Code-Review

Reviewteam:

Johannes Kraml: 01116375

Markus Ninz: 11727598

Linus Wald: 11722746

Softwareteam: Gruppe 5

Proseminargruppe: 3

Datum: 15.05.19

Hinweis: Dieses Dokument versteht sich als Guideline und dient der Strukturierung des Review-Berichts. Bitte halten Sie in diesem Bericht zusammengefassten Beobachtungen, Kritiken, Verbesserungsvorschläge in einer verständlichen und konstruktiven Art und Weise fest. Das Entwicklungsteam muss auf Ihren Review schriftlich Stellung nehmen und gegebenenfalls Änderungen am Source Code vornehmen. Wenn notwendig, untermauern Sie Ihr Review mit Source Code-Auszügen, Screenshots etc.

1. Zusammenfassung

Die Grundstruktur des Codes ist gut verständlich und die meisten Konventionen wurden eingehalten. Das Konzept wurde sehr gut umgesetzt und zeitlich werden die Meilensteine eingehalten. Klassen wurden verständlich benannt und übersichtlich strukturiert. Das Konzept wurde unter Einhaltung des MVC-Patterns umgesetzt.

In den meisten Klassen ist keine Javadoc für Methoden gegeben oder ist unvollständig. Die bereits vorhandenen Test laufen bisher fehlerfrei und können ohne Probleme automatisch ausgeführt werden.

2. Architektur

Die Klassenstruktur folgt dem Konzept, praktisch perfekt. Eine gut umgesetzte Variante ist die Datenbank auf einem externen Server. Das MVC Pattern wird in den entsprechenden Klassen (Bean, Service und Repository) eingehalten und klar strukturiert. Der Ablauf einer Abfrage erfolgt über das MVC Pattern und es werden keine Klassen übersprungen. Keine Beans greifen direkt auf Repository zu. Die HTML Dateien greifen auch immer nur auf entsprechende Beans zu und folgen den Konventionen eines MVC Patterns. Das Konzept ist im Bezug auf die Meilensteine perfekt in der Zeit.

3. Code

Der Code ist im Allgemeinen gut verständlich und hält sich auch im Großen und Ganzen an wichtige Konventionen. Es ist auch ein Vorteil, dass Parameter immer gesetzt werden, auch wenn es die default parameter sind, was dazu führt, dass der Code besser verständlich ist. Die gute Benennung beinahe aller Klassen, Methoden und Variablen unterstützt beim Verständnis des Codes. Die Klassen werden klein gehalten und es gibt keine übermächtigen Gottklassen oder globalen Variablen.

Die einzige Konvention, die gebrochen wird, ist, dass Enums im generellen groß geschrieben wird.

Im QuestionService in Zeile 102 scheint der Gruppe ein Fehler unterlaufen zu sein, da sie auf die 4. Antwort überprüfen und dann die 1. Antwort auf ihre Länge überprüfen.

Im CSVImportBean ist der Code von Zeile 160 - 175 eher schwer verständlich und unübersichtlich, hierbei würde eine for-Schleife anstatt der gehäuften if-Verzweigungen helfen.

Der QuizRoom ist eine sehr große Klasse und, sofern es mit dem Konzept der Gruppe vereinbar ist, wäre es gut diese Klasse zu unterteilen.

4. Sicherheit

Die Inputs werden immer überprüft und auch meistens korrekt weiterverarbeitet. Erstellt ein Manager ein Benutzer, kann er allerdings für den Usernamen nur Leerzeichen setzen. Dies kann zu “leeren” Benutzernamen führen.

Wenn ungültige Daten eingegeben werden, werden diese korrekt behandelt und dem Benutzer wird fast immer mitgeteilt weshalb die EIngabe nicht korrekt ist. Einzige Ausnahme ist, wenn der Manager einen neuen Benutzer mit einem bereits vergebenen Namen erstellen möchte. Dies wird dem Manager korrekterweise verweigert, allerdings wird nicht darauf hingewiesen wieso er ihn nicht erstellen kann. Um die Benutzerfreundlichkeit zu steigern wäre ein Hinweis bzw. Fehlermeldung angebracht.

Zudem können Benutzer ihr Passwort nicht ändern. Um ihre Sicherheit sicher zu stellen wäre es angebracht das Passwort ändern zu können.

5. Dokumentation

Bei den Methoden der Repositories wird nicht einheitlich dokumentiert. Es werden nicht alle eigens erstellten Abfragen dokumentiert und teilweise fehlen die Beschreibungen der Parameter und der Rückgabewerte. Die Javadoc Kommentare für die Service Methoden beschreibt die Methoden ausführlich, aber es fehlt bei fast allen Methoden die Beschreibung der Parameter und der Rückgabewerte. Auch in den Klassen des ui Ordners werden Methoden nicht dokumentiert. Dadurch ist es teilweise schwer nachvollziehbar für was eine Methode gedacht ist oder was ihre Funktion ist. Es gibt teilweise Methoden oder Code-Stücke, die auskommentiert sind beispielsweise in QuestionServiceTest. In Klassen mit vielen Attributen (beispielsweise QuizRoom), werden diese für die Verständlichkeit und Übersichtlichkeit ausführlich beschrieben.

6. Tests

Es sind 116 Tests implementiert mit einer Code Coverage von 29% aller Methoden und 27% aller Lines of Code. Bezogen auf die Meilensteine liegt das Team gut in der Zeit da die gewünschte Testabdeckung mit Meilenstein 5 erreicht werden soll.

Von den implementierten Tests ist keiner mit der Annotation @Ignore-Test annotiert. Es werden alle implementierten Tests ausgeführt und dies erfolgreich beendet. Das Ausführen funktioniert automatisch nach dem Start der Tests. Die getesteten Methoden werden sinnvoll und teilweise auch durch mehrere Tests überprüft, des Weiteren sind die einzelnen Testmetoden nicht zu lang.

Leichter Mangel (Code)	Code könnte an manchen Stellen refactored werden.
Leichter Mangel (Code)	Wenn möglich Große Klassen aufteilen (QuizRoom)
Mittlerer Mangel (Sicherheit)	Ein Manager oder User kann nicht sein Passwort ändern.
Mittlerer Mangel (Dokumentation)	Es fehlt bei vielen Methoden eine Javadoc bzw. bereits vorhandenen Kommentare sind nicht vollständig. Häufig werden Parameter und Rückgabewerte nicht beschrieben.
Leichter Mangel (Dokumentation)	In einigen Klassen gibt es auskommentierte Methoden (werden vielleicht noch verwendet?)
Leichter Mangel (Tests)	In den Testklassen gibt es teileweis auskommentierte Testmethoden