Studienarbeit im Fach „Software Engineering 2“ – Thema: Notenrechner

Markus Österle
Maximillian Schreiber
Tobias Schmidbauer
Stefan Memmel
Christoph Kammerer

Inhaltsverzeichnis

1 Einleitung & Motivation

2 Lasten- und Pflichtenheft

Anforderungsliste an das Projekt „Notenrechner“:

- 2 Frontends (Verwaltung & Studierende)
- individualisierbare Notenliste/-berechnung pro Jahrgang
- Studiengangspezifisch, d.h. Programm nur für Vinf oder allgemeingültig?
- Wenn allgemeingültig müsste der Administrator in der Lage sein Studiengänge mit Spezifikation zu erstellen - kann schwierig werden
- Ablage der Noten in einer Datenbank -> Diskussionsbedarf, SQL oder NoSQL
- Generierung von Testdaten (Mock data) für die Datenbank
  - Ist möglich durch CSV Dateien der ersten Semester
- Ablage der Noten kann nur durch den Administrator/Dozenten erfolgen
- graphische Aufbereitung der Noten in Diagrammen
- statistische Kennzahlen berechen (Standartabweichung, Durchschnittsnote,...)
- Statistikfunktionen für den Jahrgang
- Farbliche Abhebung der Noten, ob durchgefallen (rot), über- (grün)/unterdurchschnittlich (gelb) usw...
- Durchschnittsnote für alle sichtbar (kann bedenklich sein wenn nur zwei Studenten das Fach geschrieben haben)
- Authentifizierung notwendig über JAAS
- Desktop Client mit JavaFX (optional?)
- Umsetzung in "gesprochene Noten"
- Eintragung von Traumnoten"der Studenten und anschließende Berechnung der resultierenden/prognostizierendenËndnote, die überschrieben werden durch Eintragung der echten Note durch den Dozenten
- Technologie fuer die Abhaengigkeitsverwaltung?

  - Ant
  - Maven
  - Gradle
- Mobile Devices über App oder AngularJS? Wenn App, welche Plattformen?

2.1 Lastenheft

2.1.1 Zielbestimmung

Es soll eine Software entwickelt werden, die

2.1.2 Produkteinsatz

2.1.3 Produktübersicht

2.1.4 Produktfunktionen

2.1.5 Produktdaten

2.1.6 Produktleistungen

2.1.7 Qualitätsanforderungen

2.1.8 Ergänzungen

2.2 Pflichtenheft

2.2.1 Zielbestimmung

Musskriterien

Wunschkriterien

Abgrenzungskriterien

2.2.2 Produkteinsatz

Anwendungsbereiche

Zielgruppen

Betriebsbedingungen

2.2.3 Produktumgebung

Software

Hardware

Orgware

Produkt – Schnittstellen

2.2.4 Produktfunktionen

Produktspezifisch

2.2.5 Produktdaten

Produktspezifisch

2.2.6 Produkt - Leistungen

2.2.7 Benutzungsoberfläche

2.2.8 Qualitäts-Zielbestimmung

2.2.9 Globale Testszenarien/Testfälle

2.2.10 Entwicklungsumgebung

3 verwendete Technologien

Im ersten Meeting des Teams wurden die zu verwendenden Softwareversionen festgelegt, diese wurden im Verlaufe des Projekts nur noch aufgrund äußerer Begebenheiten (bekannte Fehler mit Fix in höherere Version, finale Version) angepasst. Die jeweilige festgelegte Version und warum diese unter Umständen noch angepasst wurde findet sich im jeweiligen Unterpunkt.

3.1 Entwicklung

3.1.1 Java SDK

Als Java Umgebung kam während der ganzen Projektdauer das Oracle Java SDK Version 8 Update 60 zum Einsatz.

3.1.2 Entwicklungsumgebung - Netbeans

festgelegte Version: **8.1RC2**
während des Projektverlaufs verändert? **ja**
Grund: **erscheinen der finalen Version**
geändert zu Version: **8.1 final**

3.1.3 SQL Editor - MySQL Workbench

3.1.4 Versionsverwaltung - GIT

3.2 Bibliotheksverwaltung mit Maven

3.3 Test

3.4 Unit Tests mit JUnit

3.5 Continous Integration Tests mit Travis, Jenkins und Sonarqube

3.6 Übersicht über die final verwendeten Versionen

Software	Version
Oracle Java SDK	8u60
Java EE	7
Netbeans	8.1
MySQL	
MySQL Workbench	6.3
Wildfly Application Server	9.0.1

4 Teamstruktur und Arbeitsverteilung

4.1 Gemeinsame Codeentwicklung mit GIT

Es wurde auf ein gemeinsames Github-Repository entwickelt von dem sich jeder im Team einen eigenen Fork erstellt hatte, entwickelter Code wurde per Pull Request an den Eigner des Hauptrepositories geschickt und von diesem nach erfolgreichen CI-Tests in den Hauptzweig gemerged. Diese Vorgehensweise hat sich nach einigen anfänglichen Schwierigkeiten als die sicherste herausgestellt, da Code der zu Fehlern im Build-Prozess führt vor dem Zusammenführen erkannt und nachgebessert werden kann und somit nicht der komplette Master unbrauchbar wird.

4.2 Arbeitsverteilung

Scrum like Wöchentliche Meetings

4.2.1 Protokolle der wöchentlichen Meetings

5 (realisierte) Funktionalitäten

6 Einsatz der Software

6.1 Systemvoraussetzungen

6.2 Installation