

**Dokumentationskonzept**

**Vereinbarung für die Qualitätssicherung sowie für die interne und externe Dokumentation**

****

|  |  |
| --- | --- |
| **Team:** | **Edge** |
| **Projektleiter:** | **Kevin Mangold** |
| **Teammitglieder:** | **Severin Neuner, Florian Schmidt, Nicholas Dickel, Tessa Haschtschek** |
| **Datum:** | **10.04.17** |

**Bearbeitungsverlauf**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Autor** | **Kommentar** |
| 1.00 | 23.03.17 | Tessa Haschtschek | Erstellung |
| 1.01 | 03.04.17 | Tessa Haschtschek | Formatanpassungen |
| 1.02 | 10.04.17 | Tessa Haschtschek | Logo und Verantwortlichkeiten |

1. **Coding Conventions**
   1. **Java Coding Conventions**

Als Referenz wurden die Java Coding Conventions der Firma Oracle[[1]](#footnote-1) hergenommen, von deren Website auch die folgenden Informationen stammen.

Die Coding Conventions sind aus mehreren Gründen eine wichtige Investition in die Qualität der Software. Einmal erhöhen die Conventions die Lesbarkeit der Software und erlaubt so anderen Ingenieuren das einfachere Verstehen neuer Software. Außerdem ist kaum eine Software ihre ganze Lebensdauer von ihrem originalen Author gewartet. Und da rund 80% der lebenslangen Kosten in die Wartung der Software fließen, ist es ein wichtiger Faktor, ob diese Wartung so einfach wie möglich vorgenommen werden kann.

In der Java Coding Convention sind Regeln festgeschrieben, die sich auf häufig genutzte Dateiname, Kommentare, Deklarationen, Zeilenlänge, Statements, Leerzeilen/-zeichen, Namensgebung und allgemeine Programmierpraktiken bezieht. Dadurch erhalten die Entwickler eine Übersicht, wie sie ihren Code strukturieren und gestalten sollen.

Der fertiggestellte Code soll wöchentlich auf die Java Coding Conventions überprüft werden und so dessen Qualität gesichert werden. Dies ist einmal durch die Entwickler während des Entwickelns selbst sicherzustellen, abschließend aber auch von dem Verantwortlichen für die Qualitätssicherung bestätigt werden.

**1.2 HTML**

Das Projektteam richtet sich nach den von W3-Schools vorgeschlagenen Coding Conventions[[2]](#footnote-2):

In der ersten Zeile steht immer die Dokumenttypdeklaration. Namen sollten immer komplett klein geschrieben werden, nur in Ausnahmen darf camelCase verwendet werden. Alle Elemente sollen mit einem schließenden Tag versehen werden und auch leere Elemente sollen geschlossen werden. Attribute sollen zwischen Anführungszeichen stehen und bei Bildern soll immer ein Alternativtext angegeben werden. Vor und Nach einem "=" Symbol sollen keine Leerzeichen stehen. In einer Zeile sollen maximal 80 Zeichen pro Zeile stehen und die Blöcke zwischen zwei zusammengehörigen Tags sollen mit zwei Leerzeichen eingerückt werden.

Abschließend sollen in einer HTML Datei die <html>, <head> und <body> Tags immer verwendet werden.

Auch hier soll der Code schon während des Entwickelns bestmöglich den Coding Conventions entsprechend und abschließend vom Verantwortlichen für Qualitätssicherung überprüft werden.

**1.3 CSS**

Auch hier richtet sich das Projektteam nach den Coding Conventions der W3-Schools[[3]](#footnote-3):

Diese schlagen vor, dass Regeln mit weniger als drei Definitionen in eine Zeile geschrieben werden können und alles darüber hinaus auf mehrere Zeilen verteilt werden soll, wobei pro Zeile allerdings nur eine Definition stehen darf. Die öffnende Klammer eines Selektors soll in der gleichen Zeile wie dieser stehen, allerdings mit einem Leerzeichen davor. Eine Definition soll um zwei Leerzeichen eingerückt werden, nach einem Doppelpunkt soll nur ein Leerzeichen folgen. Jede Definition soll mit einem Semikolon abgeschlossen werden. Anführungszeichen sollen nur verwendet werden, wenn der Wert Leerzeichen enthält, ansonsten nicht. Die schließende Klammer eines Selektors soll in einer neuen Zeile ohne Einrückung stehen.

Für die Überprüfung und Verantwortlichkeiten gelten die gleichen Regeln wie bei den Java und HTML Coding Conventions.

1. **Interne Dokumentation**

Die interne Dokumentation dient nur dem Gebrauch innerhalb des Entwicklerteams und soll das Arbeiten mit dem bereits erstellten Code erleichtern. Im Allgemeinen soll der Code alle benötigten Informationen enthalten und selbsterklärend sein, was durch aussagekräftige Namen sichergestellt werden kann. An besonders komplexen Codeabschnitten soll deren Funktion durch Kommentare ergänzend erklärt werden. Dabei ist darauf zu achten, dass die Kommentare dabei auf das Wesentliche beschränkt und redundante Informationen weggelassen werden. Das Hauptmerkmal der internen Dokumentation ist die Erstellung von Dokumentationsdateien durch Javadoc, worauf in Unterkapitel 4.2 noch einmal eingegangen wird.

1. **Externe Dokumentation**

Die externe Dokumentation wird in Form von zwei Dokumenten umgesetzt. Das erste Dokument stellt die Designbeschreibung dar und enthält alle wichtigen Informationen zu den Struktur- und Entwurfsprinzipien. Diese dienen dazu einem Programmierer, der sich bisher nicht mit dem Projekt befasst hat, Änderungen oder Ergänzungen des Quellcodes zu ermöglichen. Auch eine Produktübersicht mit einer Beschreibung der äußerlichen Funktionsmerkmale des Systems ist in dem Dokument enthalten. Das zweite Dokument stellt eine kurze Anleitung zur Handhabung der Anwendung dar und ist für die Benutzer der Applikation relevant, da es die äußerlichen Bestandteile und die Bedienung der Website verständlich erklärt. Zur Projektfertigstellung wird diese Anleitung auf der Website öffentlich zugänglich hinterlegt, um einen einfachen Einstieg zu gewährleisten. Das Dokument selbst kann durch eine FAQ oder ein Videotutorial ergänzt werden.

1. **Werkzeuge zur Unterstützung der Qualitätssicherung**
   1. **Asana**

Das Projektteam hat sich auf die Verwendung des Projektmanagement Tools Asana geeinigt, um die verschiedenen anfallenden Aufgaben gemeinsam organisieren zu können und die Kommunikation und Qualitätssicherung im Team zu vereinfachen. Die Website, die auch als App für mobile Geräte verfügbar ist, bietet umfassende Funktionen an. Zu diesen Funktionen gehört es, den Projektfortschritt jederzeit auf einen Blick feststellen zu können, ohne Meetings einberufen zu müssen oder von jedem Teammitglied einen persönlichen Stand erfragt zu haben. Die Aufgaben können weitreichend konfiguriert und den eigenen Präferenzen angepasst werden, um den passenden Sachverhalt darzustellen. Den Aufgaben können Notizen, Kommentare, Dateien oder Tags hinzugefügt werden und die Benutzer können Projekten oder einzelnen Aufgaben folgen, um Benachrichtigungen über den Fortschritt zu erhalten.

Insgesamt stellt Asana also ein zentrales Werkzeug zur Projektorganisation und dem übergeordneten Management dar, was dem Team Edge das Verwalten der Arbeitsaufgaben hilft.

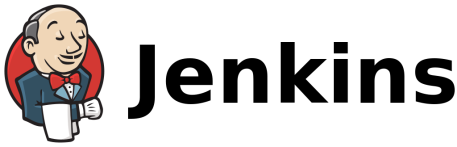
* 1. **Javadoc**

Im Zusammenhang mit der internen Dokumentation wird das Werkzeug Javadoc für dieses Projekt verwendet, welches aus Java-Quelltexten automatisch HTML-Dokumentationsdateien erstellt. Die dadurch erstellte Dokumentation kann durch spezielle Kommentare im Quelltext angereichert werden. Durch verschiedene Tags werden so zum Beispiel Interfaces, Klassen, Methoden, Felder oder ähnliches näher beschrieben. Neben der Standardausgabe in HTML sind auch alternative Ausgabeformen möglich.

Um die Dokumentationsdatei zu erstellen, parst Javadoc die Quelltexte nach allen Javadoc-Kommentaren, die mit ‚/\*\*‘ beginnen. In diesen Kommentaren wird nach den oben genannten Tags gescannt, die mit ‚@‘ oder ‚{@‘ beginnen. Auf eine simple Art und Weise kann also der Programmierer schon während der Entwicklung des Codes mit der Dokumentierung beginnen.

* 1. **JUnit und Jenkins**

JUnit und Jenkins sind zwei testspezifische Werkzeuge, die auch im Recherchebericht schon einmal Erwähnung fanden. JUnit ist ein Framework zum Testen von Java-Programmen. Wie der Name schon sagt, ist es besonders zum automatisierten Testen einzelner Units, also Klassen und Methoden, geeignet. Diese Tests liefern nur zwei Ergebnisse, nämlich entweder das Gelingen oder das Misslingen des Tests. Das Misslingen wird durch einen Fehler (Error) oder ein falsches Ergebnis (Failure) ausgelöst, die beide per Exception signalisiert werden.

Jenkins dagegen ist ein erweiterbares, webbasiertes Software-System zur kontinuierlichen Integration von Komponenten zu einem Anwendungsprogramm. Dieses System wird später zum Testen auf dem Server laufen, der auch die Projektwebsite betreibt. Es unterstützt mehrere Build-Tools, zu denen auch das automatische Testverfahren JUnit gehört.

In Kombination werden die beiden Werkzeuge vom Team Edge als Testumgebung der erstellten Website verwendet und tragen dadurch ebenso zur Qualitätssicherung bei.

1. http://www.oracle.com/technetwork/java/codeconvtoc-136057.html [↑](#footnote-ref-1)
2. https://www.w3schools.com/html/html5\_syntax.asp [↑](#footnote-ref-2)
3. https://www.w3schools.com/html/html5\_syntax.asp [↑](#footnote-ref-3)