Universität Leipzig

Softwaretechnik-Praktikum

Qualitätssicherungsplan Gruppe cz17a - Gamification

Lisa Vogelsberg, Felix Fink, Michael Fritz, Thomas Gerbert, Steven Lehmann, Fabian Ziegner, Willy Steinbach, Christian Schlecht

supervised by
Dr. Christian Zinke, Julia Friedrich, Christian Frommert

6. Januar 2018

Inhaltsverzeichnis

1 Dokumentationskonzept

1.1 Entwurfsbeschreibung

In der Entwurfsbeschreibung, welche bei Release mit ausgeliefert wird, ist der aktuelle Entwurfsfortschritt dokumentiert und begründet. Dies bezieht sich vor allem auf Datenmodell sowie verwendete Softwarearchitekturen. Die Entwurfsbeschreibung bietet einen Gesamtüberblick über das ausgelieferte Release und dessen Umsetzung. Sie ist damit das zentrale Dokument, mit der sich Dritte als Erstes auseinandersetzen. Dementsprechend muss von Details abstrahiert werden, da sonst wichtigere („größere“) Grundkonzepte verloren gehen könnten.

1.2 strukturelle Dokumentation

Die strukturelle Dokumentation des Programms erfolgt mittels JavaDoc. Somit wird eine detailierte Beschreibung automatisch als Web-Dienst zur Verfügung gestellt. Es wird vereinbart, dass jede existierende Funktion mittels JavaDoc kommentiert sein muss. Dabei

muss so dokumentiert werden, dass auch Dritte verstehen, was die Funktion ausführt. Es darf nicht auf projektinterne Konventionen zurückgegriffen werden (falls doch nötig, müssen diese mit ausgeführt werden). Es ist speziell auf Parameter und Return-Werte zu achten.

1.3 Code-Kommentierung

Neben der Dokumentation mittels JavaDoc müssen zusätzliche Inline-Kommentare existieren, um spezielle Vorgänge/Funktionen zu erläutern. Solche Kommentare sollen ebenfalls möglichst nicht auf projektinterne Konventionen zurückgreifen und möglichst einfach sein bzw. möglichst wenig Vorkenntnisse voraussetzen. Verweise auf andere Kommentare sind zu vermeiden.

1.4 Sprachgebrauch

Code, Dokumentation und Kommentare werden in englischer Sprache verfasst. Alle anderen Dokumente, wie z.b. Entwurfsbeschreibung werden in Deutsch verfasst.

2 Testkonzept

2.1 Unit-Tests

Es werden für einzelne Funktionen Unit-Test mittels JUnit durchgeführt. Diese sind von jedem zu seinem individuellen Teil anzufertigen und ins Git zu pushen. Diese sollten möglichst eine große Breite der Funktionalitäten abdecken, um Fehleranfälligkeit zu vermeiden.
Im Rahmen eines Continous Integration Prozesses werden die Unit-Test mittels Jenkins automatisch im Gitlab eingebunden.

2.2 manuelle Tests

Der Projektleiter führt in regelmäßigen Abständen manuelle Test durch, um Funktionalität “im Standardgebrauch“ zu garantieren. Besonders soll hierbei auf das User Interface geachtet werden.

3 Organisatorische Festlegungen

3.1 Fertigstellung vor Releasetermin

Die eigentliche Programmierung soll mindestens vier Tage vor Abgabezeitpunkt des Releasebündels abgeschlossen sein, um anschließend umfangreiche Test durchführen zu können, sowie die Dokumentation fertigzustellen.

3.2 Kommunikationskanal

Um angemessenen Kommunikationsfluss zu gewährleisten ist der Hauptkanal eine Whatsapp-Gruppe. Der Kontakt zu den Betreuern wird per E-Mail gehalten.

3.3 Coding Standards

Um einheitlichen und guten Code zu produzieren werden folgende Konventionen festgelegt:

- Variablen und Funktionen müssen *eindeutig* durch ihren Name beschrieben werden. Eines Ausnahme bildet Function-Overloading. Für die Namensgebung wird CamelCase verwendet.
- „Magic Numbers“ dürfen nicht auftreten, es sei denn ihre Bedeutung ist unmittelbar durch einen Kommentar geregelt. Ansonsten ist die Variablendeklaration und anschließende Verwendung dieser Variable vorzuziehen.
- guter Code ist kurz und prägnant, jedoch sollten nicht mehr als 2-3 Statements auf einer Zeile auftreten.
- Als Einrückungszeichen wird ein Tabulator pro Layer vereinbart.

3.4 Issueverwaltung

Das Issues-Board im Gitlab wurde in folgende Boards aufgeteilt: Backlog, To Do, Doing, To be Tested, Waiting, Problems und Done. Zu Beginn einer Iteration wird das aktuelle Release in Issues aufgeteilt und ins Backlog gestellt (ggf. spätere Nachverfeinerung). Sobald sich ein Assignee gefunden hat, wird die Issue in To Do verschoben und bei Beginn der Arbeit dementsprechend in Doing. Nach Beendigung verschiebt das Teammitglied seine Issue in die To be Tested Spalte. Erst nach allen Tests sowie der Sichtung des Codes durch ein anderes Teammitglied kann die Verschiebung in Done erfolgen. Zum Ende eines Release werden alle erfüllten Issues in Closed archiviert, nicht erfüllte werden ins Backlog übernommen, um beim nächsten Release umgesetzt zu werden. Die Spalte Waiting dient zur Markierung, dass die Fertigstellung der Issue von einer anderen abhängig ist. In

Problems werden sowohl aufgetretene Bugs, als auch Probleme bei der Umsetzung gelistet und gesondert bearbeitet.