LinkedSpending

SWP14-LS

- Qualitätssicherungskonzept -

Inhaltsverzeichnis

1. Dokumentationskonzept

1.1 Verwendete Sprache

Für das Projekt werden sowohl Englisch als auch Deutsch verwendet. Da LinkedSpending ein Open-Source-Projekt mit potentiell weltweiter Reichweite ist kommt für alle extern zur Verfügung gestellten Dokumente nur Englisch in Frage. Dies beinhaltet sowohl den Quellcode als auch JavaDoc. Da Deutsch die von den Teammitgliedern am besten beherrschte Sprache ist werden alle internen Dokumente in dieser verfasst.

1.2. Coding Conventions

Es wird eine Datei namens codeprofile.xml zentral auf Github hinterlegt. In dieser Datei sind alle einstellbaren Coding Conventions hinterlegt. Die Teammitglieder können diese Datei in ihre IDE importieren um die wesentlichsten Quellcodeformatierungen automatisch anzuwenden.

Für die Namensgebung wird der Camel-Style verwendet. Dabei wird bei Klassennamen der erste Buchstabe großgeschrieben. Methoden- und Variablennamen beginnen mit einem Kleinbuchstaben.

Geschweifte Klammern gehören auf eine separate Zeile.

Konstanten werden komplett großgeschrieben und deren Teilworte mit Unterstrich getrennt.

Vor und nach dem Zuweisungsoperator ist ein Leezeichen zu lassen.

Für jede Einrückung wird ein Abstand von 4 Leerzeichen festgelegt.

Die maximale Zeilenbreite wird auf 120 Zeichen gesetzt

1.3. Kommentare

Für kurze Kommentare wird /*Kommentar*/ verwendet.

Jede Klasse, jedes Attribut und jede Methode werden mit JavaDoc kommentiert.

1.4. Änderungsdokumentation

Jeder Commit wird mit einer aussagekräftigen Beschreibung versehen. Falls möglich ist die Ticket-ID mit anzugeben.

1.5. Benutzerhandbuch

Mit der Arbeit am Handbuch wird parallel zur Implementierungsphase begonnen und unter Aufsicht des Verantwortlichen für Dokumentation sukzessiv weiterentwickelt

1.6. Sonstiges

Lizenz-Disclaimer oder ähnliche Verweise werden nicht verwendet.

2. Testkonzept

2.1 Komponententests

Hierbei werden einzelne Klassen und Methoden auf ihre grundlegende Funktionalität überprüft. Nach jeder Programmmodifikation werden diese durch Maven automatisch durchgeführt. Erst nach erfolgreichem Komponententest sind weiterführende Schritte erlaubt, bei Fehlern wird der Build sofort abgebrochen. Diese Unit-Tests werden schon während der Entwicklung der jeweiligen Klasse vom gleichen Entwickler geschrieben. Es ist darauf zu achten dass diese nicht von anderen Klassen abhängig sind und außerdem schnell ausgeführt werden können.

Für die Tests der einzelnen Klassen kommt JUnit zum Einsatz. Zu den zu testenden Klassen werden Testklassen mit mehreren Testmethoden erzeugt. Die entsprechenden Testklassen befinden sich in einer identischen Ordnerhierarchie wie die Hauptklassen. Die Testklasse von src/main/$package/X.java ist hierbei unter src/test/$package/XTest.java zu finden.

2.2 Integrationstests

Nach erfolgreichem Abschluss des Komponententests erfolgt der Integrationstest. Dieser prüft, ob die einzelnen Komponenten korrekt kommunizieren und zusammenarbeiten. Bei Fehlern werden diese entsprechend vermerkt, der Build wird aber dennoch fortgesetzt. Integrations-Tests sollten von einem komponentenfremden Entwickler geschrieben werden, entweder parallel zur Komponentenentwicklung oder kurz darauf.

2.3 Systemtests

Dieser Test wird im Gegensatz zum Komponenten- und Integrationstest aus der Sicht des Anwenders durchgeführt und nicht aus Sicht des Projektteams. Das Ziel des Systemtestes ist die Überprüfung des Gesamtsystems auf Erfüllung der erwarteten Anforderungen. Es werden die funktionalen als auch die nicht-funktionalen Anforderungen geprüft. Als Basis dieses Testes dient das Pflichtenheft. Ein Teil der Systemtests kann automatisiert werden, zusätzliche manuelle Tests sind aber dennoch Pflicht.

2.4 Abnahmetest

Der Abnahmetest schließt das Testkonzept ab und testet das finale Softwareprodukt. Im Abnahmetest wird die Software vom Auftraggeber auf vertragliche Akzeptanz und Benutzerakzeptanz geprüft.

3. Organisatorische Festlegungen

Das Team trifft sich mindestens einmal in der Woche am Mittwoch um 11:00 Uhr. In der Regel sind hierbei alle Mitglieder anwesend.

Zur Kommunikation für die Zwischenzeit ist jedem Mitglied die E-Mail-Adresse jedes anderen bekannt. Des Weiteren verwendet die Gruppe Skype zur schnellen Absprache.

Zur Versionsverwaltung existiert ein Repository auf Github, auf das jedes Gruppenmitglied Zugriffsrechte hat.

Es finden regelmäßige Codereviews statt. Diese Tätigkeit wird zwar auf alle Mitglieder aufgeteilt, jedoch fällt den erfahreneren Teilnehmern diese Aufgabe häufiger zu. Paarweises Arbeiten an einem Modul unterstützt dies.

Der Verantwortliche für die Dokumentation stellt die Standards für alle Dokumente auf. Jedes Teammitglied ist zwar für die Einhaltung dieser selbst verantwortlich, die Qualität wird jedoch regelmäßig durch den jeweils Verantwortlichen überprüft.

Github stellt ein Ticketverwaltungssystem bereit welches genutzt wird. Wenn möglich sollte zu jeder Arbeit schon ein Ticket vorhanden sein. Etwaige Festlegungen oder Probleme sind dort festzuhalten.

Jedes Mitglied schreibt einen wöchentlichen Aufwandsbericht. Dieser wird dem Projektleiter übergeben welcher sich um dessen Veröffentlichung kümmert.