Qualitätssicherungskonzept

Wordpress-Plugin für Ontologiebasierte Datenevaluation

Dominik Michael, Christofer Meinecke, Lukas Janik, Noah Walle, Enno Jertschat, David Geistert, Lukas Rosenkranz

Inhaltsverzeichnis

1. Vorwort

Im Rahmen unserer Aufgabe ein Wordpress-Plugin zum Erstellen und Abrufen von Gebäudeinformationen im Bezug auf ihre Barrierefreiheit zu erstellen, sollen selbstverständlich auch alle damit verbundenen Ressourcen möglichst barrierefrei gestaltet sein.

Ein Schwerpunkt liegt somit beim Sicherstellen genau jener Eigenschaft, insbesondere in Code und Darstellung. Dies setzt ein einfaches Design bzw. leichte Benutzbarkeit und hohe Funktionalität vorraus.

Weiterhin wird auf Wiederverwertbarkeit und Zugänglichkeit des Codes geachtet, da der Auftraggeber das Programm, hinsichtlich der Weiterentwicklung des genutzten Ontologie Systems, evtl. umbauen oder weiterentwickeln will. Darüber hinaus sollte das Programm nach Möglichkeit leicht auf neue Ontologien anzupassen sein.

Dieses Ziel zieht ebenfalls eine saubere Strukturierung von Code und Kommentaren in den Mittelpunkt der Qualitätssicherung.

2. Dokumentationskonzept

Das Dokumentationskonzept erläutert die Festlegungen und Richtlinien im Bezug auf die Dokumentation von Code, Tests und externen Dokumenten. Dies ist wichtig zum schnellen Verstehen des Projektes für fremde Entwickler oder die Auftraggeber. Weiterhin ermöglicht es einen Überblick über mögliche Fehler und den Arbeitsaufwand zu erhalten.

2.1 Sprache

Mit Ausnahme mancher Fachbegriffe, werden Dokumente in Deutsch verfasst, da Auftraggeber und Team deutschsprachig sind. Code und Kommentare hingegen werden in Englisch gehalten, da das Projekt als Open Source Projekt möglichst vielen Leuten zur Verfügung gestellt werden soll.

2.2 Coding Standard

Unser Code richtet sich nach dem 'PSR-2 Coding Style'[1], sowie den 'Web Content Accessibility Guidelines 2.0'[2] der EU von 2008. Dies garantiert einen gut verständlichen Code, sowie barrierefreie Darstellung.

Sollte das Team es als notwendig sehen, von diesen Standards abzuweichen, wird dies markiert bzw. kommentiert und explizit als Ausnahmeregelung festgehalten.

2.3 Änderungsdokumentation

Alle durchlaufenden Änderungen des Projekts müssen vom jeweiligen Bearbeiter mithilfe von Git dokumentiert werden.

[1] http://www.php-fig.org/psr/psr-2/
[2] http://www.w3.org/TR/WCAG20/

2.4 Dokumentation der Tests

Durchgeführte Tests werden vom testenden Teammitglied dokumentiert. Diese Dokumentation umfasst: Name des Testers, den Inhalt des Tests, evtl. eingesetzte Programme und evtl. aufgetretene Fehler.

2.5 Interne Dokumentation

Da kein offizieller Dokumentationsstandard für PHP exisitiert, verwenden wir für die interne Dokumentation des Codes Sami[3]. Sami ist an Javadoc angelehnt, dies kommt dem Java vertrauten Team zugute.

2.6 Externe Dokumentation

Die externe Dokumentation findet in Form eines Handbuches statt, welches über das Wiki bzw. unsere Website zugänglich ist. Das Handbuch beschreibt Aufbau und Funktion des Projektes und sollte es ermöglichen fremden Entwicklern leichter zukünftige Veränderungen vorzunehmen. Zu diesem Zweck müssen auch genaue Nutzungsanweisungen und evtl. Wartungsinstruktionen im Handbuch enthalten sein.

2.7 Weitere Dokumente

Alle externen Dokumente sind über das Wiki zugänglich (http://pcai042.informatik.uni-leipzig.de:1680/Home) und auf Deutsch.

Bevorzugte Formatierung für Textdokumente ist PDF, da deren Änderungen leichter nachvollzogen werden kann, um ungewollten Zugriff abzufangen.

3. Testkonzept

Die Funktionalität des Codes wird mithilfe des Frameworks PHPUnit[4] getestet. Es wird vorrausgesetzt, dass eine Testabdeckung von mindestens 90% eingehalten wird. Des Weiteren kommt CodeSniffer[5]zum Einsatz, um sicher zu stellen, dass im kompletten Projekt der PSR-2 Coding Standard eingehalten wird.

Ein großer Vorteil von PHPUnit sowie CodeSniffer ist die einfache Anbindung der beiden Testing-Tools an die meisten gängigen PHP-Entwicklungsumgebungen in Form von herunterladbaren Plugins. Somit kann der Test einfach und schnell, sowie in kurzen Abständen, während der Entwicklung erfolgen.

3.1 PHPUnit

PHPUnit ist ein freies PHP-Framework, zum Erstellen und Ausführen von Unit-Tests. In unserem Projekt kommt PHPUnit zum Einsatz um die Grundfunktionalität des PHP-Codes automatisiert zu testen. Nach jedem *git push* werden automatisch alle Unit-Tests ausgeführt. Dieses Vorgehen

[3] https://github.com/FriendsOfPHP/Sami
[4] https://phpunit.de/
[5] https://github.com/squizlabs/PHP_CodeSniffer

garantiert, dass Fehler im Code nach einem *push* sofort erkannt werden und von dem Entwickler Team ausgebessert werden können.

Außerdem besteht die Möglichkeit, dass jeder Nutzer seinen Code selbstständig und manuell lokal auf seinem Rechner testet.

3.2 CodeSniffer

CodeSniffer ist ebenfalls ein freies Tool, welches in der Lage ist, den Code auf die Einhaltung eines festgelegten Code Standards zu überprüfen. Bei Verletzungen des Standards wird der entsprechende *git commit* als fehlerhaft markiert.

Durch die automatisierten Tests mittels PHPUnit und CodeSniffer sowie die Möglichkeit seinen Code auf beide Aspekte manuell überprüfen zu können ist ein fehlerfreier Code weitestgehend garantiert.

4. Organisatorische Festlegungen

Organisatorische Festlegungen sind alle wichtigen Regelungen zum Ablauf und der Zusammenarbeit im Team und mit dem Auftraggeber.

4.1 Wöchentliche Scrum-Treffen

Jeden Dienstag wird ein Treffen mit Team, Betreuer und Auftraggeber abgehalten, in welchem der bisherige Fortschritt des Projektes und evtl. Probleme oder Verbesserungsvorschläge besprochen werden. Das Treffen wird mithilfe eines Protokolls dokumentiert. Der Protokollleiter wechselt jede Woche.

4.2 Genutzte Dienste von Drittanbietern

Zur Erleichterung der Kommunikation und Organisation nutzt das Team mehrere Drittprogramme. Hierbei direkt das Projekt betreffend sind: Slack, Taiga und GitLab.

Tutor und Auftraggeber haben über diese Programme ebenfalls Zugriff auf das Projekt.

Zur Textverarbeitung nutzen wir Google Docs, so kann das komplette Team parallel an den Dokumentationsaufgaben arbeiten.

4.3 Bereitstellung der Homepage

Die Homepage des Teams bzw. Projektes (http://pcai042.informatik.uni-leipzig.de/~wpod16/) wird für die Zeit des Praktikums von der Universität Leipzig über ihren Server zur Verfügung gestellt. Dies ist nicht verbindlich und die Universität hat das Recht dem Team die Seite jederzeit zu entziehen.

4.4 Eigenverwaltung jedes Teammitgliedes und Verantwortung der leitenden Rollen

Jedes Mitglied ist selbst verantwortlich für den von ihm geschriebenen Code und die damit verbundenen Kommentare und Tests.

Die Tests sind von jedem Mitglied selbst zu entwerfen, durchzuführen und zu dokumentieren.

Erzielte Ergebnisse sind an den entsprechenden Leiter des Gebietes weiter zu leiten, welcher diese kontrollieren und evtl. Verbesserungen fordern kann (z.B. Testsergebnisse an den Leiter Testing, Code an den Leiter Qualitätssicherung, usw.).

Nach testen der Ergebnisse stellt der Leiter diese anschließend der Gruppe bei einem Treffen vor.