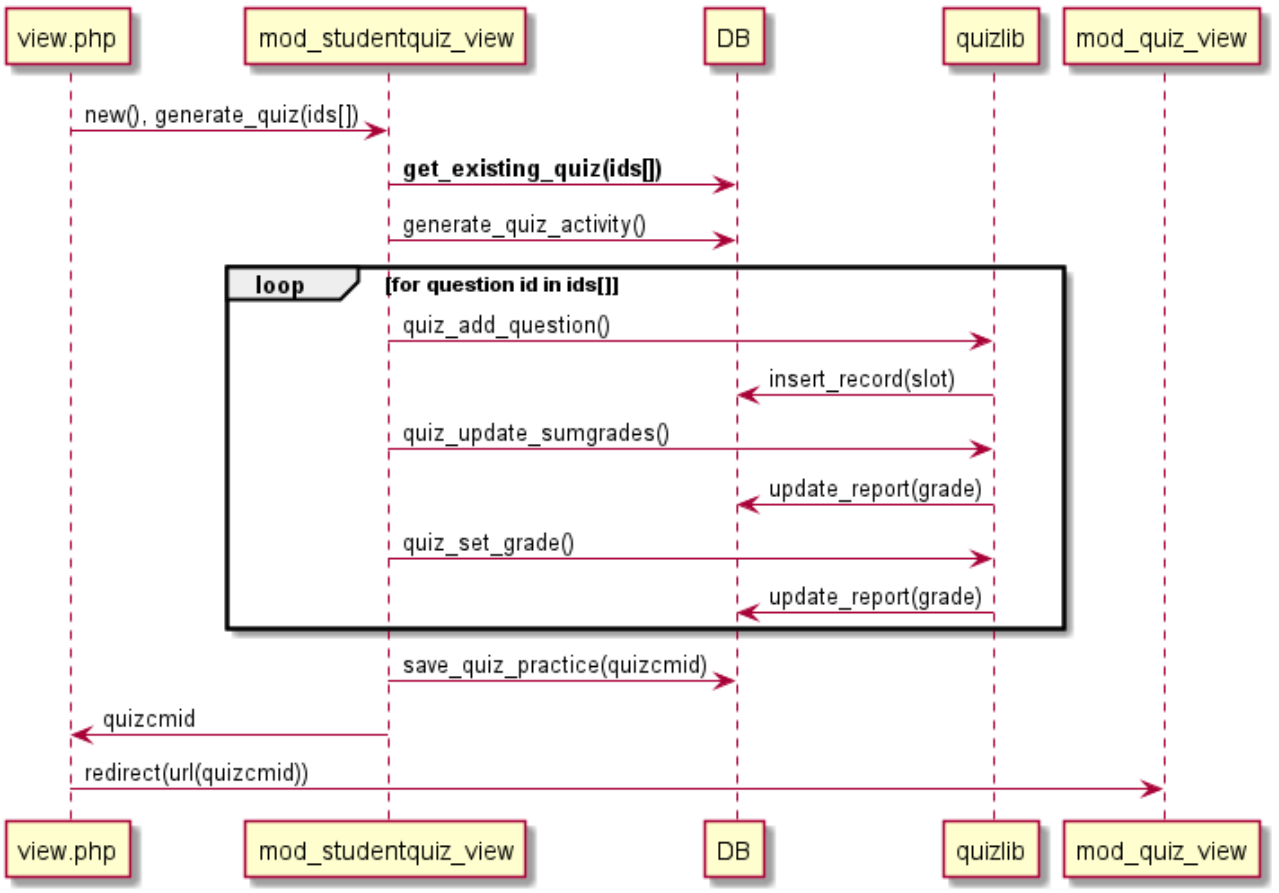


Abbildung 4: Sequenzdiagramm "Start Quiz"in StudentQuiz 2.0.3

4.3.3 Start Quiz

Bei diesem Vorgang gilt es herauszufinden, wie die durch verschiedene Nutzer beschriebenen Latenzzeiten entstehen und wie man diese verringern könnte. Im Hinblick auf die späteren Optimierungen konzentrieren wir uns bei der Analyse dieser Aktionen bezüglich ihrer Aufrufkomplexität.

Diese Prozedur erhält die Identifikationsnummern der ausgewählten Fragen als Aufrufparameter. Wie Abbildung 4 illustriert, wird bei einem Klick auf «Start Quiz»eine Abfrage an die Datenbank gestartet um ein bereits angelegtes Quiz zu finden, welches mit genau dieser Auswahl von Fragen initialisiert wurde.

Listing 1 zeigt die implementierte Abfrage, welche leider die innere Select-Abfrage für jede Zeile einzeln berechnen muss, was schlecht skaliert.

```
1 SELECT  quizid, COUNT(quizid) FROM {quiz_slots} s1
2 WHERE questionid IN (:ids)
3 AND (select count(s2.quizid) FROM {quiz_slots} s2 WHERE s1.quizid = s2.quizid) = :countids
4 GROUP BY quizid
5 HAVING COUNT(questionid) = :countids
```

Listing 1: Abfrage nach einem Quiz mit derselben Fragenauswahl

Falls die Abfrage keine identische Fragenauswahl findet, wird ein neues Quiz konfiguriert und in einer ver-

steckten Section des Kurses abgelegt. Dabei fällt auf, dass das Hinzufügen jeder Frage mehrere Aufrufe an die Datenbank abfeuert um die Aggregatswerte des Quiz zu aktualisieren. Nachdem das Quiz erfolgreich angelegt wurde, wird der Student auf dessen Startseite umgeleitet, womit StudentQuiz alle weiteren Aktionen und Schritte an das Quiz Modul übergibt. Wir vermuten dass es schlussendlich die Datenbankabfrage ist, welche aufgrund ihrer ineffizienten Struktur den grösseren Anteil an den Latenzzeiten bei grossen Datenmengen verursacht. Schlussendlich sind sowohl die Schleife wie auch die Abfrage zu überarbeiten.

4.3.4 Aktivität hinzufügen

Wenn ein Dozent seinem Kurs eine neue StudentQuiz Activity hinzufügt, wird als erstes ein Objekt der Klasse `mod_studentquiz_mod_form` geladen, welche die Einstellungen auf Kursmodulebene verwaltet. Sobald diese Einstellungen gesetzt bzw. bestätigt wurden, wird die Funktion `mod_studentquiz_add_instance` aufgerufen. Bemerkenswertes Detail dieser Methode ist die direkte Erstellung der Standardkategorie für die zukünftigen, in diesem Aktivität verwaltenden Fragen. Zudem werden in diesem Moment der Rolle `Student` für den Kontext dieses Moduls Capabilities gesetzt, welche das Hinzufügen und Bearbeiten von Fragen erlauben; allerdings wird nicht überprüft ob diese Berechtigungen denn schon gesetzt sind.

4.3.5 Implementierung durch Vererbung

Bei der Analyse verschiedener in StudentQuiz implementierten Klassen fällt auf, dass mehrfach bereits bestehende Klassen aus der Question Engine abgeleitet und massiv überschrieben wurden. Dies betrifft insbesondere die Implementierung der Klassen `studentquiz_bank_view`. Wir vermuten, dass das ursprüngliche Ziel der gewählten Implementierung die Wiederverwendung bereits implementierter Mechanismen und Datenstrukturen der Question Bank war.

4.3.6 Verschmelzung von Model und View in der Question_Bank_View

Während Moodle auf globaler Ebene und an vielen ähnlichen Stellen, die saubere Trennung zwischen Model und View dahingehend vollzieht, dass lediglich die Renderer sich mit der Ausgabe von HTML basierend auf den vom Model generierten Daten befassen, scheint diese Trennung in der Question Bank nicht durchgezogen worden zu sein. So schreiben die verschiedenen Funktionen, welche die Fragentabelle in HTML ausgeben direkt auf das PHP-Äquivalent zu `stdout`, genannt `echo`, anstatt saubere Rückgabewerte zu liefern, was die Testbarkeit deutlich erschwert.

4.3.7 Backup und Restore Prozeduren

Bei der Analyse der implementierten Backup und Restore Prozeduren überraschte uns, dass keine der spezifisch von StudentQuiz generierten Daten im Falle eines Backups abgespeichert wurden. Die versteckten Quiz Activities wurden zwar durch den Backup und Restore Prozess wiederhergestellt, doch war die Wiederherstellung der Tabelle `studentquiz_practice` nicht implementiert, wodurch die Methoden, welche die Statistik hätten berechnen sollen diese Quiz Activities nicht mehr fanden und so die entsprechenden Summen und Durchschnittswerte auf 0 zurückgesetzt wurden.

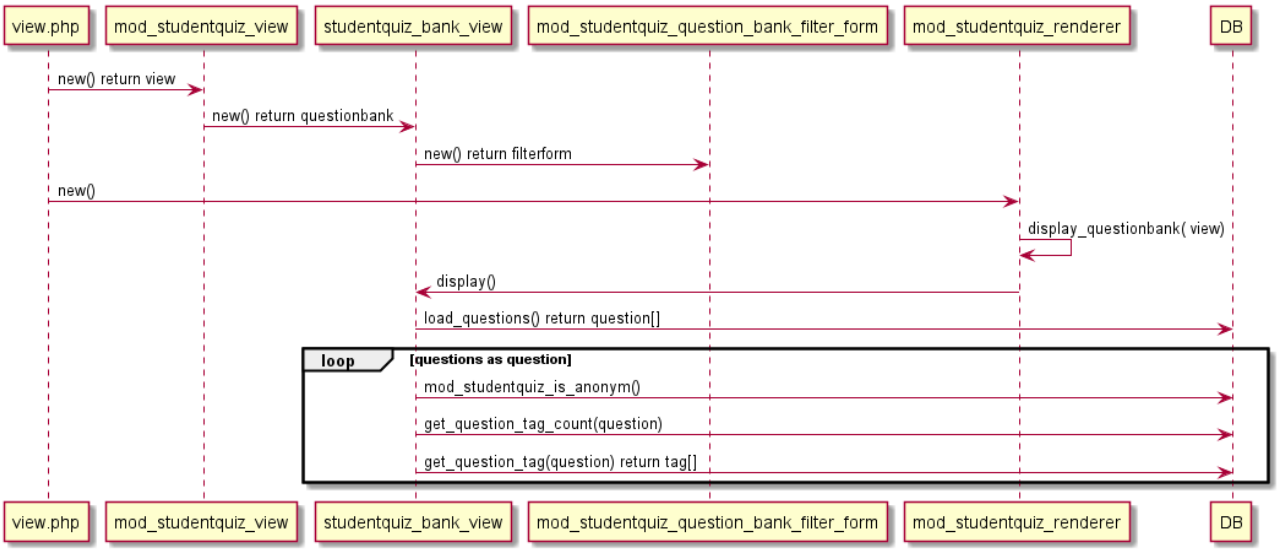


Abbildung 5: Sequenzdiagramm Seitenaufbau Fragentabelle in StudentQuiz 2.0.3

4.3.8 Fragentabelle

Nutzer von StudentQuiz werden auf einer Startseite (Einstiegspunkt view.php) begrüsst, auf welcher eine Tabelle der zur Verfügung stehenden Fragen ausgegeben wird, sowie ein ausgebautes Filterformular, welches die Sortierung und Filterung dieser Tabelle ermöglicht. Ausgehend von dieser Tabelle können zwei primäre Aktionen ausgelöst werden, nämlich das Hinzufügen einer weiteren Frage und das Starten eines Quiz mit den ausgewählten Fragen. Uns interessierte in der Analyse, wie die Liste der Fragen generiert wird. Abbildung 5 skizziert, wie bei einem Seitenaufruf sämtliche in diesem StudentQuiz befindliche Fragen aus der Datenbank geladen werden und anschliessend pro Frage noch einmal mehrere Datenbankabfragen abgefeuert werden, um die den Fragen allfällig gesetzten Tags den geladenen Fragen anzufügen. Erst bei der Ausgabe werden die durch die Pagination nicht benötigten Fragen übersprungen, sodass nur die 20 pro Seite zugelassenen Fragen gesetzt werden können.

4.3.9 Statistik und Rangliste

Eine der Kernfunktionalitäten von StudentQuiz stellt die Darstellung der Statistik dar. Das Sequenzdiagramm in Abbildung 6 visualisiert die gewählte Implementierung zur Berechnung der Rangliste. Es fällt auf, dass die Datenbankabfragen innerhalb der Schleifen abgefeuert werden und so deren Anzahl von der Anzahl der im Kurs eingeschriebenen Benutzern, der Anzahl der bereits generierten Quiz Activities und der Anzahl der Versuche pro Quiz abhängig ist. Hierzu ist anzumerken, dass der Übersichtlichkeit des Sequenzdiagramms halber weitere Datenbankabfragen in jeder Schleife weggelassen wurden. Zudem wird auch an diesem Beispiel deutlich, dass hier keine klare Separation of Concerns angewandt wird.

4.3.10 Problematik Orphaned Section und magische Konstanten

Die durch StudentQuiz generierten Quiz Activities haben zwei Aufgaben. Einerseits werden sie von StudentQuiz benötigt, um die jeweilige Auswahl von Fragen durch die Studenten beantworten zu lassen. Dabei

Abbildung 6: Sequenzdiagramm Illustration Berechnung der Rangliste in StudentQuiz 2.0.3

stehen den Studenten alle vertrauten Funktionen und Ansichten eines Quiz zur Verfügung. Damit diese Quiz Activities durch die eingeschriebenen Studenten genutzt werden können, müssen sie in einem Kurs integriert werden. Ist ein Quiz durch einen Studenten beantwortet, erfüllt es ab diesem Moment lediglich die Funktion einer durch die Statistik auswertbare Datensammlung. Dementsprechend präsentiert sich dadurch auch das in Abbildung 7 dargestellte Domain Model von StudentQuiz 2.0.3.

Diese Datensammlung muss nach wie vor mit dem Kurs verbunden bleiben, soll aber nicht von den Studenten gesehen werden. Um die Quiz Activity zu verstecken, wird in dem Moment, indem dem Kurs eine StudentQuiz Activity hinzugefügt wird, eine versteckte Section am Ende des Kurses angelegt. Diese Section erhielt eine magische Konstante als Ordnungsnummer, deren Wert auf 999 gesetzt wurde, in der Annahme, dass kein Kurs in der Praxis mehr als 998 Sections definieren würde und diese Section so immer sicher am Ende des Kurses befinden würde. Denn obwohl die Section und die darin befindlichen Quiz-Activity den eingeschriebenen Studenten eines Kurses verborgen blieben, wurde sie bei jedem Seitenaufbau der Kursansicht dem Dozenten dargestellt, wie in Abbildung 8 zu sehen ist.

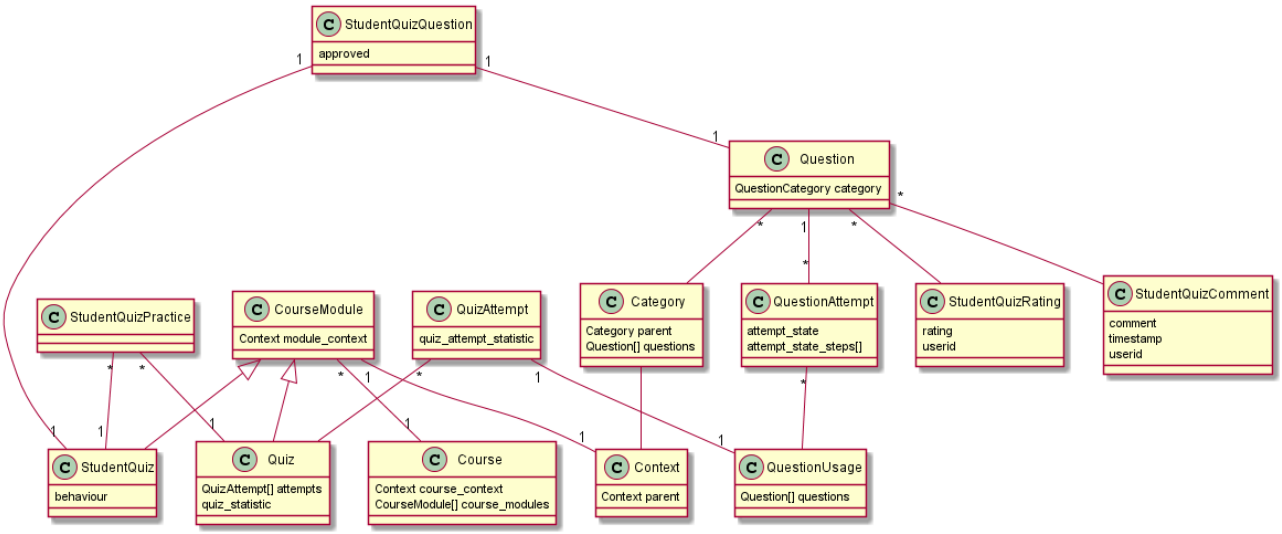


Abbildung 7: Vereinfachtes Domainmodell von StudentQuiz 2.0.3

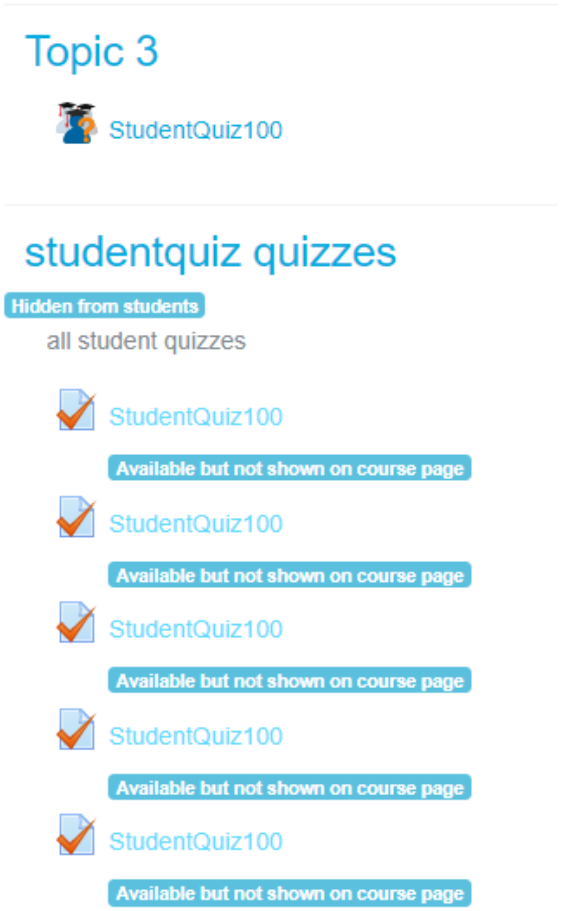


Abbildung 8: Versteckte Quiz-Activities in einer Orphaned Section unter StudentQuiz 2.0.3

4.3.11 Testbarkeit und Continuous Integration

Moodle Plugin Entwickler werden dazu ermutigt Integration Tests mit der Testumgebung Behat zu erstellen und die funktionalen Bestandteile ihres Codes mit Unit Tests abzudecken. Bei Übernahme finden wir bei StudentQuiz einen Behat Test, welcher lediglich sicherstellt, dass StudentQuiz installiert ist, was schon allein die Tatsache beweist, dass dieser Test ausgeführt werden kann. Die vorhandenen 12 Unit Tests decken nur einen Bruchteil der vorhandenen Logik ab.

Für den für Open Source Software frei verfügbaren Continuous Integration Service Travis steht der Moodle Community eine Paketsammlung namens `moodle-plugin-ci` zur Verfügung. Diese Sammlung von Hilfsprogrammen und deren Konfigurationen zur Sicherstellung der Einhaltung von Coding Style-Richtlinien definiert einen minimalen gemeinsamen Qualitätsstandard welcher der Code eines Plugins erfüllen muss. Mittels einer eigenen `.travis.yml`-Konfigurationsdatei wird Travis so konfiguriert, dass der Buildserver eine Moodle-Instanz mit StudentQuiz unter der PHP-Version 5.6 und unter der PHP-Version 7 installiert und ausführt.

Die damit offensichtlich von Moodle zur Verfügung stehenden und geförderten Methoden zur Verbesserung der Codestabilität werden von StudentQuiz kaum genutzt. Es lässt sich aus der Betrachtung des Codes auch schnell erkennen, dass die verschiedenen Klassen und Funktionen stark gekoppelt und in diesem Sinne kaum unabhängig getestet werden können. Zusätzlich verwenden die viele Methoden globale Objekte oder schreiben direkt mit der Funktion `echo` in die HTML Ausgabe, oder erwarten als Aufrufparameter komplexe in einander verschachtelte Objektbäume anstelle von einfachen Werten, was gerade die Testbarkeit für Unit Tests unnötig kompliziert macht.

4.3.12 Fazit

Unsere Einarbeitung in den bestehenden Code förderte einige strukturelle Probleme und aus Sicht der Skalierbarkeit ineffiziente Implementierungen zu Tage. Insbesondere die fehlende Abdeckung durch Unit und Acceptance Tests gestaltet eine Weiterentwicklung des Projekts problematisch, da so während der Entwicklung nicht sichergestellt werden kann, dass die gewünschte Funktionalität auf mit der neu eingeführten Anpassung erhalten bleibt.

4.4 Eigener Lösungsansatz

4.4.1 Lösungsansätze für die Problematik Orphaned Section

Wir unterteilen die Lösung des Problems grundsätzlich in zwei Stufen.

In der ersten Stufe überarbeiten wir die Architektur des Plugins dahingehend, dass die magische Konstante 999 der Vorversion nicht mehr benötigt wird. Unsere erste Lösung führt also ein neues Datenbankfeld ein, welches die Referenz auf diejenige Section enthält, in welche die Quiz Activities gespeichert werden sollen.

Zudem sollen allfällige beim Import wegen unserer magischen Konstanten entstandenen 990+ Sections automatisch entfernt werden. Diese erste Stufe soll so bald wie möglich als Minor Release 2.1.0 veröffentlicht werden, sodass die aktiven Nutzer von StudentQuiz wieder damit arbeiten können.

In der zweiten Stufe möchten wir gänzlich davon wegkommen, auf die Verknüpfung von Quiz Activities in im jeweiligen Kurs angewiesen zu sein, in welchem eine StudentQuiz Activity eingebunden wird.

Während die erste Stufe noch sehr überschaubar und bezüglich der Architektur des Plugins wenig Auswirkungen hat, müssen wir uns bei der zweiten eingehend damit auseinandersetzen, wie wir sowohl möglichst viel der vertrauten Funktionalitäten der Moodle Core Module Question nutzen können, ohne auf die genannte Verknüpfung von Quiz Activities angewiesen zu sein.

Die bisherige Architektur des StudentQuiz benutzt die Infrastruktur des Quiz Moduls nur, um dem Nutzer die Möglichkeit zu geben, alle vom Question Modul angebotenen und gegebenenfalls durch andere Plugins ergänzte Fragetypen zu nutzen. Die Startansicht des StudentQuiz bedient sich der Darstellung der Question Bank um dem Nutzer die verfügbaren Fragen und die durch StudentQuiz berechneten Werte (Anzahl der Kommentare, Schwierigkeit etc.) anzuzeigen.

Die durch das Quiz Modul erhobenen Daten (Bearbeitungsdauer, Grading, Completeness) sind für StudentQuiz nicht relevant, sondern lediglich die Antworten auf die jeweiligen Fragen. In erster Linie möchten wir uns von der Notwendigkeit, versteckte Quiz Activities generieren zu müssen lösen, andererseits lösen wir damit auch eine Abhängigkeit auf, die bei Einführung gar nicht zwingend gewesen wäre.

Wenn sich der Nutzer die gewünschten Fragen durch Filterung und Sortierung und Ergänzung der Question Bank zusammengestellt hat, wird diese Liste von Fragen zur Konfiguration eines versteckten Quiz verwendet und der Nutzer mit einem Redirect auf die Startseite eben dieses Quiz umgeleitet. Von da an übernimmt der Quiz dann die Steuerung bis hin zum Abschluss des Quiz. StudentQuiz implementiert bei diesem Vorgehen keine weiteren Funktionalitäten und hat auch keinen Einfluss auf die Bearbeitung des Quiz. Das separate und optionale Question Behaviour StudentQuiz ermöglicht es, sofern es installiert wurde, die einzelnen Fragen zu bewerten und zu kommentieren, doch hat auch Question Behaviour keinen Einfluss

Um diese Abhängigkeit zu lösen, müssen wir folgende Änderungen in den Abläufen und Abhängigkeiten vornehmen:

- **Start Quiz**: Beim Start eines StudentQuiz soll keine Quiz Activity mehr generiert werden. Die Liste der Fragen wird nicht mehr an ein Quiz übergeben. Wir übernehmen den Vorgang des Quiz Moduls, die Fragen für einen Quiz zu verwalten.
- **Attempt**: Wenn nun nicht mehr das Quiz Modul unsere Durchführung verwaltet, müssen wir selber

einen Controller und die dazugehörigen Ansichten implementieren, die für die Durchführung des Quiz Moduls relevant sind.

- **Auswertung**: Die Auswertung für die beiden Ansichten der Statistik und der Rangliste, sowie die Personal Learning Assistance, kann sich nicht mehr auf verfügbare Quiz Activities berufen, sondern werten lediglich die Fragen aus. Hierbei ist zu beachten, dass die Performance des Seitenaufbaus massiv verbessert werden kann, je mehr dieser Berechnungen nur dann gemacht werden, wenn sich die Werte effektiv ändern.

Aus diesen Überlegungen resultiert das neue Domainmodel für die zweite Stufe, wie es in Abbildung 9 zu sehen ist.

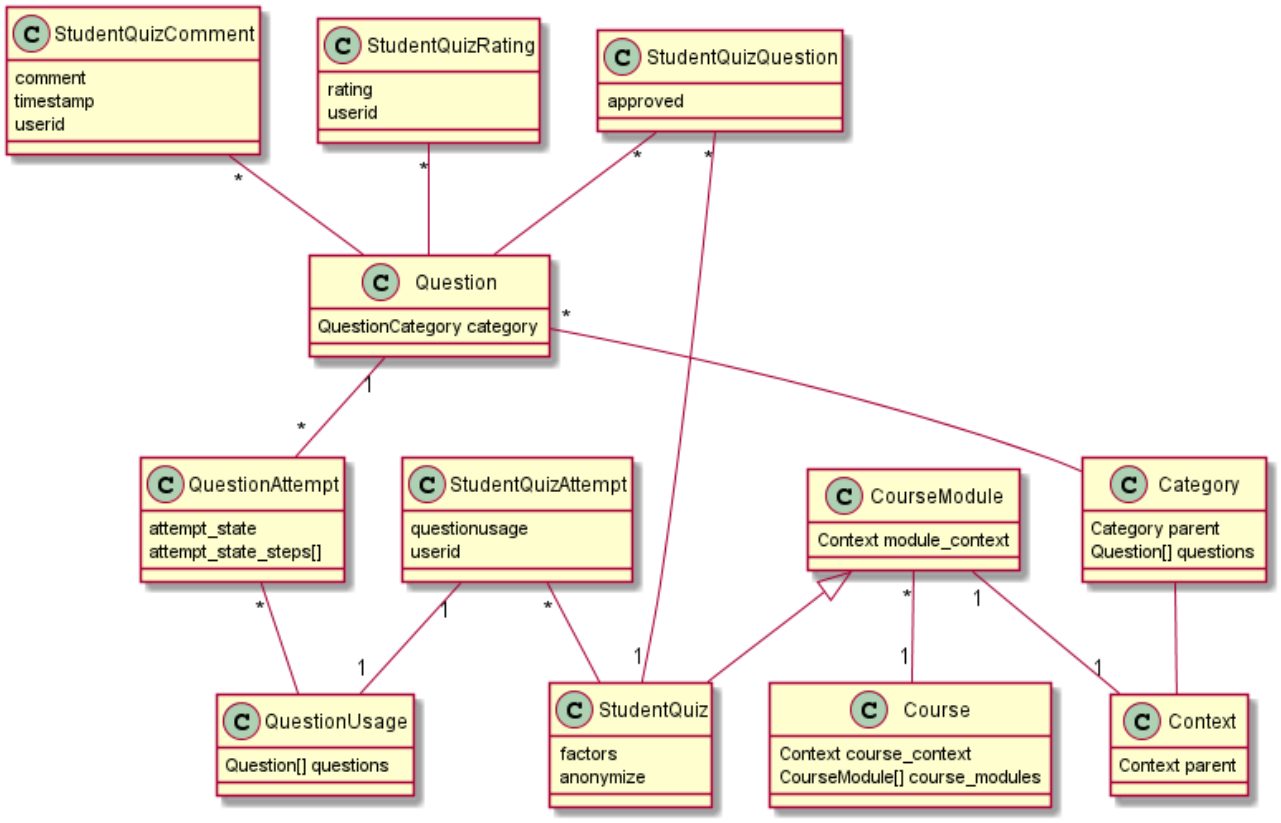


Abbildung 9: Vereinfachtes Domain Model von StudentQuiz 2.0.3

4.4.2 Konzept Personal Learning Assistance

Die Personal Learning Assistance soll es dem Studenten ermöglichen, den eigenen Lernfortschritt bezüglich der im StudentQuiz befindlichen Fragen sichtbar zu machen. Das Bedürfnis für dieses Feature leiten wir aus der Beliebtheit von Lernkarteisystemen wie AnkiWeb,[3] Memrise.[4]

Diese Systeme zeichnen sich dadurch aus, dass sie den Lernenden einerseits anzeigen, welche Fragen und Aufgaben noch nie bearbeitet wurden, sowie wie erfolgreich die bereits bearbeiteten Aufgaben gelöst wurden.

Dabei werden ganz unterschiedliche Kriterien und Abläufe angelegt. So signalisiert bei Memrise[4] jeweils ein kleines Piktogramm die Allegorie einer wachsenden Pflanze, welche bei der ersten Bearbeitung der Frage als Samen gesetzt wird, und bei der fünften erfolgreichen Beantwortung in voller Blüte steht. Fragen welche zwar schon voll geblüht haben, aber schon lange nicht mehr beantworten wurden, werden immer grauer eingefärbt. Eine einzige korrekte Beantwortung lässt sie wieder in Farbe anzeigen.

In der webbasierten Applikation Remembr.it[5] wird insbesondere Wert auf die optimierte Reihenfolge, in der die Karten abgefragt werden gelegt. Karten, welche häufig falsch beantwortet wurden, werden häufiger wiederholt, korrekt beantwortete Fragen werden zu einem vergleichsweise späteren Zeitpunkt wieder abgefragt.

Die mit diesen Applikationen umgesetzten Lerntechniken wurden schon lange vor der Entwicklung des Internets und Smartphones untersucht und optimiert.

Wir möchten diese bewährten Methoden nun auch auf das StudentQuiz in Form der Personal Learning Assistance übertragen. Daaraus ergeben sich neue Use Cases und später Anpassungen im Datenmodell und dem GUI.

Im Hinblick auf die später iterative Umsetzung dieser Personal Learning Assistance unterscheiden wir bereits in der Konzeption zwischen einer Basisversion und der Vollversion.

Basisversion Die Basisversion der Personal Learning Assistance umfasst die folgenden Use Cases:

- **Neue Fragen filtern** Als Student möchte ich mir noch nie beantwortete Fragen anzeigen lassen.
- **Anzahl Beantwortungen, Schwierigkeit** Als Student möchte ich von jeder Frage wissen, wie oft ich diese schon beantwortet habe und welchen Wert die errechnete Schwierigkeit aus der Bewertung meiner Antworten aufweist. Zudem möchte ich die Fragen nach diesem Werten filtern und sortieren können.
- **Status letzte Antwort filtern und sortieren** Als Student möchte ich Fragen danach filtern und sortieren können, ob ich sie noch nie oder beim letzten Versuch korrekt oder falsch beantwortet habe.

Um die mit diesen Informationen verbundenen Daten in der Datenbank zu speichern, müssen wir das Datenmodell um eine Relation zwischen dem Studenten und der jeweiligen Frage erweitern, oder die entsprechenden Werte beim Seitenaufruf berechnen.

In der Basisversion enthält diese Relation die folgenden Werte:

- **last_answer_correct** Bool'sches Feld, welches angibt, ob diese Frage beim jüngsten Versuch korrekt beantwortet wurde. Wurde sie noch nie beantwortet ist das Feld falsch.
- **attempts** Beschreibt die Anzahl der Versuche einer Beantwortung durch den Studenten.
- **correct_attempts** Beschreibt die Anzahl der Versuche bei denen die korrekte Antwort gegeben wurde.
- **difficulty** Funktional von attempts und correct_attempts abgeleitet und die Schwierigkeit der Frage für den jeweiligen Studenten berechnet.

Vollversion Während die Basisversion lediglich die jeweiligen Versuche protokolliert in der Übersicht die Filter- und Sortierbarkeit ermöglicht, unterstützt die Vollversion die individuelle Auseinandersetzung mit den Fragen um folgende Features:

- **Schnellzugriff** Diese Ergänzungen stellen lediglich vordefinierte Filter dar, welche aber über einen spezifischen **Start Quiz** direkt gestartet werden können. Über die Filtereinstellung, ob nur durch den Dozenten akzeptierte Fragen beachtet bzw. in den Schnellzugriff geladen werden sollen, kann das Ergebnis dieser Schnellzugriffe angepasst werden.
  - **Neue Fragen** Startet ein Quiz mit den Fragen, die ich noch nie beantwortet habe.
  - **Random** Startet das Quiz mit zufällig ausgewählten Fragen.
  - **Schwierige Fragen** Startet ein Quiz mit allen Fragen, deren persönlicher Schwierigkeitsgrad einen Schwellwert übersteigt.
  - **Review** Startet ein Quiz mit den Fragen, deren letzte Beantwortung an längsten zurückliegt.
  - **HardForMe** Startet ein Quiz mit den Fragen, bei denen mein persönlicher Schwierigkeitsgrad, denjenigen der Klasse überschreitet.
- **Visualisierung** Das aktuelle GUI der Fragentabelle ist optisch zu verschönern. Ziel soll es sein die verfügbaren Daten durch Farbkodierung und aussagekräftige Icons visuell hervorzuheben und zu gruppieren. So soll mit einem Blick erkennbar sein welche Fragen neu, besonders schwierig, besonders gut bewertet oder noch nicht bestätigt sind.
- **Rangliste auf der Startseite** Die Top10 der Rangliste könnte auf der Startseite angezeigt werden, was die Nutzer zusätzlich motivieren könnte.
- **Persönlicher Fortschritt auf der Startseite** Eine einfach zu lesende grafische Darstellung des gesamthaften Lernfortschritts bezüglich aller verfügbaren Fragen soll auf der Startseite dargestellt werden.
- **Kartenlayout** In einem weiteren Schritt werden die einzelnen Fragen nicht mehr als Zeile einer Tabelle, sondern in Form einer responsiven Liste von Karten dargestellt. Mittels Farben und anderen Gestaltungselementen werden die Daten zusätzlich optisch kodiert. Die Darstellungsform (Tabellen oder Kartenlayout) kann über einen entsprechenden Button umgeschaltet werden. Die Standardeinstellung ist das Kartenlayout.

Abbildung 10 zeigt ein erstes Designkonzept für die aufgewertete Visualisierung der Fragentabelle

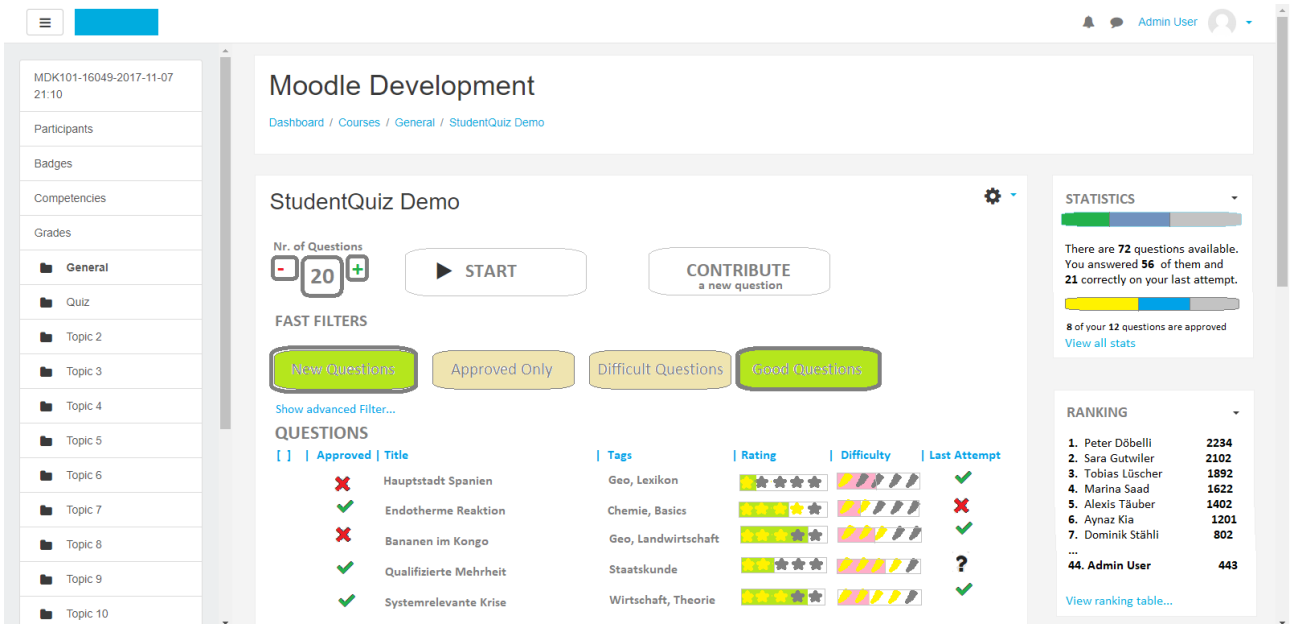


Abbildung 10: Konzeptionsskizze Personal Learning Assistance

In der Vollversion müssen weitere Werte in der Relation der Frage zum Studenten gespeichert werden:

- **last_attempt** Zeitstempel der jüngsten Beantwortung dieser Frage.
- **harder** Funktional abhängiges, bool'sches Feld, welches angibt, ob mein persönlicher Schwierigkeitsgrad grösser ist als der der Klasse ist.

4.4.3 Moodle Mobile Plugin Entwicklung

Wie so vieles in der Entwicklungsgeschichte rund um Moodle, wurden auch bei der Umsetzung einer mobilen Verwendung mehrere Ansätze von der Moodle Community ausprobiert. Die heutige und bis auf weiteres finale Lösung ist eine zentral entwickelte, offizielle Moodle Mobile App basierend auf Angular.

Um nun die Funktionalität unseres StudentQuiz auf dieser Moodle Mobile App zur Verfügung zu stellen müssen wir diese als sogenanntes Remote-Addon entwickeln. Diese werden ab Moodle Version 3.1 unterstützt. Die Implementierung der für Moodle Mobile benötigten Klassen zieht eine von der bestehenden Codebasis unabhängige Implementierung derselben Funktionalität nach sich. Im client-seitigen Teil implementiert ein Remote Addon alle Angular-Componenten und Konfigurationsobjekte, welche von Moodle Mobile mit dynamisch nachgeladen und integriert werden sollen. Für StudentQuiz müsste in jedem Fall Komponenten zur Darstellung der Übersichtsseite mit Fragetabelle implementiert werden. Für den server-seitigen Teil eines Remote Addons müssen die gewünschten Funktionalitäten, wie zum Beispiel die Filterabfragen an die Fragetabelle, oder der Event Start Quiz in sogenannten Webservices implementiert werden. Der server-seitige Teil der Applikation könnte so primär für die Moodle Mobile App aber indirekt auch schon Basis für eine grundlegende Neuentwicklung des StudentQuiz Plugins verwendet werden.

Für die Installation der dazugehörigen Entwicklungsumgebung verweisen wir auf die Moodle Dokumentation

und unsere Ergänzungen in der Anleitung für Entwickler (siehe Abschnitt 5.5 in der Softwaredokumentation).

4.4.4 Backup und Restore Prozeduren

Moodle bietet Administratoren und Dozierenden die Möglichkeit an, sämtliche Inhalte eines Kurses in einem Backup zu speichern. Da das bisherige StudentQuiz essenzielle Daten beim Backup entweder nicht berücksichtigt oder die Relation verloren geht, sind als Erstes die fehlenden Prozeduren zu implementieren, damit der Restore den exakten Zustand wiederherstellen kann.

Für Backups mit dem alten StudentQuiz stellt sich somit die Herausforderung, die verfügbaren Daten im Backup so zu rekonstruieren, dass zumindest alle verfügbaren Daten wiederhergestellt werden können. So sind die Quiz Aktivitäten im Kurs Backup vorhanden, da dieses Quiz Modul seine eigene Backup Prozedur vollständig implementiert hat. Zufälligerweise, mithilfe der magischen Konstanten (siehe Abschnitt 4.3.10) sowie die Wiederverwendung des StudentQuiz Aktivitätsnamen im Quiz Aktivitätsnamen, können die Quiz Aktivitäten gefunden und die Relation wiederhergestellt werden. Fehlende Daten, wie die Bewertungen und Kommentare, können somit nicht rekonstruiert werden.

Moodle bietet auch eine Import Prozedur, welche im Wesentlichen wie folgt zu verstehen ist. Ein Import kopiert einen bestehenden Kurs in einen anderen bestehenden oder neuen Kurs. Hierbei wird Moodle-intern auf die Backup und Restore Prozedur zurückgegriffen, indem es dieses Backup-File in einem Zwischenspeicher erstellt und anschliessend den Restore durchführt. Der Import ist somit automatisch unterstützt, wenn der Backup und die Restore Prozedur implementiert sind.

4.4.5 Upgrade Prozedur

Moodle bietet Plugins eine Upgrade Prozedur an, die insbesondere für die Migration der Datenbank und ihren Daten genutzt werden kann. Während die Installationsprozedur immer den Endzustand der Datenbank beschreibt, müssen für einen Upgrade etwaige Änderungen deklariert werden um diesen Endzustand zu erreichen. Hierfür steht dem Entwickler unter anderem ein Datenbank Manager und ein Snapshot System zur Verfügung. Den einzelnen Upgrade Schritten müssen die zum Updatezeitpunkt zugehörende Plugin Versionsnummer als Snapshot hinterlegt werden. So kann Moodle sicherstellen, welche Schritte erledigt sind und welche noch durchzuführen sind. Damit ist es möglich, im Fehlerfall die Prozedur für diesen Schritt zu korrigieren und zu wiederholen, wie auch mehrere Upgrade Schritte sequenziell durchführen zu lassen. Für diese Arbeit ist die Berücksichtigung der Upgrade Prozedur wichtig, um jeweilige Datenbankmigrationen nachführen zu können ohne den Administrator auffordern zu müssen, das Plugin neu zu installieren. Eine Deinstallation des Plugins hätte Datenverlust zur Folge.

4.4.6 Capabilities

Um die verschiedenen Zugriffsrechte sauber zu verwalten zu können, müssen wie die bisherige Lösung komplett überarbeiten. Wir führen für StudentQuiz spezifische Capabilities ein, welche den Kursteilnehmern gesetzt werden, um deren Berechtigung im Bezug auf StudentQuiz einheitlich zu regeln. Zudem müssen auch für die Einbindung des Notificationsystem weitere Capabilities definiert werden. Beachtet werden muss hierbei, dass die von uns hinzugefügten Capabilities und die zusätzlich im Kontext unseres Kursmoduls

hinzugefügten Capabilities für die Question Bank, das Bearbeiten und Hinzufügen von Fragen erlauben nicht missbraucht werden können.

4.4.7 Lösungsansätze Testbarkeit

Um die Testbarkeit der bestehenden Codebasis zu verbessern schlagen wir vor, eine Reihe von bewährten Best Practices und Design Prinzipien der Objektorientierten Softwareprogrammierung anzuwenden.

- **Layers** In einem ersten Schritt sollte eine Gliederung umgesetzt werden, welche konsequent angewandt eine klare Struktur schafft um zum Beispiel die Abfragen an die Datenbank in eine Persistenzschicht zu gruppieren und andererseits die Businesslogik vom Rendering zu trennen.
- **MVC** Das Model-View-Controller Pattern wird offensichtlich schon beim Quiz Modul angewendet. Überhaupt ist dieses sehr zugänglich, da dort eine klare Struktur herrscht.
- **Separation of Concerns** Die implementierten Klassen wirken überladen haben viele Klassenvariablen, was die Testbarkeit beeinträchtigt. So muss nicht eine einzige Rendererklasse die Rendermethoden sämtlicher Elemente implementieren, sondern kann mit einer Vererbungshierarchie gruppiert werden, was sowohl die Leserlichkeit als auch die Testbarkeit der einzelnen Klassen verbessert.
- **DRY** Dont-Repeat-Yourself. Es lohnt sich, dieselbe Methode nicht an mehreren Orten sondern an genau einem sinnvollen Ort zu implementieren. So wird bei StudentQuiz 2.0.3 die Abfrage nach dem aktuellen Kursmodul an mehreren Orten aufgerufen, anstatt nach einem Aufruf pro Seitenaufbau an geeigneter Stelle eine Referenz zu verwalten.
- **Innere Zustände vermeiden** Je weniger zustandsbehaftete Klassen sich in einem Projekt befinden, um so einfacher lässt sich die Testbarkeit erhöhen.
- **Keine globalen Objekte** Moodle definiert die globalen Objekte USER, DB, CFG. Diese stehen auch in einer Unit Test Umgebung zur Verfügung, doch sind Methoden mit globalen Objekten inhärent schlechter testbar.
- **Generatoren Implementieren** Um für Unit Tests schnell einfache Szenarien testen zu können, werden vernünftige Generatoren benötigt. Insbesondere für die Versuchsdaten.

4.5 Architekturentscheide für StudentQuiz 3.0.0

4.5.1 Strukturierung des Backups

Da vorhergehende StudentQuiz Versionen keine Backup-Funktionalität implementierten, war es an uns zu entscheiden, wie StudentQuiz Backups bis auf Weiteres strukturiert sein sollen. Ausgehend von unserem Domain Model (siehe Abschnitt 5.1.1) ergeben sich zwei grundsätzliche Ansätze, die von StudentQuiz erfassten relationalen Daten in die im Backup zur Verfügung stehende Struktur einer XML Datei abzulegen.

Ansatz: Hierarchisch Bei diesem Ansatz wandeln wir die 1:n-Relationen zwischen Kommentar, Bestätigung, Bewertung, Fortschrittsdaten und Frage in Eltern-Kind-Beziehungen um, sodass diese Daten, sofern vorhanden als Kind Elemente einer Frage abgelegt werden. Der Vorteil dieser Struktur wäre aus unserer Sicht, dass alle mit einer Frage verknüpften Elemente innerhalb des Backups beieinanderliegen.

Ansatz: Horizontal Bei diesem Ansatz ignorieren wir unser aktuelles Wissen über die Beziehungen zwischen den verschiedenen Relationen und speichern die Inhalte der Tabellen nacheinander als Kinder des xml-Root Elements ab. Als Vorteil dieser Struktur identifizieren wir eine höhere Flexibilität, falls gewisse Daten in Zukunft nicht mehr wiederhergestellt werden sollen, oder neue Daten hinzugefügt werden sollen.

Entscheidung Schlussendlich haben wir uns für den horizontalen Ansatz entschieden. Ausschlaggebend war für uns die Tatsache, dass diese Struktur auch nach zukünftigen Änderungen des Datenmodells einfacher zu verwalten ist, als beim hierarchischen Ansatz. Wohl auch aus der konkreten Erfahrung aus der Programmierung der Migrationsskripte war uns der Wert einer einfachen Datenstruktur wichtiger als die Eleganz einer mehrschichtigen Hierarchie.

4.5.2 Reportlib

Unter diesem Titel beschreiben wir die verschiedenen Auswirkungen der Umbauten der Pluginarchitektur auf die Berechnung der statistischen Daten. Während wir unsere Implementierung begründen, besprechen wir im Abschnitt 5.6.2 die anschliessend gemessenen Kennwerte und Messdaten.

Eine Vielzahl von Aggregierten Werten Betrachtet man StudentQuiz aus mit etwas Abstand, so werden alle Daten die für die Personal Learning Assistance und die Gesamtstatistik der Community berechneten Werte Aggregate. Die vollständige Liste aller von StudentQuiz berechneten Aggregate und die Aktivitäten welche diese verändern können, findet sich in der Softwaredokumentation im Abschnitt 5.1.3. Die korrekte und performante Berechnung dieser Werte stellt also eine Kernfunktionalität des Plugins dar.

Die Schwierigkeit entsteht darin, aus dem komplexen Zustand einer Frage innerhalb einer Question Usage abzuleiten, ob diese nun als korrekt oder inkorrekt beantworteter Versuch oder überhaupt nicht gezählt werden soll. Andererseits werden die aggregierten Werte vom jederzeit möglichen Hinzufügen und entfernen von Fragen und der ebenfalls jederzeit möglichen Einschreibung von weiteren Studenten bzw. deren Austritt aus dem Kurs beeinflusst.

Ziele der Optimierung Unsere Optimierung hatte zwei Ziele: Zum einen die Anzahl der benötigten Datenbankabfragen von der Anzahl der eingeschriebenen Studenten und der Anzahl der Fragen gänzlich unabhängig zu machen. Wir mussten einen Weg finden, alle benötigten Werte in einer konstanten Anzahl von Abfragen zu ermitteln. Zum anderen der Dynamik der sich sich ändernden Daten Rechnung zu tragen.

Ad hoc Aggregation vs. Redundanz An dieser Stelle mussten wir einen Grundsatzentscheid fällen, ob unsere zukünftige Lösung die gewünschten Aggregatswerte jeweils anhand des jeweiligen Datensatzes von Fragen, Question Usages und der Menge von eingeschriebenen Benutzern pro Seitenaufbau berechnet oder, ob die Aggregate in entsprechenden Datenbankfeldern zwischengespeichert und bei jeder der oben beschriebenen Aktionen beziehungsweise einem regelmässigen Synchronisierungstask aktualisiert werden.

Folgende Argumente sprachen für die Einführung von Aggregatswerten

- Die Aggregate werden häufiger gelesen als geschrieben.
- Die nur über mehrfache Join berechenbare Aggregate (Anzahl und Klassifizierung der Antworten) können direkt bei der Beantwortung einer Frage aktualisiert werden und müssen nicht mehr neu abgefragt werden.
- Unveränderte Aggregatswerte müssen nicht bei jedem Seitenaufbau neu berechnet werden.

Die Speicherung aller benötigten Aggregatswerte, würde die Einführung oder Ergänzung der folgenden Relationen notwendig machen:

- **StudentQuiz-Student-Frage**: Letzte Antwort, Anzahl Antworten, Anzahl korrekter Antworten, Gegebenes Rating
- **StudentQuiz-Frage**: Anzahl Antworten, Anzahl korrekter Antworten, Durchschnittlich erhaltenes Rating, Approved
- **StudentQuiz-Student**: Personal Progress, Punkte
- **StudentQuiz** Anzahl verfügbare Fragen, Anzahl gegebener Antworten, Anzahl eingeschriebener Studenten, Anzahl Frageautoren

Folgende Argumente sprachen für eine ad hoc Berechnung:

- Zwischengespeicherte Aggregate führen zusätzliche und nicht notwendige Redundanz ein, welche inkonsistenzen verursachen kann.
- Gerade die Aktionen, welche den grössten Einfluss auf die Aggregatswerte haben (Einschreiben eines Benutzers, Entfernen einer Frage, können nicht direkt in von StudentQuiz übersteuert und zur Aktualisierung der Aggregatswerte genutzt werden.).
- Die Verifikation einer ad hoc Berechnung kann mit einem einfachen Soll-Ist Vergleich durchgeführt werden.

- Diese Aggregate könnten datenbankseitig elegant mit Views und Stored Procedures umgesetzt werden. Leider stellt die von Moodle verwendete Datenbankabstraktion keine entsprechenden Möglichkeiten zur Verfügung um die möglichst generelle Portabilität auf die verschiedenen unterstützten Datenbanksysteme zu gewährleisten.
- Sofern die Aktualisierung der Aggregatswerte direkt nach dem Beantworten/Kommentieren/Bewerten einer Frage stattfinden soll, müssen bei jeder dieser Aktionen mehrere Datenbankabfragen und -aktualisierungen durchgeführt werden, da bei jeder dieser Aktionen in jeder der oben beschriebenen Relationen sich mindestens ein Wert ändert, was den Seitenaufbau während dem Üben ausbremst.

Folgende Argumente/Kriterien waren nicht anwendbar:

- **Anzahl Datenbankaufrufe** Beide Varianten benötigen dieselbe Anzahl von Datenbankabfragen um den Seitenaufbau der Startseite zu laden.

Die Entscheidung fiel uns nicht leicht. Wir haben uns schlussendlich für die Ad-hoc-Berechnung entscheiden, da wir das Risiko, die zusätzliche Komplexität und Inkonsistenzen zwischen dem eigentlichen Datenstand und den berechneten Aggregaten nicht mehr in der zur Verfügung stehenden Zeit umzusetzen nicht in Kauf nehmen wollten. Zudem waren wir zuversichtlich, dass wir die Abfragen so optimieren konnten, dass sie die Stärken des Datenbanksystems nutzen können, sodass die geforderten nicht funktionalen Anforderungen erreicht werden können. An dieser Stelle möchten wir aber darauf hinweisen, dass die ad hoc Berechnung bei besonders grossen Datensätzen und häufigen Readonly-Aufrufen gegenüber der Aggregatswerten massiv mehr Leistung von einer Datenbank verlangt. Wir empfehlen für die zukünftige Weiterentwicklung von StudentQuiz diesen Weg einzuschlagen, falls der Bedarf nach weiterer Optimierung besteht.

4.6 Implementierung

Unseren ersten Schritte während der Einarbeitung beinhalteten die Einrichtung einer verlässlichen Deploymentumgebung auf dem HSR-Server. Diese diente auch als gleich als Demonstrationsplattform für unseren Austausch mit unserem Betreuer. Ziel war es, mittels der Moodle Development Kits (MDK) beliebige Moodle Versionen und StudentQuiz Versionen installieren zu können, ohne dies über zeitraubende Installation über manuelle Uploads oder oder gar das Webinterface vornehmen zu müssen. Parallel dazu setzten wir unsere lokale Entwicklungsumgebung auf. Die dafür nötigen Schritte sind nun in der Softwaredokumentation im Abschnitt 5.5 beschrieben und stehen zukünftigen Projekten als Anschauungsbeispiel zur Verfügung.

4.6.1 Verwendete Software und Dienste

Für dieses Projekt sind für die unterschiedliche Aufgaben die folgenden Dienste und Software im Einsatz:

Anwendung	Software	Kurze Beschreibung
Dokumentation	MikTex	Open-Source LaTeX Bibliothek für Windows
Dokumentation	TexLive	Open-Source LaTeX Bibliothek für Linux
Dokumentation	Kile	Open-Source LaTeX Editor für Linux
Dokumentation	TexStudio	Open-Source LaTeX Editor für Windows
Dokumentation	Draw.io	Proprietärer Online Diagramm Editor & als Chrome App
Generell	Git	Open-Source Versionsverwaltungssystem
Generell	Github	Proprietäres Online Git Repository Management
Konfiguration	Ubuntu	Open-Source Linux Betriebssystem
Container	Docker	Virtualisierungssoftware auf Containerebene
Testing	Travis-CI	Open-Source Automation Server für Continuous Integration
Testing	jMeter	Java basierte Applikation zur Durchführung von Belastungstests
Testing	Selenium	Software zur Durchführung von automatisierten Browser Tests
Datenbank	PostgreSQL	Von Moodle unterstützte Datenbankumgebung
Webserver	XAMPP	Lokaler Apache Stack für die lokale Installation von Moodle
IDE	PhpStorm	Integrierte Entwicklungsumgebung für PHP von JetBrains
Debugging	xDebug	Debugging von PHP Applikationen
Grafik	PlantUML	Tool zur Darstellung von UML- und Sequenzdiagrammen
Projektdokumentation	Redmine	Bewährte Projektverwaltungssoftware
Kommunikation	Skype	Software für Videokonferenzen von Microsoft
Kommunikation	Telegram	Messengerdienst für kurzfristige Absprachen

Tabelle 1: Liste verwendeter Dienste und Software in dieser Arbeit

4.7 Berechnung der Werte für die Personal Learning Assistance

Wie in Abschnitt 4.5.2 beschrieben, haben wir uns dafür entschieden, die Berechnung der statistischen Werte weiterhin pro Seitenaufruf durchzuführen. Somit müssen auch die neuen aus dem Konzept der Personal Learning Assistance (siehe Abschnitt 4.4.2) möglichst effizient abgefragt werden. Wir illustrieren unsere Lösung anhand unserer Ergänzungen für den Wert My Difficulty in der Fragentabelle auf der Startseite.

Die Berechnung der anderen neuen Werte folgte demselben Design. Die Spezifikation der Berechnung kann in der Software Dokumentation 5 nachgelesen werden. Die Fragentabelle wird mit einer einzigen Abfrage berechnet, welche sowohl sämtliche durch den Filter definierte Kriterien, wie auch die gewählte Sortierung berücksichtigt. Unsere Lösung implementiert ein zusätzliches durch die Identifikationsnummer des betreffenden Studenten und der betreffenden StudentQuiz parametrisierten Aktivität, `JOIN` Statement, welches die Anzahl seiner Antworten dieser Frage, und die davon korrekten zählt. An dieser Stelle wird das komplexe Datenmodell der Question Engine zur Modellierung von Fragen und deren Beantwortung sichtbar. Die Beantwortung wird durch einen `question_attempt` modelliert. Dieser implementiert eine State Machine, welche sich zu jeder Zeit in einem von dreizehn verschiedenen Zuständen befindet und deren Übergänge durch sieben verschiedene `question_attempt_steps` ausgelöst werden können. Je nach Fragetyp kann ein einzelner`question_attempt_step` weitere `question_attempt_step_data` Einträge oder sogar zusätzliche `question_attempt_steps` generieren. Für unsere Zwecke interessieren uns nur die bewerteten Zustände und diejenigen Übergänge, welche durch einen Klick des Benutzers ausgelöst wurden. Zudem muss der Join so geschrieben werden, dass er nur nur einmal für die gesamte Tabelle berechnet werden muss und somit durch die Datenbank optimiert werden kann. Listing 2 stellt den resultierenden `LEFT JOIN` dar.

```
1  LEFT JOIN (
2  SELECT
3      ROUND(1-avg(case state when 'gradedright' then 1 else 0 end),2) as mydifficulty,
4      sum(case state when 'gradedright' then 1 else 0 end) as mycorrectattempts,
5      questionid
6  FROM {studentquiz_attempt} sqa
7      JOIN {question_usages} qu ON qu.id = sqa.questionusageid
8      JOIN {question_attempts} qa ON qa.questionusageid = qu.id
9      JOIN {question_attempt\_steps} qas ON qas.questionattemptid = qa.id
10 LEFT JOIN {question_attempt_step_data} qasd ON qasd.attemptstepid = qas.id
11 WHERE sqa.userid = :currentuserid
12 AND sqa.studentquizid = :studentquizid
13 AND qasd.name='-submit'
14 AND (qas.state = 'gradedright' OR qas.state = 'gradedwrong' OR qas.state='gradedpartial')
15 GROUP BY qa.questionid
16 ) mydiffs ON mydiffs.questionid = q.id
```

Listing 2: Zusätzlicher LEFT JOIN zur Berechnung von My Difficulty

4.7.1 Vektorgrafiken

Um den persönlichen Fortschritt, die Schwierigkeit einer Frage oder den Durchschnitt der dafür eingegangenen Bewertungen zu visualisieren implementierten wir diese als Vektorgrafiken im SVG Format. Alternative Formate sind JPG und PNG. Die Vorteile von Vektorgrafiken im Gegensatz zu anderen Formaten ist die Tatsache, dass Dimensionen, Farbwerte, Liniendicke und weitere Merkmale direkt in der Definition der Vektorgrafiken angepasst werden können. Dies ist auch noch im Browser mittels Javascript möglich, wodurch interaktive Elemente gebaut werden können. Zudem können sie von jedem Browser beliebig präzise berechnet werden sodass sie in der optimalen Auflösung erscheinen. Der einzige Nachteil ist wohl, dass ein Browser je nach Komplexität oder Anzahl der Vektorelemente mehr Rechenzeit dafür verwenden muss. Doch die von uns benötigten Grafiken sind weniger als 7 Elementen pro Stück sehr überschaubar. Abbildung 11 zeigt den neu eingeführten Block zur Anzeige des persönlichen Lernfortschritts auf der Startseite, auf welchem die ausgegebenen Werte durch eine Vektorgrafik in Form eines Ladebalkens visualisiert werden.

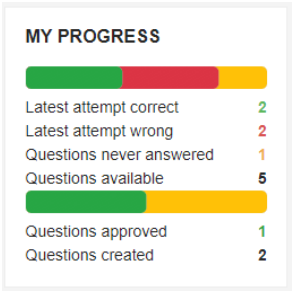


Abbildung 11: Der persönliche Lernfortschritt wird mit einer Vektorgrafik visualisiert

4.7.2 Spalten der Fragentabelle

Die auf der Startseite angezeigte Fragentabelle wurde von unseren Vorgängern aus der Implementierung der Question Bank kopiert und an verschiedenen Stellen ergänzt oder überschrieben. Zusätzlich zu den von der Question Bank ausgegebenen Spalten, wurden fünf zusätzliche Spalten hinzugefügt, welche die Auswertungen der Übungsdaten und Bewertungen sowie den Bestätigungsstatus anzeigten. Abbildung 12 illustriert die somit neun Spalten bei der Fragentabelle unter StudentQuiz 2.0.3. Nun sollten mit der Umsetzung der Personal Learning Assistance mindestens drei weitere Werte pro Fragen verfügbar werden. Es musste eine Lösung gefunden werden, wie die Zahl der Spalten nicht weiter erhöht und gleichzeitig weiterhin nach allen ausgegebenen Werten sortiert werden kann.

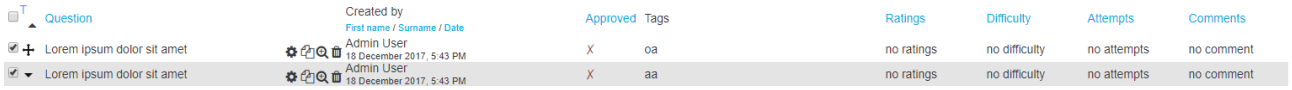


Abbildung 12: Spalten der Fragentabelle auf der Startseite in StudentQuiz 2.0.3

Unsere Lösung besteht darin, dass wir die verschiedenen Spalten kombinierten und mit Vektorgrafiken (siehe Abschnitts 4.7.1) optisch darstellten. Die Sortierung der Werte wurde weiterhin gewährleistet, dass mehrere dem Spaltentitel untergeordnete Untertitel nun das Sortieren nach einzelnen Werten erlaubt. Wie in Abbildung 13 zu sehen ist, hat die resultierende Tabelle nun insgesamt 9 Spalten. Obwohl drei weitere Werte dazugekommen sind wirkt sie aufgeräumter und attraktiver.

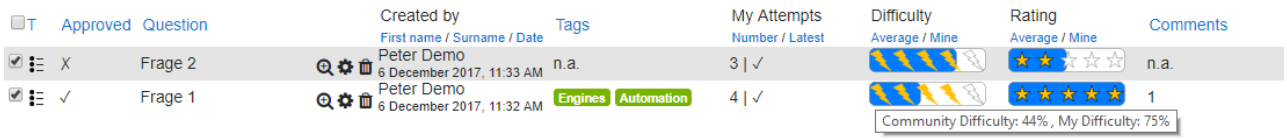


Abbildung 13: Neue kompakte Spalten der Fragentabelle

4.7.3 Erweiterung Travis CI

Bekanntlich dürfen Open Source Projekte die automatisierte Continuous Integration Test Infrastruktur von Travis-CI kostenfrei nutzen. Dazu genügt eine im Hauptordner des Quellcodes abgelegte Konfigurationsdatei, welche die Spezifikation der gewünschten Testumgebung und die Befehle zur Ausführung beliebiger Testskripte beinhaltet. Von der Moodle Community wird dazu ein Python-Script unter dem Namen

`moodle-plugin-ci` angeboten, welches die den Moodle Coding Style Guides entsprechenden Konfigurationen für diverse statischen Codeanalysetools enthält. Mittels Travis-CI kann dieses Skript problemlos auf den Code eines Plugins losgelassen werden. Auch StudentQuiz nutzt seit Herbst 2016 diese Dienstleistung. Die damals definierte Testumgebung prüfte die insgesamt vier verschiedenen Konstellationen von PHP 5.6 und 7.0 und den beiden Datenbanken PostgreSQL und MySQLi mit einer Moodle Instanz in der Version 31. Wir erweiterten diese auf insgesamt vierzehn verschiedene Installationen um zusätzlich die Moodle Versionen 3.2, 3.3, 3.4 und auch PHP 7.1 automatisiert abzudecken. Dank weiteren Anpassungen ist es nun auch möglich Integrationstests mit Selenium laufen zu lassen, was insbesondere für Behat-Tests relevant ist.

4.7.4 Backup, Restore und Upgrade

Wie in der Analyse in Abschnitt 4.3.7 berichtet, ist beim Restore eines Backups von einem Kurs mit altem StudentQuiz eine Rekonstruktion von Daten nötig. Durch die Umstellung auf die neue Datenstruktur aufgrund des Lösungsansatzes für die Orphaned Section 4.4.1 ist zeitgleich auch eine Migration der Daten notwendig. Des weiteren musste die Zwischenlösung von StudentQuiz v2.1.0 ebenfalls berücksichtigt werden, da die Orphaned Section dort nicht mehr die Section-Nummer 999 aufweist, sondern eine beliebige Section sein kann. Kurz vor dem Release der neuen Version ist uns durch das zuspielen eines alten Kurs-Backups aufgefallen, dass wohl vor StudentQuiz v2.0.3 die Quiz Activities mit einem Suffix versehen waren.

Nachfolgend der Überblick der StudentQuiz Versionen mit Ihren Eigenschaften bezüglich Backup, Restore und Upgrade:

- Allgemein
  - Es können mehrere StudentQuizzes in einem Kurs vorkommen
  - Es können mehrere Quizzes in einem Kurs vorkommen, die auch nicht von StudentQuiz erstellt wurden
  - Es können auch keine dieser Module im Kurs vorkommen
- StudentQuiz v2.0.3
  - Orphaned Section ist Section-Nummer 999, Section-Name ist “studentquiz quizzes” (magische Konstante)
  - StudentQuiz verwaltet Quiz-Aktivitäten für die Durchführung der Fragen
  - Quiz-Aktivitäten verwalten Beantwortung der Fragen
  - Quiz-Aktivitäten erhalten denselben Namen wie die von der StudentQuiz Aktivität
  - Enthält ab Moodle 3.3 leere und unbenutzte Sections, falls dieser Kurs importiert wird oder importiert worden ist
  - Bei einem Import geht die Relation zu den Quizzes verloren
  - Ein Quiz kann von mehreren Benutzern verwendet worden sein, da die Quiz Aktivitäten wiederverwendet werden
- StudentQuiz < v2.0.3
  - Wie StudentQuiz v2.0.3, jedoch ergänzend:

  - Quiz-Aktivitäten enthalten in ihrem Namen auch das Suffix des Studenten

- StudentQuiz v2.1.0
  - Wie StudentQuiz v2.0.3, jedoch ergänzend:
  - Orphaned Section kann irgend eine Section Nummer sein, da der Dozent diese in den StudentQuiz Einstellungen verschieben kann
  - Beim Import werden die leeren, unbenutzten Sections aufgeräumt
  - Beim Upgrade auf diese Version bleiben die Sections wie sie waren
- StudentQuiz v3.0.0
  - Import Prozedur ist zu definieren (siehe unten)
  - Upgrade Prozedur ist zu definieren (siehe unten)
  - Verwendet die Question-Engine direkt, keine Abhängigkeit auf Quiz mehr

Die Aufgabe, die beim Restore implementiert werden soll, ist dem beim Upgrade ähnlich: Der Restore importiert einen bestimmten Kurs, beim Upgrade bedarf es die Berücksichtigung aller Kurse. Die Migration kann somit für beide implementiert werden, wenn beiden Fällen die gleiche Datenbestand-Ausgangslage vorliegt und so nur in der Anzahl der Kurse variieren würde. Glücklicherweise bietet die Moodle Backup Schnittstelle eine Prozedur an, den Restore Prozess ganz am Schluss des Restores zu ergänzen, somit ist die gleiche Datenbestands-Ausgangslage gegeben.

Die Import Prozedur ist somit:

- Am Schluss eines Restores
- Führe die Migration-Prozedur im importierten Kurs aus (siehe unten)

Und beim Upgrade ist die Prozedur:

- Für jeden Kurs mit einem StudentQuiz
- Führe die Migration-Prozedur für jeden gefundenen Kurs aus (siehe unten)

Die Migrations-Prozedur ist in folgende Schritte aufgebaut (alle Aktionen hier implizieren immer die Bedingung, dass die Werte dem aktuellen Kurs angehören):

- Existiert eine Orphaned Section?
  - Indem eine Section mit den Namen “studentquiz quizzes” existiert
  - Die Section kann irgend eine Nummer sein
- Kommentar: Dann beginne mit Abschnitt Migration
- Suche alle StudentQuiz Activities in diesem Kurs
- Für jede StudentQuiz Activity suche die zugehörigen Quiz Activities

  - Indem der Quiz Name mindestens dem Namen dieses StudentQuizzes beginnt
- Für jeden Quiz Activity suche die zugehörigen Question Usages
  - Hierbei pro Question Usage und Benutzer Kombination ein Resultat
- Verschiebe jede Question Usage pro Benutzer zum StudentQuiz
- Kommentar: Abschnitt Migration abgeschlossen
- Kommentar: Beginne mit Abschnitt Cleanup
- Finde die letzte, ordentlich benutzte, Section
  - Welche nicht die Orphaned Section von oben ist
  - Welche einen individuellen Namen oder Beschreibung besitzt
  - Oder Activities beinhaltet
- Und lösche alle Sections, deren Section Nummer grösser ist als diese
- Kommentar: Abschnitt Cleanup abgeschlossen

Diese Prozess funktioniert somit immer für alle Eigenschaften der bisherigen StudentQuiz Versionen, wenn der Administrator oder der Dozent diese essenzielle Information nicht angepasst hat: Section Name der Orphaned Section. Die beiden Abschnitte Migration und Cleanup funktionieren unabhängig von einander, beide erfordern lediglich das existieren einer Orphaned Section. Somit kann ein Cleanup auch durchgeführt werden, ohne dass ein StudentQuiz existiert.

Dieser Migrationsprozess hinterlässt zum Zeitpunkt Datenleichen: Die alten Quiz Activities, da ihre Section durch den Cleanup gelöscht wurde. Da die Prozedur während eines Upgrades stattfinden kann, ist mit einer hohen Anzahl Kursen zu rechnen und somit den Migrationsaufwand möglichst auf ein Minimum zu halten.

Mithilfe den Scheduled Tasks von Moodle, welche über den Moodle Cronjob ausgeführt werden, werden Schritt für Schritt die alten Quiz Aktivitäten entfernt, ohne die Systemperformance negativ zu beeinflussen.

4.8 Acceptance Tests

Wir verweisen auf die im Anhang C.2 beigelegten manuellen Acceptance Test Instruktionen, welche in der vorausgehenden Studienarbeit 2.0.3 definiert wurden, ebenfalls als Grundlage dienten, die grundlegenden Funktionen von StudentQuiz manuell zu testen. Ausgehend von diesen Acceptance Tests und anhand der Aufgabenstellung testete Prof. Frank Koch zu Beginn der Transition in seiner Rolle als Lead Maintainer StudentQuiz 3.0.0. Dabei meldete er uns rund 50 Beobachtungen von Mängeln und Anpassungswünschen. Diese wurden so schnell wie möglich umgesetzt. Im Anhang C.3 findet sich eine kommentierte Zusammenfassung dieser Beobachtungen.

4.9 Resultate und Weiterentwicklung

Die mit der Aufgabenstellung erhaltene Todo Liste und der Bug Report sind abgearbeitet. Anstatt sämtliche der dort aufgeführten Punkte durchzugehen, fassen wir die wichtigsten Ergebnisse hier zusammen.

4.9.1 Erstelltes Produkt

Zwei veröffentlichte Versionen

Am 8. November 2017 veröffentlichten wir einen Minor Release v2.1.0 unter dem Titel **Orphaned Section Fix**. Gemäss offizieller Downloadstatistik wurde dieser im November bereits auf 121 Moodle Instanzen installiert. Am 11. Dezember 2017 veröffentlichten wir den Major Release v3.0.0 unter dem Titel **StudentQuiz 3.0.0**. Die Downloadstatistik zum Dezember ist ab Januar 2018 verfügbar. StudentQuiz 3.0.0 wurde ebenfalls am 11. Dezember 2017 erfolgreich auf dem HSR Server installiert.

Question Behaviour StudentQuiz obsolet

StudentQuiz 3.0.0 integriert die Bewertungs- und Kommentarfunktionen des bis anhin separaten Question Behaviour Plugins. Es kann mittelfristig aus dem Plugin Directory gelöscht werden. Wir empfehlen beim Release von Moodle 3.6 im November 2018 zu tun. Entsprechende Hinweise haben wir im Moodle Plugin Directory vermerkt.

Keine Quiz-Activities mehr

StudentQuiz 3.0.0 kommt komplett ohne Abhängigkeiten auf Quiz-Activities aus. Damit lösten wir sowohl einige Effizienzprobleme und eliminierten die Notwendigkeit eine versteckte Section anzulegen.

Performance

Die Performance von StudentQuiz wurde deutlich verbessert. Die in der Aufgabenstellung definierte Nicht funktionale Anforderung von möglichen 50 Zugriffen pro Minute bei 500 eingeschriebenen Studenten und 5'000 Fragen wurde übertroffen, wie auch Abbildungen 14 und 15 illustrieren und die Belastungstests in Abschnitt 5.6.2 ausführlich beweisen.

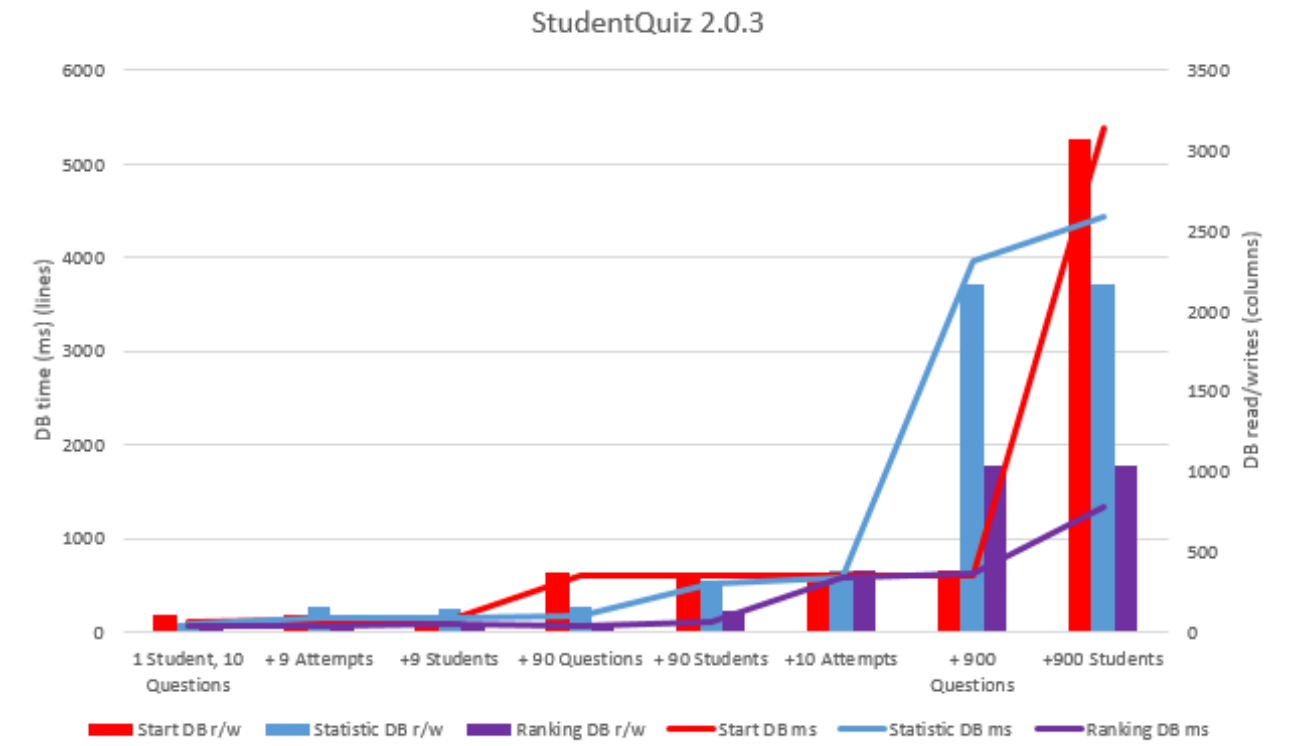


Abbildung 14: Manueller Belastungstest StudentQuiz 2.0.3 zeigt problematische Skaleneffekte

Personal Learning Assistance

Die neue Funktionalität der Personal Learning Assistance ist mit StudentQuiz 3.0.0 veröffentlicht. Ein Student kann nun jederzeit seinen persönlichen Lernfortschritt mit dem der Community vergleichen und sieht bei jeder Frage nicht nur die durchschnittlichen Werte von Schwierigkeit und Bewertung, sondern auch seine eigenen. Die neuen Schnellfilter wie sie in Abbildung 16 zu sehen sind, erlaubten eine zügige Auswahl der gewünschten Fragen.

Auf der überarbeiteten Startseite (siehe Abbildung 17) kommt die Fragetabelle nun mit weniger Seitenbreite aus, was die Darstellung von zusätzlichen Daten (Rangliste, Persönlicher Fortschritt) in seitlichen Blöcken erlaubt. Die Schwierigkeit und die Bewertung einer Frage werden nun durch Grafiken dargestellt, sodass besonders schwierige oder und besonders gute Fragen auch in einer langen Frageliste optisch auffallen. Die Statistik und die Rangliste wirken aufgeräumter und enthalten die neuen Werte wie zum Beispiel den Personal Progress.

Testing

Mittels 12 Behat Szenarien decken wir momentan die Überhänge zwischen allen verfügbaren Ansichten sowie die Durchführung eines StudentQuiz mit 3 Fragen ab. Die 11 Unit Tests decken nur einen kleinen Anteil des vorhandenen Codes ab. Die Testbarkeit des Codes ist weiterhin grenzwertig.

Upgrade mit Migration

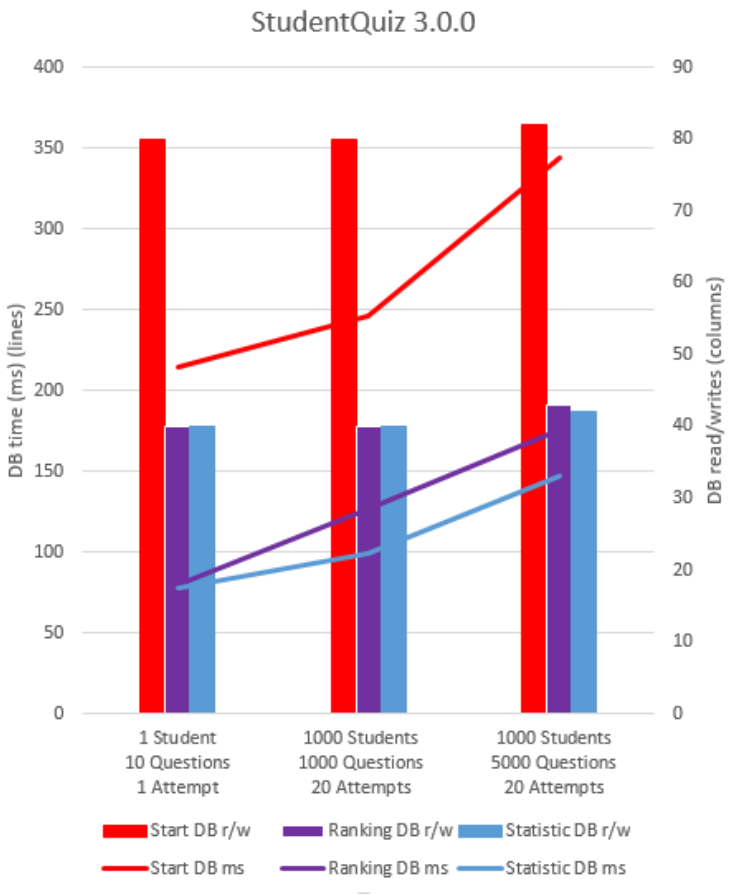


Abbildung 15: Manueller Belastungstest StudentQuiz 3.0.3

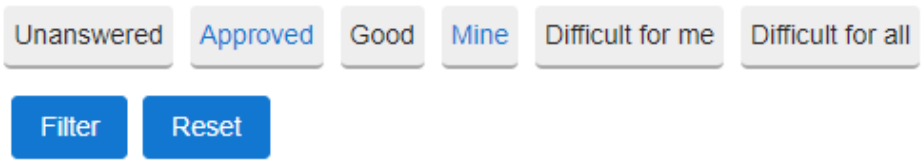


Abbildung 16: Neue Schnellfilter der Personal Learning Assistance

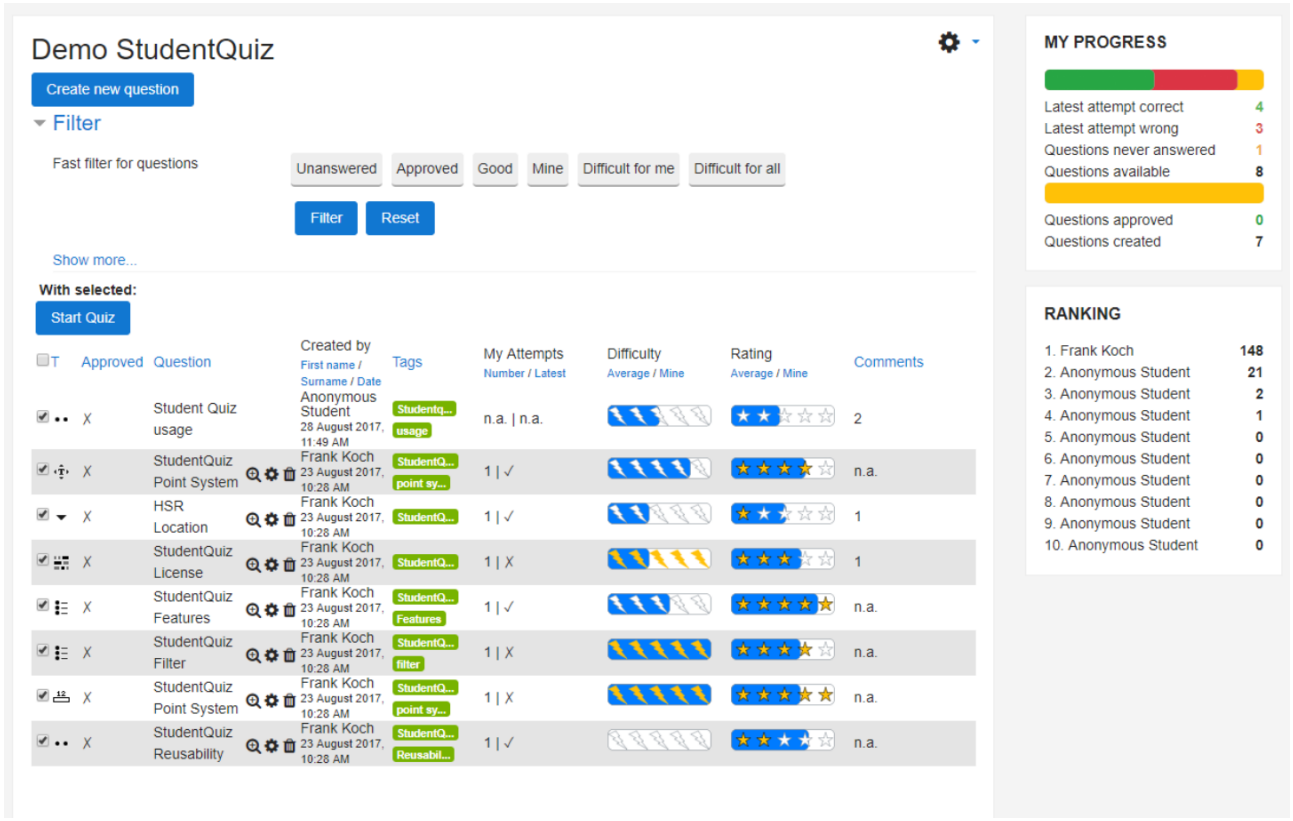


Abbildung 17: Überarbeitete Startseite StudentQuiz mit Personal Learning Assistance

Der Upgradeprozess räumt nun nicht nur die allfällig überschüssigen 990 Sections auf, sondern migriert zusätzlich die mit vorigen StudentQuiz Versionen abgespeicherten Attemptdaten. Auch erhalten Kommentierende eine Mail, wenn sie durch den Dozenten gelöscht wurde.

Backup und Restore

Die neuste StudentQuiz Version speichert nun in einem Kursbackup sämtliche Daten, welche StudentQuiz zu einer Frage kennt. So werden bei einem Restore sowohl Bewertungen, Bestätigungen, Kommentare und sämtliche Antworten wiederhergestellt.

Notifications

StudentQuiz versendet nun über die Message API von Moodle Benachrichtigungen an die betreffenden Autoren einer Frage, wenn diese bestätigt, gelöscht oder kommentiert wird.

Übersetzung

StudentQuiz ist nun vollständig durch das von Moodle betriebene Übersetzungssystem AMOS übersetzbar. Wir haben die Deutsche Übersetzung bereits selbst beigesteuert, erfreulicherweise wurden auch schon die Spanische Übersetzung von einem Übersetzer aus der AMOS Netzwerk übersetzt.

Nicht erreichte Ziele

Aus Zeitgründen mussten wir die Implementierung folgender Ziele und Verbesserungen zurückweisen

- **Moodle Mobile Kompatibilität**
- **Benutzerdefinierte Completion API Integration**

4.9.2 Ausblick Weiterentwicklung

Test Coverage

Unsere dringende Empfehlung ist es, als nächstes in die Weiterentwicklung der Testbarkeit des Plugins zu investieren. Die Stabilität des Projektes und sämtliche zukünftigen Anpassungen werden davon profitieren.

Moodle Mobile

Als nächstes sollte die Implementierung des Remote Addons zur Integration in Moodle Mobile App angegangen werden. Dies wird die Attraktivität des Plugins für Studenten massiv erhöhen, da StudentQuiz so bequem unterwegs genutzt werden kann.

Roadmap für das Datenmodell

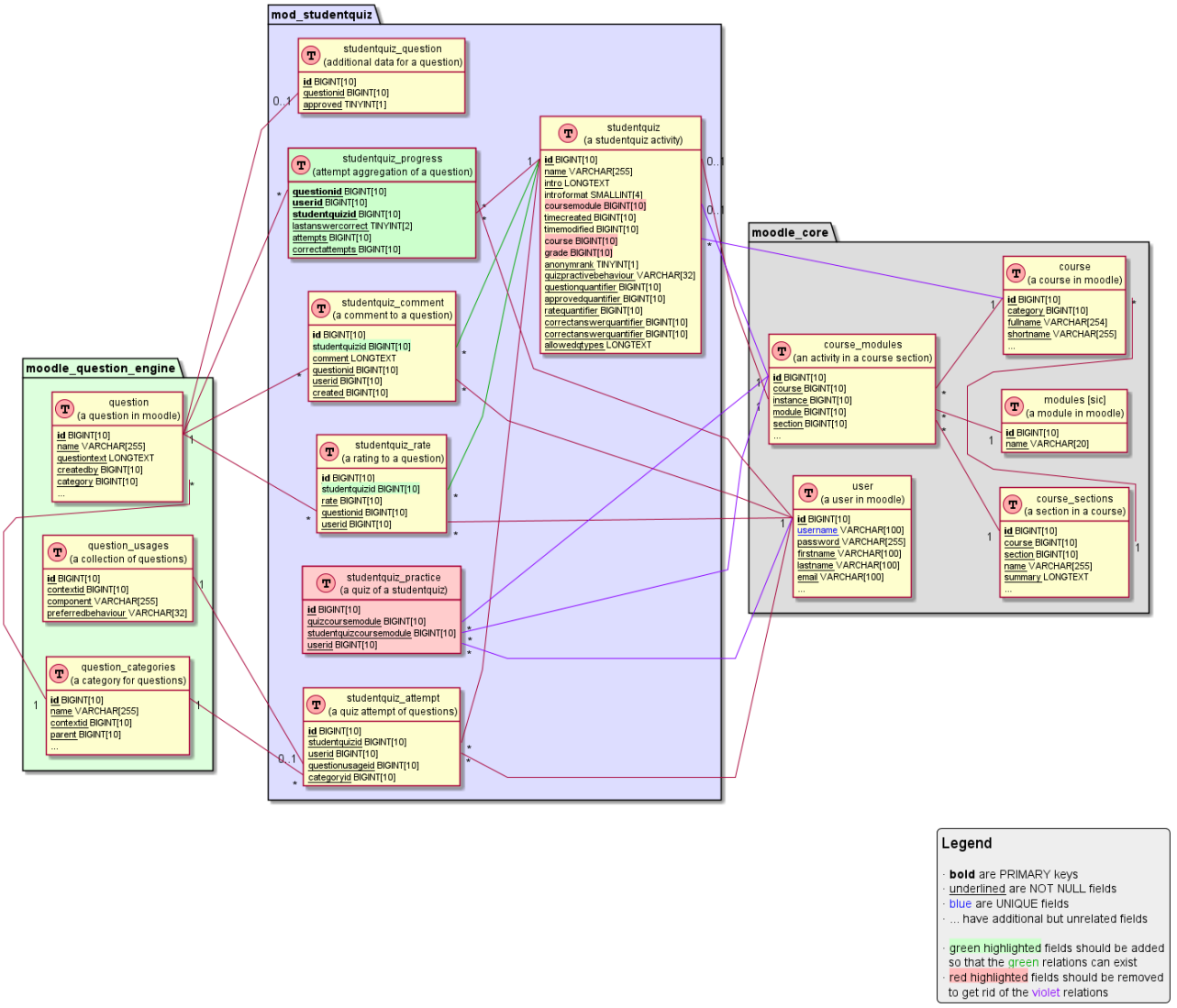


Abbildung 18: Datenmodell Roadmap für künftiges StudentQuiz

Die aktuelle Version von StudentQuiz konnte noch nicht alle Probleme der Vorversion lösen. Erstens, ist uns schleierhaft, warum man 3 Felder benötigt um sich im Kurs zurechtzufinden. Eine eindeutige Beziehung zur Tabelle `course_modules` kann hergestellt werden, indem das Feld `id` vom StudentQuiz zum Feld `instance` in Beziehung gebracht wird, indem man zusätzlich das Feld `module` zum StudentQuiz Modul Eintrag verknüpft. In allen von Moodle kommenden Views ist die Course Module ID der übergebene Wert, welche man auch intern immer verwenden könnte, und somit ist die Beziehung sogar noch einfacher herzustellen. Korrigiert man alle Stellen im Code wird somit das Feld `coursemodule` im StudentQuiz überflüssig. Diese Course Module ID gibt auch eindeutig an, in welchem Kurs sich das StudentQuiz befindet, somit kann man sich das Feld `course` ebenfalls sparen. Letzteres muss man jedoch zur Verteidigung erwähnen, dass das Quiz Modul dieses Feld ebenfalls pflegt.

Die Tabelle `studentquiz_practice` wurde genutzt um sich die Beziehung von StudentQuiz Activities zu

Quiz Activities zu merken. Die Quizzes sind nun durch den Migrations-Cleanup definitiv weg. Zu klären ist, ob die Anzahl der Quizzes in irgend einer Form noch dem Studenten angerechnet werden soll oder die Tabelle ersatzlos entfernt werden kann.

Bei `studentquiz_progress` handelt sich um eine bereits vorbereitete Tabelle für die aggregierten Attempt Daten als nächsten Performance Boost. Die Werte `lastanswercorrect`, `attempts` und `correctattempts` sind nämlich genau die essentiellen Daten für die Auswertung, die das aktuelle StudentQuiz live aus der Datenbank aus den Rohdaten berechnen muss. Eine solche Tabelle würde eine wirksame Abkürzung zu diesen Daten bedeuten. Ein Eintrag dieser Werte gehört einem Studenten in Verbindung zu einer Frage in einem bestimmten StudentQuiz.

Gegen Ende unserer Arbeit ist der Wunsch aufgekommen, dass Dozenten Fragen aus anderen Quellen in ein StudentQuiz verschieben können. Dies impliziert nun auch, dass man Fragen von einem StudentQuiz in ein anderes StudentQuiz verschieben kann, also auch in eines, welches nicht im gleichen Kurs ist. Dass der Frage-Ersteller nicht im Kurs eingeschrieben sein muss, ist deswegen berücksichtigt, sowohl in der Darstellung, als auch in der Ranking-Bewertung. Es ergibt sich dann aber die Situation, dass Bewertungen und Kommentare zu diesen Fragen sich in das neue StudentQuiz ebenfalls mit verschieben. Der letzte Stand der Anforderung ist, dass die Bewertungen und Kommentare im StudentQuiz bleiben, somit müsste man sich in diesen Tabellen die ID des StudentQuizzes merken.

Weiterführende Optimierungen der Performance

Auch wenn wir die Performance deutlich verbessert haben, können weitere Optimierungen vorgenommen werden. So würden die in Abschnitt 4.5.2 besprochene Speicherung der Aggregatswerte die Abfrage der Statistik und der Rangliste weiter beschleunigen. Listing 3 illustriert dies mit der Gegenüberstellung des `LEFT JOIN`s für die Berechnung des Wertes `mydifficulty` in der Fragentabelle. Dabei ist allerdings zu beachten, dass die korrekte Aktualisierung dieser Werte nicht trivial ist.

```
%% Abfrage My Difficulty:
%
LEFT JOIN (
 SELECT
 ROUND(1-avg(case state when 'gradedright' then 1 else 0 end),2) as mydifficulty,
 sum(case state when 'gradedright' then 1 else 0 end) as mycorrectattempts,
 questionid
 FROM {studentquiz_attempt} sqa
 JOIN {question_usages} qu ON qu.id = sqa.questionusageid
 JOIN {question_attempts} qa ON qa.questionusageid = qu.id
 JOIN {question_attempt\_steps} qas ON qas.questionattemptid = qa.id
 LEFT JOIN {question_attempt_step_data} qasd ON qasd.attemptstepid = qas.id
 WHERE sqa.userid = :currentuserid
 AND sqa.studentquizid = :studentquizid
 AND qasd.name='-submit'
 AND (qas.state = 'gradedright' OR qas.state = 'gradedwrong' OR qas.state='gradedpartial')
 GROUP BY qa.questionid
) mydiffs ON mydiffs.questionid = q.id
%
%% Abfrage My Difficulty mit Aggregatstabelle studentquiz_progress:
%
LEFT JOIN (
 SELECT
 mydifficulty as mydifficulty,
 correct_attempts as mycorrectattempts,
 questionid
 FROM {studentquiz_progress} sqp
 WHERE sqp.userid = :currentuserid
 AND sqp.studentquizid = :studentquizid
) mydiffs ON mydiffs.questionid = q.id
```

Listing 3: Vereinfachte Abfrage mit Aggregatstabellen

Ausserdem sind die JMeter Testpläne im Sourcecode Repository abgelegt, sodass diese wiederverwendet werden können.

Punktestand und Rang

Keiner der in StudentQuiz vorhandenen Aggregatswerte ist von dermassen vielen verschiedenen Datensätzen betroffen die der Punktestand und den damit verbundenen Rang eines Studenten. Eine gekonnte Zwischenspeicherung dieses Wertes wird die Rangliste noch einmal massiv beschleunigen. Dabei ist zu beachten, dass gerade aktive Benutzer jeweils die Auswirkungen ihrer Aktionen auf die Rangliste und den Punktestand direkt sehen möchten. Die Zwischenspeicherung könnte so gestaltet werden, dass die Anteile des Punktestandes, die durch die Aktionen anderer verändert werden (Bewertungen, Bestätigungen) durch einen minütlichen Cronjob aktualisiert werden und diejenigen die durch die eigenen Aktionen beeinflusst wird (Beantwortung von Fragen, Erstellen von Fragen) direkt eine neue Berechnung des Punktestandes auslöst. Die Rangliste kann auch mit einer leichten Verzögerung in einem Cronjob aktualisiert werden, da davon auszugehen ist, dass sich ein Grossteil der Studenten eines grossen Kurses nicht in den oberen Top10 der Rangliste anzutreffen sind und somit sowieso nicht sehen können, dass ihr Rang und ihr Punktestand während einer halben Minute nicht zusammenpassen.

Dozentendefinierte Completion

Um die Interaktion mit StudentQuiz weiter zu fördern, könnten Dozenten eine Mindestschwelle für eine erfolgreiche Beteiligung definieren. Mögliche Werte dafür wären die Anzahl erstellter Fragen, oder die Anzahl bestätigter Fragen. Prinzipiell könnte jeder statistische Wert als Grundlage genommen werden, wobei aus unserer Sicht aber zum Beispiel ein Mindestschwellwert bei der Durchschnittlichen Bewertung der erstellten Fragen wenig Sinn macht.

Weiterentwicklung der Visualisierung der Startseite

Die Startseite von StudentQuiz kann weiterentwickelt werden. Ansätze dazu haben wir im Konzept zur Personal Learning Assistance angedacht. So könnte man sich von der Tabellendarstellung verabschieden und noch mehr in die Metapher der Lernkartei investieren dass der Benutzer nicht mehr eine Liste von Fragen sondern einen virtuellen Stapel von Lernkarten vor sich sieht.

StudentQuiz als hochperformante Single Page Application auf der Startseite

Eine zukünftige Weiterentwicklung von StudentQuiz könnte sich damit beschäftigen, sämtliche Ansichten von einem kompletten Moodle Seitenaufbau zu lösen und lediglich als Komponenten einer Single Page Applikation zu implementieren. Mit moderner Webentwicklung könnten damit die Interaktionen mit StudentQuiz auf neue Höchstleistungen getrimmt werden, da der serverseitige Pageload von Moodle wegfällt.

5 Softwaredokumentation

Die folgenden Abschnitte beschreiben die Funktionsweise von StudentQuiz zum Zeitpunkt des Release 3.0.0.

5.1 Anforderungen und Design im Detail

5.1.1 Domain Model

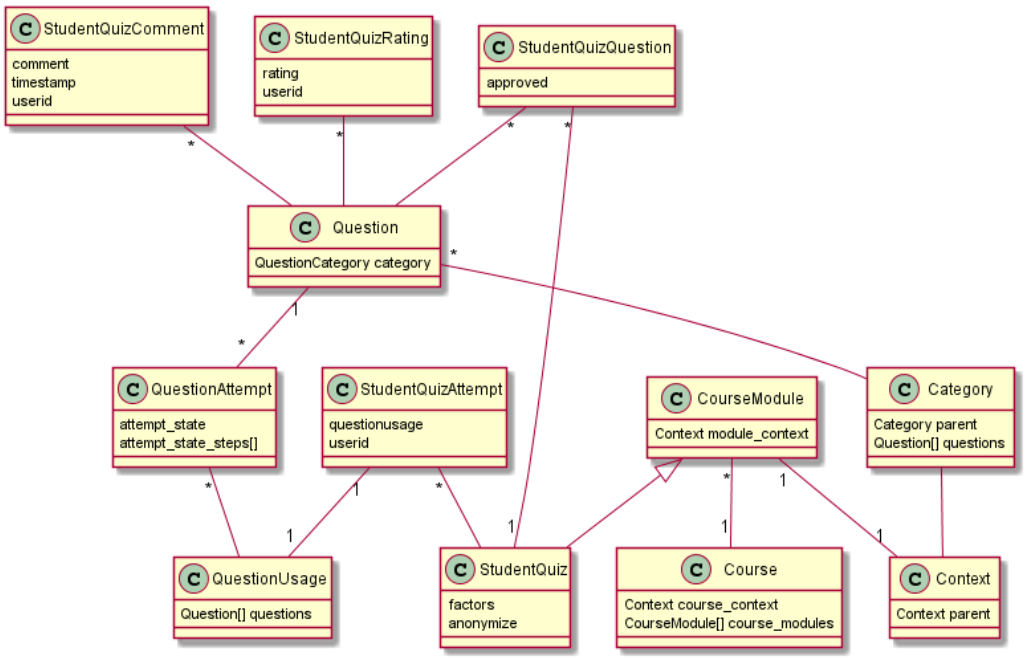


Abbildung 19: Domainmodel StudentQuiz 3.0.0

5.1.2 Use Cases

Die hier beschriebenen Use Cases beschränken sich auf die Szenarien, welche für die Entwicklung und Nutzung von StudentQuiz relevant sind. Wir haben die ursprüngliche Nummerierung der in der Vorarbeit beschriebenen Use Cases übernommen, die Beschreibung aber neu geschrieben. Eine Vielzahl von weiteren Use Cases, Rollen und Szenarien würde die Beschreibung aller verschiedenen Use Cases rund um die Verwendung von Moodle beinhalten. Für Moodle typische Schritte werden komprimiert erwähnt.

Aktoren

- **Dozent** Möchte in seinem Kurs StudentQuiz verwenden.
- **Student** Ist in einem Kurs eingeschrieben und möchte StudentQuiz nutzen.
- **Non-Editing-Teacher** Ist wie ein Student zu behandeln.

- **Administrator** Pflegt in der Regel eine produktiv eingesetzte Moodle Instanz und möchte eine stabile Verfügbarkeit des gesamten Systems garantieren.
- **Entwickler** Möchte das Plugin weiterentwickeln, ohne dabei die Stabilität der bestehenden Funktionalitäten zu gefährden.
- **Translator** Bietet seine Unterstützung zur Übersetzung des Plugins in seine Sprache an.

Anmerkung: Die in einem Kurs eingeschriebenen Studenten und Dozenten werden hier zusammengenommen als Kursteilnehmer bezeichnet.

Legende

- **[Brief]** Kurze Beschreibung des Use Cases.
- **[Pre]** Zu erfüllende Precondition / Vorbedingung vor erfolgreicher Durchführung des Use Cases.
- **[Post]** Zu erfüllende Postcondition / Bedingung nach erfolgreicher Durchführung des Use Cases.
- **[Steps]** Einzelne Schritte des Use Cases im brief format.
- **[Incl]** Untergeordneter Use Case
- **[Note]** Kommentar

UC01: StudentQuiz Activity CRUD

Brief Dozenten und Administratoren können eine oder mehrere StudentQuiz Activities in einem Kurs erstellen, konfigurieren oder löschen.

Pre User ist eingeloggt und hat ausreichende Berechtigungen um einen Kurs zu bearbeiten.

Pre StudentQuiz ist installiert.

Steps Create

1. User schaltet Kursübersicht in den Bearbeitungsmodus.
2. User wählt einen beliebigen Abschnitt aus und klickt auf Aktivität hinzufügen
3. User wählt StudentQuiz in der Liste der möglichen Aktivitäten
4. User erhält die Möglichkeit die neue StudentQuiz Aktivität zu konfigurieren.
5. User speichert die Einstellungen.

Steps Update

1. User öffnet die Einstellungen der StudentQuiz Activity
2. User passt die Einstellungen an.

Steps Delete

1. User schaltet Kursübersicht in den Bearbeitungsmodus
2. User löscht über das Einstellungsmenü der StudentQuiz Activity selbige.

Post Sobald eine StudentQuiz Activity erstellt wurde, können alle Kursteilnehmer auf die Activity zugreifen.

Post Nach dem Löschen einer StudentQuiz Activity sind sämtliche in Daten der Beantwortungen, Bestätigungen Bewertungen, Kommentare und Fragen gelöscht. Die eingeschriebenen Studenten verlieren die im Zusammenhang mit dieser Frage erhaltenen Punkte.

Post Nachdem die Konfiguration eines StudentQuiz angepasst wurde, werden die erhaltenen Punkte und die Rangliste mit den neuen Werten berechnet.

Post Die Änderung der Auswahl von zugelassenen Fragetypen hat keine Auswirkungen auf die im StudentQuiz verfügbaren Fragen, sondern nur auf die Auswahl von Fragetypen bei der Erstellung von neuen Fragen.

Folgende Werte einer StudentQuiz Activity können konfiguriert werden (In Klammern der empfohlene Standardwert)

- Punkte pro erstellte Frage (10)
- Punkte pro bestätigte Frage (5)
- Punkte für durchschnittliches Rating einer erstellten Frage (3)
- Punkte für zuletzt korrekte Antworten (2)
- Punkte für zuletzt falsche oder teilweise falsche Antworten (1)
- Erlaubte Fragetypen (Alle)
- Anonymisierung von Studenten (Aktiv)

UC02: StudentQuiz Activity teilnehmen

Brief Alle in den Kurs eingeschriebenen Studenten und Dozenten können am StudentQuiz teilnehmen.

Note Nicht eingeschriebene User werden nicht in der Rangliste geführt. Ihre Fragen werden aber wie die anderen Fragen eingerechnet. Für die Durchschnittswerte der Community werden nicht eingeschriebene User nicht berücksichtigt.

UC03: Fragen CRUD

Brief Alle Studenten und Dozenten können neue Fragen erfassen, eigene bearbeiten, in der Vorschau betrachten und löschen.

Post Wird eine Frage bearbeitet, sodass eine andere Antwort als bisherige korrekt gewertet wird, werden die bisherigen korrekten Antworten nicht als falsche Antwort gezählt.

UC04: Fragen filtern und sortieren

Brief Studenten und Dozenten können die angezeigten Fragen im StudentQuiz nach verschiedenen Kriterien filtern und sortieren.

Note Die aktive Filterung oder Sortierung kann mit einem Klick auf <Reset> Zurückgesetzt werden.

Note Ist keine Filterung aktiviert werden die verfügbaren Fragen nach absteigend nach Erstelldatum sortiert.

Note Mit Schnellfiltern können typische Filtereinstellungen mit einem Klick gesetzt werden.

Note Die Anzeige der Fragen ist grundsätzlich portioniert. Es können aber auf Wunsch alle Fragen angezeigt werden.

Note Die Grösse der Portionen kann angepasst werden.

Note Mit Checkboxen können Fragen der Auswahl hinzugefügt oder daraus entfernt werden.

Note Beim Seitenaufbau sind die Checkboxen aller angezeigten Fragen ausgewählt.

Die Filterkriterien umfassen folgende Punkte:

- Fragetyp
- Bewertung, Meine Bewertung
- Meine Versuche, Mein letzter Versuch
- Schwierigkeit, Meine Schwierigkeit
- Tag (nach Anzahl Tags sortierbar, Filterung case insensitiv)
- Anzahl Kommentare
- Fragetitel, Frageinhalt (nicht sortierbar)
- Erstelldatum
- Frageautor:
  - Als Dozent: Vorname, Nachname
  - Als Student: Meine Fragen

UC05: Eine Auswahl von Fragen beantworten

Brief Studenten und Dozenten können Fragen aussuchen und diese der Reihe nach beantworten.

Pre User ist im Kurs eingeschriebener Student oder Dozent

Steps
1. Der User wählt mit UC04 eine Auswahl von Fragen aus
2. Der User klickt auf <Start Quiz>

3. Für jede Ausgewählte Frage:
   (a) Ein Fortschrittsbalken zeigt an, wie viele Fragen schon beantwortet wurden.
   (b) Der User beantwortet die Frage. (Abhängig vom Typ der Frage).
   (c) Der User überprüft seine Antwort mit Klick auf <Check>
   (d) Der User sieht vorhandene Kommentare anderer User. Diese sind anonymisiert sofern die Anonymisierung aktiviert ist.
   (e) Falls der User die Frage noch nie bewertet hat, muss er die Frage bewerten bevor <Next>, <Previous> und <Finish> aktiviert werden.
   (f) Der User kann die Frage beliebig oft kommentieren.
   (g) Falls er eine Frage noch nicht beantwortet hat, gelangt er zur nächsten Frage mit einem Klick auf <Next>.
4. Der User klickt auf <Finish> und wird auf die Übersichtsseite umgeleitet.

Note Nach einer erfolgreichen Beantwortung ist der Filter wieder zurückgesetzt.

UC06 Dieser Use Case wird nicht mehr unterstützt.

UC07: Fragen verschieben

Brief Dozenten und Administratoren können ausgewählte Fragen in eine andere Kategorie verschieben.

Pre Der Dozent muss berechtigt sein, Fragen in der anderen Kategorie zu bearbeiten.

Steps
1. Der Dozent oder Admin stellt sich wie in UC04 beschrieben eine Auswahl von Fragen zusammen.
2. Er wählt eine Zielkategorie in der angezeigten Auswahl von Kategorien aus.
3. Mit einem Klick auf <MoveTo> werden die Fragen in diese Kategorie verschoben.

Post Sofern die Zielkategorie in der Hierarchie über der Kategorie der StudentQuiz Aktivität liegt, verhält sich StudentQuiz so, als ob die Fragen gelöscht worden wären.

Post Sofern die Zielkategorie in der Hierarchie unter der Kategorie der StudentQuiz Aktivität liegt, verhält sich StudentQuiz so, als ob die Fragen nicht verschoben worden wären.

UC08: Import/Export von Fragen

Brief Dozenten und Administratoren können Fragen über die Question Bank Fragen exportieren und importieren.

Post Importierte Fragen verhalten sich so, als wären sie vom Dozenten/Administratoren erstellt worden.

UC09: Plugin upgrade verwalten

Brief Der Administrator kann das StudentQuiz Plugin aktualisieren.

Post Die mit der vorherigen Version erfassten Daten werden migriert.

UC10: Statistik betrachten

Brief Kursteilnehmer können die Statistik anschauen.

Die Statistik zeigt dem Kursteilnehmer folgende personalisierte Werte an:

- Anzahl selbst erstellter Fragen
- Anzahl bestätigter selbst erstellter Fragen
- Durchschnitt der durchschnittlichen Bewertungen der erstellten Fragen
- Anteil der richtigen Antworten an allen eigenen Antworten
- Anteil der zuletzt richtig beantworteten Fragen an allen verfügbaren Fragen (persönlicher Lernfortschritt)

Die Statistik zeigt dem Studenten folgende Community Werte an:

- Anzahl verfügbarer Fragen
- Anzahl verfügbarer, bestätigten Fragen
- Durchschnitt der durchschnittlichen Bewertung aller Fragen
- Anteil richtiger Antworten an allen Antworten
- Durchschnitt des persönlichen Lernfortschritts aller Kursteilnehmer

UC11: Mehrsprachigkeit

Brief StudentQuiz unterstützt den von Moodle vorgesehenen AMOS Mechanismus zur Übersetzung sämtlicher angezeigten Texte.

UC12 Dieser Use Case wird nicht mehr unterstützt.

UC13: Fragen bestätigen

Brief Um die Qualität der Fragen sicherzustellen, können Dozenten eine Frage als bestätigt markieren oder die Bestätigung rückgängig machen.

5.1.3 Berechnung der statistischen Werte

Aggregatswerte pro eingeschriebenem Student

- Erstellte Fragen (Anzahl)
- Bestätigte Fragen (Anzahl)
- Antworten (Anzahl)
- Zuletzt korrekt beantwortete Fragen (Anzahl)
- Zuletzt falsch oder teilweise falsch beantwortete Fragen (Anzahl)
- Durchschnitt der durchschnittlichen Bewertungen der eigenen Fragen durch die Community
- Total erhaltene Punkte
- Rang

Aggregatswerte pro Frage

- Bestätigt (Eigenschaft)
- Durchschnittlich erhaltenes Rating (Durchschnitt)
- Ratings (Anzahl)
- Gegebene Antworten (Anzahl)
- Korrekt gegebene Antworten (Anzahl)
- Kommentare (Kommentare)

Aggregatswerte pro eingeschriebenem Student und Frage

- Rating (Eigenschaft)
- Letzter Versuch (Eigenschaft)
- Anzahl gegebener Antworten (Summe)
- Anzahl korrekt gegebener Antworten (Summe)

Aggregatswerte pro StudentQuiz

- **Anzahl verfügbare Fragen** Summe aller durch eingeschriebene Studenten/Dozenten erstellte Fragen + Anzahl importierte/hineingeschobene Fragen von nicht (mehr) eingeschriebenen Benutzern.
- **Anzahl bestätigter Fragen**
- **Anzahl gegebener Antworten**
- **Anzahl eingeschriebener Studenten/Dozenten**
- **Durchschnittliche Anzahl gegebener Antworten pro eingeschriebenem Benutzer**
- **Durchschnittliche Anzahl korrekt gegebener Antworten pro eingeschriebenem Benutzer**
- **Durchschnittliche Anzahl zuletzt korrekt beantworteten Fragen pro eingeschriebenem Benutzer**

Skaleneffekte

Die Statistik und die Rangliste werten alle Daten aus, welche durch die Beteiligung einer Fragebeantwortung des Studenten generiert wurden. Für die Auswertung betrachten wir folgende Aktionen als atomar, obwohl diese aus Sicht von Moodle noch in weitere Teilschritte zerlegt werden könnten:

- **Erstellen einer Frage**
- **Importieren einer Frage**
- **Löschen einer Frage**
- **Verschieben einer Frage in eine übergeordnete Kategorie**
- **Einschreiben eines Studenten**
- **Einschreibung eines Studenten auflösen**
- **Un-/Bestätigen einer Frage**
- **Bewerten einer Frage**
- **Kommentieren einer Frage**
- **Beantworten einer Frage** Wie in Abbildung 20 veranschaulicht, kann eine einzelne Frage innerhalb einer Question Usage in einem von überraschend vielen Zuständen sein. Diese Vielschichtigkeit entsteht bei Moodle durch die Flexibilität, welche einerseits durch erweiterbare Fragetypen und andererseits durch erweiterbare Question Behaviours erwünscht ist. Die Übergänge zwischen diesen Zuständen hinterlassen abhängig vom gewählten Question Behaviour unterschiedliche Spuren in der Datenbank. Dies macht die Auswertung der Attempt Daten schwieriger und umständlich.

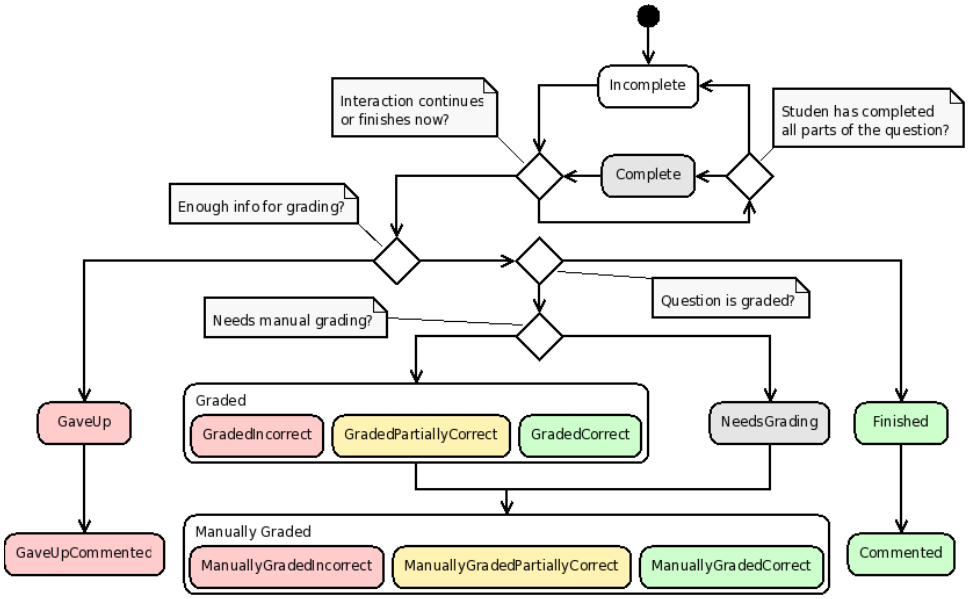


Abbildung 20: Zustandsdiagramm der Beantwortung einer Frage (Quelle: moodle.org)

5.1.4 Nichtfunktionale Anforderungen

Portierbarkeit

- **Installierbarkeit** Die vorgesehenen Prozeduren des Basissystems sind anzuwenden.
- **Abhängigkeiten** Die Anzahl der Abhängigkeiten zu basissystemfremden Bibliotheken sollte möglichst vermieden werden.
- **Umstellungsfähigkeit** Die Kompatibilitätsliste des Basissystems ist möglichst zu berücksichtigen. Bei der Datenbank müssen mindestens MySQL und PostgreSQL unterstützt werden.
- **Fremdanwendungen** Es ist keine Kompatibilität zu Fremdanwendungen ausser des Basissystems vorgesehen.

Leistung

- **Durchschnittswerte Anfragen** Das System kann 15'000 Zugriffe innerhalb eines Semesters verarbeiten.
- **Höchstwerte Anfragen** Das System kann 500'000 Zugriffe innerhalb eines Semesters verarbeiten.
- **Erlaubte Antwortzeiten** Die Antwortzeiten sollen zu Vergleichsseiten des Basissystem entsprechen.
- **Ressourcennutzung** Der Betrieb ist über mehrere Jahre hinweg vorgesehen.
- **Maximale Kapazität** Dieses Produkt soll Gruppen bis zu 500 Studierenden und 5'000 Fragen bedienen können.

Benutzerfreundlichkeit

- **Wahrnehmung** Die Wahrnehmung der Endbenutzer darf sich durch den Einsatz dieser Software nur unmerklich beeinflussen.
- **Fehleingaben** Endbenutzer können keine falschen Aktionen ausführen.
- **Falscheingaben** Fehleingaben sind nach den Vorgaben des Basissystems hilfreich zu kennzeichnen und mit einem nützlichen Hinweistext zu versehen.
- **Barrierefreiheit** Es sind keine spezifischen Unterstützungs-Funktionen vorgesehen. Für die Übersetzungen ist das kollaborative Translation-Tool des Basissystems zu unterstützen.
- **Benutzbarkeit** Das Projekt muss auf allen Themes des Basissystems laufen. Auf mobiles Handling wird Wert gelegt.

Wartbarkeit

- **Frameworks** Die Funktionen des Basissystems sind nach neuster Dokumentation zu verwenden.
- **Dritt-Anbieter** Im Resultat sind Abhängigkeiten zu Diensten von Dritt-Anbietern zu vermeiden. Für die Entwicklung und für die Testinfrastruktur sind diese erwünscht.
- **Modularität** Schnittstellen des Basissystems sind anzuwenden.
- **Backup** Die Backup Prozeduren des Basissystems sind zu implementieren.
- **Änderbarkeit** Als OpenSource Projekt ist jeglicher Quellcode dem Projekt mitzugeben.
- **Testbarkeit** Die Testinfrastruktur ist zu erweitern und die Testbibliotheken des Basissystems sind anzuwenden.

Sicherheit

- **Benutzereingaben** Jegliche Benutzereingaben sind zu validieren und zu säubern (sanitize).
- **Vertraulichkeit** Daten dürfen nur mit korrekter Authentifizierung und Authorisierung zugänglich gemacht werden.
- **Integrität** Es sind keine besonderen Massnahmen in dieser Arbeit vorgegeben. Die üblichen Vorgehensweisen und Konfigurationen sind anzuwenden.

5.2 Architektur und Design

5.2.1 Datenmodell

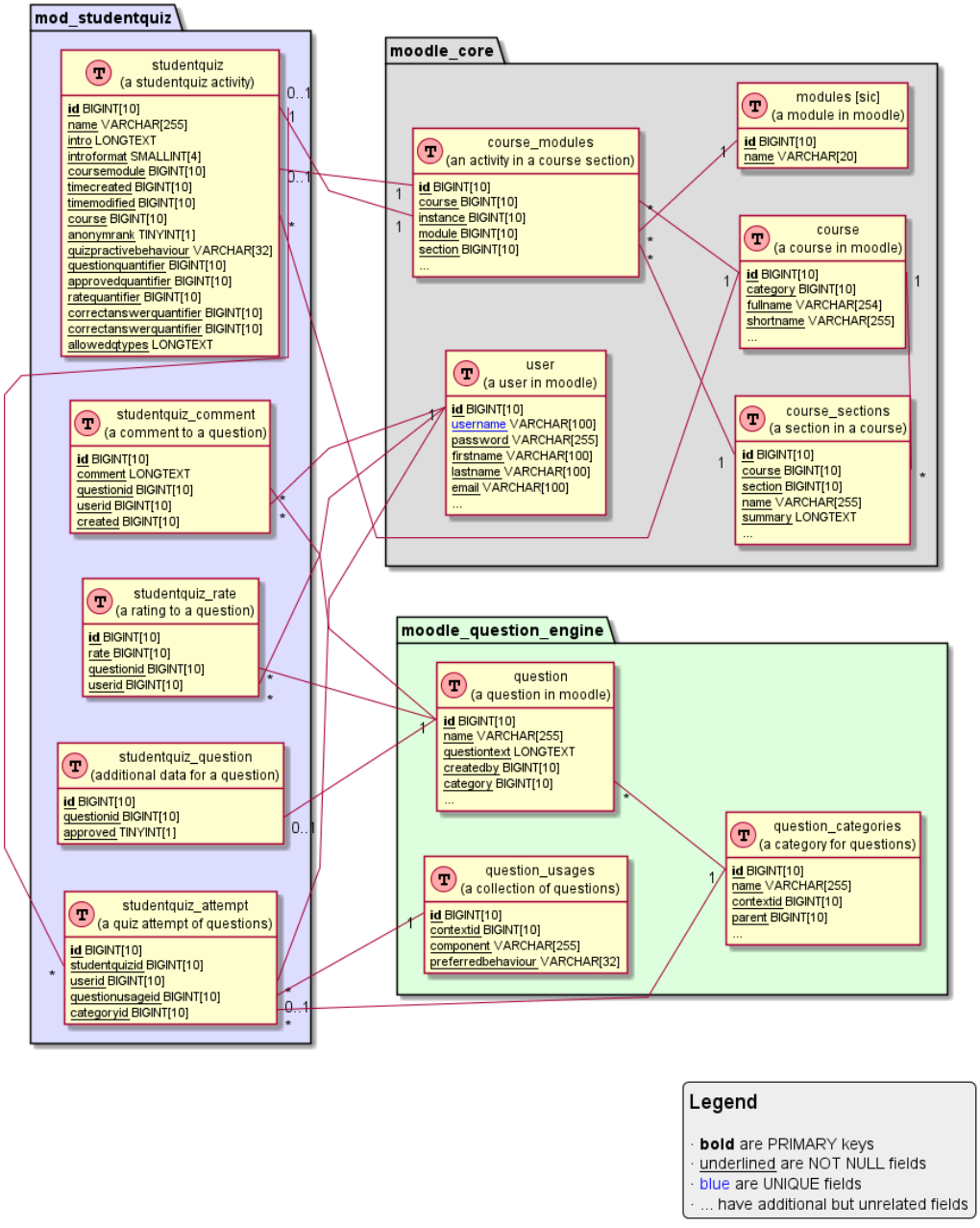


Abbildung 21: Datenmodell Stand StudentQuiz 2.0.3

Das Datenmodell in Abbildung 21 zeigt die effektive Umsetzung der Datenbank mit allen expliziten Beziehungen. Die wichtigste Änderung zur vorherigen StudentQuiz Version ist die direkte Anbindung der Question Usage an das StudentQuiz. StudentQuiz hat weiterhin die Wahl, welches Question Behaviour verwendet werden soll. Hiermit ist nun aber StudentQuiz im Verantwortungsbereich, den Studenten durch

seine Frageliste der Question Usage zu führen. Sie muss somit die entsprechenden Views anbieten, dafür spart sie sich die Abhängigkeit zum Quiz Modul. Aufgrund der Notwendigkeit, die View für die Durchführung anzubieten, ist das Question Behaviour Plugin überflüssig geworden, da nun StudentQuiz diese selbst darstellen kann. Die Verbindung von einem StudentQuiz zur Question Usage wird über die Tabelle `studentquiz_attempt` gespeichert.

Die Tabelle `studentquiz` hat ein paar Felder erhalten, damit die Punktefaktoren pro Activity gespeichert werden können. Ausserdem ist die Beziehung zwischen den Tabellen `studentquiz_question` und `question` gelockert worden. Dadurch ist es möglich, dass Fragen aus einer anderen Quelle im Moodle kommen dürfen als direkt vom StudentQuiz.

In der Weiterentwicklungs-Roadmap im Abschnitt 18 sind überflüssige und fehlende sowohl für Felder wie auch für Beziehungen erläutert, deren Änderung nicht mehr in die aktuelle Version geschafft haben.

5.3 Komponenten

In diesem Abschnitt stellen wir die einzelne spezifische Komponenten des StudentQuiz Plugins vor. Diese Beschreibungen sollen eine Einarbeitung in den Code unterstützen. Wir setzen dabei voraus, dass der offizielle Moodle Plugin File Guide[15] gelesen wurde, welcher die generischen Inhalte jedes Moodle Plugins beschreibt.

5.3.1 Renderer

Für jede der vier Seiten (Startseite, Attempt, Statistik und Rangliste) existiert ein eigener Renderer in der Datei `renderer.php` . Diese sind alle mit Absicht zustandslos und enthalten allesamt Funktionen zur Ausgabe von HTML, abhängig von den als Parameter übergebenen Daten.

5.3.2 Startseite

Der Seitenaufbau der Startseite von StudentQuiz stellt innerhalb von StudentQuiz das komplexeste Zusammenspiel verschiedener Klassen dar. Die in `view.php` instanziierte und `viewlib.php` definierte Klasse `mod_studentquiz_view` enthält in sich kaum Logik, sondern instanziert hauptsächlich die StudentQuiz Bank View. Die Separation of Concerns zwischen der im nächsten Abschnitt beschriebenen StudentQuiz Bank View und der `mod_studentquiz_view` muss bei einer nächsten Iteration sicherlich diskutiert werden.

5.3.3 StudentQuiz spezifische Question Bank View

Die Startseite von StudentQuiz müsste ohne die Fragentabelle neu erfunden werden. Diese Tabelle wird von einer Ableitung und Übersteuerung des Moodle Packages `\core_question\bank\view` aus der `questionlib` implementiert.

Fragentabelle

Die StudentQuiz Fragentabelle `studentquiz_bank_view` ist im Ordner `classes/question/bank` zu finden. Diese Klasse übernimmt einerseits die Abfrage der gewünschten Fragen, als auch deren Darstellung als Tabelle. Zudem initialisiert, konfiguriert und verwaltet sie das Filterformular mit einer StudentQuiz Condition und verhält sich diesem gegenüber als Controller, in dem sie sich auch noch gleich um die Auswertung der im Filter gesetzten Werte kümmert. Allein schon dieser Beschreibung ist anzumerken, dass diese Klasse viel zu viele Aufgaben und Zustände übernimmt und möglichst bald entsprechend umstrukturiert und aufgeteilt werden sollte. Moodle setzt jedoch die Question Bank View genau so ein.

Filterformular

Das StudentQuiz Filterformular `mod_studentquiz_question_bank_filter_form` ist im Ordner `classes/question/bank` zu finden. Im Vergleich zur StudentQuiz Bank View ist diese Klasse sehr überschaubar und implementiert, wie an vielen anderen Stellen in Moodle, ein benutzerdefiniertes Formular. Ein Filterformular verwendet eine Liste von Feldern.

Für die Fast Filter wurde ein neuer Feldtyp `toggle_filter_checkbox` definiert, welcher eine Spezialisierung der bewährten `user_filter_checkbox` darstellt. Die Spezialisierung war notwendig, um die Fast Filters im Filterformular als horizontal angeordnete Gruppe von Feldern darzustellen.

Auch der Feldtyp `user_filter_tag` wurde für StudentQuiz implementiert um Fragen auch nach Tags filtern können, ohne spürbare Performance-Verluste in Kauf nehmen zu müssen.

Filterauswertung

Die StudentQuiz Condition befindet sich in `classes/condition`. Diese Klasse implementiert die Auswertung der gesetzten Filtereinstellungen. Sie stellt sicher, dass die resultierende Abfrage die gewünschten `WHERE` Statements und die entsprechenden Parameter erhält.

Spalten der Fragentabelle

Die einzelnen Spalten der Fragentabelle werden als eigenständige Klassen modelliert. Sie erben alle von der Basisklasse `column_base` und definieren sowohl benötigten Werte zur Abfrage in Form von zusätzlichen `JOIN` Statements und Sortierkriterien, als auch die Darstellung dieser Werte, wenn die Tabelle in HTML ausgegeben werden soll.

- Unverändert eingesetzte Standardspalten der Question Bank
  - `checkbox_column`
  - `question_type_column`
  - `question_name_column`
  - `edit_action_column`
  - `delete_action_column`
- Für StudentQuiz überschriebene Standardspalten aus der Question Bank
  - `anonym_creator_name_column`

  - `preview_column`
  - `question_text_row`

- Für StudentQuiz zusätzlich implementierte Spalten
  - `approved_column`
  - `tag_column`
  - `practice_column`
  - `difficulty_level_column`
  - `rate_column`
  - `comment_column`

5.3.4 Tasks

Jedes Plugin kann für Moodle unter `db/tasks.php` Tasks definieren, die regelmässig oder unter bestimmten Bedingungen, aber vor allem unabhängig von einem Seitenaufbau, ausgeführt werden. Wir haben einen solchen Task `delete_quiz_after_migration` implementiert, um die, nach dem Upgrade auf StudentQuiz 3.0.0 obsolet gewordenen, Quiz Activities zu löschen. Wir gehen davon aus, dass zukünftige Optimierungen weitere Tasks definieren, welche die Aktualisierung der statistischen Daten von den einzelnen Seitenaufrufen entkoppeln.

5.4 Benutzeranleitungen

Die verschiedenen Manuals für Studenten, Dozenten und Administratoren sind nun als ReadTheDocs[30] Dokumentsammlung verfügbar. Hiermit kann das Format für den Export frei gewählt werden, wie z.B. HTML oder auch PDF. Die Anleitung kann somit unter `https://studentquiz.hsr.ch/docs/` betrachtet werden, siehe Abbildung 22 für einen Screenshot. Der PDF-Export ist im Anhang C.4 beigelegt.

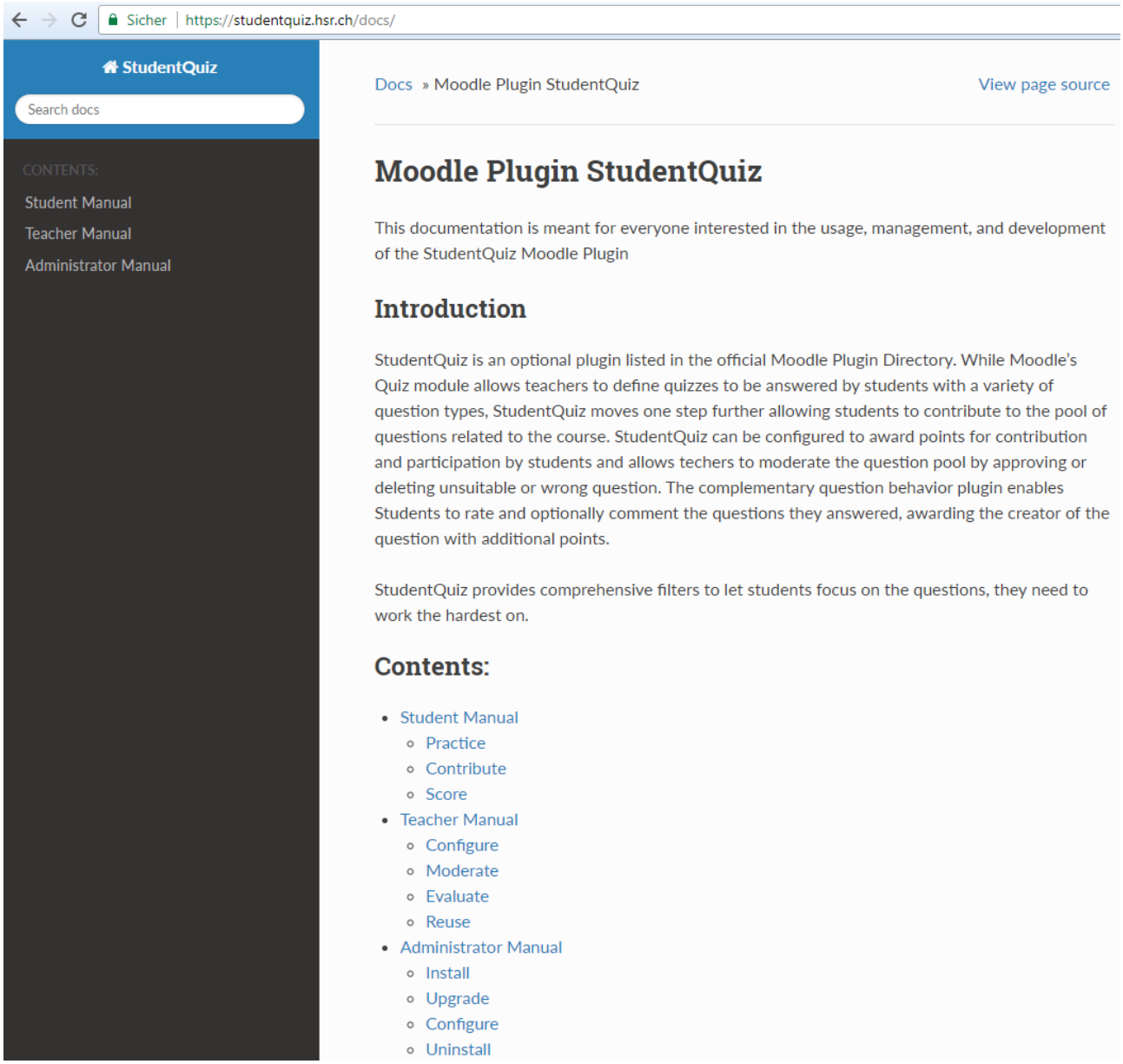


Abbildung 22: Benutzeranleitungen als HTML-Seite

Sie ermöglicht so auch die Suche, siehe Abbildung 23, nach einem Begriff für das schnellere Finden der gesuchten Stelle.

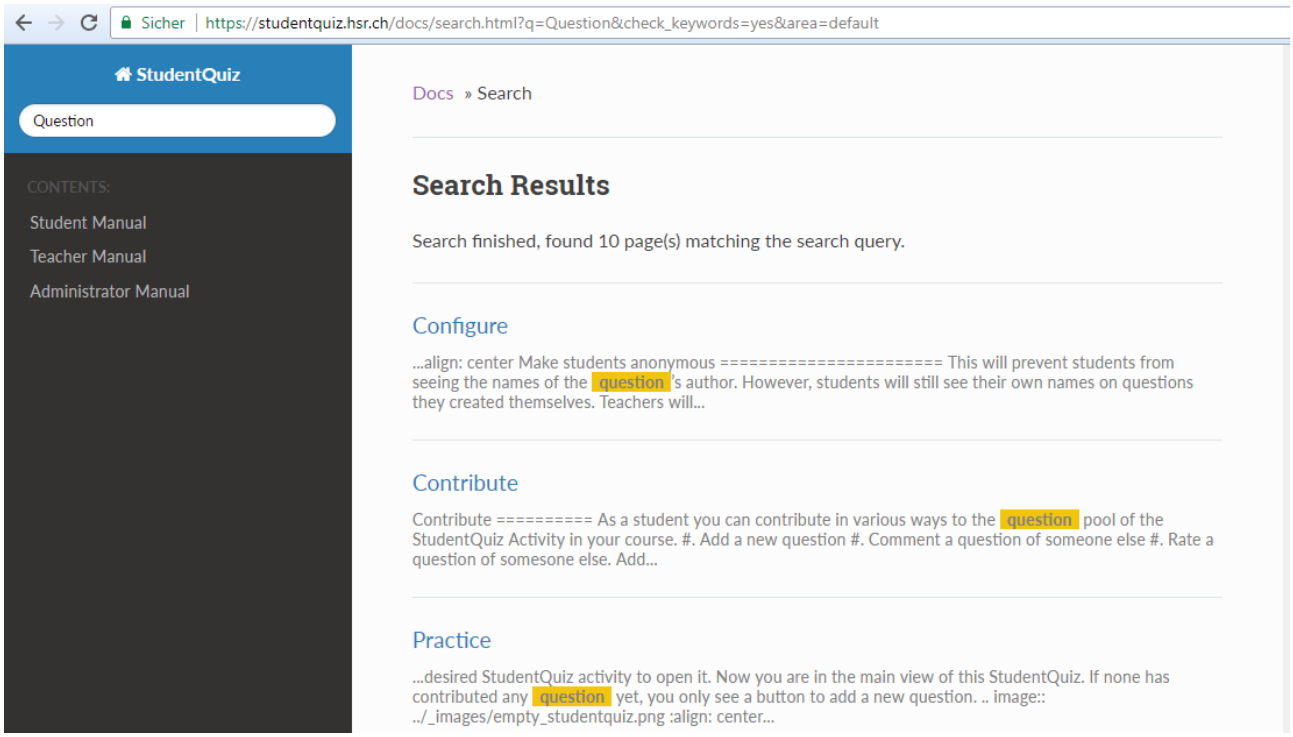


Abbildung 23: Vorschau der Suchfunktion der HTML-Seite

5.5 Anleitung für Entwickler für die HSR VM-Infrastruktur

Diese Softwaredokumentation enthält bezüglich der Einrichtung der Staging Umgebung diverse HSR-spezifische Angaben. Diese Angaben sind als Anschauungsbeispiel und Umsetzungsvorschlag gedacht, sollte eine ähnliche Umgebung wieder aufgestellt werden.

Moodle ist eine in PHP geschriebene Webapplikation, somit benötigt man einen Web-Server, den PHP-Interpreter und eine Datenbank. Zusätzlich sind diverse Debugging-Tools hilfreich. Die Anleitung hilft beim Einrichten der lokalen Umgebung, sowie auf einem HSR-Server auf Basis des SE2-Images (in welchem die vorbereiteten Dienste mit Docker realisiert sind).

Alle Angaben sind nur für eine Test-Umgebung optimiert, für den produktiven Einsatz entsprechende Massnahmen treffen. Die verwendeten Datenbankdienste variieren zwischen den Umgebungsoptionen absichtlich um eine passive Testabdeckung zu erhalten.

5.5.1 Dienste einrichten

Lokal auf Windows

Für Windows verwendet man am besten einen Web-Stacks, wie z.B. XAMPP oder eine andere äquivalente Software.

Eckdaten:

- Beispiel mit Apache, PHP 5.6 oder 7 (je nach Wahl), MySQL
- DocumentRoot ist vorgabe-des-stacks/moodle
- Moodle wird manuell heruntergeladen und entpackt

Für die Einrichtung sind die folgenden Schritte notwendig. Für die genauen Einzelheiten der Schritte konsultiere die entsprechende Anleitung des Web-Stacks:

- Erstelle einen VirtualHost mit dem obigen DocumentRoot
- Erstelle einen Benutzer für MySQL für die Verwendung der Moodle Instanz
- Aktiviere gegebenenfalls benötigte PHP-Erweiterungen
  - Die Benötigten werden später bei der Installation von Moodle angezeigt
  - Üblicherweise fehlen diese: `php-xml` `php-zip` `php-curl` `php-intl` `php-mbstring` `php-openssl`

Lokal auf Linux

Diese Anleitung richtet sich an die Linux Distribution Ubuntu 16.04, dennoch sollte diese ohne Probleme bei allen Debian-basierten Systemen angewendet werden können.

- Beispiel mit Nginx, PHP 7, MariaDB
- DocumentRoot ist `/var/www/html/moodle`
- Moodle Instanzen mit MDK[1]

Als Erstes kümmern wir um die Installation der benötigten Dienste:

```
1  apt-add-repository ppa:ondrej/php #for php7.1 packages
2  apt-get install nginx # webserver
3  apt-get install php-fpm php-xml php-gd php-zip php-curl php-intl php-mbstring php-mysql # php
   ↪ as fastcgi process manager, database-extension alternatively: php-pgsql. mariadb also uses
   ↪ the mysql-client. the additional packages are required by moodle
4  apt-get install mariadb-server # database, alternatively: postgresql or mysql-server
```

Listing 4: Installation der benötigten Dienste auf der lokalen Umgebung

Dank dem Betriebssystem und dem Paketmanager sind die Dienste bereits vorkonfiguriert und gestartet. Eine Ausnahme ist eine Anpassung an der Nginx Konfiguration, die wegen Moodle's benötigen Slash-Arguments[13] notwendig sind. Die Konfiguration `/etc/nginx/sites-enabled/default` sollte am Schluss etwa so aussehen:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    root /home/your-user/some-path/moodle;

    index index.php index.html index.htm index.nginx-debian.html;

    server_name _;
    client_max_body_size 10M;

    location / {
        try_files $uri $uri/ =404;
        autoindex on;
    }

    location ~ [^/].php(/|$) {
        fastcgi_split_path_info  ^(.+.php)(/.+)$;
        include snippets/fastcgi-php.conf;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_pass unix:/run/php/php7.1-fpm.sock;
    }
}
```

Listing 5: Nginx Konfiguration für Moodle in der lokalen Umgebung

Auch zwingt Moodle MySQL-basierte Systeme das Barracuda-Dateiformat für die volle Unicode-Unterstützung zu aktivieren. Hierzu sind die folgenden Werte in der MariaDB-Konfiguration `/etc/mysql/mariadb.conf.d/50-server.cnf` zu ergänzen:

```
[mysqld]
innodb_file_format=BARRACUDA
innodb_large_prefix=1
```

Listing 6: Konfiguration für MySQL betreffend Dateiformaten

Desweiteren kann die Upload-Limite zu niedrig sein. Ein Referenz-Quiz-Import für die Tests ist z.B. 3MB gross. Hierfür muss in PHP `/etc/php/7.1/fpm/conf.d/40-moodle.ini` sowie gegebenfalls im Webserver (für Nginx oben schon enthalten) die Upload-Limite erhöht werden:

```
post_max_size=10M
upload_max_filesize=10M
```

Listing 7: Konfiguration für PHP betreffend Upload-Limiten

Darauf nicht vergessen die jeweiligen Dienste neu zu starten:

```
service nginx restart
service mysql restart
service php7.1-fpm restart
```

Listing 8: Dienste neustarten um die neue Konfiguration zu laden

Als nächstes legen wir für die MariaDB einen administrativen Benutzer an, damit später MDK die Datenbanken für die Moodle Instanzen anlegen kann:

```
mysql -e "GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost' IDENTIFIED BY 'admin' WITH GRANT OPTION; FLUSH PRIVILEGES;"
```

Listing 9: Einrichten des Datenbank-Administratorkontos

Linux-Server auf Basis des SE2-Image der HSR

Das SE2-Image der HSR baut die vorgefertigten Dienste mit Docker. Die Container sind in der Crane-Konfiguration `/etc/crane/crane.yaml` spezifiziert. Da die Datei relativ gross für dieses Dokument ist, sind die notwendigen Anpassungen als Diff spezifiziert:

```
@@ -145,5 +145,43 @@
       - jenkins-data_var_lib_jenkins:/d/10:rw
       - wiki-data_data:/d/11:rw
+      - moodle-db-data_var_lib_postgresql_data:/d/12:rw
+      - moodle-data_var_www_html:/d/13:rw
+      - moodle-data_var_log_nginx:/d/14:rw
   volimport: { <<: *dumpref, run: { <<: *dumprefrun, cmd: ["/bin/bash", "-c", "bsdtar xf /xfer/dumpvol.$$(date +%Y%m%d).tgz -C /d"] } }
+  moodle-db: &mcredref
+    image: sameersbn/postgresql:latest
+    run: &mcredrunref
+      detach: false
+      env:
+      - DB_NAME=admin
+      - DB_USER=admin
+      - DB_PASS=admin
+      hostname: moodle-db
+      publish:
+      - 5432:5432
+      stop-signal: SIGTERM
+      volume:
+      - moodle-db-data_var_lib_postgresql:/var/lib/postgresql:rw
+  moodle-db-dumpdb: { <<: *mcredref, run: { <<: *mcredrunref, user: "postgres", cmd: ["/bin/bash", "-c", "eval \"pg_dump -U $$DB_USER -h pgdb -d $$DB_NAME --encoding=UTF-8 --file=/dbdata/$${DB_NAME}_$$(date +%Y%m%d).sql\""], detach, tty: true, interactive: true, volume: ["/tmp/moodle:/dbdata:rw"], link: ["moodle-db:pgdb"] } }
+  moodle-db-loaddb: { <<: *mcredref, run: { <<: *mcredrunref, user: "postgres", cmd: ["/bin/bash", "-c", "eval \"psql -U $$DB_USER -h pgdb -d $$DB_NAME --file=/dbdata/$${DB_NAME}_$$(date +%Y%m%d).sql\""], detach, tty: true, interactive: true, volume: ["/tmp/moodle:/dbdata:ro"], link: ["moodle-db:pgdb"] } }
+  moodle:
+    image: richarvey/nginx-php-fpm:latest
+    run:
+      detach: false
+      env:
+      - VIRTUAL_HOST=sinv-56015.edu.hsr.ch
+      - VIRTUAL_PATH=/moodle
+      #- LOCATION_REWRITE=/(.*)|/$$1
+      - WEBROOT=/var/www/html/moodle
+      - ERRORS=1
+      - RUN_SCRIPTS=1
+      hostname: moodle
```

```
+      link:
+      - moodle-db:postgres
+      stop-signal: SIGTERM
+      volume:
+      - moodle-data_var_www_html:/var/www/html:rw
+      - moodle-data_var_log_nginx:/var/log/nginx:rw
+      - /etc/localtime:/etc/localtime:ro
 hooks:
   ngxpx:
@@ -156,4 +194,6 @@
   - jenkins
   - wiki
+  - moodle
+  - moodle-db
 volumes:
   ngxpx-data-confd: null
@@ -168,2 +208,5 @@
   jenkins-data_var_lib_jenkins: null
   wiki-data_data: null
+  moodle-db-data_var_lib_postgresql: null
+  moodle-data_var_www_html: null
+  moodle-data_var_log_nginx: null
```

Listing 10: Differenz der Crane-Konfiguration zu der ausgelieferten Fassung

Die Anpassung umfasst im groben die folgenden Ergänzungen:

- PostgreSQL Container für die Datenbanken der Moodle Instanzen
  - Die Zugangsdaten sind entsprechend für das MDK-Tool zu wählen
  - Der Port wird exposed, damit das Hostsystem sich ebenfalls verbinden kann
  - Sowie zugehörige Dump-Abläufe referenziert
- Nginx mit PHP-FPM Container als Webserver für die Moodle Instanzen
  - Erhält den PostgreSQL Container gelinkt das damit den Datenbankzugriff ermöglicht
  - Wird mit entsprechenden Umgebungsvariablen an den NGXPX-Container als Subpfad angehängt
  - Diverse Containeroptionen aktiviert um Modifikationen vor dem Start zu erlauben

Dazu werden die passenden Systemservice-Konfigurationen generiert und gestartet. Gleichzeitig mounten wir das DocumentRoot-Volume des Moodle-Containers auf den gleichnamigen Pfad des Host-Systems, damit MDK keine Fehlinterpretation bei der Instanziierung durch den Pfad erfährt:

```
crane generate --config /etc/crane/crane.yaml --template /etc/crane/systemd.tmpl --output
↪  /etc/systemd/system/docker-%s.service moodle-db #service file for moodle-db
crane generate --config /etc/crane/crane.yaml --template /etc/crane/systemd.tmpl --output
↪  /etc/systemd/system/docker-%s.service moodle #service file for moodle
systemctl daemon-reload #reload since new service files available
systemctl start docker-moodle-db.service
systemctl start docker-moodle.service
mount --bind /var/lib/docker/volumes/moodle-data\_var\_www\_html/\_data/ /var/www/html/
↪  #imitate path behavior by mounting instead of symlinking of the document root
```

Listing 11: Einrichten der Dienste für den Moodle- und Datenbank-Container

Wie in der Crane-Anpassung angekündigt, sind Modifikationen vor dem Starten des Containers notwendig. Zum Einen muss auch hier wieder die Nginx-Konfiguration so angepasst werden, dass die Slash-Arguments[13] akzeptiert werden. Für diese Anpassung wird die vom Container zur Verfügung gestellte[16] benutzerdefinierte Nginx-Konfiguration genutzt, indem diese am richtigen Pfad `/var/www/html/conf/nginx/nginx-site.conf` bereitgestellt wird:

```
    server {
        listen   80;
        listen   [::]:80

        root /var/www/html/moodle;
        index index.php index.html index.htm;

        server_name _;
        charset utf-8;
        sendfile off;

        error_log /dev/stdout info;
        access_log /dev/stdout;

        location / {
            try_files $uri $uri/ =404;
            autoindex on;
        }

        error_page 404 /404.html;
        location = /404.html {
            root /var/www/errors;
            internal;
        }

        location ~ [^/].php(/|$) {
            fastcgi_split_path_info ^(.+.php)(/.+)$;
            fastcgi_pass unix:/var/run/php-fpm.sock;
            fastcgi_param PATH_INFO       $fastcgi_path_info;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param SCRIPT_NAME     $fastcgi_script_name;
            fastcgi_index index.php;
            include fastcgi_params;
            fastcgi_read_timeout 900;
        }
    }
```

Listing 12: Nginx Konfiguration für den Moodle-Container

Desweiteren fehlt dem aktuellen Moodle-Container die PHP-Erweiterung um sich überhaupt zu einem Postgres-Server verbinden zu können. Dafür kann die Templating-Funktionalität[17] des Containers genutzt werden, indem ein Script an einem bestimmten Pfad `/var/www/html/scripts/00-phpextensions.sh` bereitgestellt wird:

```
#!/usr/bin/env bash
apk add postgresql-dev postgresql-client #install postgesql client and headers
docker-php-ext-install pgsql #compile and install pgsql php extension
```

Listing 13: Skript zur Installation von PostgreSQL und ihrer PHP-Extension für innerhalb des Moodle-Containers

Und damit diese Änderungen nun übernommen werden und kontrolliert werden kann, muss der Container neu gestartet werden:

```
systemctl restart docker-moodle.service
```

Listing 14: Neustarten des Moodle Docker-Dienstes

5.5.2 Moodle Development Kit einrichten

Als nächstes muss das MDK eingerichtet werden, welches für das Administrieren der verschiedenen Moodle Instanzen verwendet wird. Der schwierigere Teil ist die Konfiguration dessen, damit man nachher mit wenigen Befehlen ganze Instanzen erstellen, neu bauen und automatisch mit Skripten erweitern kann.

Für Windows

Für Windows steht MDK leider nicht zur Verfügung. Hiermit ist man zur manuellen Einrichtung nach der offiziellen Anleitung von Moodle angewiesen.

Für Linux

Für die Installation folgt man am besten ihrer Anleitung auf Github.[18] Für die Initialisierung verwenden wir die Pfade `/var/www/html/storage` als Ablage und `/var/www/html/moodle` für die Instanzen. MDK ruft PHP-Skripte der Moodle Instanzen für die Installation auf, deshalb sind einige PHP-Pakete dafür erforderlich:

```
apt-get install php-cli postgresql-client php-pgsql php-xml php-curl php-gd php-zip php-intl
↪  php-mbstring
mkdir /var/www/html/storage
mkdir /var/www/html/moodle
# and the installation and initialisation steps for mdk itself
```

Listing 15: Installation und Einrichten der Abhängigkeiten von MDK

5.5.3 Moodle Development Kit konfigurieren

Der Befehl `mdk init` setzt noch nicht alle gewünschten Optionen, hierfür muss die Datei `~/.moodle-sdk/config.json` angepasst werden.

Lokales Windows und Linux

Für den lokalen Linux und Windows-Rechner (mit entsprechend angepassten Pfaden) reicht die folgende Konfiguration:

```
{
    "dirs": {
        "www": "/home/your-user/some-path/moodle",
        "storage": "/home/your-user/some-path/storage"
    },
    "ci": {
        "token": null
    },
    "db": {
        "mariadb": {
            "passwd": "admin",
            "user": "admin"
        }
    },
    "editor": null,
    "host": "localhost:8000", "_comment": "or whatever port you're using",
    "path": "",
    "login": "admin",
    "passwd": "admin", "_comment": "change this pw!"
}
```

Listing 16: Konfiguration für MDK für die lokale Umgebung

Linux-Server auf Basis des SE2-Image der HSR

Für den HSR-Server sind weitere Einstellungen notwendig, damit das Resultat überhaupt funktionieren kann. Zum Einen ist ausgehend der SSH-Port 22 nicht benutzbar, also musste auf Git-Repositories über HTTP zugegriffen werden, zum Anderen muss der PostgreSQL-Container als Host eingetragen werden, sowie die Moodle-Konfiguration "ReverseProxy" aktiviert werden, weil sich der Moodle-Container hinter dem NGXPX-Container befindet:

```
{
    "dirs": {
        "www": "/var/www/html/moodle",
        "storage": "/var/www/html/storage"
    },
    "ci": {
        "token": null
    },
    "db": {
        "pgsql": {
            "host": "postgres",
            "user": "admin",
            "passwd": "admin"
        }
    },
    "host": "sinv-56015.edu.hsr.ch",
    "remotes": {
        "stable": "https://git.in.moodle.com/moodle/moodle.git",
        "integration": "https://git.in.moodle.com/moodle/integration.git"
    },
    "editor": null,
    "path": "moodle",
    "forceCfg": {
        "reverseproxy": 1
    },
    "login": "admin",
    "passwd": "admin", "_comment": "change this pw!"
}
```

Listing 17: Konfiguration für MDK für den HSR-Server

Desweiteren wird ausgehenden Mailversand über Moodle benötigt. Da hierfür keine Konfigurationsoption zur Verfügung steht, muss diese Änderung in der Datenbank der Moodle Instanz eingetragen werden. Dafür ist die Verwendung eines eigenen Skripts `~/.moodle-sdk/scripts/hsrsmtp.php` für MDK zu empfehlen:

```
<?php

/**
* Sets the smtp host settings to HSR smarthost
*/

define('CLI_SCRIPT', true);
require(dirname(__FILE__).'/config.php');

// Fix smtp to use hsr address
$smtphosts = $DB->get_record('config', array('name' => 'smtphosts'));
if ($smtphosts && $smtphosts->value != $hsrsmarthost) {
    mtrace('Fixing SMTP to use HSR smarthost');
    $smtphosts->value = $hsrsmarthost;
    $DB->update_record('config', $smtphosts);
    // have to clear cache now
    mtrace('Clearing cache');
    passthru('php admin/cli/purge_caches.php');
}
```

Listing 18: PHP Skript für MDK zur Einrichtung der HSR SMTP-Einstellungen

Moodle Instanzen erstellen

Endlich sind alle Vorbereitungen abgeschlossen, die Instanzen können erstellt und verwendet werden. Gleichzeitig bietet MDK vorgefertigte Skripte um die Instanzen für Entwickler optimal einzurichten:

```
#for local installations
mdk create --version 34 --install --engine mysql --integration --identifier myinstance #use latest beta probably with an instance name matching your webservers document root path
#for the server
mdk create --version 33 --install --engine pgsql # creates instance called stable_33
#for all
mdk run setup stable_33 # enables some dev options, creates dummy users and a course
mdk run webservices stable_33 # entirely enables the web services
```

Listing 19: Einrichtung von Moodle Instanzen

Für den HSR-Server muss auch die geschriebene SMTP-Korrektur ausgeführt werden. Gleichzeitig kann man noch ein Plugin installieren lassen, womit man die Mail-Einstellungen in der Instanz testen kann:

```
mdk run hsrsmtp stable_33 # custom script to set smtphosts config in db
mdk plugin install local_mailtest stable_33
```

Listing 20: Konfiguration des Moodles für die HSR SMTP-Einstellungen

Hat alles geklappt, ist die Instanz fertig eingerichtet und kann über die Adresse lokal über `http://local-addr/stable_33/` und beim Server über `http://server-addr/moodle/stable_33/` aufgerufen werden.

5.5.4 StudentQuiz Plugin installieren

Das StudentQuiz-Modul muss über das Git-Repo ausgecheckt werden, damit man auf den `develop`-Branch wechseln kann, in welchem sich die aktive Entwicklung befindet und jederzeit aktualisiert werden kann.

Lokales Windows und Linux

Für die Installation des StudentQuiz Plugins mit grafischen Tools sind die folgenden Schritte notwendig:

- Mithilfe der Git-Software das StudentQuiz Repo nach `some-path/moodle/mod/studenquiz` exakt auschecken
- Den notwendigen Datenbank-Upgrade über die lokale Webseite durchführen

Ansonsten sind die folgenden Befehle auf der Kommandozeile zu benutzen:

```
cd /home/your-user/some-path/moodle/myinstance #navigate to your instance
git clone git@github.com:frankkoch/moodle-mod_studentquiz.git mod/studentquiz #clone the repo to this exact folder
php admin/cli/upgrade.php --non-interactive #upgrade database according to new plugin
```

Listing 21: StudentQuiz Installation auf der Kommandozeile

5.5.5 Testing

Unit-Tests

Am schnellsten und einfachsten ist die Initialisierung der Unit-Test-Umgebung mit MDK zu realisieren:

```
1    mdk phpunit myinstance #initialize phpunit environment
2    mdk phpunit --help #see options how to run the tests
```

Listing 22: Einrichten der Unit Testumgebung

Alternativ kann mit diesem Moodle Guide[20] die Umgebung manuell eingerichtet und durchgeführt werden.

Beachte: Ein gesamter Durchlauf aller Unit-Tests dauert mehrere Stunden! Ein einzelner Test kann zum Beispiel folgendermassen ausgeführt werden:

```
1    #run single unit test with mdk
2    mdk phpunit -r hsrbamoodle -t mod_studentquiz_viewlib_testcase -u
     ↪  mod/studentquiz/tests/viewlib_test.php
3    #run single unit test manually
4    ./vendor/bin/phpunit mod_studentquiz_viewlib_testcase mod/studentquiz/tests/viewlib_test.php
```

Listing 23: Ausführen eines Unit Tests

Acceptance-Tests

Am schnellsten und einfachsten ist die Initialisierung der Behat Testumgebung mit MDK zu realisieren:

```
1    mdk behat myinstance #initialize behat environment
2    mdk behat --help #see options how to run the tests
```

Listing 24: Einrichten der Behat Testumgebung

Alternativ kann mit diesem Moodle Guide[21] die Umgebung manuell eingerichtet und durchgeführt werden.

Beachte: Ein gesamter Durchlauf aller Acceptance Tests dauert wenige Stunden! Ein einzelner Test kann zum Beispiel folgendermassen ausgeführt werden:

```
1    #run single acceptance test with mdk
2    mdk behat -r hsrbamoodle --feature=mod/studentquiz/tests/behat/add_studentquiz.feature
3    #run single acceptance test manually
4    php admin/tool/behat/cli/run.php
     ↪  --feature=$(pwd)/mod/studentquiz/tests/behat/add_studentquiz.feature
5    #or
6    vendor\bin\behat.bat --config \Users\Dionysius\Projects\bht_moodledata\behatrun\behat\behat.yml
     ↪  \Users\Dionysius\Projects\hsrbamoodle\mod\studentquiz\tests\behat\backup.feature
```

Listing 25: Ausführen eines Acceptance Tests

5.5.6 Lokale Entwicklungsumgebung

IDE

Als Referenz-IDE wird PhpStorm verwendet. Die nachfolgenden Anleitungsschritte sind auf diese IDE abgestimmt. Es ist natürlich möglich andere IDEs zu verwenden, doch nachfolgende Konfigurationsschritte werden natürlich abweichen.

Debugging

XDebug[6] ist eine PHP Erweiterung, welche Debugging- und Profiling-Techniken anbietet. Die Integration in PhpStorm vereinfacht das Debuggen wesentlich. Für Installation und Anwendung sollte am Besten der Anleitung von JetBrains[7] gefolgt werden.

Profiling

Tideways[23] ist eine PHP Erweiterung, welche ebenfalls Profiling-Techniken anbietet. Sie ist zu Facebook's XHProf API kompatibel und wird von Moodle unterstützt.[22] Leider bietet PhpStorm keine Integration für diese Erweiterung an, auch ist die Analyse auf der Herstellerseite kostenpflichtig, deshalb muss für die Ansicht auf XHGUI zurückgegriffen werden.

Beginne mit der Installation der Extension nach der Anleitung von Tideways.[24] Um diese ohne die Tideways-Plattform zu nutzen, füge die folgende Zeile zur Konfigurationsdatei `/etc/php/7.1/mods-available/tideways.ini` hinzu:

```
1    tideways.auto_prepend_library=0
```

Listing 26: Konfiguration von Tideways

Sowie starte php-fpm neu um diese Einstellung zu übernehmen:

```
1    service php7.1-fpm restart
```

Listing 27: Neustarten des PHP Dienstes

Aktiviere Profiling in Moodle über `SiteAdministration>Development>Profiling` und setze `profilingincluded` zu `/mod/studentquiz/*`. Sollte der Navigationspunkt fehlen, ist die Erweiterung in PHP nicht richtig aktiviert. Navigiere im StudentQuiz, damit der Profiler erste Daten sammeln kann. Zum Schluss kann über `SiteAdministration>Development>ProfilingRuns` die angelegten Profile betrachtet werden. Die einzelnen Durchläufe können auch exportiert werden und KCachegrind[25] betrachtet werden.

Slow Query

Vermutlich haben die oberen Debugging-Techniken bereits auf langsame Requests aufmerksam gemacht. Diese Technik erlaubt das Analysieren von sogenannten langsamen Datenbankabfragen. Als erstes muss das Reporting von langsamen Queries aktiviert werden, folge hier am Besten dieser Anleitung.[26] Der Abschnitt dieser Konfiguration sieht ungefähr so aus:

```
# Enable the slow query log to see queries with especially long duration
slow_query_log          = 1
slow_query_log_file     = /var/log/mysql/mariadb-slow.log
long_query_time         = 1
#log_slow_rate_limit    = 1000
#log_slow_verbosity     = query_plan
log-queries-not-using-indexes
```

Listing 28: Konfiguration der Datenbank für Slow-Queries

Für die Analyse des Logs bietet sich das Percona-Toolkit[27] an, welches wie folgt installiert werden kann:

```
apt-get install percona-toolkit
```

Listing 29: Installation des Percona-Toolkits

Es ist zu empfehlen, eine strenge Analyse nach einem Acceptance-Test-Durchlauf durchzuführen.

5.6 Leistungsmessungen

5.6.1 Code Metrik

Verglichen mit den rund 30'000 Dateien und 3 Millionen Zeilen Code der Moodle Version 3.4 ist der Code von StudentQuiz sehr überschaubar. StudentQuiz enthält nach sowie vor unserer Arbeit 36 PHP Dateien auf welche sich rund 3'000 Zeilen PHP Instruktionen verteilen. In Tabelle 2 werden die Anzahl PHP Dateien und die Anzahl der PHP Codezeilen zwischen der Ausgangslage und dem Endzustand des Projektes verglichen. Dabei ist zu beachten, dass bei StudentQuiz 3.0.0 davon ausgegangen wird, dass das Question Behaviour StudentQuiz nicht mehr dazu gezählt werden soll, da dessen Logik nun in StudentQuiz 3.0.0 integriert wurde.

Plugin	Anzahl PHP Dateien	Zeilen PHP Code
Modul StudentQuiz 3.0.0	36	3201
Question Behaviour StudentQuiz	4	256
Modul StudentQuiz 2.0.3	32	2397
Delta	0	548
Delta (ohne Question Behaviour)	4	804

Tabelle 2: Code Statistik mit phpStat für phpStorm (ohne Kommentarzeilen)

5.6.2 Performance und Lasttest

Eine wichtige Anforderung der Aufgabenstellung war die Verbesserung der Performance der verschiedenen Views des StudentQuiz Plugins, sowie indirekt die der Kursübersichtsseite, welche durch die versteckten Quiz-Instanzen beeinträchtigt war. Namentlich sind dies folgende Testszenarien:

- Ladezeit der Kurs Übersichtsseite
- Ladezeit der StudentQuiz Übersichtsseite
- Ladezeit der StudentQuiz Statistikseite
- Ladezeit der StudentQuiz Rankingseite
- Einrichtungszeit der Aktion "Start Quiz"

Die Lasttests und Performance-Messungen werden mit JMeter durchgeführt. Jedes Testszenario wird einzeln auf ein Moodle mit dem alten StudentQuiz in der Version 2.0.3 und darauffolgend auf dem gleichen Serversystem in ein Moodle mit dem neuen StudentQuiz in der Version 3.0.0 ausgeführt. Das Testsetup umfasst:

- Beide Moodle Instanzen sind nach Werkseinstellung und im Developer-Modus eingerichtet, jeweils in der neusten Version 3.4, die beide Plugin-Versionen beherrschen.
- Diese Moodle Instanzen sind auf dem gleichen nginx-Docker-Container um dieselbe Leistung zu garantieren.
- Der Ziel-Server ist ein von der HSR-Abteilung Informatik eingerichteter virtueller Server auf Basis des SE2 Images, gemäss erhaltenem Datenblatt folgende Eigenschaften aufweist: VMWare virtualisiert, Ubuntu 14.04, 1 vCPU max. 2.2 GHz, 1 GB RAM, 15 GB Disk, 1 VMXNET3 LAN-Adapter. Offensichtlich ist das erhaltene Datenblatt nicht korrekt beschrieben, denn innerhalb der virtuellen Maschine können wir folgende Eigenschaften feststellen: 2 vCPU max. 2.2 GHz, 4 GB RAM, 80 GB Disk. Die Internetgeschwindigkeit ist nicht angegeben, geschätzt wird 1 Gbit/s symmetrisch, da eine kurze manuelle Messung ~800 Mbit/s ergeben hat.
- Der Quell-Rechner ist ein privater PC mit folgender Ausstattung: Bare-Metal, Windows 10, 4-Kerne CPU max. 3.2 GHz, 16 GB RAM, 512GB SSD Disk, 1 GBit/s LAN-Adapter und ebenfalls 1 Gbit/s symmetrischem Internet-Anschluss. Dieser Quellrechner verfügt somit über ausreichend Ressourcen um diese Tests alleine ausführen zu dürfen.
- Die Latenz zwischen Quell- und Zielsystem schwankt zwischen 7 und 10ms. Somit ist diese für die Messung geeignet und durch die stabile niedrige Latenz vernachlässigbar.
- In beide Moodle Instanzen ist der "Wirtschaftsinformatik 1 FS2017"-Kurs von Frank Koch als Referenzkurs importiert worden. Dieser enthält ausreichend umfangreiche und praxisnahe Testdaten für die Performance-Messung:
  - 82 Benutzer
  - 323 Fragen

- 1'851 Fragesets
- 2'401 Attempts auf die Fragesets
- 56'662 Beantwortungen auf Fragen

JMeter Szenario Template

Die Vorbereitung des Testplans gestaltete sich schwieriger als erhofft, da für eine aussagekräftige Messung das Caching von Moodle, die Variation der Benutzerkonten, sowie die Benutzer-Logins berücksichtigt werden müssen.

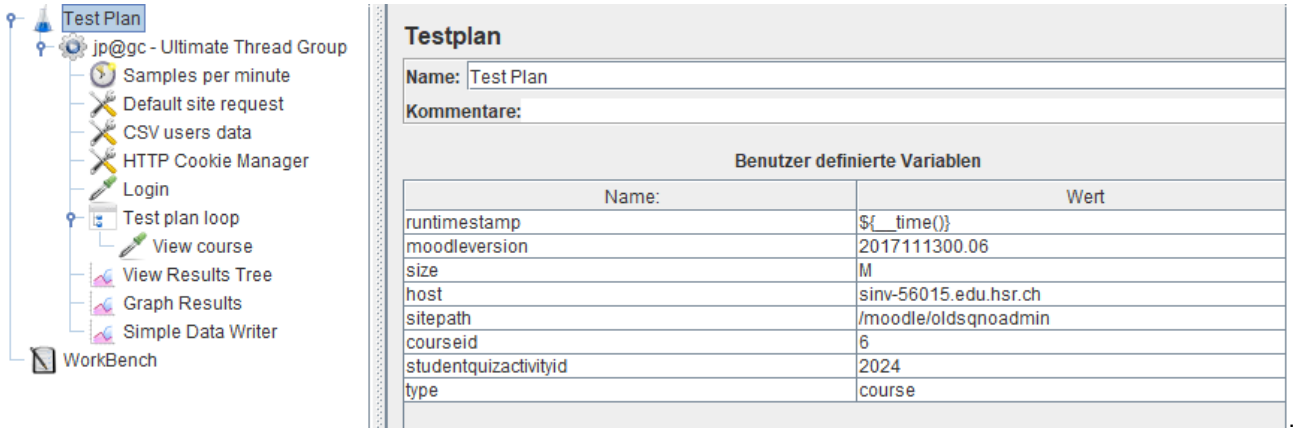


Abbildung 24: Template der JMeter Konfiguration zur Lasttest und Performance-Messung

Hierfür gliedert sich ein Testplan (siehe Abbildung 24) in die folgenden Abschnitte:

- Die Warm-Up Phase: Ein kompletter Login-, eine Kursansicht und Logout-Prozess (nicht abgebildet) vor der eigentlichen Messung, um Moodle die Möglichkeit zu bieten, systemweit relevante und für jeden Benutzer spezifische Informationen cachen zu können. (Nebenbemerkung: Von einem Plugin zur Verfügung gestellten externen Ressourcen wie das Javascript vom StudentQuiz Behavior-Teil wird in den systemweiten Cache geladen)
- Die Durchführung der jeweiligen Tests: Mit einem Login und der zu messender Seite. Hierbei wird nur die zu messende Seite in der Auswertung berücksichtigt.

In JMeter ist eine Thread Group so konfiguriert, dass eine stetig wachsende Belastung realisiert werden kann, welche bei Volllast bis zu 100 Threads gleichzeitig laufen lässt (Siehe Abbildung 25).

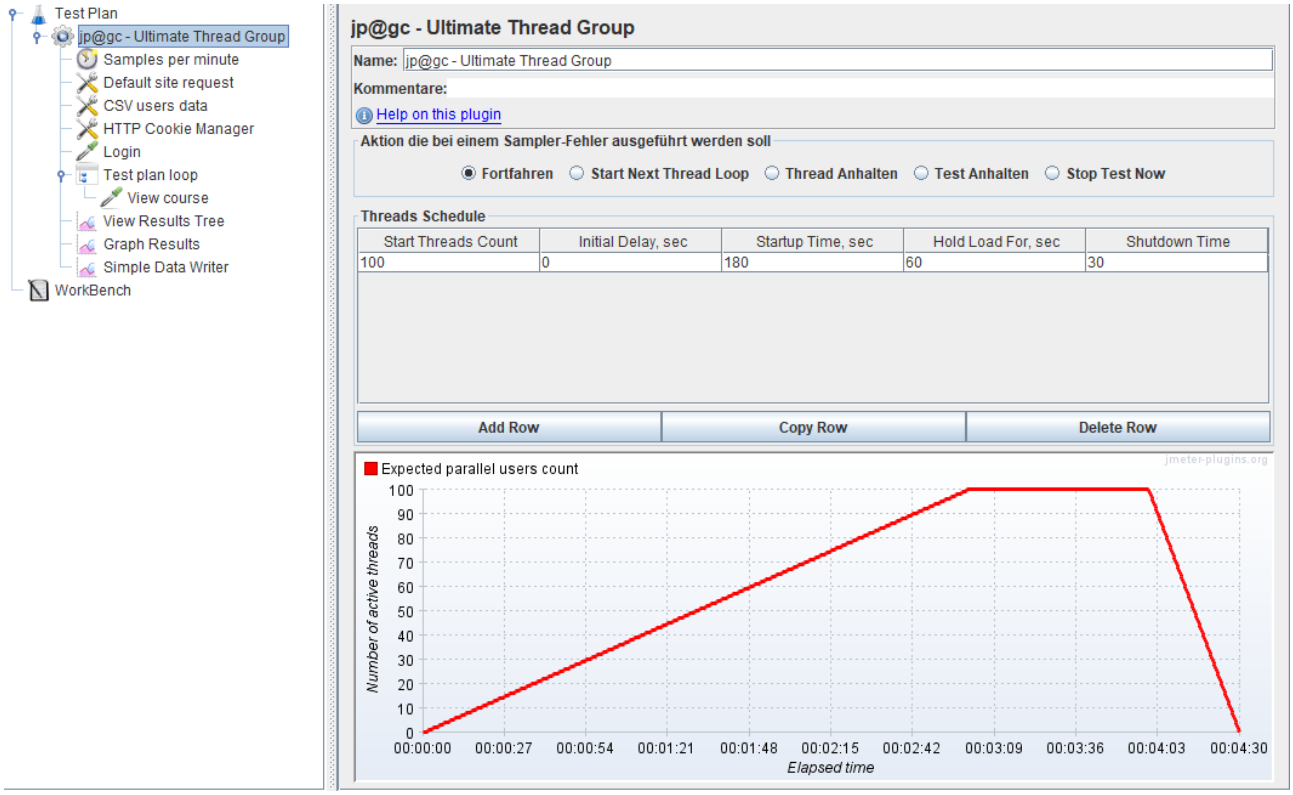


Abbildung 25: JMeter Konfiguration für steigende Anzahl Ausführungs-Threads

Ein Thread entspricht somit einem virtuellen User, welcher 5 Requests pro Minute abschickt (Siehe Abbildung 26). Dieser Wert dient primär nicht dazu, den echten User zu imitieren, sondern Pausen in Threads zu erlauben, sodass die anderen Threads ihrer Aufgabe nachkommen können.

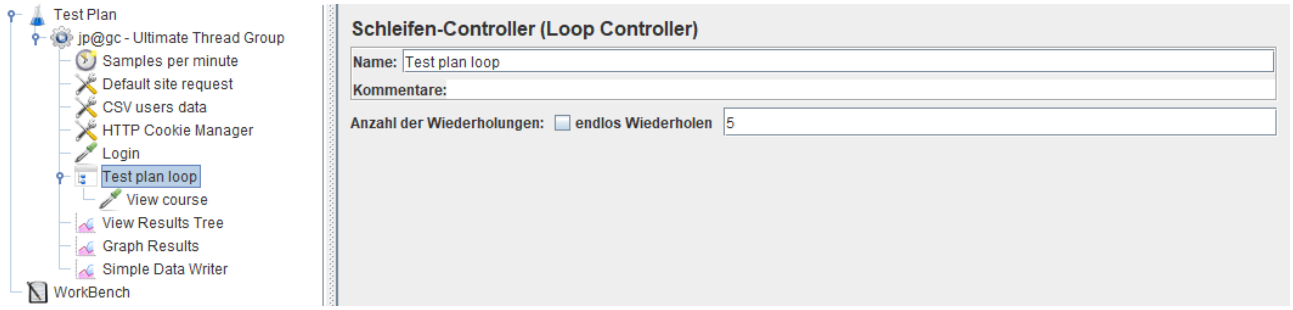


Abbildung 26: JMeter Konfiguration für Anzahl Requests pro Thread pro Minute

Der Umfang des Testplans umfasst 100 Benutzer aus einer CSV-Datei (Siehe Abbildung 27). Um diese Anzahl Benutzer zu ermöglichen, wurde der Kurs um 18 weitere Benutzer ergänzt. Jeder Thread nimmt sich beim Durchlauf den nächsten Eintrag und falls dieser am Ende angekommen ist, beginnt er wieder von vorne. Somit kann ein Benutzer diese Aktionen gegebenenfalls mehrmals durchführen.

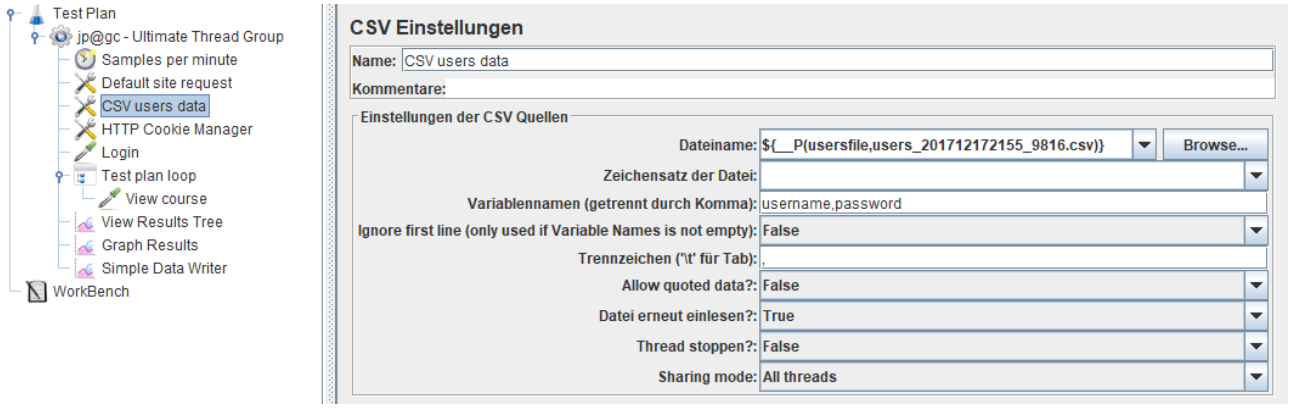


Abbildung 27: JMeter Konfiguration für die Verwendung verschiedener Benutzer pro Thread

Beginnend muss einmalig der Benutzer-Login (Siehe Abbildung 28) ausgeführt werden, bevor er die zu prüfende Zielseite öffnen kann.

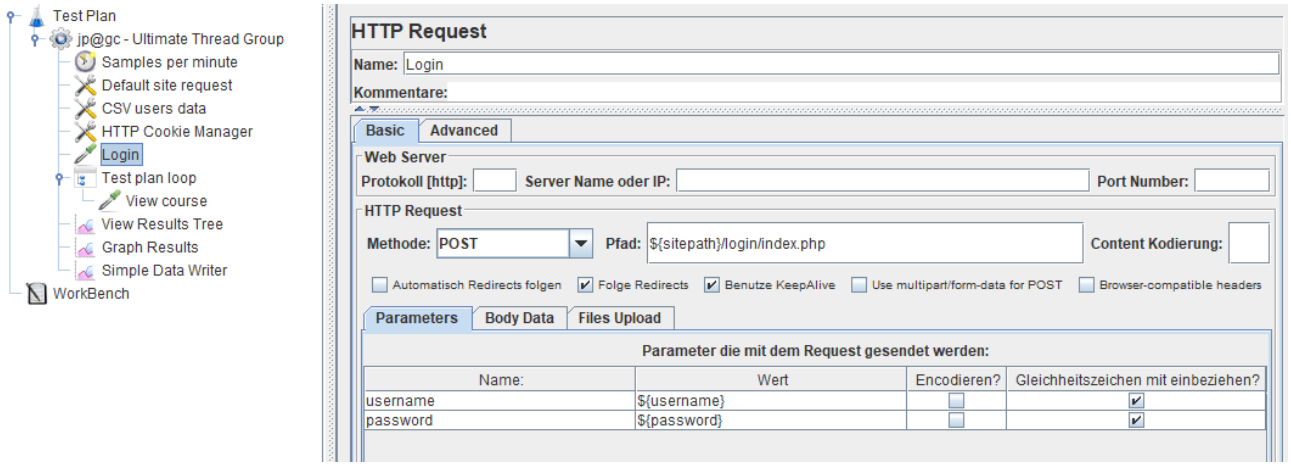


Abbildung 28: JMeter Konfiguration Request für den Benutzer-Login

Durch die sehr niedrig gewählte Anzahl Requests pro Thread pro Minute soll der Thread seine Aktion auf die zu testende Zielseite insgesamt 5 Mal durchführen (Siehe Abbildung 29), sodass eine verwertbare steigende Überlappung von Threads mit steigender Anzahl Threads erreicht wird. Die Warm-Up Phase ist davon nicht betroffen und erhält eine Durchführung konfiguriert.

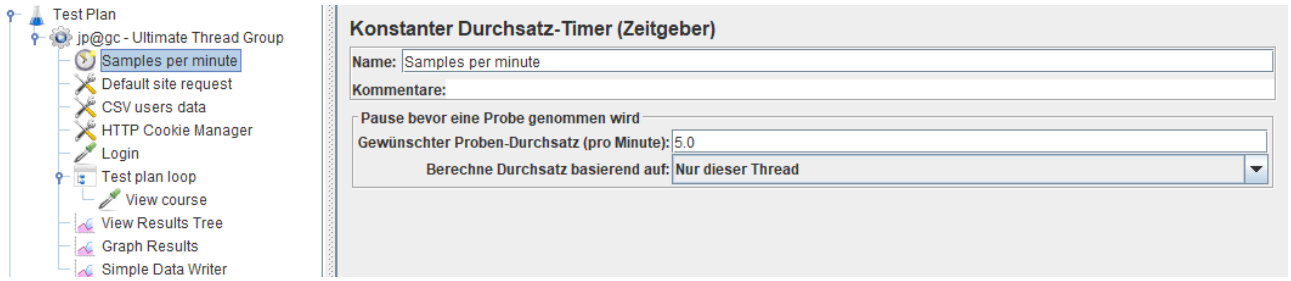


Abbildung 29: JMeter Konfiguration für die Anzahl Wiederholungen pro Thread

Schlussendlich die Konfiguration der zu prüfenden Seite. Der Request ist im voraus so zusammengestellt, dass beim Aufruf dieses Schritts die gewünschte Zielseite geladen wird (Siehe Abbildung 30). Dieser Schritt unterscheidet sich zwischen den einzelnen Szenarien.

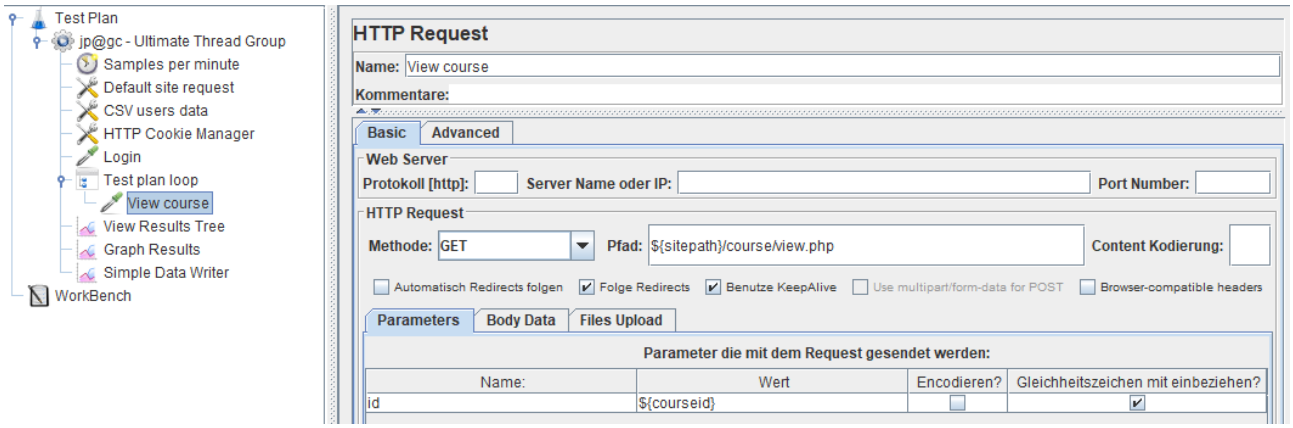


Abbildung 30: JMeter Konfiguration Request Endpoint und Request Body Daten

Alle weiteren Elemente des Test Plans dienen entweder zum Erfassen des Default-Verhaltens, zur Auswertung der Ergebnisse zu Debug-Zwecken oder zur effektiven Speicherung in eine Datei.

Sobald der Testplan dem Gewünschten entspricht, kann dieser auf der CLI mit folgendem Befehl ausgeführt werden:

```
1 meter -n -t testplan_<scenario>.jmx -l result.txt -e -o report
```

Dadurch erhält man die Metainformationen der einzelnen Requests im result.txt sowie einen aggregierten HTML-Report. Die Messergebnisse sind für die präzisere und anschaulichere Auswertung jedoch über BlazeMeter[32] verarbeitet.

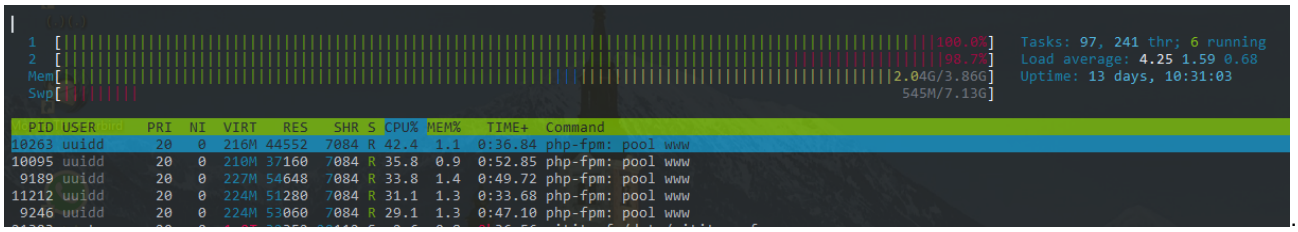


Abbildung 31: Performance Test Momentaufnahme der Volllast auf dem Ziel-System

Wie in Abbildung 31 zu sehen is, ist schon durch die Warm-Up Phase der Ziel-Server ausgelastet. Somit werden generell keine herausragenden Ergebnisse durch die kontinuierliche Erhöhung der Belastung erwartet. Durch die hohe Grundbelastung werden Antwortzeiten innerhalb 5'000 ms als vertretbar betrachtet. Nachfolgend sind die Ergebnisse der einzelnen Szenarien zu finden.

Kurs Übersichtsseite

Ab Moodle Version 3.3 und StudentQuiz bis Version 2.0.3 hat sich das Problem ergeben, dass ein Restore eines Kurses, welches ein StudentQuiz enthält, 999 Sections erstellt. Diese entstehen auch, wenn das Backup-Archiv keine zutreffenden Daten der einzelnen Sections enthält. Die Erklärung des Problems ist in der Analyse im Abschnitt 4.3.10 näher beschrieben, die Implementierung der Lösung ist in Abschnitt 4.4.1 zu finden. Die Ladezeit hat sich auch negativ auf das Rendern der Seite im Browser bemerkbar gemacht, da entsprechend der Anzahl Sections Anzeige-Instruktionen vom Browser dargestellt werden mussten. Auch hat die Übermittlung dieser Anzeige-Instruktionen zusätzliche Aufwände in der Datenübertragung erfordert.

Diese Kurs-Übersichtsseite wird auschliesslich von Moodle generiert, auf welches ein Plugin keinen Einfluss hat. Die Lösung des Section-Problems erspart die beiden überflüssigen Teilaspekte, Rendering- und Übermittlungs-Aufwand, in der gesamten Ladezeit.

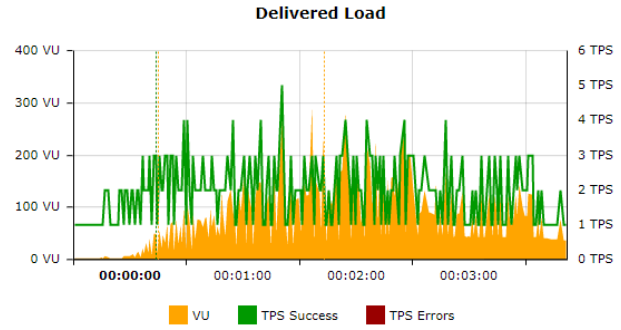
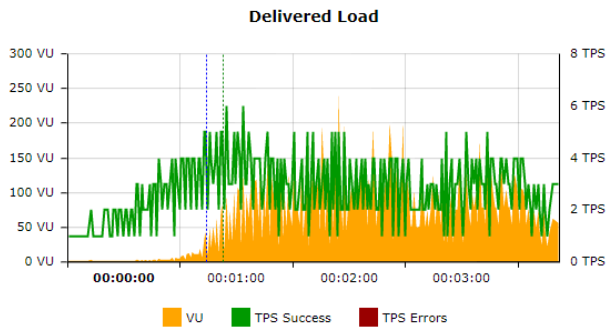
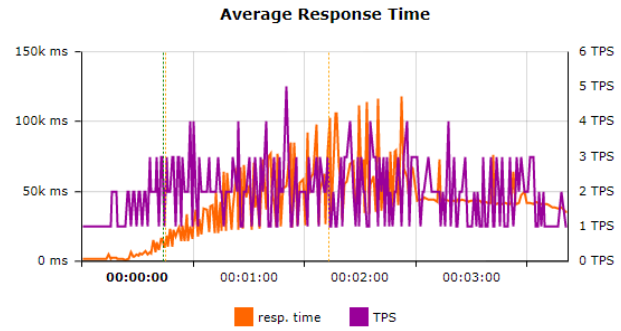
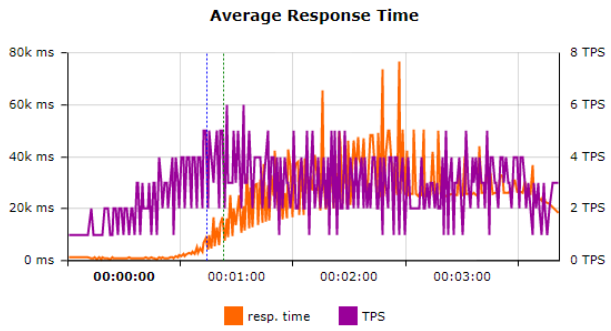
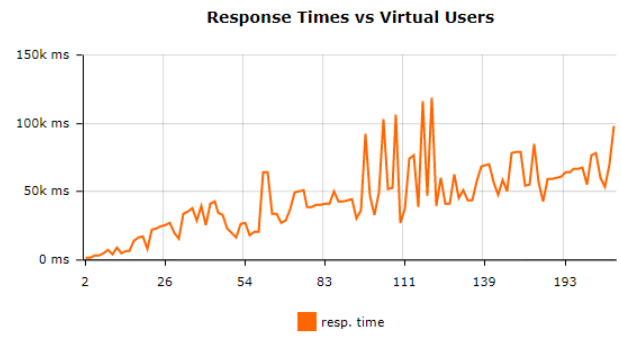
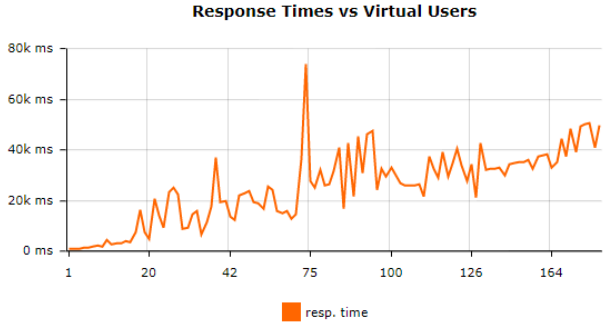
StudentQuiz v2.0.3	StudentQuiz v3.0.0
	
	
	

Tabelle 3: Performance-Test Ergebnisse: Kurs Übersichtsseite, isoliert

Im direktem Vergleich

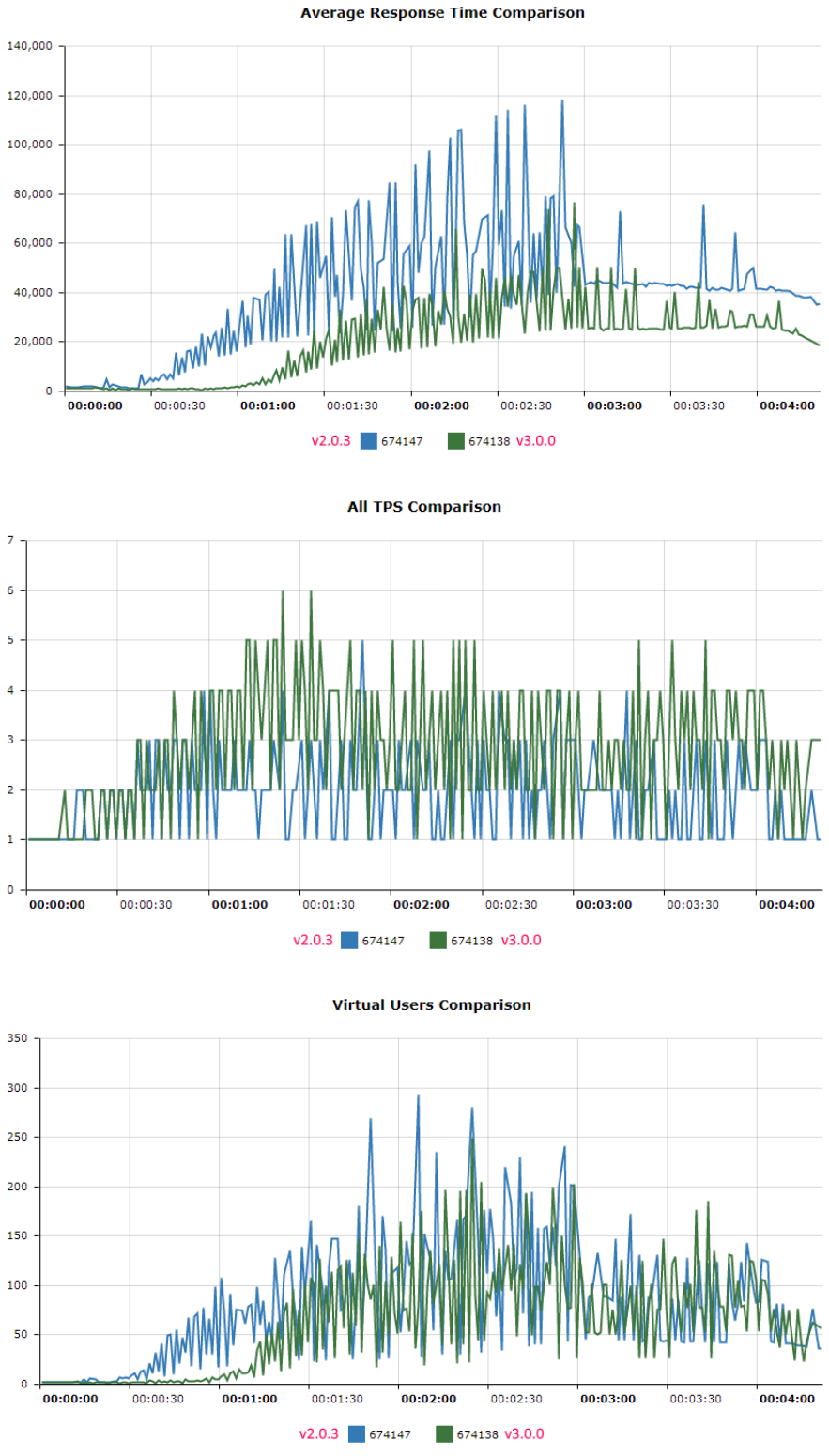


Tabelle 4: Performance-Test Ergebnisse: Kurs Übersichtsseite, Vergleich

Metrik	**v2.0.3**	**v3.0.0**	**Verbesserung**
TPS bei 2'000 ms Antwortzeit	1-2	2-4	+1-2
VU bei 2'000 ms Antwortzeit	5	9	+4
TPS bei 5'000 ms Antwortzeit	1-3	2-5	+1-2
VU bei 5'000 ms Antwortzeit	8	16	+8
Sättigungspunkt erreicht am	~00:43	~01:15	+00:32
Antwortzeit am Sättigungspunkt	~16'000 ms	~8'000 ms	+8'000 ms

Tabelle 5: Performance-Test Ergebnisse: Kurs Übersichtsseite, Verbesserungen

Diese Messungen ergeben, dass sowohl die Antwortzeit wie auch das Verarbeitungsvolumen deutlich verbessert werden konnten. Sehr vermutlich geht dies Hand in Hand, durch die effizientere Verarbeitung kann ein höheres Volumen bearbeitet werden. Der Sättigungspunkt zeigt auf, dass dieser deutlich später auftritt und unter voller Last auch doppelt so schnell geantwortet werden kann.

StudentQuiz Übersichtsseite

Die Übersichtsseite von StudentQuiz hat bei der Anzeige des Fragepools in der Implementierung ein paar Verbesserungen, wie in Abschnitt 4.4.2 beschrieben, erhalten. Durch die Personal Learning Assistance hat diese Seite neue persönliche, zu dieser Frage zugehörigen, Werte in der Tabelle und einen neuen Block mit einem Top10 Ranking erhalten, welche ihre Daten frisch aggregieren. Insofern werden sich die Performance-Gewinne der Korrekturen und -Einbussen durch die neuen Features in etwa ausgleichen.

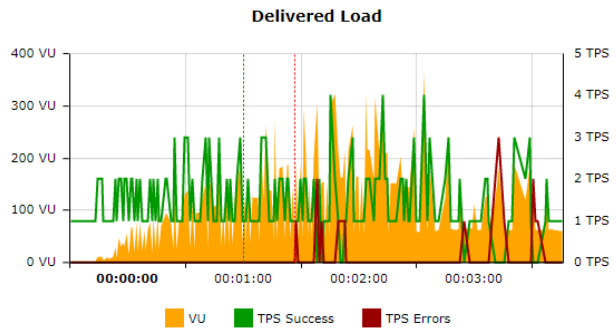
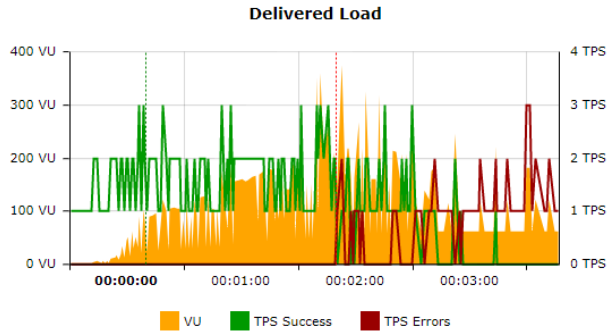
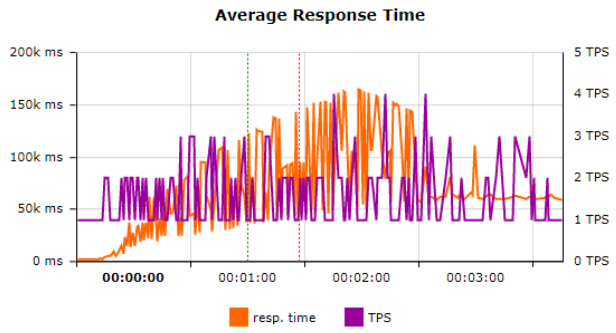
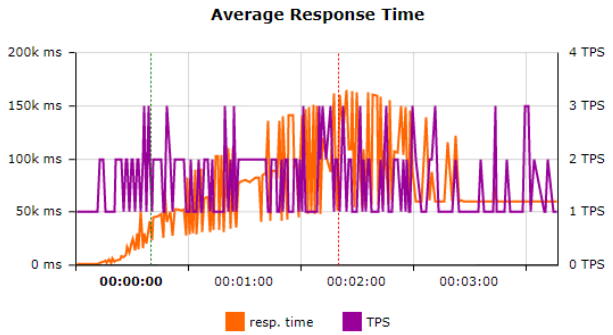
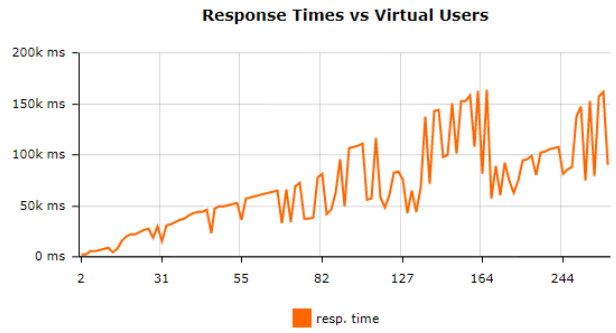
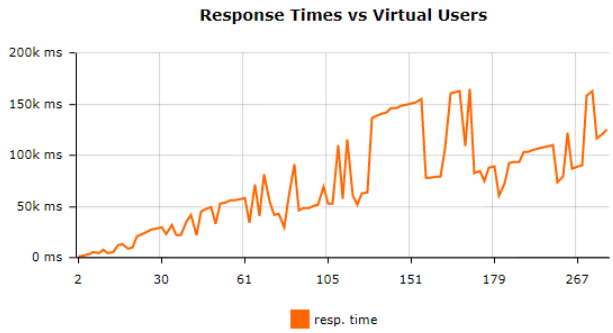
StudentQuiz v2.0.3	StudentQuiz v3.0.0
	
	
	

Tabelle 6: Performance-Test Ergebnisse: StudentQuiz Übersichtsseite, isoliert

Im direktem Vergleich

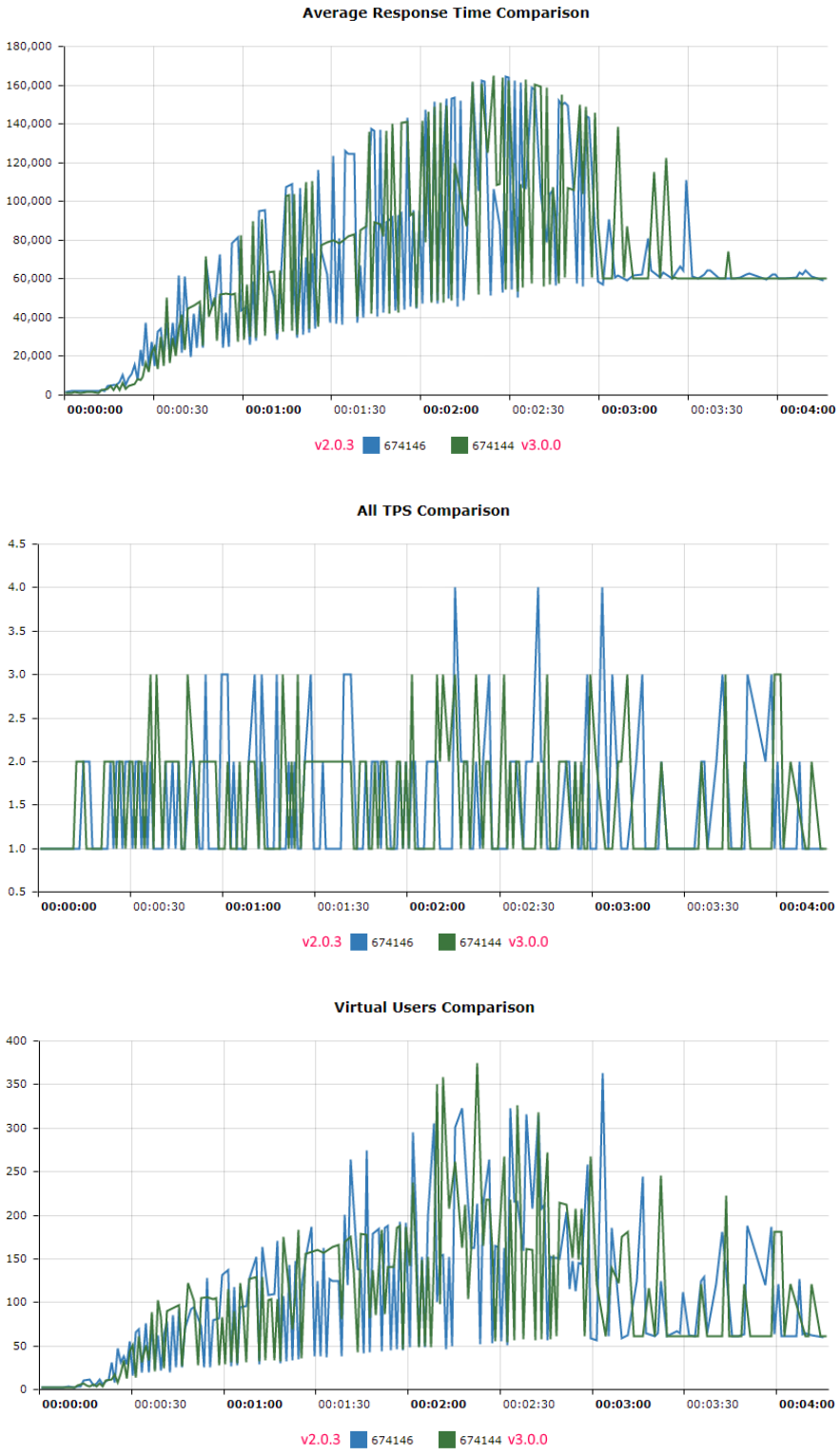


Tabelle 7: Performance-Test Ergebnisse: StudentQuiz Übersichtsseite, Vergleich

Metrik	**v2.0.3**	**v3.0.0**	**Verbesserung**
TPS bei 2'000 ms Antwortzeit	1	1-2	+0-1
VU bei 2'000 ms Antwortzeit	3	5	+2
TPS bei 5'000 ms Antwortzeit	1-2	1-2	0
VU bei 5'000 ms Antwortzeit	11	12	+1
Sättigungspunkt erreicht am	~00:54	~00:39	-00:15
Antwortzeit am Sättigungspunkt	~37'000 ms	~35'000 ms	+2'000 ms

Tabelle 8: Performance-Test Ergebnisse: Kurs Übersichtsseite, Verbesserungen

Diese Messungen ergeben, dass die Verbesserung dieser Seite für eine kleine Anzahl von Fragen wie erwartet marginal ist, obwohl neuerdings zusätzlich zu den bis anhin berechneten Werten auch noch My Difficulty, My Last Attempt und My Rating und zur Darstellung der Top10 Rangliste und des Personal Progress sämtliche Berechnungen der Statistik und der Rangliste berücksichtigt werden. Unter diesem Aspekt ist auch ein marginaler Gewinn ein grosses Plus, da mit der ursprünglichen Implementierung diese Werte in jedem Fall nicht günstiger zu haben gewesen wären. StudentQuiz 2.0.3 startete zudem für jede verfügbare Frage eine neue Datenbankabfrage, sobald sie mindestens einen Tag besass, was zusätzliche Ressourcen anforderte. Die kontinuierlich steigende Last weist auch dieselben Eigenschaften in allen Bereichen ab. Die neue Version hat zwar früher und relativ generell eine höhere TPS vorzuweisen, damit ist jedoch auch der Sättigungspunkt schon früher erreicht. Die Skalierbarkeit durch die Verwendung eines grösseren Fragepools ist in Abschnitt 4.9.1 beschrieben. Sollte man künftig auf dieser Seite eine Performancesteigerung wünschen, sollte man sich überlegen, auf den Top10-Block zu verzichten oder auf diese Daten über einen Cache zuzugreifen.

StudentQuiz Statistikseite

Die Performance ist in der alten StudentQuiz Version miserabel, was auf die in der Analyse im Abschnitt 4.3.9 beschriebene Implementierung zurückzuführen ist. Um zu verifizieren, dass unsere Implementierung die Performance verbessert hat, führen wir auch für die Statistikseite einen Belastungstest durch.

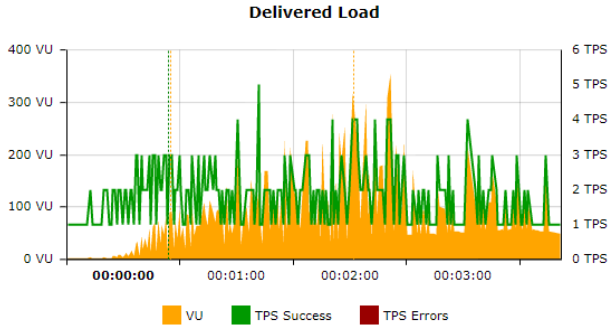
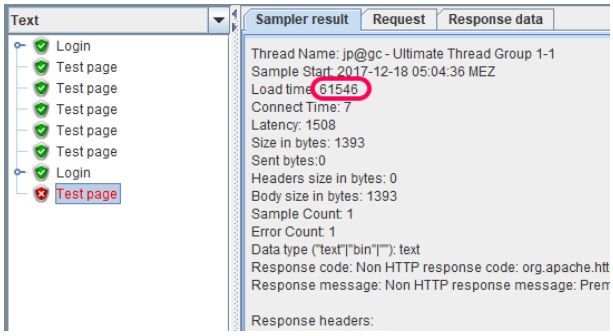
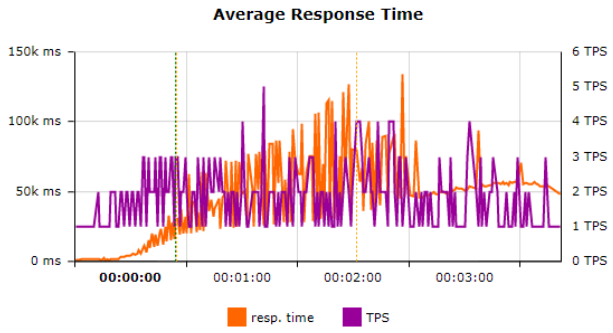
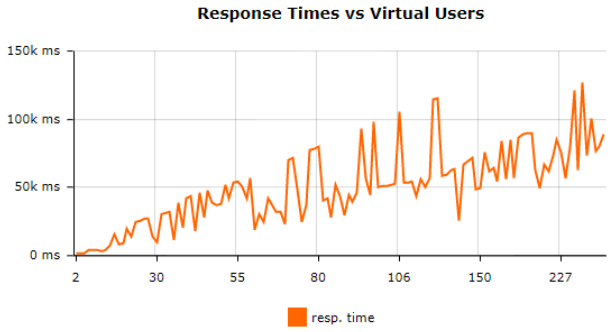
StudentQuiz v2.0.3	StudentQuiz v3.0.0
Testergebnis-Erfassung unmöglich, da bereits nach dem zweiten Benutzer der Server überlastet.	
	
	

Tabelle 9: Performance-Test Ergebnisse: StudentQuiz Statistikseite, isoliert

Metrik	v2.0.3	v3.0.0	Verbesserung
TPS bei 2'000 ms Antwortzeit	k.A.	1-2	k.A.
VU bei 2'000 ms Antwortzeit	k.A.	7	k.A.
TPS bei 5'000 ms Antwortzeit	k.A.	2	k.A.
VU bei 5'000 ms Antwortzeit	k.A.	12	k.A.
Sättigungspunkt erreicht am	k.A.	~00:56	k.A.
Antwortzeit am Sättigungspunkt	k.A.	~29'000 ms	k.A.

Tabelle 10: Performance-Test Ergebnisse: Kurs Übersichtsseite, Verbesserungen

Ohne verwendbare Messwerte der alten Version kann kein Vergleich auf die Performance gemacht werden. Mit diesen Messergebnissen kann zumindest ausgesagt werden, dass diese Seite benutzt werden kann und im groben Vergleich mit der StudentQuiz Übersichtsseite auch, dass sie genauso gut benutzt werden kann.

StudentQuiz Rangliste

Die Rangliste ist vollständigkeitshalber ebenfalls aufgelistet, sie litt jedoch unter denselben Problemen wie die Statistikseite, da beide auf die gleichen Daten auf der Datenbank zugreifen, nur etwas anders aggregiert.

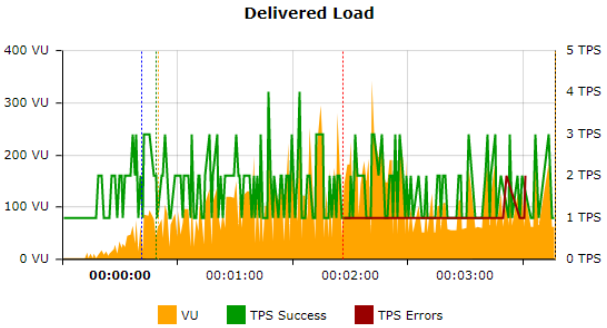
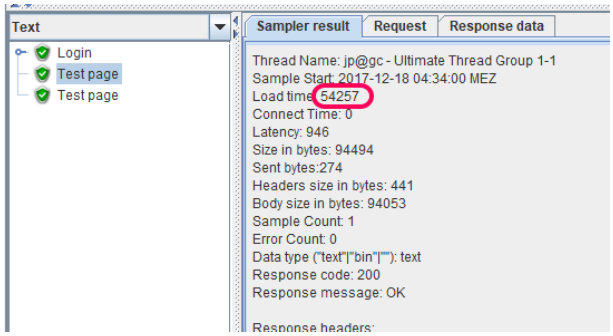
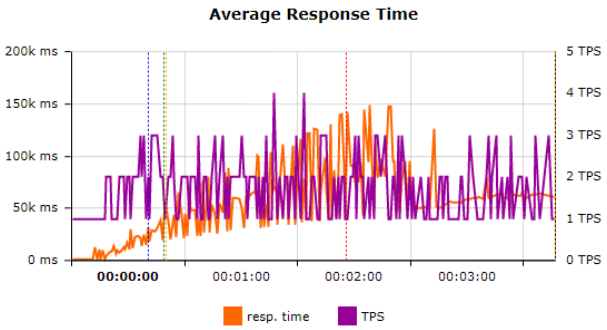
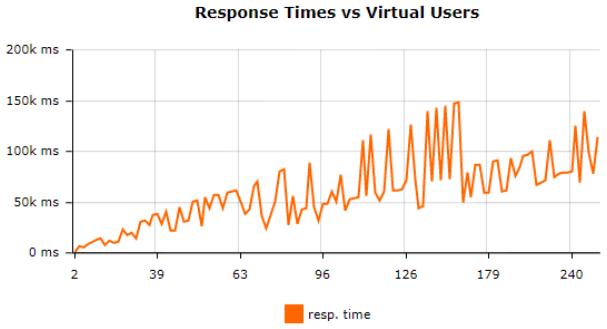
StudentQuiz v2.0.3	StudentQuiz v3.0.0
Testergebnis-Erfassung unmöglich, da jeder Aufruf den Server überlastet.	
	
	

Tabelle 11: Performance-Test Ergebnisse: StudentQuiz Rankingseite, isoliert

Metrik	v2.0.3	v3.0.0	Verbesserung
TPS bei 2'000 ms Antwortzeit	k.A.	1-2	k.A.
VU bei 2'000 ms Antwortzeit	k.A.	9	k.A.
TPS bei 5'000 ms Antwortzeit	k.A.	2	k.A.
VU bei 5'000 ms Antwortzeit	k.A.	18	k.A.
Sättigungspunkt erreicht am	k.A.	~00:51	k.A.
Antwortzeit am Sättigungspunkt	k.A.	~38'000 ms	k.A.

Tabelle 12: Performance-Test Ergebnisse: Kurs Übersichtsseite, Verbesserungen

Auch hier gelten im groben die Erklärungen wie bei der StudentQuiz Statistikseite. Im Vergleich zu ihr, scheint diese Seite zwar etwas mehr VU bedienen und TPS bieten zu können, während der Sättigungszeitpunkt etwas früher mit höherer Antwortzeit erreicht wird.

Aktion "Start Quiz"

Die Aktion "Start Quiz" ist ein zentrales Element dieses Plugins, deshalb darf diese nicht unter Performanceschwierigkeiten leiden. Das alte StudentQuiz hat hierfür bei einem neuen Frageset eine neue Quiz Aktivität erstellt, konfiguriert und dieses Frageset mitgegeben. Entweder um die Anzahl der Quiz Aktivitäten zu reduzieren oder vermutlich als erste Massnahme gegen die Performanceproblemen, wird zuvor geprüft, ob es bereits eine Quiz Aktivität mit diesem Frageset existiert. Im Erfolgsfall wird diese verwendet, statt eine Neue zu generieren. Dieser Problemfall ist in Abschnitt 4.3.3 genauer erläutert.

Die Wahrscheinlichkeit, dass ein Quiz zu einem individuellen Frageset erstellt werden muss, ist durch die Learning Assistance noch höher als es schon ist. Hiermit muss in erster Linie ein Quiz mit einem neuen Frageset performant sein. Das neue StudentQuiz greift nicht auf das Quiz Modul zurück, sondern interagiert mit der Question-Engine direkt, wie in Abschnitt 4.4.1 detailliert beschrieben ist. Die aktuelle Implementierung kennt auch keine Wiederverwendung, somit erstellt diese immer eine neue sogenannte Question Usage.

Im Szenario ist der Test so konfiguriert, dass dem Redirect gefolgt wird und das Ziel des Redirects mit aufgerufen wird. Diese gesamte Aktion gilt als ein Request in der Messung.

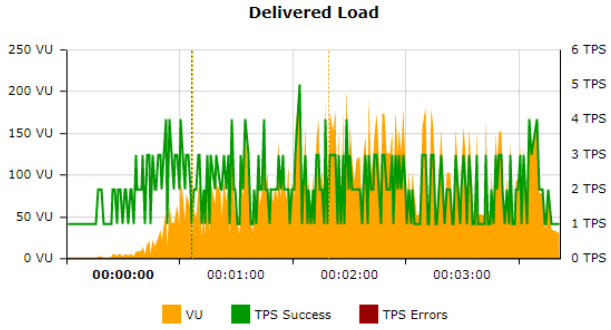
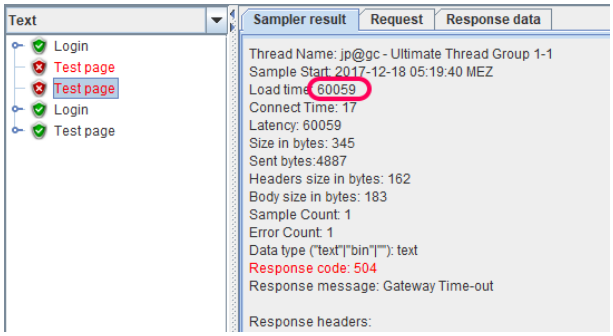
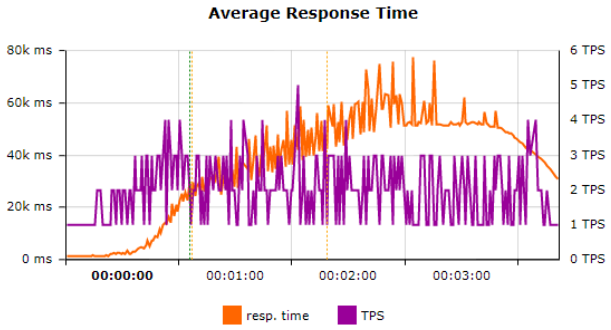
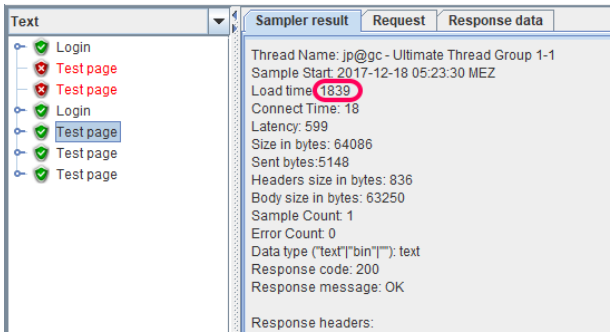
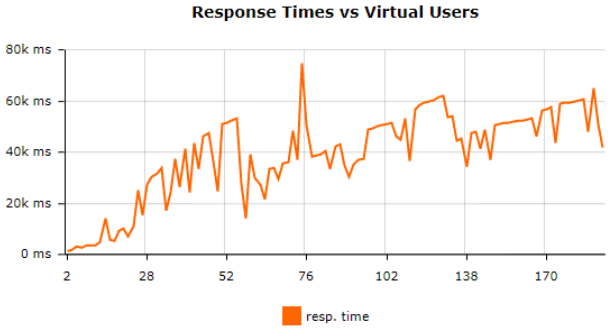
StudentQuiz v2.0.3	StudentQuiz v3.0.0
Testergebnis-Erfassung unmöglich, da der erste Aufruf eines bestimmten Fragesets den Server überlastet. Dieser läuft zwar für wenige Minuten im Hintergrund weiter und erst wenn dieser fertig ist, ist dieses Frageset für einen Benutzer brauchbar.	
	
	

Tabelle 13: Performance-Test Ergebnisse: Aktion “Start Quiz”, isoliert

Metrik	v2.0.3	v3.0.0	Verbesserung
TPS bei 2’000 ms Antwortzeit	k.A.	1-2	k.A.
VU bei 2’000 ms Antwortzeit	k.A.	5	k.A.
TPS bei 5’000 ms Antwortzeit	k.A.	2-3	k.A.
VU bei 5’000 ms Antwortzeit	k.A.	17	k.A.
Sättigungspunkt erreicht am	k.A.	~01:07	k.A.
Antwortzeit am Sättigungspunkt	k.A.	~22’000 ms	k.A.

Tabelle 14: Performance-Test Ergebnisse: Kurs Übersichtsseite, Verbesserungen

Obwohl bei jedem Frageset eine neue Question Usage erstellt wird, können sich diese Werte im Vergleich zu den anderen Testseiten messen und die Aktion ist auf jeden Fall so benutzbar wie diese. Insbesondere ist der Sättigungspunkt relativ spät erreicht unter verhältnismässig etwas geringerer Antwortzeit.

5.6.3 User Acceptance Tests

An dieser Stelle verweisen wir auf die im Anhang C.2 abgelegten Acceptance Tests der vorhergehenden Studienarbeit.

Literatur

[1] Moodle Development Kit. *A collection of tools meant to make developers' lives easier.*
URL: `https://docs.moodle.org/dev/Moodle_Development_kit`
(besucht am 15. September 2017).

[2] Moodle Wikipedia *Deutscher Wikipedia Artikel über Moodle*
URL: `https://de.wikipedia.org/wiki/Moodle`
(besucht am 19. September 2017).

[3] AnkiWeb *Vielseitiges Lernkarteisystem, welches browserbasiert, über eine native App oder auch als Desktopsoftware verwendet werden kann.*
URL: `https://www.ankiweb.net`
(besucht am 19. September 2017).

[4] Memrise *Populärer Web- und App-basierter, multimedialer Vokabeltrainer*
URL: `https://www.memrise.com`
(besucht am 19. September 2017)

[5] Remembr *Webbasierter Vokabeltrainer als Karteikartensystem mit anschaulichen Flashanimationen*
URL: `https://www.remembr.it`
(besucht am 19. September 2017)

[6] Xdebug *Php Erweiterung, welche das Debugging mit Webstorm vereinfacht*
URL: `https://xdebug.org`
(besucht am 22. September 2017)

[7] JetBrains *Anleitungen und Tutorials für die Integration von xDebug mit phpStorm*
URL: `https://www.jetbrains.com/help/phpstorm/configuring-xdebug.html`
(besucht am 25. September 2017)

[8] Moodle *Einführung zum Moodle Badge System*
URL: `https://docs.moodle.org/33/en/Badges`
(besucht am 26. September 2017)

[9] Moodle *Weltweite Nutzungsstatistik*
URL: `https://www.moodle.org`
(besucht am 10. Dezember 2017)

[10] Moodle *Moodle Completion API*
URL: `https://docs.moodle.org/dev/Activity_completion_API`
(besucht am 26. September 2017)

[11] Moodle *Administration via command line*
URL: `https://docs.moodle.org/33/en/Administration_via_command_line`
(besucht am 30. September 2017))

[12] Moodle *Using slash arguments*
URL: `https://docs.moodle.org/33/en/Using_slash_arguments`
(besucht am 30. September 2017)

[13] Moodle *MySQL full unicode support*
URL: https://docs.moodle.org/33/en/MySQL_full_unicode_support
(besucht am 1. Oktober 2017)

[14] Moodle *Mobile Plugin Development*
URL: https://docs.moodle.org/dev/Moodle_Mobile_Plugins_Development
(besucht am 3. Oktober 2017)

[15] Moodle *Plugin files*
URL: https://docs.moodle.org/dev/Plugin_files
(besucht am 5. Oktober 2017)

[16] Github *richarvey/nginx-php-fpm: Repository Layout Guidelines*
URL: https://github.com/richarvey/nginx-php-fpm/blob/master/docs/repo_layout.md
(besucht am 1. Oktober 2017)

[17] Github *richarvey/nginx-php-fpm: Scripting*
URL: https://github.com/richarvey/nginx-php-fpm/blob/master/docs/scripting_templating.md
(besucht am 1. Oktober 2017)

[18] Github *FMCorz/mdk: Moodle Development Kit*
URL: https://github.com/FMCorz/mdk
(besucht am 1. Oktober 2017)

[19] University of California, Irvine *Unofficial Windows Binaries for Python Extension Packages*
URL: http://www.lfd.uci.edu/~gohlke/pythonlibs/#mysql-python
(besucht am 1. Oktober 2017)

[20] Moodle *PHPUnit*
URL: https://docs.moodle.org/dev/PHPUnit
(besucht am 1. Oktober 2017)

[21] Moodle *Running acceptance test*
URL: https://docs.moodle.org/dev/Running_acceptance_test
(besucht am 6. Oktober 2017)

[22] Moodle *Profiling PHP*
URL: https://docs.moodle.org/dev/Profiling_PHP#Tideways_for_php7
(besucht am 7. Oktober 2017)

[23] Github *Tideways PHP Profiler Extension*
URL: https://github.com/tideways/php-profiler-extension
(besucht am 7. Oktober 2017)

[24] tideways *Install Tideways Extension and Daemon*
URL: https://tideways.io/profiler/docs/setup/installation
(besucht am 7. Oktober 2017)

[25] Gihub *programming tool for memory debugging, memory leak detection, and profiling*
URL: https://kcachegrind.github.io/html/Home.html
(besucht am 7. Oktober 2017)

[26] nixCraft *How to set and enable MariaDB slow query log*
URL: https://www.cyberciti.biz/faq/how-to-set-and-enable-mariadb-slow-query-log-linux-unix
(besucht am 8. Oktober 2017)

[27] Percona Toolkit *pt-query-digest, Analyze MySQL queries from logs, processlist, and tcpdump*
URL: https://www.percona.com/doc/percona-toolkit/2.2/pt-query-digest.html
(besucht am 8. Oktober 2017)

[28] Det Norske Veritas Ltd (2002) *Application of QRA in operational safety issues*
URL: http://www.hse.gov.uk/research/rrpdf/rr025.pdf
ISBN: 0 7176 2570 2 (Kapitel 4, Seite 25)

[29] Semantic Versioning 2.0.0 *Semantic Versioning Specification*
URL: https://www.semver.org
(besucht am 11. Dezember 2017)

[30] Read the Docs *Dokumentation erstellen, hosten und durchsuchen.*
URL: https://readthedocs.org
(besucht am 20. Dezember 2017)

[31] Moodle Plugin StudentQuiz *Studienarbeit von Lukas Dürrenberger, Steven Ryser, Alexis Suter*
URL: http://eprints.hsr.ch/555
(besucht am 18. September 2017)

[32] BlazeMeter *JMeter and Performance Testing for DevOps*
URL: https://www.blazemeter.com/
(besucht am 18. Dezember 2017)