PS Softwareentwicklung und Projektmanagement

Code-Review

Reviewteam: Review-Team 1B

Name: Christoph Wirnsperger Matrikelnummer: 1416422

Name: Pfeifhofer Martin Matrikelnummer: 1416200

Softwareteam: qepssepmss2017 / ps4_team1

Proseminargruppe: 703036-4 17S PS3

Datum: 17.05.2017

Hinweis: Gestalten Sie den Review in konstruktiver Art und Weise. Konkrete Vorschläge können anhand von Kommentaren und Vorschläge im Konzeptpapier selbst gemacht werden. Bitte beachten Sie, dass das Projektteam auf Ihren Review schriftlich Stellung nehmen muss.

1. Zusammenfassung

Im Großen und Ganzen ist der Code gut strukturiert und verständlich. Uns ist aufgefallen, dass der Code sehr gut dokumentiert ist (fast bei jeder Methode findet man eine passende Beschreibung, welche kurz und knapp erklärt, was hier passiert). Leider hatten wir große Probleme beim Ausführen des Projekts, es war uns nicht möglich das Projekt überhaupt zum Laufen zu bringen, unserer Meinung nach ist die Portierbarkeit derzeit noch stark eingeschränkt. Im Git wurde leider auch nicht besonders viel dokumentiert, wie man am besten vorgehen sollte um das Projekt ohne großen Aufwand zum Laufen zu bringen. Dann wäre da noch eine mysteriöse "bootstrap.sh"-Datei. Wieso braucht man diese? In dieser Datei sind Shell-Befehle, die ein Windows Rechner nicht ausführen kann.

Im Allgemeinen ist der Gesamteindruck jedoch sehr positiv. Das Projektteam ist schon sehr fortgeschritten und hat (vermutlich) als einziges Team JUnitTest bereits eingebaut. Außerdem sollte man die Grafische Oberfläche erwähnen, da ein roter Faden erkennbar ist und ein besonderes Augenmerk auf die „User Expereince“ geworfen wird. Grundsätzlich sollte das Team redundante Dateinamen überdenken, da diese doch sehr verwirrend sind. Außerdem bestehen noch einige Mängel bei der Sicherheit der Applikation. Zum Schluss würden wir uns noch wünschen für das letzte Review eine lauffähige Version zu bekommen oder zumindest weitere Instruktionen. Falls es Unklarheiten gibt, bitten wir das Team ein Gespräch mit uns zu suchen, wir stehen bei Fragen gerne zur Verfügung.

2. Architektur

Gegenüber der uns zum Review vorgelegten Konzept-Architektur vom 29.03.2017 gibt es auf den ersten Blick wenige Unterschiede. Das Spring Framework bildet die Basis des Systems (Skeleton Projekt) und liefert auch das User-Interface mit PrimeFaces bzw. CSS.

Ein Vagrantfile liegt dem Projekt wie angefordert bei, welches eine VM für die Datenbank installiert. Die Architektur ähnelt sehr stark dem des Konzepts, auch wenn im Konzept immer noch „H2“ als Datenbank angeführt wird.

Der Model View Controller wird beim Spring Framework fast schon vorausgesetzt. Vor allem das Skeleton Projekt trennt „Models“ von den „Controllern“. Bis zum jetzigen Zeitpunkt wurde dieses Konzept auch so beibehalten. Generell wirkt die interne Struktur des Projekts sehr aufgeräumt und die einzelnen Klassen sind nicht mit Code überfüllt.

Uns verwundert jedoch die Datei „bootstap.sh“ ein wenig, da wir ja grundsätzlich kein Bootstrap verwenden dürfen. Auch wenn die Datei offensichtlich nicht mit einem Bootstrap zu tun hat, ist die Bezeichnung doch sehr irreführend.

Des Weiteren ist die Namensgebung von einigen „.xhtml“-Dateien redundant. Zum Beispiel, wenn ein Ordner mit „Admin“ betitelt wird muss in den einzelnen Dateinamen der Zusatz „Admin“ (sogar doppelt) nicht mehr verwendet werden.

Zum jetzigen Zeitpunkt sind 30 „Issues“ noch ausständig (von gesamt 63 „Issues“), das entspricht einem Projektfortschritt von knapp 52%. Natürlich sollte man solche Angaben ein wenig kritisch betrachten, da zum Ende hin weitere Probleme auftauchen könnten, die vielleicht viel Zeit in Anspruch nehmen können.

Der Zeitplan stimmt mit dem jetzigen Stand des Projekts größtenteils überein, da noch einige Features fehlen. Es ist grundsätzlich schwer zu sagen, ob dieses Team nach dem Zeitplan/Meilensteinplan ihres Konzepts arbeitet, da einige offene Features noch fehlen. Im Meilensteinplan des Konzepts wurde auf die Umsetzung der einzelnen Features nur sehr oberflächlich eingegangen.

3. Code

Im Großen und Ganzen ist der Code gut strukturiert und verständlich aufgebaut. Zudem hält das Team sich an die Oracle Java Code Conventions. Redundanter Code hält sich in Grenzen, da die Klassen sehr gut aufeinander abgestimmt sind.

Es gibt verhältnismäßig viele Klassen die oftmals nur wenig Code beinhalten. Durch die Ordnerstruktur sind die Zusammenhänge dieser Klassen jedoch intuitiv verständlich. Übervolle Klassen gibt es unserer Meinung nach keine. In Hinsicht auf Design Patterns konnten wir nur das MVC-Pattern erkennen, welches durch das Spring Framework realisiert wird.

Unnötige Globale Variablen wurden keine verwendet und auch sogenannte Gott-Klassen konnten wir keine finden. Dies kommt wahrscheinlich daher, da die Klassen generell gut strukturiert und aufeinander abgestimmt sind. Im vorhandenen Code könnte man unserer Meinung nichts durch eine Library ersetzen.

Wir sind auf einige unnötige Debug-Funktionen gestoßen, welche durch den Befehl "System.out.println()" Werte auf der Konsole ausgeben. In der Klasse "LoginHandler" wird ein Logger verwendet. Methoden- und Klassennamen sind größtenteils aussagekräftig und verständlich, obwohl in den meisten Klassen mit Kommentaren meist sparsam umgegangen wird. Bei jeglicher Namensgebung wurde konsequent die Camelcase-Schreibweise umgesetzt.

Auf die Frage, ob der Code Funktionalitäten implementiert, die von einer Library ersetzt werden könnten, müssen wir eindeutig mit JA antworten. Es gibt heutzutage doch Libraries für jede noch so Kleinigkeit, also wird es bestimmt auch Libraries geben, die das gleiche machen, wie Codesegmente in dem Projekt. Uns ist jedoch nichts Dramatisches aufgefallen, das man unbedingt mit einer Library implementieren hätte müssen.

4. Sicherheit

In den „.xhtml“-Dateien werden die meisten Input-Felder - bei denen es auch notwendig ist - mit einer Maske versehen, um die Eingabe zu überprüfen. Diese Daten werden dann weitergereicht, aber in den Controllern wird sonst nichts weiter überprüft. Im Repository werden Inputs wiederum sehr genau überprüft.

E-Mail Adressen werden nicht auf ihre Genauigkeit überprüft, beispielsweise ob sich ein @-Zeichen im eingegebenen String befindet.

Passwörter werden verschlüsselt gespeichert mit „BCrypt“, sodass teilweise Datensicherheit gewährleistet wird. Andere Daten, wie zum Beispiel das Geburtsdatum wird nicht „gehasht“ und somit eins zu eins in die Datenbank gespeichert. Somit ist das System anfällig für Datenklau.

Falls durch die Eingabe falscher Daten eine Exception geworfen wird, kommt es zu keiner adäquaten Behandlung dieser.

In diesem Stadium des Projekts wurde wahrscheinlich Priorität auf andere Dinge gesetzt, jedoch sollte bis zur Veröffentlichung der Applikation, ein gewisser Sicherheitsstandard erreicht werden.

5. Dokumentation

In diesem Projekt ist eine großteils vorbildliche Dokumentation vorzufinden. Durch eine gute Strukturierung des Projekts wird dieser Eindruck abermals verstärkt. Einige Klassen wurden sehr gut dokumentiert, jedoch andere Klassen gar nicht. Wir vermuten aber, dass sich dies von Programmierer zu Programmierer unterscheidet. Die einzelnen „Getter"- und „Setter-Methoden" werden nicht beschrieben, das ist aber in unseren Augen keinesfalls notwendig.

Wir denken, dass sich dieses Team noch wenig mit Corner Cases befasst hat, bzw. noch keine konkreten Fälle implementiert hat und somit auch keine Dokumentation zu diesem Punkt vorweisen kann.

In der Datei „pom.xml" wurden einige „Dependencies" hinzugefügt, jedoch wurde nicht dokumentiert, welchen Nutzen diese Libraries haben. Es wurde ausschließlich kommentiert, wenn eine Library bisher noch nicht verwendet wurde.

Es werden einige Datenstrukturen verwendet, jedoch wurde auch davon nichts dokumentiert.

Uns ist aufgefallen, dass in mehreren Klassen auskommentierter Code existiert, und zwar in einigen Models, jedoch mussten wir danach suchen. Dies ist in diesem Stadium des Projekts nachvollziehbar.

6. Tests

Es sind einige Testklassen vorhanden, wie zum Beispiel „ChildServiceTest" und „ParentConstraintTest", es gibt aber auch Testklassen, die ausschließlich auskommentierten Code beinhalten. Wir nehmen an, dass diese Testklassen alle korrekt funktionieren, da diese sonst auskommentiert sein würden. @Ignore-Annotation bei Tests wurden nicht verwendet.

Der Abdeckungsgrad der Tests ist recht niedrig, da nur für bestimmte wenige Klassen Tests geschrieben wurden. Grundsätzlich finden wir aber, dass der Code testbar programmiert wurde.

Die meisten Tests sind sehr kurz und verständlich geschrieben. Ein paar wenige sind länger, aber auch diese sind verständlich.

Wir hatten große Probleme bei der Ausführung des Systems und deswegen war es uns nicht möglich die Tests auf Lauffähigkeit zu überprüfen. Wir denken jedoch, dass es nicht unsere Aufgabe ist, nach Fehlern im System zu suchen, um herauszufinden, warum das Projekt für uns nicht ausführbar ist.

Hinweis:
Die Verwendung von PMD[1], Findbugs[2], etc bereits zur Entwicklungszeit erleichtert die Einhaltung einheitlicher Programmierstandards und den Review des Source Codes. Sollte Ihr Team eines dieser Tools verwenden, so ist dies den Reviewern mitzuteilen und die entsprechende Konfiguration bereitzustellen.

---

[1] http://pmd.sourceforge.net/eclipse/

[2] http://findbugs. http://pmd.sourceforge.net/eclipse/ sourceforge.net/